# Classic Flang Call - Agenda and Minutes

**Meeting Link**:
https://teams.microsoft.com/l/meetup-join/19%3ameeting_ZTgyNGZkYzQtMjY4Yy00ZTBmLTg3ODgtZWNmMWJiZjE2MDdl%40thread.v2/0?context=%7b%22Tid%22%3a%223dd8961f-e488-4e60-8e11-a82d994e183d%22%2c%22Oid%22%3a%22d5724e89-a25b-4595-b660-ba2e5d968db0%22%7d

**Time**: 10:30AM EDT, 3:30PM BST, 8:00PM IST

**Call Coordinators**
Pawel/Arm : December, March, June, September
Bryan/Huawei : January, April, July, October
Shivaram/AMD : February, May, August, November

## Feb 19, 2025

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan

- Arm is stopping participating in Classic Flang maintenance. The next meeting will be organized in email format.
- LLVM 20 port:
  - Bryan created the release_20x branch with CI workflows, based on llvmorg-20.1.0-rc2 (the downstream branch will be rebased on llvmorg-20.1.0 when it is finalized); PR with remaining LLVM 20 patches will be submitted soon.
- Issue flang#1447: Port Classic Flang to RISC-V
  - Bryan got approval to open-source this code
  - PR flang#1462 is submitted for review. Petr has tried to review/test, but he ran into problems cross-compiling libomp for RISC-V (llvm#210). Bryan will help debug, and try (if time allows) to confirm that Classic Flang can be built on RISC-V native**ly.**
  - **Updates:** Building natively in a RISC-V VM is not feasible. Petr's problem is now resolved.
- Issue flang#1429: 521.wrf_r segfaults when built with -flto=full -Ofast
  - Caused by upstream changes in LLVM 17
  - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
  - ARM could reproduce the issue after the downstream Loop unswitching optimization has been removed during the downstream tech-debt reduction activities.
    - The downstream patch that prevents the wrf segfault https://reviews.llvm.org/D108138

    - The issue **will be on HOLD** now pending further investigation and identifying possible solutions.
    - LLVM flang works fine with the same flagsets.
    - **No updates**
- Issue #1437: malloc(): corrupted top size after allocating array of polymorphic data type elements
    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
    - **Issue is not yet assigned.**
- Issue #1436: Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
    - **Issue is not yet assigned.**
- Issue flang#399: bind(C) function with c_char leads to ICE
  Using BIND(C) with CHARACTER(1), which is an interoperable type, causes a crash in flang2. For non-interoperable types like CHARACTER(N), we need to throw a semantic error until F2018 is supported. Bryan hopes to do further analysis about this issue.
    - Problem discovered while building DBCSR.
    - Reference: Modern Fortran: Features for High-Performance Computing (stevelionel.com)
    - **No update this week. This is a low-priority issue.**
- Other PRs
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; the author promised to address it after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
        - **No update this week.**

## Feb 5, 2025

**Meeting Cancelled due to insufficient agenda**

## Jan 22, 2025

**Meeting Cancelled due to insufficient agenda**

## Jan 8, 2025

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan

- Issue flang#1447: Port Classic Flang to RISC-V
    - Bryan got approval to open-source this code
    - PR flang#1462 is submitted for review. Petr has tried to review/test, but he ran into problems cross-compiling libomp for RISC-V. Bryan will help debug, and try (if time allows) to confirm that Classic Flang can be built on RISC-V natively.
- Issue flang#1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
    - ARM could reproduce the issue after the downstream Loop unswitching optimization has been removed during the downstream tech-debt reduction activities.
        - The downstream patch that prevents the wrf segfault https://reviews.llvm.org/D108138
    - The issue **will be on HOLD** now pending further investigation and identifying possible solutions.
    - **No updates**
- Issue #1437: malloc(): corrupted top size after allocating array of polymorphic data type elements
    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
    - **Issue is not yet assigned.**
- Issue #1436: Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
    - **Issue is not yet assigned.**
- Issue flang#399: bind(C) function with c_char leads to ICE
  Using BIND(C) with CHARACTER(1), which is an interoperable type, causes a crash in flang2. For non-interoperable types like CHARACTER(N), we need to throw a semantic error until F2018 is supported. Bryan hopes to do further analysis about this issue.
    - Problem discovered while building DBCSR.

    - Reference: Modern Fortran: Features for High-Performance Computing (stevelionel.com)
    - **No update this week. This is a low-priority issue.**
- Other PRs
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; the author promised to address it after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
        - **No update this week.**

## Dec 11, 2024

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan

- Next meeting is on 8th January 2025.
- Issue flang#1458 (**Closed**): Incorrect no alias by TBAA due to a problem with metadata tags.Currently, functions within the same module are assigned unique TBAA type system IDs based on their order, but functions in different modules can unintentionally receive the same ID, leading to incorrect non-aliasing results.
    - PR flang#1460 (**Merged**) addresses this issue.
- Issue flang#1447: Port Classic Flang to RISC-V
    - Bryan got approval to open-source this code;
    - PR flang#1462 is submitted for review.
    - AArch64 CI is failing (should be passing as llvm#205 has been merged) due to a CMake version problem. Submitted llvm#208 to address this in release_18x. Also added a commit to flang#1462 to replace the release_17x CI jobs with release_19x jobs.
- Issue flang#1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
    - ARM could reproduce the issue after the downstream Loop unswitching optimization has been removed during the downstream tech-debt reduction activities.
        - The downstream patch that prevents the wrf segfault https://reviews.llvm.org/D108138
    - The issue **will be on HOLD** now pending further investigation and identifying possible solutions.
    - **No updates**

- Issue #1437: malloc(): corrupted top size after allocating array of polymorphic data type elements
    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
    - **Issue is not yet assigned.**
- Issue #1436: Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
    - **Issue is not yet assigned.**
- Issue flang#399: bind(C) function with c_char leads to ICE
  Using BIND(C) with CHARACTER(1), which is an interoperable type, causes a crash in flang2. For non-interoperable types like CHARACTER(N), we need to throw a semantic error until F2018 is supported. Bryan hopes to do further analysis about this issue.
    - Problem discovered while building DBCSR.
    - Reference: Modern Fortran: Features for High-Performance Computing (stevelionel.com)
    - **No update this week. This is a low-priority issue.**
- Other PRs
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; the author promised to address it after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
        - **No update this week.**

## Nov 27, 2024

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan
NVIDIA:

- 2nd meeting of December (25th December) will be cancelled.
- Issue flang#1458: Incorrect no alias by TBAA due to a problem with metadata tags.Currently, functions within the same module are assigned unique TBAA type system IDs based on their order, but functions in different modules can unintentionally receive the same ID, leading to incorrect non-aliasing results. PR flang#1460 addresses this issue. **-**

**Reviews needed and CI is failing. Bryan will help Author to remove the dependency on lld and that will resolve the issue.**
- **Bryan and Shivarama has comments - need to be addressed by author**

- Issue flang#1447: Port Classic Flang to RISC-V
  - Bryan got approval to open-source this code; PR flang#1462 is submitted for review. CI is failing (will pass after llvm#205).
    Need reviews.
  - llvm#205 Enable Classic Flang cross-compilation, e.g. from X86 to AArch64
  - Driver changes to enable cross compilation of compiler.
    **Approved and merged.**

- Issue flang#1429: 521.wrf_r segfaults when built with -flto=full -Ofast
  - Caused by upstream changes in LLVM 17
  - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
  - ARM could reproduce the issue after the downstream Loop unswitching optimization has been removed during the downstream tech-debt reduction activities.
    - The downstream patch that prevents the wrf segfault https://reviews.llvm.org/D108138
  - The issue **will be on HOLD** now pending further investigation and identifying possible solutions.
  - **No updates**
- Issue #1437: malloc(): corrupted top size after allocating array of polymorphic data type elements
  - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
  - **Issue is not yet assigned.**
- Issue #1436: Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
  - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
  - **Issue is not yet assigned.**
- Issue flang#399: bind(C) function with c_char leads to ICE
  Using BIND(C) with CHARACTER(1), which is an interoperable type, causes a crash in flang2. For non-interoperable types like CHARACTER(N), we need to throw a semantic error until F2018 is supported. Bryan hopes to do further analysis about this issue.
  - Problem discovered while building DBCSR.

    - Reference: [Modern Fortran: Features for High-Performance Computing (stevelionel.com)]()
    - **No update this week. This is a low-priority issue.**
- Other PRs
    - [flang#1418](): Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; the author promised to address it after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
        - **No update this week.**

## Nov 13, 2024

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan
NVIDIA:

- LLVM 19 port:
    - [llvm#196](), [llvm#197](), [llvm#198](): Three PRs to update the CI workflows in release_17x, release_18x, and release_19x, to stop using deprecated GitHub actions. **Already Merged**
    - Proposal: Switch default branch to release_19x. **Already done.**
- The upstream LLVM PR [upstream#80874]() has introduced #ifdef's in the omp_lib.h.var file, making it unusable for Classic Flang, which requires it to build its runtime library.
    - Commented here: https://github.com/llvm/llvm-project/commit/fb5fd2d82f9befba9cf5152d1a0c5e6f91ee48f0#commitcomment-143551133
    - Bryan tried adding #ifdef's into omp_lib.h in release_18x but couldn't reproduce the problem. Pawel explains that the 2nd build of OpenMP (with LIBOMP_FORTRAN_MODULES=ON) is the one that fails.
    - **Update:** Bryan has documented the issue and concluded that Paul's workaround seems most suitable. Submitted [llvm#201](). The PR has been reviewed by Paul and Shivarama and  merged.
- Issue  [flang#1458](): Incorrect no alias by TBAA due to a problem with metadata tags.Currently, functions within the same module are assigned unique TBAA type system IDs based on their order, but functions in different modules can unintentionally receive the same ID, leading to incorrect non-aliasing results. PR [flang#1460]() addresses this issue. **- Reviews needed**

- Issue flang#1453: OpenMP test mp_correct/lit/pv01.sh failure in GitHub workflow
    - This has started failing intermittently in the CI workflow. Not sure if it is due to changes in the GitHub X86 runners, or some unknown bug in release_18x.
    - PR flang#1457 was submitted to disable the only CI job that seems to fail this test. Investigation is still needed to find out why this test fails (at least more frequently) when LLVM is built with Clang 15. - **Approved. To be Merged if the issue gets reproduced again.**
- Issue flang#1447: Port Classic Flang to RISC-V
    - Bryan got approval to open-source this code; will start upstreaming soon. Working on setting up a RISC-V VM.
    - **In progress**
- Issue flang#1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
    - ARM could reproduce the issue after the downstream Loop unswitching optimization has been removed during the downstream tech-debt reduction activities.
        - The downstream patch that prevents the wrf segfault https://reviews.llvm.org/D108138
    - The issue **will be on HOLD** now pending further investigation and identifying possible solutions.
    - **No updates**
- Issue #1437: malloc(): corrupted top size after allocating array of polymorphic data type elements
    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
    - **Issue is not yet assigned.**
- Issue #1436: Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
    - **Issue is not yet assigned.**
- Issue #1435: Wrong understanding of the rules for lbound and ubound for non-pointer dummy arguments

        - **Response from the Fortran committee was that there is no undefined behaviour. Classic Flang generates incorrect code for the given test case. Bryan had a question in the issue.**
        - Robert Corbett answered the query. **https://mailman.j3-fortran.org/pipermail/j3/2024-July/014788.html**
- Issue flang#399: bind(C) function with c_char leads to ICE
  Using BIND(C) with CHARACTER(1), which is an interoperable type, causes a crash in flang2. For non-interoperable types like CHARACTER(N), we need to throw a semantic error until F2018 is supported. Bryan hopes to do further analysis about this issue.
    - Problem discovered while building DBCSR.
    - Reference: Modern Fortran: Features for High-Performance Computing (stevelionel.com)
    - **No update this week. This is a low-priority issue.**
- Other PRs
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; the author promised to address it after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
        - **No update this week.**

## Oct 16, 2024

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan
NVIDIA:

- LLVM 19 port:
    - llvm#193: Classic Flang patches have been cherry-picked on top of LLVM 19.1.0. **Merged.**
    - flang#1454: Backward-compatible changes to prepare the build for LLVM 19. **Merged.**
    - flang#1455: Adapt DebugInfo tests for LLVM 19. **Merged.**
    - llvm#196, llvm#197, llvm#198: Three PRs to update the CI workflows in release_17x, release_18x, and release_19x, to stop using deprecated GitHub actions. **Need reviews.**
    - Proposal: Switch default branch to release_19x. **No objection.**
- The upstream LLVM PR upstream#80874 has introduced #ifdef's in the omp_lib.h.var file, making it unusable for Classic Flang, which requires it to build its runtime library.

    - Commented here: https://github.com/llvm/llvm-project/commit/fb5fd2d82f9befba9cf5152d1a0c5e6f91ee48f0#commitcomment-143551133
    - Bryan tried adding #ifdef's into omp_lib.h in release_18x but couldn't reproduce the problem. Pawel explains that the 2nd build of OpenMP (with LIBOMP_FORTRAN_MODULES=ON) is the one that fails.
    - **Update:** Bryan has documented the issue and concluded that Paul's workaround seems most suitable. Working on a PR.
- Two PRs to add Flang test cases for flang#1413 and flang#1028 have been **merged**.
    - flang#1448:  [NFC] modifying the testcase by adding Mbackslash and Mnobackslash options together
    - flang#1450: [NFC] Testcase for Issue #1413
- Huawei looking into a potential Classic Flang bug in TBAA metadata generation, which could cause TBAA to conclude incorrectly that two pointers do not alias when they in fact do. Will submit an issue.
- Issue flang#1453: OpenMP test mp_correct/lit/pv01.sh failure in GitHub workflow
    - This has started failing intermittently in the CI workflow. Not sure if it is due to changes in the GitHub X86 runners, or some unknown bug in release_18x.
    - PR flang#1457 was submitted to disable the only CI job that seems to fail this test. Investigation is still needed to find out why this test fails (at least more frequently) when LLVM is built with Clang 15.
- Issue flang#1447: Port Classic Flang to RISC-V
    - Bryan got approval to open-source this code; will start upstreaming soon. Working on setting up a RISC-V VM.
- Issue flang#1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
    - ARM could reproduce the issue after the downstream Loop unswitching optimization has been removed during the downstream tech-debt reduction activities.
        - The downstream patch that prevents the wrf segfault https://reviews.llvm.org/D108138
    - The issue **will be on HOLD** now pending further investigation and identifying possible solutions.
- Issue #1437: malloc(): corrupted top size after allocating array of polymorphic data type elements
    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
    - **Issue is not yet assigned.**

- Issue #1436: Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
    - **Issue is not yet assigned.**
- Issue #1435: Wrong understanding of the rules for lbound and ubound for non-pointer dummy arguments
    - **Response from the Fortran committee was that there is no undefined behaviour. Classic Flang generates incorrect code for the given test case. Bryan had a question in the issue.**
    - Robert Corbett answered the query. **https://mailman.j3-fortran.org/pipermail/j3/2024-July/014788.html**
- Issue flang#399: bind(C) function with c_char leads to ICE
  Using BIND(C) with CHARACTER(1), which is an interoperable type, causes a crash in flang2. For non-interoperable types like CHARACTER(N), we need to throw a semantic error until F2018 is supported. Bryan hopes to do further analysis about this issue.
    - Problem discovered while building DBCSR.
    - Reference: Modern Fortran: Features for High-Performance Computing (stevelionel.com)
    - **No update this week. This is a low-priority issue.**
- Other PRs
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; the author promised to address it after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
        - **No update this week.**

## Oct 2, 2024

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan
NVIDIA:

- LLVM 19 port:
    - llvm#193: Classic Flang patches have been cherry-picked on top of LLVM 19.1.0. **Reviews needed.**
    - flang#1454: Backward-compatible changes to prepare the build for LLVM 19. **Reviews needed.**

    - flang#1455: Adapt DebugInfo tests for LLVM 19. **Reviews needed.**
- The upstream LLVM PR upstream#80874 has introduced #ifdef's in the omp_lib.h.var file, making it unusable for Classic Flang, which requires it to build its runtime library.
    - Commented here: https://github.com/llvm/llvm-project/commit/fb5fd2d82f9befba9cf5152d1a0c5e6f91ee48f0#commitcomment-143551133
    - Bryan tried adding #ifdef's into omp_lib.h in release_18x but couldn't reproduce the problem. Pawel explains that the 2nd build of OpenMP (with LIBOMP_FORTRAN_MODULES=ON) is the one that fails.
    - **Update:** Bryan has documented the issue and concluded that Paul's workaround seems most suitable. Will submit a PR after llvm#193 is merged.
- Two PRs to add Flang test cases for flang#1413 and flang#1028 are **waiting for Pawel's reviews**:
    - flang#1448: [NFC] modifying the testcase by adding Mbackslash and Mnobackslash options together
    - flang#1450: [NFC] Testcase for Issue #1413
- Issue #1453: OpenMP test mp_correct/lit/pv01.sh failure in GitHub workflow
    - This has started failing intermittently in the CI workflow. Not sure if it is due to changes in the GitHub X86 runners, or some unknown bug in release_18x.
    - **Any taker?**
- Issue #399: bind(C) function with c_char leads to ICE
  Using BIND(C) with CHARACTER(1), which is an interoperable type, causes a crash in flang2. For non-interoperable types like CHARACTER(N), we need to throw a semantic error until F2018 is supported. Bryan hopes to do further analysis about this issue.
    - Problem discovered while building DBCSR.
    - Reference: Modern Fortran: Features for High-Performance Computing (stevelionel.com)
    - **No update this week. This is a low-priority issue.**
- Issue flang#1447: Port Classic Flang to RISC-V
    - Bryan got approval to open-source this code; will start upstreaming soon.
- Issue flang#1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
    - ARM could reproduce the issue after the downstream Loop unswitching optimization has been removed during the downstream tech-debt reduction activities.
        - The downstream patch that prevents the wrf segfault https://reviews.llvm.org/D108138
    - The issue **will be on HOLD** now pending further investigation and identifying possible solutions.
- Issue #1437: malloc(): corrupted top size after allocating array of polymorphic data type elements

    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
    - **Issue is not yet assigned.**
- Issue [#1436](): Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
    - **Issue is not yet assigned.**
- Issue #1435: Wrong understanding of the rules for lbound and ubound for non-pointer dummy arguments
    - **Response from the Fortran committee was that there is no undefined behaviour. Classic Flang generates incorrect code for the given test case. Bryan had a question in the issue.**
    - Robert Corbett answered the query. **https://mailman.j3-fortran.org/pipermail/j3/2024-July/014788.html**
- Other PRs
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; the author promised to address it after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
        - **No update this week.**

## Sep 18, 2024

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan
NVIDIA:

- Proposals:
    - LLVM 19 branch has been created upstream; 19.1.0 RCs are available; we should start upgrading
    - **Update:** Bryan has started porting patches to llvmorg-19.1.0.
- The upstream LLVM PR upstream#80874 has introduced #ifdef's in the omp_lib.h.var file, making it unusable for Classic Flang, which requires it to build its runtime library.

    - Commented here: https://github.com/llvm/llvm-project/commit/fb5fd2d82f9befba9cf5152d1a0c5e6f91ee48f0#commitcomment-143551133
    - Bryan tried adding #ifdef's into omp_lib.h in release_18x but couldn't reproduce the problem. Pawel explains that the 2nd build of OpenMP (with LIBOMP_FORTRAN_MODULES=ON) is the one that fails.
    - **Update:** Bryan has documented the issue and concluded that Paul's workaround seems most suitable.
- Issue #399: bind(C) function with c_char leads to ICE
  Using BIND(C) with CHARACTER(1), which is an interoperable type, causes a crash in flang2. For non-interoperable types like CHARACTER(N), we need to throw a semantic error until F2018 is supported. Bryan hopes to do further analysis about this issue.
    - Problem discovered while building DBCSR.
    - Reference: Modern Fortran: Features for High-Performance Computing (stevelionel.com)
    - **No update this week. This is a low-priority issue.**
- Issue flang#1447: Port Classic Flang to RISC-V
    - Bryan got approval to open-source this code; will start upstreaming soon.
- Issue flang#1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
    - ARM could reproduce the issue after the downstream Loop unswitching optimization has been removed during the downstream tech-debt reduction activities.
        - The downstream patch that prevents the wrf segfault https://reviews.llvm.org/D108138
    - The issue **will be on HOLD** now pending further investigation and identifying possible solutions.
- Issue #1437: malloc(): corrupted top size after allocating array of polymorphic data type elements
    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
    - **Issue is not yet assigned.**
- Issue #1436: Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
    - **Issue is not yet assigned.**

- Issue #1435: Wrong understanding of the rules for lbound and ubound for non-pointer dummy arguments
    - **Response from the Fortran committee was that there is no undefined behaviour. Classic Flang generates incorrect code for the given test case. Bryan had a question in the issue.**
    - Robert Corbett answered the query. **https://mailman.j3-fortran.org/pipermail/j3/2024-July/014788.html**
- Other PRs
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; the author promised to address it after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
        - **No update this week.**

## Sep 4, 2024

Attendees
AMD: Shivarama
Arm:
Huawei: Bryan
NVIDIA:

- Proposals:
    - LLVM 19 branch has been created upstream; 19.1.0 RCs are available; we should start upgrading. Paul or Bryan?
- The upstream LLVM PR upstream#80874 has introduced #ifdef's in the omp_lib.h.var file, making it unusable for Classic Flang, which requires it to build its runtime library.
    - Commented here: https://github.com/llvm/llvm-project/commit/fb5fd2d82f9befba9cf5152d1a0c5e6f91ee48f0#commitcomment-143551133
    - Bryan tried adding #ifdef's into omp_lib.h in release_18x but couldn't reproduce the problem. Pawel explains that the 2nd build of OpenMP (with LIBOMP_FORTRAN_MODULES=ON) is the one that fails.
    - **Update:** Bryan has documented the issue and concluded that Paul's workaround seems most suitable.
- Issue #399: bind(C) function with c_char leads to ICE
  Using BIND(C) with CHARACTER(1), which is an interoperable type, causes a crash in flang2. For non-interoperable types like CHARACTER(N), we need to throw a semantic error until F2018 is supported. Bryan hopes to do further analysis about this issue.
    - Problem discovered while building DBCSR.
    - Reference: Modern Fortran: Features for High-Performance Computing (stevelionel.com)

    - **No update this week. This is low priority issue.**
- Issue flang#1447: Port Classic Flang to RISC-V
    - Bryan got approval to open-source this code; will start upstreaming soon.
- Issue flang#1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - Downstream compilers are not having this issue. Need to be investigated.
        - Arm will be dropping as much of the downstream-only LLVM code as possible in order to reduce the downstream delta (lots of upstreaming commits are under public review), hopefully one day we'll get to the point where either upstream got fixed or downstream got hit by this issue
    - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
    - ARM could reproduce the issue when downstream Loop unswitching optimization is removed.
        - The downstream patch that prevents the wrf segfault https://reviews.llvm.org/D108138
    - The issue **will be on HOLD** now pending further investigation and identifying possible solutions.
- Issue #1437: malloc(): corrupted top size after allocating array of polymorphic data type elements
    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
    - **Issue is not yet assigned.**
- Issue #1436: Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
    - **Issue is not yet assigned.**
- Issue #1435: Wrong understanding of the rules for lbound and ubound for non-pointer dummy arguments
    - Nathan Sircombre from Arm is considering a response to Wanbinchen's comment. Nathan will discuss this during the Fortran Language committee meeting. Pawel will follow up, perhaps find a different contact for the committee.
    - **Response from the Fortran committee was that there is no undefined behaviour. Classic Flang generates incorrect code for the given test case. Bryan had a question in the issue.**
    - Robert Corbett answered the query. **https://mailman.j3-fortran.org/pipermail/j3/2024-July/014788.html**
- Other PRs

    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; the author promised to address it after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
        - **No update this week.**

## Aug 21, 2024

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan
NVIDIA:

- Proposals:

- The upstream LLVM PR upstream#80874 has introduced #ifdef's in the omp_lib.h.var file, making it unusable for Classic Flang, which requires it to build its runtime library.
    - Commented here: https://github.com/llvm/llvm-project/commit/fb5fd2d82f9befba9cf5152d1a0c5e6f91ee48f0#commitcomment-143551133
    - Bryan tried adding #ifdef's into omp_lib.h in release_18x but couldn't reproduce the problem. Pawel explains that the 2nd build of OpenMP (with LIBOMP_FORTRAN_MODULES=ON) is the one that fails.
    - **Update:** Bryan will document the issue and the possible workaround.
- Issue #399: bind(C) function with c_char leads to ICE
  Using BIND(C) with CHARACTER(1), which is an interoperable type, causes a crash in flang2. For non-interoperable types like CHARACTER(N), we need to throw a semantic error until F2018 is supported. Bryan hopes to do further analysis about this issue.
    - Problem discovered while building DBCSR.
    - Reference: Modern Fortran: Features for High-Performance Computing (stevelionel.com)
    - **No update this week. This is low priority issue.**
- Issue flang#1447: Port Classic Flang to RISC-V
    - **No update this week.Bryan will push the relevant patches once approved.**
- Issue flang#1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - Downstream compilers are not having this issue. Need to be investigated.
        - Arm will be dropping as much of the downstream-only LLVM code as possible in order to reduce the downstream delta (lots of upstreaming commits are under public review), hopefully one day we'll get to the point where either upstream got fixed or downstream got hit by this issue

    - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
    - ARM could reproduce the issue when downstream Loop unswitching optimization is removed.
        - The downstream patch that prevents the wrf segfault https://reviews.llvm.org/D108138
    - The issue **will be on HOLD** now pending further investigation and identifying possible solutions.
- Issue #1437: malloc(): corrupted top size after allocating array of polymorphic data type elements
    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
    - **Issue is not yet assigned.**
- Issue #1436: Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
    - **Issue is not yet assigned.**
- Issue #1435: Wrong understanding of the rules for lbound and ubound for non-pointer dummy arguments
    - Nathan Sircombre from Arm is considering a response to Wanbinchen's comment. Nathan will discuss this during the Fortran Language committee meeting. Pawel will follow up, perhaps find a different contact for the committee.
    - **Response from the Fortran committee was that there is no undefined behaviour. Classic Flang generates incorrect code for the given test case. Bryan had a question in the issue.**
    - Robert Corbett answered the query. **https://mailman.j3-fortran.org/pipermail/j3/2024-July/014788.html**
    - 
- Issue #1413 related to pgmath
    - Workaround patch is provided.
    - Need rework of VecFuncs.def file for AVX2/AVX512 support.
    - **Issue is closed for now, but AMD will revisit this after the LLVM 17 upgrade.**
        - This has already happened, LLVM17 is the default branch for classic-flang now…
        - AMD will work on a fix in the weeks to come.
    - The issue happens when -mavx512 is not specified in the command line. The compiler is vectorizing with 8 vector factor  [8 x double]. Since the zmm registers

    - are not available, it is using ymm registers. But this will not have definitions in pgmath library and at runtime application aborts.
    - Workaround is to use -mllvm -force-vector-width=4 on subtargets without avx512 support. This is not a correct solution though.
    - TargetLibraryInfo doesn't have Subtarget info, so we don't know whether avx512 is enabled. Still investigating potentially fixing it here. Another option may be to add Driver logic to force -mllvm -force-vector-width=4 when pgmath is enabled and avx512 is not available (according to -march or -mcpu).
    - Removing the pgmath vector functions will make the generated code inefficient for avx512 target, so the original workaround patch is not suitable.
    - llvm#179 has been merged into release_17x. Bryan's comments will be incorporated as part of a separate pull request.
    - **Changes still need to be ported to release_18x. Rework is needed as LLVM 18 has moved some code from Clang into LLVM.**
    - **Merged to release_18_x**
- Issue #1028: Make backslash behaviour compatible with gfortran
    - flang#1440: [test] pass Mnobackslash explicitly to the tests that assume default…
    - flang#1440 has been merged. The classic-flang-llvm-project patch that makes Mbackslash option default (llvm#180) has also been merged into release_17x.
    - **Changes still need to be ported to release_18x.**
    - **Merged to release_18_x**
- Other PRs
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; the author promised to address it after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
        - **No update this week.**

## Jul 24, 2024

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan
NVIDIA:

- Proposals:
    - Make release_18x the default branch for classic-flang-llvm-project.
    - Tag current tips of release_17x and release_18x (after merging necessary PRs) as stable. Also the relevant flang commits.
    - **Approved.**

- GitHub has stopped supporting macOS-11 runners (actions/runner-images#10198). Bryan has updated the workflows in llvm#183. **Merged.**
- The upstream LLVM PR upstream#80874 has introduced #ifdef's in the omp_lib.h.var file, making it unusable for Classic Flang, which requires it to build its runtime library.
    - Commented here: https://github.com/llvm/llvm-project/commit/fb5fd2d82f9befba9cf5152d1a0c5e6f91ee48f0#commitcomment-143551133
    - Bryan tried adding #ifdef's into omp_lib.h in release_18x but couldn't reproduce the problem. Pawel explains that the 2nd build of OpenMP (with LIBOMP_FORTRAN_MODULES=ON) is the one that fails.
- Issue llvm#181: Many options are treated as unknown in release_18x.
    - llvm#182: [Driver] Allow Classic Flang driver to accept more Clang options
    - **Merged.**
- Issue #399: bind(C) function with c_char leads to ICE
  Using BIND(C) with CHARACTER(1), which is an interoperable type, causes a crash in flang2. For non-interoperable types like CHARACTER(N), we need to throw a semantic error until F2018 is supported. Bryan hopes to do further analysis about this issue.
    - Problem discovered while building DBCSR.
    - Reference: Modern Fortran: Features for High-Performance Computing (stevelionel.com)
    - **No update this week.**
- Issue flang#1447: Port Classic Flang to RISC-V
    - **No update this week.**
- Issue flang#1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - Downstream compilers are not having this issue. Need to be investigated.
        - Arm will be dropping as much of the downstream-only LLVM code as possible in order to reduce the downstream delta (lots of upstreaming commits are under public review), hopefully one day we'll get to the point where either upstream got fixed or downstream got hit by this issue
    - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
    - ARM could reproduce the issue when downstream Loop unswitching optimization is removed.
        - The downstream patch that prevents the wrf segfault https://reviews.llvm.org/D108138
    - The issue **will be on HOLD** now pending further investigation and identifying possible solutions.
- Issue #1437: malloc(): corrupted top size after allocating array of polymorphic data type elements
    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never

    returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
  - **Issue is not yet assigned.**
- Issue #1436: Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
  - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
  - **Issue is not yet assigned.**
- Issue #1435: Wrong understanding of the rules for lbound and ubound for non-pointer dummy arguments
  - Nathan Sircombre from Arm is considering a response to Wanbinchen's comment. Nathan will discuss this during the Fortran Language committee meeting. Pawel will follow up, perhaps find a different contact for the committee.
  - **Response from the Fortran committee was that there is no undefined behaviour. Classic Flang generates incorrect code for the given test case. Bryan had a question in the issue.**
- Issue #1413 related to pgmath
  - Workaround patch is provided.
  - Need rework of VecFuncs.def file for AVX2/AVX512 support.
  - **Issue is closed for now, but AMD will revisit this after the LLVM 17 upgrade.**
    - This has already happened, LLVM17 is the default branch for classic-flang now…
    - AMD will work on a fix in the weeks to come.
  - The issue happens when -mavx512 is not specified in the command line. The compiler is vectorizing with 8 vector factor  [8 x double]. Since the zmm registers are not available, it is using ymm registers. But this will not have definitions in pgmath library and at runtime application aborts.
  - Workaround is to use -mllvm -force-vector-width=4 on subtargets without avx512 support. This is not a correct solution though.
  - TargetLibraryInfo doesn't have Subtarget info, so we don't know whether avx512 is enabled. Still investigating potentially fixing it here. Another option may be to add Driver logic to force -mllvm -force-vector-width=4 when pgmath is enabled and avx512 is not available (according to -march or -mcpu).
  - Removing the pgmath vector functions will make the generated code inefficient for avx512 target, so the original workaround patch is not suitable.
  - llvm#179 has been merged into release_17x. Bryan's comments will be incorporated as part of a separate pull request.
  - **Changes still need to be ported to release_18x. Rework is needed as LLVM 18 has moved some code from Clang into LLVM.**
- Issue #1028: Make backslash behaviour compatible with gfortran
  - flang#1440: [test] pass Mnobackslash explicitly to the tests that assume default…

    - flang#1440 has been merged. The classic-flang-llvm-project patch that makes Mbackslash option default (llvm#180) has also been merged into release_17x.
    - **Changes still need to be ported to release_18x.**
- Other PRs
    - flang#1449: [flang2] Generate loads of complex variables with correct alignment
        - Comment on flang#1322 pointed out that complex type variables are loaded with a pessimistic 1-byte alignment. flang2 usually generates load instructions with alignment set to the size of the data type being loaded.
        - **Merged.**
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; the author promised to address it after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
        - **No update this week.**
    - flang#642: AArch64: significantly improve formatted input performance by using optimized libc functions on ARM64
        - Looks like a good idea; needs reviews.
        - Huawei to take a look. **No update.**

## Jul 10, 2024

Attendees (**Canceled**)
AMD:
Arm:
Huawei:
NVIDIA:

- Proposal: Make release_18x the default branch for classic-flang-llvm-project.
- GitHub has stopped supporting macOS-11 runners (actions/runner-images#10198). Bryan will update the workflows.
- The upstream LLVM PR upstream#80874 has introduced #ifdef's in the omp_lib.h.var file, making it unusable for Classic Flang, which requires it to build its runtime library.
    - Commented here: https://github.com/llvm/llvm-project/commit/fb5fd2d82f9befba9cf5152d1a0c5e6f91ee48f0#commitcomment-143551133
- Issue llvm#181: Many options are treated as unknown in release_18x.
    - **Bryan is investigating.**
- Issue #399: bind(C) function with c_char leads to ICE
  Using BIND(C) with CHARACTER(1), which is an interoperable type, causes a crash in flang2. For non-interoperable types like CHARACTER(N), we need to throw a semantic error until F2018 is supported. Bryan hopes to do further analysis about this issue.
    - Problem discovered while building DBCSR.

    - Reference: [Modern Fortran: Features for High-Performance Computing (stevelionel.com)]
    - **No update this week.**
- Issue [#1429]: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - Downstream compilers are not having this issue. Need to be investigated.
        - Arm will be dropping as much of the downstream-only LLVM code as possible in order to reduce the downstream delta (lots of upstreaming commits are under public review), hopefully one day we'll get to the point where either upstream got fixed or downstream got hit by this issue
    - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
    - ARM could reproduce the issue when downstream Loop unswitching optimization is removed.
        - The downstream patch that prevents the wrf segfault https://reviews.llvm.org/D108138
    - The issue **will be on HOLD** now pending further investigation and identifying possible solutions.
- Issue [#1437]: malloc(): corrupted top size after allocating array of polymorphic data type elements
    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
    - **Issue is not yet assigned.**
- Issue [#1436]: Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
    - **Issue is not yet assigned.**
- Issue [#1435]: Wrong understanding of the rules for lbound and ubound for non-pointer dummy arguments
    - Nathan Sircombre from Arm is considering a response to Wanbinchen's comment. Nathan will discuss this during the Fortran Language committee meeting. Pawel will follow up, perhaps find a different contact for the committee.
    - **Response from the Fortran committee was that there is no undefined behaviour. Classic Flang generates incorrect code for the given test case.**
- Issue [#1413] related to pgmath
    - Workaround patch is provided.
    - Need rework of VecFuncs.def file for AVX2/AVX512 support.
    - **Issue is closed for now, but AMD will revisit this after the LLVM 17 upgrade.**

        - This has already happened, LLVM17 is the default branch for classic-flang now…
        - AMD will work on a fix in the weeks to come.
    - The issue happens when -mavx512 is not specified in the command line. The compiler is vectorizing with 8 vector factor [8 x double]. Since the zmm registers are not available, it is using ymm registers. But this will not have definitions in pgmath library and at runtime application aborts.
    - Workaround is to use -mllvm -force-vector-width=4 on subtargets without avx512 support. This is not a correct solution though.
    - TargetLibraryInfo doesn't have Subtarget info, so we don't know whether avx512 is enabled. Still investigating potentially fixing it here. Another option may be to add Driver logic to force -mllvm -force-vector-width=4 when pgmath is enabled and avx512 is not available (according to -march or -mcpu).
    - Removing the pgmath vector functions will make the generated code inefficient for avx512 target, so the original workaround patch is not suitable.
    - llvm#179 has been merged into release_17x. Bryan's comments will be incorporated as part of a separate pull request.
    - **No updates this week. Changes still need to be ported to release_18x.**
- Issue #1028: Make backslash behaviour compatible with gfortran
    - flang#1440: [test] pass Mnobackslash explicitly to the tests that assume default…
    - flang#1440 has been merged. The classic-flang-llvm-project patch that makes Mbackslash option default (llvm#180) has also been merged into release_17x.
    - **Changes still need to be ported to release_18x.**
- Other PRs
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; the author promised to address it after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
        - **No update this week.**
    - flang#642: AArch64: significantly improve formatted input performance by using optimized libc functions on ARM64
        - Looks like a good idea; needs reviews.
        - Huawei to take a look. **No update.**

## Jun 26, 2024

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan
NVIDIA:

- The upstream LLVM PR upstream#80874 has introduced #ifdef's in the omp_lib.h.var file, making it unusable for classic flang, which requires it to build its runtime library.
    - Commented here: https://github.com/llvm/llvm-project/commit/fb5fd2d82f9befba9cf5152d1a0c5e6f91ee48f0#commitcomment-143551133
- Issue #399: bind(C) function with c_char leads to ICE
  Using BIND(C) with CHARACTER(1), which is an interoperable type, causes a crash in flang2. For non-interoperable types like CHARACTER(N), we need to throw a semantic error until F2018 is supported. Bryan hopes to do further analysis about this issue.
    - Problem discovered while building DBCSR.
    - Reference: Modern Fortran: Features for High-Performance Computing (stevelionel.com)
    - **No update this week.**
- Issue #1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - Downstream compilers are not having this issue. Need to be investigated.
        - Arm will be dropping as much of the downstream-only LLVM code as possible in order to reduce the downstream delta (lots of upstreaming commits are under public review), hopefully one day we'll get to the point where either upstream got fixed or downstream got hit by this issue
    - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
    - ARM could reproduce the issue when downstream Loop unswitching optimization is removed.
        - The downstream patch that prevents the wrf segfault https://reviews.llvm.org/D108138
    - The issue **will be on HOLD** now pending further investigation and identifying possible solutions.
- Issue #1437: malloc(): corrupted top size after allocating array of polymorphic data type elements
    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
    - **Issue is not yet assigned.**
- Issue #1436: Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
    - **Issue is not yet assigned.**

- Issue #1435: Wrong understanding of the rules for lbound and ubound for non-pointer dummy arguments
    - Nathan Sircombre from Arm is considering a response to Wanbinchen's comment. Nathan will discuss this during the Fortran Language committee meeting. Pawel will follow up, perhaps find a different contact for the committee.
    - **No updates this week.**
- Issue #1413 related to pgmath
    - Workaround patch is provided.
    - Need rework of VecFuncs.def file for AVX2/AVX512 support.
    - **Issue is closed for now, but AMD will revisit this after the LLVM 17 upgrade.**
        - This has already happened, LLVM17 is the default branch for classic-flang now…
        - AMD will work on a fix in the weeks to come.
    - The issue happens when -mavx512 is not specified in the command line. The compiler is vectorizing with 8 vector factor  [8 x double]. Since the zmm registers are not available, it is using ymm registers. But this will not have definitions in pgmath library and at runtime application aborts.
    - Workaround is to use -mllvm -force-vector-width=4 on subtargets without avx512 support. This is not a correct solution though.
    - TargetLibraryInfo doesn't have Subtarget info, so we don't know whether avx512 is enabled. Still investigating potentially fixing it here. Another option may be to add Driver logic to force -mllvm -force-vector-width=4 when pgmath is enabled and avx512 is not available (according to -march or -mcpu).
    - Removing the pgmath vector functions will make the generated code inefficient for avx512 target, so the original workaround patch is not suitable.
    - **llvm#179 has been merged into release_17x . Bryan's comments will be incorporated as part of a separate pull request.**
    - **No updates this week.**
- Issue #1028: Make backslash behaviour compatible with gfortran
    - flang#1440: [test] pass Mnobackslash explicitly to the tests that assume default…
    - flang#1440 has been merged. The classic-flang-llvm-project patch that makes Mbackslash option default (llvm#180) is still undergoing reviews.
- Other PRs
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; the author promised to address it after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
        - **No update this week.**
    - flang#642: AArch64: significantly improve formatted input performance by using optimized libc functions on ARM64
        - Looks like a good idea; needs reviews.

            - Huawei to take a look. **No update.**

## Jun 12, 2024

Attendees
AMD:
Arm: Pawel
Huawei: Bryan
NVIDIA:

- Issue #1445: Fix bugs with implied do
    - **Merged**
- Issue #1444: Pawel found an issue with flang#1422, which he had to revert downstream because it caused one of the dependencies of Quantum Espresso (libmbd) to stop building correctly. **Issue is closed.**
- Issue #399: bind(C) function with c_char leads to ICE
  Using BIND(C) with CHARACTER(1), which is an interoperable type, causes a crash in flang2. For non-interoperable types like CHARACTER(N), we need to throw a semantic error until F2018 is supported. Bryan hopes to do further analysis about this issue.
    - Problem discovered while building DBCSR.
    - Reference: Modern Fortran: Features for High-Performance Computing (stevelionel.com)
    - **No update this week.**
- Issue #1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - Downstream compilers are not having this issue. Need to be investigated.
        - Arm will be dropping as much of the downstream-only LLVM code as possible in order to reduce the downstream delta (lots of upstreaming commits are under public review), hopefully one day we'll get to the point where either upstream got fixed or downstream got hit by this issue
    - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
    - ARM could reproduce the issue when downstream Loop unswitching optimization is removed.
        - The downstream patch that prevents the wrf segfault https://reviews.llvm.org/D108138
    - The issue will be on HOLD now pending further investigation and identifying possible solutions.
- Issue #1437: malloc(): corrupted top size after allocating array of polymorphic data type elements
    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.

    - **Issue is not yet assigned.**
- Issue #1436: Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
    - **Issue is not yet assigned.**
- Issue #1435: Wrong understanding of the rules for lbound and ubound for non-pointer dummy arguments
    - Nathan Sircombre from Arm is considering a response to Wanbinchen's comment. Nathan will discuss this during the Fortran Language committee meeting. Pawel will follow up, perhaps find a different contact for the committee.
    - **No updates this week.**
- Issue #1413 related to pgmath
    - Workaround patch is provided.
    - Need rework of VecFuncs.def file for AVX2/AVX512 support.
    - **Issue is closed for now, but AMD will revisit this after the LLVM 17 upgrade.**
        - This has already happened, LLVM17 is the default branch for classic-flang now…
        - AMD will work on a fix in the weeks to come.
    - The issue happens when -mavx512 is not specified in the command line. The compiler is vectorizing with 8 vector factor [8 x double]. Since the zmm registers are not available, it is using ymm registers. But this will not have definitions in pgmath library and at runtime application aborts.
    - Workaround is to use -mllvm -force-vector-width=4 on subtargets without avx512 support. This is not a correct solution though.
    - TargetLibraryInfo doesn't have Subtarget info, so we don't know whether avx512 is enabled. Still investigating potentially fixing it here. Another option may be to add Driver logic to force -mllvm -force-vector-width=4 when pgmath is enabled and avx512 is not available (according to -march or -mcpu).
    - Removing the pgmath vector functions will make the generated code inefficient for avx512 target, so the original workaround patch is not suitable.
    - llvm#179 submitted for review. Bryan needs to review.
    - **The code is merged into release_17x . Bryan's comments will be incorporated as part of a separate pull request.**
    - **No updates this week.**
- Other PRs
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; the author promised to address it after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.

        - **Author has moved on; Bryan will commandeer this patch.**
        - **No update this week.**
    - flang#1440: [test] pass Mnobackslash explicitly to the tests that assume default…
        - Prerequisite for flang#1028: Make backslash behaviour compatible with gfortran
        - This patch is merged. The classic-flang-llvm-project patch that makes Mbackslash option default (llvm#180) need reviews.
        - **Review comments need to be addressed. Will address by next meeting.**
    - flang#642: AArch64: significantly improve formatted input performance by using optimized libc functions on ARM64
        - Looks like a good idea; needs reviews.
        - Huawei to take a look. **No update.**

## May 29, 2024

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan
NVIDIA:

- Issue #1445: Fix bugs with implied do
    - To be reviewed
- Issue #1444: Pawel found an issue with flang#1422, which he had to revert downstream because it caused one of the dependencies of Quantum Espresso (libmbd) to stop building correctly. **Issue is fixed. Can it be closed?**
  **Will be closed after #1445 is merged.**
- Issue #1382: Petr Penzin has asked about restoring the ability to build Classic Flang in-tree.
    - Bryan will be responding to the ticket.
- Issue #1376: Petr Penzin has asked about why .c files were simply renamed to .cpp files in flang2.
    - The decision was made by NVIDIA years ago and it is irreversible. The issue will be closed.
- Issue #399: bind(C) function with c_char leads to ICE
  Using BIND(C) with CHARACTER(1), which is an interoperable type, causes a crash in flang2. For non-interoperable types like CHARACTER(N), we need to throw a semantic error until F2018 is supported. Bryan hopes to do further analysis about this issue.
    - Problem discovered while building DBCSR.
    - Reference: Modern Fortran: Features for High-Performance Computing (stevelionel.com)
    - **No update this week.**
- Issue #1429: 521.wrf_r segfaults when built with -flto=full -Ofast

    - Caused by upstream changes in LLVM 17
    - Downstream compilers are not having this issue. Need to be investigated.
        - Arm will be dropping as much of the downstream-only LLVM code as possible in order to reduce the downstream delta (lots of upstreaming commits are under public review), hopefully one day we'll get to the point where either upstream got fixed or downstream got hit by this issue
    - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
    - ARM could reproduce the issue when downstream Loop unswitching optimization is removed. Need to be discussed further.
    - The downstream patch that prevents the wrf segfault https://reviews.llvm.org/D108138
    - Issue will be on HOLD now pending further investigation and identifying possible solutions.
- Issue #1437: malloc(): corrupted top size after allocating array of polymorphic data type elements
    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
    - **Issue is not yet assigned.**
- Issue #1436: Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
    - **Issue is not yet assigned.**
- Issue #1435: Wrong understanding of the rules for lbound and ubound for non-pointer dummy arguments
    - Nathan Sircombre from Arm is considering a response to Wanbinchen's comment. Nathan will discuss this during the Fortran Language committee meeting. Pawel will follow up, perhaps find a different contact for the committee.
    - **No updates this week.**
- Issue #1413 related to pgmath
    - Workaround patch is provided.
    - Need rework of VecFuncs.def file for AVX2/AVX512 support.
    - **Issue is closed for now, but AMD will revisit this after the LLVM 17 upgrade.**
        - This has already happened, LLVM17 is the default branch for classic-flang now…
        - AMD will work on a fix in the weeks to come.
    - The issue happens when -mavx512 is not specified in the command line. The compiler is vectorizing with 8 vector factor [8 x double]. Since the zmm registers

are not available, it is using ymm registers. But this will not have definitions in pgmath library and at runtime application aborts.
    - Workaround is to use -mllvm -force-vector-width=4 on subtargets without avx512 support. This is not a correct solution though.
    - TargetLibraryInfo doesn't have Subtarget info, so we don't know whether avx512 is enabled. Still investigating potentially fixing it here. Another option may be to add Driver logic to force -mllvm -force-vector-width=4 when pgmath is enabled and avx512 is not available (according to -march or -mcpu).
    - Removing the pgmath vector functions will make the generated code inefficient for avx512 target, so the original workaround patch is not suitable.
    - llvm#179  submitted for review. Bryan needs to review.
    - **The code is merged into release_17x . Bryan's comments will be incorporated as part of a separate pull request.**
    - **No updates this week.**
- Other PRs
    - llvm#178: Port Classic Flang to LLVM 18.1.1
        - Resolved minor cherry-pick conflicts; added one change for libpgmath function mappings. Test failures (2 libomptarget tests and a number of Driver option tests) have been fixed; CI is currently running. **Merged.**
        - **Some driver options like -m64, -m32 are not working. The options -funroll-loops etc.. also not getting recognized. AMD to create issue for this problem.**
    - flang#1440: [test] pass Mnobackslash explicitly to the tests that assume default…
        - Prerequisite for flang#1028: Make backslash behaviour compatible with gfortran
        - This patch is merged. The classic-flang-llvm-project patch that makes Mbackslash option default (llvm#180) need reviews.
        - **Review comments need to be addressed. Will address by next meeting.**
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; the author promised to address it after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
        - **No update this week.**
    - flang#642: AArch64: significantly improve formatted input performance by using optimized libc functions on ARM64
        - Looks like a good idea; needs reviews.
        - Huawei to take a look. **No update yet.**

## May 15, 2024

Attendees
AMD: Shivarama
Arm: Pawel
Huawei:
NVIDIA:

- Issue #1446: Update CI to test release_18x, and stop testing release_16x
    - **Approved and merged.**
- Issue #1445: Fix bugs with implied do
    - To be reviewed
- Issue #1442: Pawel found an issue with flang#1422, which he had to revert downstream because it caused one of the dependencies of Quantum Espresso (libmbd) to stop building correctly. **Issue is fixed. Can it be closed? Will be closed after #1445 is merged.**
- llvm#173: All tags in classic-flang-llvm-project were deleted. Does anyone know why? **Bryan has re-created the tags. Issue closed.**
- Issue #1382: Petr Penzin has asked about restoring the ability to build Classic Flang in-tree.
- Issue #1376: Petr Penzin has asked about why .c files were simply renamed to .cpp files in flang2.
- Issue #399: bind(C) function with c_char leads to ICE
  Using BIND(C) with CHARACTER(1), which is an interoperable type, causes a crash in flang2. For non-interoperable types like CHARACTER(N), we need to throw a semantic error until F2018 is supported. Bryan hopes to do further analysis about this issue.
    - Problem discovered while building DBCSR.
    - Reference: Modern Fortran: Features for High-Performance Computing (stevelionel.com)
    - **No update this week.**
- Issue #1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - Downstream compilers are not having this issue. Need to be investigated.
        - Arm will be dropping as much of the downstream-only LLVM code as possible in order to reduce the downstream delta (lots of upstreaming commits are under public review), hopefully one day we'll get to the point where either upstream got fixed or downstream got hit by this issue
    - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
- Huawei will open a new issue about OpenMP hang in CESM (only when linking statically); a reproducer should be available soon.
    - Pawel suggests trying OpenMP runtime environment variables regarding locking behaviour.

    - **Bryan built the static OpenMP runtime but cannot reproduce this problem; there seems to be something wrong with the reproducer provided by his team.**
- Issue [#1437](): malloc(): corrupted top size after allocating array of polymorphic data type elements
    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
    - **Issue is not yet assigned.**
- Issue [#1436](): Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
    - **Issue is not yet assigned.**
- Issue [#1435](): Wrong understanding of the rules for lbound and ubound for non-pointer dummy arguments
    - Nathan Sircombre from Arm is considering a response to Wanbinchen's comment. Nathan will discuss this during the Fortran Language committee meeting. Pawel will follow up, perhaps find a different contact for the committee.
    - **No updates this week.**
- Issue [#1413]() related to pgmath
    - Workaround patch is provided.
    - Need rework of VecFuncs.def file for AVX2/AVX512 support.
    - **Issue is closed for now, but AMD will revisit this after the LLVM 17 upgrade.**
        - This has already happened, LLVM17 is the default branch for classic-flang now…
        - AMD will work on a fix in the weeks to come.
    - The issue happens when -mavx512 is not specified in the command line. The compiler is vectorizing with 8 vector factor  [8 x double]. Since the zmm registers are not available, it is using ymm registers. But this will not have definitions in pgmath library and at runtime application aborts.
    - Workaround is to use -mllvm -force-vector-width=4 on subtargets without avx512 support. This is not a correct solution though.
    - TargetLibraryInfo doesn't have Subtarget info, so we don't know whether avx512 is enabled. Still investigating potentially fixing it here. Another option may be to add Driver logic to force -mllvm -force-vector-width=4 when pgmath is enabled and avx512 is not available (according to -march or -mcpu).
    - Removing the pgmath vector functions will make the generated code inefficient for avx512 target, so the original workaround patch is not suitable.
    - [llvm#179]()  submitted for review. Bryan needs to review.

    - **The code is merged into release_17x . Bryan's comments will be incorporated as part of a separate pull request.**
- Other PRs
    - llvm#178: Port Classic Flang to LLVM 18.1.1
        - Resolved minor cherry-pick conflicts; added one change for libpgmath function mappings. Test failures (2 libomptarget tests and a number of Driver option tests) have been fixed; CI is currently running. **Merged.**
        - **Some driver options like -m64, -m32 are not working. The options -funroll-loops etc.. also not getting recognized. AMD to create issue for this problem.**
    - flang#1440: [test] pass Mnobackslash explicitly to the tests that assume default…
        - Prerequisite for flang#1028: Make backslash behaviour compatible with gfortran
        - This patch is merged. The classic-flang-llvm-project patch that makes Mbackslash option default (llvm#180) need reviews.
        - **Merged.**
    - ARM has attempted to build classic flang with LLVM18. Two PRs were issued and **need reviews**:
        - flang#1438 With LLVM18 a loop preheader is being exposed, hence the preds must be more liberal. Bryan reviewed.
        - flang#1439 The checks for AArch64 target features must be more flexible
        - **Both merged.**
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; the author promised to address it after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
        - **No update this week.**
    - flang#642: AArch64: significantly improve formatted input performance by using optimized libc functions on ARM64
        - Looks like a good idea; needs reviews.
        - Huawei to take a look. **No update yet.**

## Apr 17, 2024

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan
NVIDIA:

- Issue #1442: Pawel found an issue with flang#1422, which he had to revert downstream because it caused one of the dependencies of Quantum Espresso (libmbd) to stop building correctly. **Asking author for help.**
- llvm#173: All tags in classic-flang-llvm-project were deleted. Does anyone know why? It should be possible for Bryan/Pawel to recreate the tags from their local clones. Bryan will ask Kiran to check who has write access to the repo.
- Issue #1382: Petr Penzin has asked about restoring the ability to build Classic Flang in-tree.
- Issue #1376: Petr Penzin has asked about why .c files were simply renamed to .cpp files in flang2.
- Issue #399: bind(C) function with c_char leads to ICE
  Using BIND(C) with CHARACTER(1), which is an interoperable type, causes a crash in flang2. For non-interoperable types like CHARACTER(N), we need to throw a semantic error until F2018 is supported. Bryan hopes to do further analysis about this issue.
  - Problem discovered while building DBCSR.
  - Reference: Modern Fortran: Features for High-Performance Computing (stevelionel.com)
  - **No update this week.**
- Issue #1429: 521.wrf_r segfaults when built with -flto=full -Ofast
  - Caused by upstream changes in LLVM 17
  - Downstream compilers are not having this issue. Need to be investigated.
    - Arm will be dropping as much of the downstream-only LLVM code as possible in order to reduce the downstream delta (lots of upstreaming commits are under public review), hopefully one day we'll get to the point where either upstream got fixed or downstream got hit by this issue
  - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
- Huawei will open a new issue about OpenMP hang in CESM (only when linking statically); a reproducer should be available soon.
  - Pawel suggests trying OpenMP runtime environment variables regarding locking behaviour.
  - **Bryan built the static OpenMP runtime but cannot reproduce this problem; there seems to be something wrong with the reproducer provided by his team.**
- Issue #1437: malloc(): corrupted top size after allocating array of polymorphic data type elements
  - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
  - **Issue is not yet assigned.**
- Issue #1436: Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length

    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
    - **Issue is not yet assigned.**
- Issue #1435: Wrong understanding of the rules for lbound and ubound for non-pointer dummy arguments
    - Nathan Sircombre from Arm is considering a response to Wanbinchen's comment. Nathan will discuss this during the Fortran Language committee meeting. Pawel will follow up, perhaps find a different contact for the committee.
    - **No updates this week.**
- Issue #1413 related to pgmath
    - Workaround patch is provided.
    - Need rework of VecFuncs.def file for AVX2/AVX512 support.
    - **Issue is closed for now, but AMD will revisit this after the LLVM 17 upgrade.**
        - This has already happened, LLVM17 is the default branch for classic-flang now…
        - AMD will work on a fix in the weeks to come.
    - The issue happens when -mavx512 is not specified in the command line. The compiler is vectorizing with 8 vector factor [8 x double]. Since the zmm registers are not available, it is using ymm registers. But this will not have definitions in pgmath library and at runtime application aborts.
    - Workaround is to use -mllvm -force-vector-width=4 on subtargets without avx512 support. This is not a correct solution though.
    - TargetLibraryInfo doesn't have Subtarget info, so we don't know whether avx512 is enabled. Still investigating potentially fixing it here. Another option may be to add Driver logic to force -mllvm -force-vector-width=4 when pgmath is enabled and avx512 is not available (according to -march or -mcpu).
    - Removing the pgmath vector functions will make the generated code inefficient for avx512 target, so the original workaround patch is not suitable.
    - **Still working on this; have not yet found the right solution. No update this week; will try to revisit this before the next meeting.**
- Other PRs
    - llvm#178: Port Classic Flang to LLVM 18.1.1
        - Resolved minor cherry-pick conflicts; added one change for libpgmath function mappings. Test failures (2 libomptarget tests and a number of Driver option tests) have been fixed; CI is currently running. **Merged.**
    - llvm#176: fix declaration does not match definition when enable classic flang
        - Work-in-progress patch. Author has not updated since March 18. Asking if this is still a work-in-progress. **Author closed the PR.**
    - flang#1440: [test] pass Mnobackslash explicitly to the tests that assume default…
        - Prerequisite for flang#1028: Make backslash behaviour compatible with gfortran

        - A classic-flang-llvm-project patch will follow.
        - **Needs reviews. Bryan has reviewed.**
    - ARM has attempted to build classic flang with LLVM18. Two PRs were issued and **need reviews**:
        - flang#1438 With LLVM18 a loop preheader is being exposed, hence the preds must be more liberal. Bryan reviewed.
        - flang#1439 The checks for AArch64 target features must be more flexible
        - **Both merged.**
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; the author promised to address it after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
        - **No update this week.**
    - flang#642: AArch64: significantly improve formatted input performance by using optimized libc functions on ARM64
        - Looks like a good idea; needs reviews.
        - Huawei to take a look. **No update yet.**

## Apr 3, 2024

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan
NVIDIA:

- Pawel found an issue with flang#1422, which he had to revert downstream because it caused one of the dependencies of Quantum Espresso (libmbd) to stop building correctly. He will open an issue.
- Issue #399: bind(C) function with c_char leads to ICE
  Using BIND(C) with CHARACTER(1), which is an interoperable type, causes a crash in flang2. For non-interoperable types like CHARACTER(N), we need to throw a semantic error until F2018 is supported. Bryan hopes to do further analysis about this issue.
    - Problem discovered while building DBCSR.
    - Reference: Modern Fortran: Features for High-Performance Computing (stevelionel.com)
    - **No update this week.**
- Issue #1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - Downstream compilers are not having this issue. Need to be investigated.

        - Arm will be dropping as much of the downstream-only LLVM code as possible in order to reduce the downstream delta (lots of upstreaming commits are under public review), hopefully one day we'll get to the point where either upstream got fixed or downstream got hit by this issue
    - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
- Huawei will open a new issue about OpenMP hang in CESM (only when linking statically); a reproducer should be available soon.
    - Pawel suggests trying OpenMP runtime environment variables regarding locking behaviour.
    - **Trying to reproduce this; creating a script to build OpenMP statically using classic-flang-llvm-project. On hold while working on the LLVM 18 upgrade.**
- Issue #1437: malloc(): corrupted top size after allocating array of polymorphic data type elements
    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
    - **Issue is not yet assigned.**
- Issue #1436: Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
    - **Issue is not yet assigned.**
- Issue #1435: Wrong understanding of the rules for lbound and ubound for non-pointer dummy arguments
    - Nathan Sircombre from Arm is considering a response to Wanbinchen's comment. Nathan will discuss this during the Fortran Language committee meeting.
    - **No updates this week.**
- Issue #1413 related to pgmath
    - Workaround patch is provided.
    - Need rework of VecFuncs.def file for AVX2/AVX512 support.
    - **Issue is closed for now, but AMD will revisit this after the LLVM 17 upgrade.**
        - This has already happened, LLVM17 is the default branch for classic-flang now…
        - AMD will work on a fix in the weeks to come.
    - The issue happens when -mavx512 is not specified in the command line. The compiler is vectorizing with 8 vector factor  [8 x double]. Since the zmm registers are not available, it is using ymm registers. But this will not have definitions in pgmath library and at runtime application aborts.

    - Workaround is to use -mllvm -force-vector-width=4 on subtargets without avx512 support.
    - TargetLibraryInfo doesn't have Subtarget info, so we don't know whether avx512 is enabled. Still investigating potentially fixing it here. Another option may be to add Driver logic to force -mllvm -force-vector-width=4 when pgmath is enabled and avx512 is not available (according to -march or -mcpu).
    - Removing the pgmath vector functions will make the generated code inefficient for avx512 target, so the original workaround patch is not suitable.
    - **Still working on this; have not yet found the right solution. No update this week; will try to revisit this before the next meeting.**
- Issue: Windows CI builds are failing in the Download Artifact step. Bryan will investigate.
    - Manually triggering the release_17x workflow did not fix it. The root cause of the problem was some syntax errors in the Windows PowerShell script in the GitHub workflow. Fixed in flang#1441. **Needs reviews.**
- Other PRs
    - llvm#178: Port Classic Flang to LLVM 18.1.1
        - Resolved minor cherry-pick conflicts; added one change for libpgmath function mappings. Test failures (2 libomptarget tests and a number of Driver option tests) have been fixed; CI is currently running. **Needs reviews.**
    - llvm#176: fix declaration does not match definition when enable classic flang
        - Work-in-progress patch. Author has not updated since March 18. Asking if this is still a work-in-progress.
    - flang#1440: [test] pass Mnobackslash explicitly to the tests that assume default…
        - Prerequisite for flang#1028: Make backslash behaviour compatible with gfortran
        - A classic-flang-llvm-project patch will follow.
        - **Needs reviews. Bryan has reviewed.**
    - ARM has attempted to build classic flang with LLVM18. Two PRs were issued and **need reviews**:
        - flang#1438 With LLVM18 a loop preheader is being exposed, hence the preds must be more liberal. Bryan reviewed.
        - flang#1439 The checks for AArch64 target features must be more flexible
        - No other problems encountered. Bryan reviewed.
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; the author promised to address it after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
        - **No update this week.**
    - flang#642: AArch64: significantly improve formatted input performance by using optimized libc functions on ARM64

        - Looks like a good idea; needs reviews.
        - Huawei to take a look. **No update yet.**

## Mar 20, 2024

Attendees
AMD: Shivarama
Arm:
Huawei: Bryan
NVIDIA:

- Clocks go forward: March 10 for Eastern Daylight-saving Time, March 31 for British Summer Time. **Shivarama will schedule the next meeting 1 hour earlier.**
- Issue #399: bind(C) function with c_char leads to ICE
  Using BIND(C) with CHARACTER(1), which is an interoperable type, causes a crash in flang2. For non-interoperable types like CHARACTER(N), we need to throw a semantic error until F2018 is supported. Bryan hopes to do further analysis about this issue.
    - Problem discovered while building DBCSR.
    - Reference: Modern Fortran: Features for High-Performance Computing (stevelionel.com)
    - **No update this week.**
- Issue #1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - Downstream compilers are not having this issue. Need to be investigated.
        - Arm will be dropping as much of the downstream-only LLVM code as possible in order to reduce the downstream delta (lots of upstreaming commits are under public review), hopefully one day we'll get to the point where either upstream got fixed or downstream got hit by this issue
    - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
- Huawei will open a new issue about OpenMP hang in CESM (only when linking statically); a reproducer should be available soon.
    - Pawel suggests trying OpenMP runtime environment variables regarding locking behaviour.
    - **Trying to reproduce this; creating a script to build OpenMP statically using classic-flang-llvm-project.**
- Issue #1437: malloc(): corrupted top size after allocating array of polymorphic data type elements
    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
    - **Issue is not yet assigned.**

- Issue #1436: Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
    - **Issue is not yet assigned.**
- Issue #1435: Wrong understanding of the rules for lbound and ubound for non-pointer dummy arguments
    - Nathan Sircombre from Arm is considering a response to Wanbinchen's comment. Nathan will discuss this during the Fortran Language committee meeting.
    - **No updates this week.**
- Issue #1413 related to pgmath
    - Workaround patch is provided.
    - Need rework of VecFuncs.def file for AVX2/AVX512 support.
    - **Issue is closed for now, but AMD will revisit this after the LLVM 17 upgrade.**
        - This has already happened, LLVM17 is the default branch for classic-flang now…
        - AMD will work on a fix in the weeks to come.
    - The issue happens when -mavx512 is not specified in the command line. The compiler is vectorizing with 8 vector factor  [8 x double]. Since the zmm registers are not available, it is using ymm registers. But this will not have definitions in pgmath library and at runtime application aborts.
    - Workaround is to use -mllvm -force-vector-width=4 on subtargets without avx512 support.
    - TargetLibraryInfo doesn't have Subtarget info, so we don't know whether avx512 is enabled. Still investigating potentially fixing it here. Another option may be to add Driver logic to force -mllvm -force-vector-width=4 when pgmath is enabled and avx512 is not available (according to -march or -mcpu).
    - Removing the pgmath vector functions will make the generated code inefficient for avx512 target, so the original workaround patch is not suitable.
    - **Still working on this; have not yet found the right solution.**
- Issue: Windows CI builds are failing in the Download Artifact step. Bryan will investigate.
- Other PRs
    - llvm#178: Port Classic Flang to LLVM 18.1.1
        - Resolved minor cherry-pick conflicts; added one change for libpgmath function mappings. There will probably be test failures.**Needs reviews.**
    - llvm#176: fix declaration does not match definition when enable classic flang
        - Work-in-progress patch.
    - flang#1440: [test] pass Mnobackslash explicitly to the tests that assume default…
        - Prerequisite for flang#1028: Make backslash behaviour compatible with gfortran

        - A classic-flang-llvm-project patch will follow.
        - **Needs reviews.**
    - ARM has attempted to build classic flang with LLVM18. Two PRs were issued and **need reviews**:
        - flang#1438 With LLVM18 a loop preheader is being exposed, hence the preds must be more liberal
        - flang#1439 The checks for AArch64 target features must be more flexible
        - No other problems encountered
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; the author promised to address it after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
        - **No update this week.**
    - flang#642: AArch64: significantly improve formatted input performance by using optimized libc functions on ARM64
        - Looks like a good idea; needs reviews.
        - Huawei to take a look. **No update yet.**

## Mar 6, 2024

Attendees
AMD: Shivarama
Arm: Pawel
Huawei:Bryan
NVIDIA:

- Issue #399:
  Issue with BIND(C) with CHARACTER(1) which is interoperable but causes a crash in flang2. For non-interoperable types like CHARACTER(N), we need to throw semantic error until f2018 is supported. Bryan will do further analysis about this issue.
  Reference: Modern Fortran: Features for High-Performance Computing (stevelionel.com)

- Issue #1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - Downstream compilers are not having this issue. Need to be investigated.
        - Arm will be dropping as much of the downstream-only LLVM code as possible in order to reduce the downstream delta (lots of upstreaming commits are under public review), hopefully one day we'll get to the point where either upstream got fixed or downstream got hit by this issue

    - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
- Huawei will open a new issue about OpenMP hang in CESM (only when linking statically); a reproducer should be available soon.
    - Pawel suggests trying OpenMP runtime environment variables regarding locking behaviour.
    - **No update this week**
- Issue [#1437](#1437): malloc(): corrupted top size after allocating array of polymorphic data type elements
    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
    - **Issue is not yet assigned**
- Issue [#1436](#1436): Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
    - **Issue is not yet assigned**
- Issue [#1435](#1435): Wrong understanding of the rules for lbound and ubound for non-pointer dummy arguments
    - Nathan Sircombre from Arm is considering a response to Wanbinchen's comment. Nathan will discuss this during the Fortran Language committee meeting.
    - **No updates this week**
- Issue [#1413](#1413) related to pgmath
    - Workaround patch is provided.
    - Need rework of VecFuncs.def file for AVX2/AVX512 support.
    - **Issue is closed for now, but AMD will revisit this after the LLVM 17 upgrade.**
        - This has already happened, LLVM17 is the default branch for classic-flang now…
        - AMD will work on a fix in the weeks to come.
    - The issue happens when -mavx512 is not specified in the command line. The compiler is vectorizing with 8 vector factor [8 x double]. Since the zmm registers are not available, it is using ymm registers. But this will not have definitions in pgmath library and at runtime application aborts.
    - Workaround is to use -mllvm -force-vector-width=4.
    - TargetLibraryInfo doesn't have Subtarget info, so we don't know whether avx512 is enabled. Still investigating potentially fixing it here.
    - Removing the pgmath vector functions will make the generated code inefficient for avx512 target, so the original workaround patch is not suitable.

    - **Still working on this. Not yet found right solution**
- Issue #1028: Make backslash behaviour compatible with gfortran
    - No objection from all parties. Shivarama will take a stab.
    - Patch creation in progress. Facing issues with git login.
- Other PRs
    - ARM has attempted to build classic flang with LLVM18. Two PRs were issued:
        - flang#1438 With LLVM18 a loop preheader is being exposed, hence the preds must be more liberal
        - flang#1439 The checks for AArch64 target features must be more flexible
        - No other problems encountered
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; the author promised to address it after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
        - **No updates this week**
    - flang#642: AArch64: significantly improve formatted input performance by using optimized libc functions on ARM64
        - Looks like a good idea; needs reviews.
        - Huawei to take a look. **No update yet.**

## Feb 21, 2024

Attendees
AMD: Shivarama
Arm: Pawel
Huawei:Bryan
NVIDIA:

- Issue #399:
  Issue with BIND(C) with CHARACTER(1) which is interoperable but causes a crash in flang2. For non-interoperable types like CHARACTER(N), we need to throw semantic error until f2018 is supported. Bryan will do further analysis about this issue.
  Reference: Modern Fortran: Features for High-Performance Computing (stevelionel.com)

- Issue #1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - Downstream compilers are not having this issue. Need to be investigated.
        - Arm will be dropping as much of the downstream-only LLVM code as possible in order to reduce the downstream delta (lots of upstreaming

        commits are under public review), hopefully one day we'll get to the point where either upstream got fixed or downstream got hit by this issue
    - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
- Huawei will open a new issue about OpenMP hang in CESM (only when linking statically); a reproducer should be available soon.
    - Pawel suggests trying OpenMP runtime environment variables regarding locking behaviour.
    - **No update this week**
- Issue #1437: malloc(): corrupted top size after allocating array of polymorphic data type elements
    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
    - **Issue is not yet assigned**
- Issue #1436: Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
    - **Issue is not yet assigned**
- Issue #1435: Wrong understanding of the rules for lbound and ubound for non-pointer dummy arguments
    - Nathan Sircombre from Arm is considering a response to Wanbinchen's comment. Nathan will discuss this in Fortran Language committee.
    - **No updates this week**
- Issue #1413 related to pgmath
    - Workaround patch is provided.
    - Need rework of VecFuncs.def file for AVX2/AVX512 support.
    - **Issue is closed for now, but AMD will revisit this after the LLVM 17 upgrade.**
        - This has already happened, LLVM17 is the default branch for classic-flang now…
        - AMD will work on a fix this week.
    - The issue happens when -mavx512 is not specified in the command line. The compiler is vectorizing with 8 vector factor [8 x double]. Since the zmm registers are not available, it is using ymm registers. But this will not have definitions in pgmath library and at runtime application aborts.
    - Workaround is to use -mllvm -force-vector-width=4.
    - TargetLibraryInfo doesn't have Subtarget info, so we don't know whether avx512 is enabled. Still investigating potentially fixing it here.

    - Removing the pgmath vector functions will make the generated code inefficient for avx512 target, so the original workaround patch is not suitable.
    - **Still working on this. Not yet found right solution**
- Issue #1028: Make backslash behaviour compatible with gfortran
    - No objection from all parties. Shivarama will take a stab.
    - Patch creation in progress
- Other PRs
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; author promised to address after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
        - **No updates this week**
    - flang#642: AArch64: significantly improve formatted input performance by using optimized libc functions on ARM64
        - Looks like a good idea; needs reviews.
        - Huawei to take a look. **No update yet.**

## Feb 7, 2024

Attendees
AMD: Shivarama
Arm: Pawel
Huawei:
NVIDIA:

- Issue #1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - Downstream compilers are not having this issue. Need to be investigated.
        - Arm will be dropping as much of the downstream-only LLVM code as possible in order to reduce the downstream delta (lots of upstreaming commits are under public review), hopefully one day we'll get to the point where either upstream got fixed or downstream got hit by this issue
    - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
- Huawei will open a new issue about OpenMP hang in CESM (only when linking statically); a reproducer should be available soon.
    - Pawel suggests trying OpenMP runtime environment variables regarding locking behaviour.
    - **No update this week**
- Issue #1437: malloc(): corrupted top size after allocating array of polymorphic data type elements

    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
    - **Issue is not yet assigned**
- Issue [#1436](): Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
    - **Issue is not yet assigned**
- Issue [#1435](): Wrong understanding of the rules for lbound and ubound for non-pointer dummy arguments
    - Nathan Sircombre from Arm is considering a response to Wanbinchen's comment. Nathan will discuss this in Fortran Language committee.
    - **No updates this week**
- Issue [#1413]() related to pgmath
    - Workaround patch is provided.
    - Need rework of VecFuncs.def file for AVX2/AVX512 support.
    - **Issue is closed for now, but AMD will revisit this after the LLVM 17 upgrade.**
        - This has already happened, LLVM17 is the default branch for classic-flang now…
        - AMD will work on a fix this week.
    - The issue happens when -mavx512 is not specified in the command line. The compiler is vectorizing with 8 vector factor [8 x double]. Since the zmm registers are not available, it is using ymm registers. But this will not have definitions in pgmath library and at runtime application aborts.
    - Workaround is to use -mllvm -force-vector-width=4.
    - TargetLibraryInfo doesn't have Subtarget info, so we don't know whether avx512 is enabled. Still investigating potentially fixing it here.
    - Removing the pgmath vector functions will make the generated code inefficient for avx512 target, so the original workaround patch is not suitable.
- Issue [#1028](): Make backslash behaviour compatible with gfortran
    - No objection from all parties. Shivarama will take a stab.
    - Patch creation in progress
- Other PRs
    - [llvm#172](): Work around OpenMP test failures on GitHub
        - **Merged.**
    - [flang#1434](): fix the expected number of parameters for __pd_asin_1
        - **Merged.**
    - [flang#1433](): Update CI workflows to test the right branches
        - Pawel reviewed but the branch has been updated.

        - Something is still wrong with the JSON queries for prebuilt artifacts. Will restart CI after the llvm#172 CI is complete.
    - flang#1427: Add align pragma for scalar data types (integer, logical, real, complex)
        - Extension of flang#1360. Author has rebased his work. Needs reviews.
        - **Merged**
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; author promised to address after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
        - **No updates this week**
    - flang#642: AArch64: significantly improve formatted input performance by using optimized libc functions on ARM64
        - Looks like a good idea; needs reviews.
        - Huawei to take a look. **No update yet.**

## Jan 24, 2024

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan
NVIDIA:

- Issue #1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - Downstream compilers are not having this issue. Need to be investigated.
        - Arm will be dropping as much of the downstream-only LLVM code as possible in order to reduce the downstream delta (lots of upstreaming commits are under public review), hopefully one day we'll get to the point where either upstream got fixed or downstream got hit by this issue
    - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
- Huawei will open a new issue about OpenMP hang in CESM (only when linking statically); a reproducer should be available soon.
    - Pawel suggests trying OpenMP runtime environment variables regarding locking behaviour.
- Issue #1437: malloc(): corrupted top size after allocating array of polymorphic data type elements
    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never

    returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
- Issue #1436: Uninitialized stack variable during string initialization in a function that returns a string of the parametrized length
    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
- Issue #1435: Wrong understanding of the rules for lbound and ubound for non-pointer dummy arguments
    - Nathan Sircombre from Arm is considering a response to Wanbinchen's comment.
- Issue #1413 related to pgmath
    - Workaround patch is provided.
    - Need rework of VecFuncs.def file for AVX2/AVX512 support.
    - **Issue is closed for now, but AMD will revisit this after the LLVM 17 upgrade.**
        - This has already happened, LLVM17 is the default branch for classic-flang now…
        - AMD will work on a fix this week.
    - The issue happens when -mavx512 is not specified in the command line. The compiler is vectorizing with 8 vector factor  [8 x double]. Since the zmm registers are not available, it is using ymm registers. But this will not have definitions in pgmath library and at runtime application aborts.
    - Workaround is to use -mllvm -force-vector-width=4.
    - TargetLibraryInfo doesn't have Subtarget info, so we don't know whether avx512 is enabled. Still investigating potentially fixing it here.
    - Removing the pgmath vector functions will make the generated code inefficient for avx512 target, so the original workaround patch is not suitable.
- Issue #1028: Make backslash behaviour compatible with gfortran
    - No objection from all parties. Shivarama will take a stab.
- Other PRs
    - llvm#172: Work around OpenMP test failures on GitHub
        - **Merged.**
    - flang#1434: fix the expected number of parameters for __pd_asin_1
        - **Merged.**
    - flang#1433: Update CI workflows to test the right branches
        - Pawel reviewed but the branch has been updated.
        - Something is still wrong with the JSON queries for prebuilt artifacts. Will restart CI after the llvm#172 CI is complete.
    - flang#1427: Add align pragma for scalar data types (integer, logical, real, complex)
        - Extension of flang#1360. Author has rebased his work. Needs reviews.
    - flang#1418: Fix a min-max bug

        - MIN/MAX intrinsics were not conforming to the Fortran 2008 standard.
        - The patch is incomplete; author promised to address after vacation.
        - Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.
        - **Author has moved on; Bryan will commandeer this patch.**
    - flang#642: AArch64: significantly improve formatted input performance by using optimized libc functions on ARM64
        - Looks like a good idea; needs reviews.
        - Huawei to take a look. **No update yet.**

## Jan 10, 2024

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan
NVIDIA:

- Issue #1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - Downstream compilers are not having this issue. Need to be investigated.
        - Arm will be dropping as much of the downstream-only LLVM code as possible in order to reduce the downstream delta (lots of upstreaming commits are under public review), hopefully one day we'll get to the point where either upstream got fixed or downstream got hit by this issue
    - **A reduced test case would be necessary and helpful to debug the issue further. No update this week.**
- Huawei will open a new issue about OpenMP hang in CESM (only when linking statically); a reproducer should be available soon.
    - Pawel suggests trying OpenMP runtime environment variables regarding locking behaviour.
- Arm will open two issues:
    - Ubuntu 22.04 causes malloc-after-malloc to fail; ALLOCATE for polymorphic types always sends 0 as the length parameter; the allocation function never returns NULL so the problem never caused a crash or malloc error, but this changed in Ubuntu 22.04.
    - In f90_str_cpy1, memset is given an incorrect pointer after upgrading to Ubuntu 22.04. The execution path resulting in calling memset is never taken on older operating systems for some reason. Problem was observed when running NAG test suite.
- Issue #1413 related to pgmath

    - Workaround patch is provided.
    - Need rework of VecFuncs.def file for AVX2/AVX512 support.
    - **Issue is closed for now, but AMD will revisit this after the LLVM 17 upgrade.**
        - This has already happened, LLVM17 is the default branch for classic-flang now…
        - AMD will work on a fix this week.
    - The issue happens when -mavx512 is not specified in the command line. The compiler is vectorizing with 8 vector factor  [8 x double]. Since the zmm registers are not available, it is using ymm registers. But this will not have definitions in pgmath library and at runtime application aborts.
    - Workaround is to use -mllvm -force-vector-width=4
    - Removing the pgmath vector functions will make the generated code inefficient for avx512 target.
- Other PRs
    - llvm#172: Work around OpenMP test failures on GitHub
        - Shivarama will review.
    - flang#1434: fix the expected number of parameters for __pd_asin_1
        - Shivarama will review.
    - flang#1427: Add align pragma for scalar data types (integer, logical, real, complex)
        - Extension of flang#1360. Author needs to rebase his work.
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming the Fortran 2008 standard.
        - The patch is incomplete; author promised to address after vacation.
        - **Reviews needed. Bryan reviewed and commented; Pawel left a comment. Change required.**
    - flang#1360: Add ALIGN pragma for derived type and fixed-shape array/character type
        - **Merged.**
    - flang#647: Introduce buffered I/O and replace getc with buffered read
        - **Needs reviews, and a rebase.**
        - Rebasing will cause lots of conflict; Pawel has no time but we agree to keep it around and study its relevance some more.
        - **No more work will be done. Closing.**
    - flang#642: AArch64: significantly improve formatted input performance by using optimized libc functions on ARM64
        - Looks like a good idea; **needs reviews.**
        - **Huawei to take a look.**

## Dec 13, 2023

Attendees
AMD: Shivarama

Arm: Pawel
Huawei: Bryan
NVIDIA:
Last meeting of 2023 will be canceled.
Others:

- Issue #1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - Downstream compilers are not having this issue. Need to be investigated.
        - Arm will be dropping as much of the downstream-only LLVM code as possible in order to reduce the downstream delta (lots of upstreaming commits are under public review), hopefully one day we'll get to the point where either upstream got fixed or downstream got hit by this issue
    - **A reduced testcase would be necessary and helpful to debug the issue further.**
- Issue #1413 related to pgmath
    - Workaround patch is provided.
    - Need rework of VecFuncs.def file for AVX2/AVX512 support.
    - **Issue is closed for now, but AMD will revisit this after the LLVM 17 upgrade.**
        - This has already happened, LLVM17 is the default branch for classic-flang now…
        - Possible update on the next meeting
- Other PRs
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming the Fortran 2008 standard.
        - The patch is incomplete; author promised to address after vacation.
        - **Author pushed new patch**
        - **Reviews needed. Bryan reviewed and commented**
    - flang#1360: Add ALIGN pragma for derived type and fixed-shape array/character type
        - **AMD and Arm - approved the code changes**.
        - There are tests failing, but it doesn't seem to be related to this patch. Can we merge this?
    - flang#647: Introduce buffered I/O and replace getc with buffered read
        - **Needs reviews, and a rebase.**
        - Rebasing will cause lots of conflict; Pawel has no time but we agree to keep it around and study its relevance some more.
        - **No updates this week.**
    - flang#642: AArch64: significantly improve formatted input performance by using optimized libc functions on ARM64
        - Looks like a good idea; **needs reviews.**
        - **No updates**

## November 29, 2023

Attendees
AMD: Shivarama
Arm: Pawel
Huawei:
NVIDIA:
Others:

- Timing of the call - Do we need to change to accommodate daylight savings?
  - Pawel agrees. OK with AMD
  - Will be rescheduled after confirmation from Bryan
- Issue #1429: 521.wrf_r segfaults when built with -flto=full -Ofast
  - Caused by upstream changes in LLVM 17
  - Downstream compilers are not having this issue. Need to be investigated.
  - **A reduced testcase would be necessary and helpful to debug the issue further.**
- Issue #1413 related to pgmath
  - Workaround patch is provided.
  - Need rework of VecFuncs.def file for AVX2/AVX512 support.
  - **Issue is closed for now, but AMD will revisit this after the LLVM 17 upgrade.**
- A set of PRs by Dmitry Mikushin
  - Closed a few PRs that are not accepted; pinged the author on a few others.
  - Dmitry responded on the PRs; will follow up.
- QuadFP Complex Support Plan
  - Compiler support patches have been merged.
  - Huawei's partner will upstream more patches to implement intrinsics, but this is not high priority at the moment.
  - **No update.**
- Windows related PRs
  - Windows GitHub workflow is now completely implemented
    - Currently tests cannot run on Windows due to portability issues in the test suite; PRs have been raised to adapt the test suite.
- Other PRs
  - flang#1418: Fix a min-max bug
    - MIN/MAX intrinsics were not conforming the Fortran 2008 standard.
    - The patch is incomplete; author promised to address after vacation.
    - **Author pushed new patch**
    - **Reviews needed. Bryan reviewed and commented**
  - flang#1360: Add ALIGN pragma for derived type and fixed-shape array/character type
    - Kiran raised a question about the design of the pragma and advises adopting the syntax of existing compilers (e.g. Intel's)
    - Large patch; needs more reviews

        - AMD will review this patch
        - Bryan has given review comments
        - Author has refactored the patch in September to address comments; **needs new reviews. No updates this week**
        - There is another pull request 1427 that has similar implementation and authors are coming up with combined patch.
        - Need review. Bryan reviewed.
        - **AMD - approves the code changes**.
    - flang#647: Introduce buffered I/O and replace getc with buffered read
        - **Needs reviews, and a rebase.**
        - Rebasing will cause lots of conflict; Pawel has no time but we agree to keep it around and study its relevance some more.
        - **No updates this week.**
    - flang#642: AArch64: significantly improve formatted input performance by using optimized libc functions on ARM64
        - Looks like a good idea; **needs reviews.**
        - **No updates**

## November 15, 2023

Attendees

AMD: Shivarama

Arm: Pawel

Huawei:

NVIDIA:

Others:

- Issue #1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - Downstream compilers are not having this issue?. Need to be investigated.
    - **A reduced testcase would be required to debug the issue further.**
- Issue #1413 related to pgmath
    - Workaround patch is provided.
    - Need rework of VecFuncs.def file for AVX2/AVX512 support.
    - **Issue is closed for now, but AMD will revisit this after the LLVM 17 upgrade.**
- A set of PRs by Dmitry Mikushin
    - Closed a few PRs that are not accepted; pinged the author on a few others.
    - Dmitry responded on the PRs; will follow up.
- QuadFP Complex Support Plan
    - Compiler support patches have been merged.
    - Huawei's partner will upstream more patches to implement intrinsics, but this is not high priority at the moment.
    - **No update.**

- Windows related PRs
  - Windows GitHub workflow is now completely implemented
    - Currently tests cannot run on Windows due to portability issues in the test suite; PRs have been raised to adapt the test suite.
  - flang#1408: Fix the dllimport code generation for opaque type
    - Symbols defined in a module should have dllimport attribute.
    - Bryan asked for a test case; Adam will provide.
    - **Approved and merged**
- Other PRs
  - flang#1422: Fix semantic analysis failures of LBOUND/UBOUND in type specification
    - LBOUND/UBOUND is not handled correct in the type spec of a function return value, leading to incorrect code and/or ICE crashes.
    - Lots of test cases have been provided.
    - **Approved and merged**
  - flang#1418: Fix a min-max bug
    - MIN/MAX intrinsics were not conforming the Fortran 2008 standard.
    - The patch is incomplete; author promised to address after vacation.
    - **Author pushed new patch**
    - **Reviews needed. Bryan reviewed and commented**
  - flang#1417: Fix an entry statement debuginfo bug
    - Bryan suggested more test cases and a reworded commit message.
    - **Approved and merged**
  - flang#1360: Add ALIGN pragma for derived type and fixed-shape array/character type
    - Kiran raised a question about the design of the pragma and advises adopting the syntax of existing compilers (e.g. Intel's)
    - Large patch; needs more reviews
    - AMD will review this patch
    - Bryan has given review comments
    - Author has refactored the patch in September to address comments; **needs new reviews. No updates this week**
    - There is another pull request 1427 that has similar implementation and authors are coming up with combined patch.
    - Need review. Bryan reviewed
  - flang#1259: loop metadata: add two more test cases and improve loop discovery
    - Bryan reviewed but suggested reducing code duplication.
    - **Closed by Pawel. The pull request is not relevant.**
  - flang#919: Fix issue #670: Internal compiler error
    - Gary has retired, but this patch seems valuable; someone should check if it is still needed.
    - Bryan checked and found that the problem is already fixed. **Closed.**
  - flang#887: Support pointers to scalar variables during debug info generation

        - Author (bhuvanendrakumarn) has not responded in many years
        - Not sure if the issue is still a problem; Shivarama will check.
        - **Author has closed the PR as the fix is already checked in.**
    - flang#877: flang generated executable shows wrong value for character type in debugger
        - Asked Alok whether this was still needed in July 2022; no answer.
        - **The pull request is closed by Author (Alok)**
    - flang#647: Introduce buffered I/O and replace getc with buffered read
        - **Needs reviews, and a rebase.**
        - Rebasing will cause lots of conflict; Pawel has no time but we agree to keep it around and study its relevance some more.
    - flang#642: AArch64: significantly improve formatted input performance by using optimized libc functions on ARM64
        - Looks like a good idea; **needs reviews.**
    - **No updates**
- Issue #1426 : F2008 CONTIGUOUS attribute is not correct for array slice arguments to C functions
    - When passing a slice to a procedure, the entire array and slice information are passed, but from the callee's perspective, the data is no longer contiguous.
    - Issue: Fortran procedure calls a C function from NetCDF; the C function assumes that the data is contiguous in the array it receives, and as a result the data in the array appears corrupted.
    - gfortran makes a copy of the array slice and passes the copy correctly.
    - AMD may have a fix (originally for a CP2K 8.1 problem); Shivarama will send an email and follow up with a PR.
    - Related problem: IS_CONTIGUOUS always returns true if a dummy argument is marked CONTIGUOUS, whether the actual argument is contiguous or not.
    - AMD provided a patch to Pawel and it works for netcdf. A pull request will be created soon.
    - **Pull request is created. Approved and merged.**

## November 01, 2023

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan
NVIDIA:
Others:

- Issue #1429: 521.wrf_r segfaults when built with -flto=full -Ofast
    - Caused by upstream changes in LLVM 17
    - Bryan will update the ticket with more details
- Issue #1413 related to pgmath

    - Workaround patch is provided.
    - Need rework of VecFuncs.def file for AVX2/AVX512 support.
    - **Issue is closed for now, but AMD will revisit this after the LLVM 17 upgrade.**
- A set of PRs by Dmitry Mikushin
    - Closed a few PRs that are not accepted; pinged the author on a few others.
    - Dmitry responded on the PRs; will follow up.
- QuadFP Complex Support Plan
    - Compiler support patches have been merged.
    - Huawei's partner will upstream more patches to implement intrinsics, but this is not high priority at the moment.
    - **No update.**
- Windows related PRs
    - Windows GitHub workflow is now completely implemented
        - Currently tests cannot run on Windows due to portability issues in the test suite; PRs have been raised to adapt the test suite.
    - flang#1408: Fix the dllimport code generation for opaque type
        - Symbols defined in a module should have dllimport attribute.
        - Bryan asked for a test case; Adam will provide.
        - **Approved; will be merged.**
- Other PRs
    - flang#1422: Fix semantic analysis failures of LBOUND/UBOUND in type specification
        - LBOUND/UBOUND is not handled correct in the type spec of a function return value, leading to incorrect code and/or ICE crashes.
        - Lots of test cases have been provided.
        - **Needs reviews. Approved by Bryan. Will be merged**
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming the Fortran 2008 standard.
        - The patch is incomplete; author promised to address after vacation.
        - **No update.**
    - flang#1417: Fix an entry statement debuginfo bug
        - Bryan suggested more test cases and a reworded commit message.
        - **Needs reviews from AMD and ARM**
    - flang#1360: Add ALIGN pragma for derived type and fixed-shape array/character type
        - Kiran raised a question about the design of the pragma and advises adopting the syntax of existing compilers (e.g. Intel's)
        - Large patch; needs more reviews
        - AMD will review this patch
        - Bryan has given review comments
        - Author has refactored the patch in September to address comments; **needs new reviews. No updates this week**

        - There is another pull request 1427 that has similar implementation and authors are coming up with combined patch.
    - flang#1259: loop metadata: add two more test cases and improve loop discovery
        - Bryan reviewed but suggested reducing code duplication.
        - **Needs review from AMD.**
        - **No updates this week. Will be completed before next meeting**
    - flang#919: Fix issue #670: Internal compiler error
        - Gary has retired, but this patch seems valuable; someone should check if it is still needed.
        - Bryan checked and found that the problem is already fixed. **Closed.**
    - flang#887: Support pointers to scalar variables during debug info generation
        - Author (bhuvanendrakumarn) has not responded in many years
        - Not sure if the issue is still a problem; Shivarama will check.
        - **Author has closed the PR as the fix is already checked in.**
    - flang#877: flang generated executable shows wrong value for character type in debugger
        - Asked Alok whether this was still needed in July 2022; no answer.
        - **This pull request will be closed by Alok**
    - flang#647: Introduce buffered I/O and replace getc with buffered read
        - **Needs reviews, and a rebase.**
        - Rebasing will cause lots of conflict; Pawel has no time but we agree to keep it around and study its relevance some more.
    - flang#642: AArch64: significantly improve formatted input performance by using optimized libc functions on ARM64
        - Looks like a good idea; **needs reviews.**
- Issue #1426 : F2008 CONTIGUOUS attribute is not correct for array slice arguments to C functions
    - When passing a slice to a procedure, the entire array and slice information are passed, but from the callee's perspective, the data is no longer contiguous.
    - Issue: Fortran procedure calls a C function from NetCDF; the C function assumes that the data is contiguous in the array it receives, and as a result the data in the array appears corrupted.
    - gfortran makes a copy of the array slice and passes the copy correctly.
    - AMD may have a fix (originally for a CP2K 8.1 problem); Shivarama will send an email and follow up with a PR.
    - Related problem: IS_CONTIGUOUS always returns true if a dummy argument is marked CONTIGUOUS, whether the actual argument is contiguous or not.
    - AMD provided a patch to Pawel and it works for netcdf. A pull request will be created soon.

## October 18, 2023

Attendees

AMD: Shivarama, Kiran TP
Arm: Pawel
Huawei: Bryan
NVIDIA:
Others:

- Bryan is submitting two PRs for LLVM 17 upgrade.
    - llvm#166: Port Classic Flang to LLVM 17.0.2
        - Update LLVM version passed to flang1/flang2.
        - Update header locations in #include directives.
        - Fix some downstream (Classic Flang-specific) tests.
        - **Merged.**
    - flang#1425: Upgrade to LLVM 17
        - Fix -Wcast-qual errors.
        - Fix test cases for LLVM 17 compatibility
        - Prerequisite for the classic-flang-llvm-project PR.
        - **Merged.**
- Issue #1413 related to pgmath
    - Workaround patch is provided.
    - Need rework of VecFuncs.def file for AVX2/AVX512 support.
    - **Issue is closed for now, but AMD will revisit this after the LLVM 17 upgrade.**
- A set of PRs by Dmitry Mikushin
    - Closed a few PRs that are not accepted; pinged the author on a few others.
    - Dmitry responded on the PRs; will follow up.
- QuadFP Complex Support Plan
    - Compiler support patches have been merged.
    - Huawei's partner will upstream more patches to implement intrinsics, but this is not high priority at the moment.
    - **No update.**
- Windows related PRs
    - Windows GitHub workflow is now completely implemented
        - Currently tests cannot run on Windows due to portability issues in the test suite; PRs have been raised to adapt the test suite.
    - flang#1408: Fix the dllimport code generation for opaque type
        - Symbols defined in a module should have dllimport attribute.
        - Bryan asked for a test case; Adam will provide.
        - **Approved; will be merged.**
- Other PRs
    - flang#1422: Fix semantic analysis failures of LBOUND/UBOUND in type specification
        - LBOUND/UBOUND is not handled correct in the type spec of a function return value, leading to incorrect code and/or ICE crashes.
        - Lots of test cases have been provided.
        - **Needs reviews.**

    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming the Fortran 2008 standard.
        - The patch is incomplete; author promised to address after vacation.
        - **No update.**
    - flang#1417: Fix an entry statement debuginfo bug
        - Bryan suggested more test cases and a reworded commit message.
        - **Needs reviews.**
    - flang#1360: Add ALIGN pragma for derived type and fixed-shape array/character type
        - Kiran raised a question about the design of the pragma and advises adopting the syntax of existing compilers (e.g. Intel's)
        - Large patch; needs more reviews
        - AMD will review this patch
        - Bryan has given review comments
        - Author has refactored the patch in September to address comments; **needs new reviews**
    - flang#1259: loop metadata: add two more test cases and improve loop discovery
        - Bryan reviewed but suggested reducing code duplication.
        - **Needs review from AMD.**
    - flang#919: Fix issue #670: Internal compiler error
        - Gary has retired, but this patch seems valuable; someone should check if it is still needed.
        - Bryan checked and found that the problem is already fixed. **Closed.**
    - flang#887: Support pointers to scalar variables during debug info generation
        - Author (bhuvanendrakumarn) has not responded in many years
        - Not sure if the issue is still a problem; Shivarama will check.
        - **Author has closed the PR as the fix is already checked in.**
    - flang#877: flang generated executable shows wrong value for character type in debugger
        - Asked Alok whether this was still needed in July 2022; no answer.
        - Shivarama will follow up.
    - flang#647: Introduce buffered I/O and replace getc with buffered read
        - **Needs reviews, and a rebase.**
        - Rebasing will cause lots of conflict; Pawel has no time but we agree to keep it around and study its relevance some more.
    - flang#642: AArch64: significantly improve formatted input performance by using optimized libc functions on ARM64
        - Looks like a good idea; **needs reviews.**
    - flang#529: runtime: PoC: generate call to LLVM intrinsic instead of calling runtime functions when possible; handling NINT for a beginning
        - This was never implemented downstream and can be closed.
- Pawel will file an issue: F2008 CONTIGUOUS attribute is not correct for array slice arguments to C functions

    - When passing a slice to a procedure, the entire array and slice information are passed, but from the callee's perspective, the data is no longer contiguous.
    - Issue: Fortran procedure calls a C function from NetCDF; the C function assumes that the data is contiguous in the array it receives, and as a result the data in the array appears corrupted.
    - gfortran makes a copy of the array slice and passes the copy correctly.
    - AMD may have a fix (originally for a CP2K 8.1 problem); Shivarama will send an email and follow up with a PR.
    - Related problem: IS_CONTIGUOUS always returns true if a dummy argument is marked CONTIGUOUS, whether the actual argument is contiguous or not.

## October 4, 2023

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan
NVIDIA:
Others:

- LLVM 17-related PRs
    - flang#1415: Fix failures for imports with LLVM-17 (followup patch)
        - With flang#1414, runtime is not getting built. The above patch fixes the issue.
        - What is our backward compatibility policy?
            - Maybe we don't need to remove the "imports" entry?
            - Pawel will create a GitHub issue to cover further discussion.
        - This patch is an error and not actually needed; **closed.**
- Bryan is submitting two PRs for LLVM 17 upgrade.
    - llvm#???
        - Update LLVM version passed to flang1/flang2.
        - Update header locations in #include directives.
    - flang#1425: Upgrade to LLVM 17
        - Fix -Wcast-qual errors.
        - Fix test cases for LLVM 17 compatibility
        - Prerequisite for the classic-flang-llvm-project PR.
- Issue #1413 related to pgmath
    - Workaround patch is provided.
    - Need rework of VecFuncs.def file for AVX2/AVX512 support.
    - **Issue is closed for now, but AMD will revisit this after the LLVM 17 upgrade.**
- A set of PRs by Dmitry Mikushin
    - No update.
- QuadFP Complex Support Plan
    - Compiler support patches have been merged.

    - Huawei's partner will upstream more patches to implement intrinsics, but this is not high priority at the moment.
- Windows related PRs
    - Windows GitHub workflow is now completely implemented
        - Currently tests cannot run on Windows due to portability issues in the test suite; PRs have been raised to adapt the test suite.
    - flang#1408: Fix the dllimport code generation for opaque type
        - Symbols defined in a module should have dllimport attribute.
        - Approved (Bryan and Pawel reviewed).
        - Bryan asked for a test case; Adam will provide.
- Other PRs
    - flang#1422: Fix semantic analysis failures of LBOUND/UBOUND in type specification
        - LBOUND/UBOUND is not handled correct in the type spec of a function return value, leading to incorrect code and/or ICE crashes.
        - Lots of test cases have been provided.
        - **Needs reviews.**
    - flang#1420: [flang2] byval and sret attributes should have associated types
        - Fixes flang#1419 and flang#1423; extends and replaces flang#1416.
        - The test case is split across architectures due to ABI differences.
        - **Merged.**
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming the Fortran 2008 standard
        - The patch is incomplete; author will address after vacation
    - flang#1417: Fix an entry statement debuginfo bug
        - Bryan suggested more test cases and a reworded commit message.
        - **Needs reviews.**
    - flang#1360: Add ALIGN pragma for derived type and fixed-shape array/character type
        - Kiran raised a question about the design of the pragma and advises adopting the syntax of existing compilers (e.g. Intel's)
        - Large patch; needs more reviews
        - AMD will review this patch
        - Bryan has given review comments
        - Author has refactored the patch in September to address comments; **needs new reviews**
    - flang#1220: libpgmath: add support for generic OSX builds
        - Is OSX support important? Should we try to get this PR merged, or just close it? Consensus is to close it.
    - flang#919: Fix issue #670: Internal compiler error
        - Gary has retired, but this patch seems valuable; someone should check if it is still needed. Bryan will follow up.
    - flang#887: Support pointers to scalar variables during debug info generation

        - Author (bhuvanendrakumarn) has not responded in many years
        - Not sure if the issue is still a problem; **Shivarama will check.**
    - Closed some stale PRs.

## September 20, 2023

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan
NVIDIA:
Others:

- LLVM-17 related PRs
    - flang#1415: Fix failures for imports with LLVM-17 (followup patch)
        - With flang#1414, runtime is not getting built. The above patch fixes the issue.
        - What is our backward compatibility policy?
            - Maybe we don't need to remove the "imports" entry?
            - An github issue will be created to cover further discussion
        - Need to be reviewed

- Issue #1413 related to pgmath
    - Workaround patch is provided
    - Need rework of vecFuncs.def file for avx2/avx512 support
    - check with llvm-17
    - **The issue is closed for now, but this subject will return after LLVM17 settles**

- A set of PRs by Dmitry Mikushin
    - No update

- QuadFP Complex Support Plan
    - Semantic analyses for type conversion and operand type promotion
    - I/O support
    - C_LONG_DOUBLE_COMPLEX
    - Trigonometry intrinsic functions
    - MATMUL and TRANSPOSE
    - A couple of optimizations
    - Huawei's partner will upstream more patches for intrinsics, but not high priority at the moment.

- Windows related PRs
    - Windows GitHub workflow is now completely implemented

            - Currently tests cannot run on Windows due to portability issues in the test suite; PRs have been raised to adapt the test suite
        - flang#1408: Fix the dllimport code generation for opaque type
            - Symbols defined in a module should have dllimport attribute
            - Pawel reviewed; Bryan asked for a test case
            - Approved - to be merged (but what about the test case?)

- Other PRs
    - llvm#165:[Driver] Fix flang driver preprocessor issue
        - More reviewers needed? Can it be merged already?
        - Merged.
    - flang#1360: Add ALIGN pragma for derived type and fixed-shape array/character type
        - Kiran raised a question about the design of the pragma and advises adopting the syntax of existing compilers (e.g. Intel's)
        - Large patch; needs more reviews
        - AMD will review this patch
        - Bryan has given review comments
        - The author is trying to find the time to refactor this PR based on our feedback and resolve the conflicts
            - Author was occupied until September
            - It's September and the author has presented a new batch of changes; needs new reviews
    - flang#1418: Fix a min-max bug
        - MIN/MAX intrinsics were not conforming the Fortran 2008 standard
        - Bryan requested some minor changes
        - More reviewers are welcomed
    - flang#1417: Fix an entry statement debuginfo bug
        - Bryan suggested more test cases
        - Reviewers added as assignees?
    - flang#1416: ~~Fix a sret attribute bug~~
        - **Closed and replaced by flang#1420**
        - When the return value of the function is a structure, sometimes the compiler will use the structure as an input argument and add the sret attribute in the IR generation stage. In this case, the IR generated by flang will lack the actual structure type.
        - Changes are requested (handling of the byval attribute)
    - flang#1420: [flang2] byval and sret attributes should have associated types
        - Extends and replaces flang#1416
        - Fails in CI, a test case (llvm_ir_correct/sret.f90) needs updating?
            - The test case will be split across architectures

## September 6, 2023

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan
NVIDIA:
Others: Dmitry?

- LLVM-17 related PRs
    - flang#1415: Fix failures for imports with LLVM-17 (followup patch)
        - With flang#1414, runtime is not getting built. The above patch fixes the issue.
        - Need to be reviewed
- Issue #1413 related to pgmath
    - Workaround patch is provided
    - Need rework of vecFuncs.def file for avx2/avx512 support
    - check with llvm-17
    - The issue is closed, but is the subject concluded? Yes for now, but this subject will return when llvm-17 settles
- A set of PRs by Dmitry Mikushin
    - flang#1378 (decomposed): Fix classic flang compilation with in-tree flang
        - This PR contains a lot of patches that are unrelated to each other.
        - This set of patches fixes various issues spotted during compilation of flang-classic with clang and LLVM in-tree flang.
        - For C/C++ it addresses #1374 and #1375 and fixes fatal clang issues, such as mismatching function pointer signature.
        - For Fortran it mostly adds a lot of `intent(in)` attributes.
        - **This PR has been decomposed into the list of PRs below.**
    - flang#1379: Add docker container for building flang
        - Do we really need this? It won't be tested unless we fully integrate it into our CI workflows. **Let's discuss with Dmitry in person next meeting.**
        - Related: Should we consolidate our build scripts?
            - **Bryan will push a PR to remove the .sh scripts and use .py scripts in CI, and update the wiki page**
        - **No update**
    - flang#1383: Use preprocessor in traditional mode
        - Fortran compiler is supposed to use traditional C preprocessor only. Therefore, we can't use modern C preprocessor features, such as concatenation or stringizing
        - Modifies just the runtime/flang/mmul_dir.h file

            - Stringizing and concatenation are not "traditional", but they seem to be supported by flang1 (fpp.c, accpp.c) natively? Should we remove functionality from flang1?
                - Pawel: Should we aim for full gfortran compatibility? Reaching out to NVIDIA. Previous conclusion was a "No".
                - Pawel: Making runtime code more portable is good; removing functionality is not necessary and may surprise existing users.
            - LLVM Flang supports stringization. We unanimously agree we shouldn't proceed with this PR.
            - **Rejected**
        - flang#1386: Emphasize && operator precedence with extra brackets, as suggested by clang warning
            - Single line patch
            - No reviewers assigned yet, but feel free to review
            - **No update this week (Formatting is needed)**
        - flang#1392: Adding intent(in)
            - Planting intent(in) all over the code, per strict modern Fortran compilers requirements.
            - Do not duplicate eq and == , ne and /= operators definitions, as they seem to be already aliases of each other, and compilers are not happy if we try to overdefine this fact.
            - Runtime library issue
            - No reviewers assigned yet, but feel free to review
            - Any thought on potential ill impact?
            - **No update this week (need testing details)**
        - flang#1400: Do not dump debug location if debug info is not available
            - Problem only occurs for GPU offload compilation? Worry is that the use of flags guarding debug offload compilation is inconsistent
            - Missing test case
            - Shivarama will review - need a testcase to validate
            - No update this week.
        - flang#1401: Link argument parsing functions from libraries, instead of duplicating their source files
            - Windows build is broken, but the refactoring seems okay.
            - **Rejected**
    - QuadFP Complex Support Plan
        - Semantic analyses for type conversion and operand type promotion
        - I/O support
        - C_LONG_DOUBLE_COMPLEX
        - Trigonometry intrinsic functions
        - MATMUL and TRANSPOSE
        - A couple of optimizations

        - Huawei's partner will upstream more patches for intrinsics, but not high priority at the moment.
        - There are new PRs waiting for the reviews
    - Windows related PRs
        - Windows GitHub workflow is now completely implemented
            - Currently tests cannot run on Windows due to portability issues in the test suite; PRs have been raised to adapt the test suite
        - flang#1408: Fix the dllimport code generation for opaque type
            - Symbols defined in a module should have dllimport attribute
            - Pawel reviewed; Bryan asked for a test case
            - Approved - to be merged
        - flang#1405: Add Windows support to lit.cfg
            - Bryan and Shivarama approved
            - merged
        - flang#1366: Add Windows support for f90_correct tests
            - Needs review
            - Awful lot of tests touched!
            - Merged.
        - llvm#161: Add Flang include directory to MSVC toolchain (release_15x)
            - Needs review
            - Merged.
    - Other PRs

            - flang#1360: Add ALIGN pragma for derived type and fixed-shape array/character type
                - Kiran raised a question about the design of the pragma and advises adopting the syntax of existing compilers (e.g. Intel's)
                - Large patch; needs more reviews
                - AMD will review this patch
                - Bryan has given review comments
                - The author is trying to find the time to refactor this PR based on our feedback and resolve the conflicts
                    - Author is occupied until September
                    - No update
            - flang#1314: Automatically deploy Sphinx documentation to GitHub Pages after a successful push to master
                - What happened to this PR and how? The status is unclear…
                    - Closed accidentally, will be re-issued again
                        - Still being closed
                    - **No update**

## August 23, 2023

Attendees
AMD: Shivarama
Arm:
Huawei: Bryan
NVIDIA:
Others:

- LLVM-17 related PRs
  - flang#1414: Fix failures for imports with LLVM-17
    - This PR contains patch that fixes the compatibility issues with LLVM-17
    - Approved and merged.
  - flang#1415: Fix failures for imports with LLVM-17 (followup patch)
    - With flang#1414, runtime is not getting built. The above patch

  Fixes the issue.
    - Need to be reviewed
- Issue #1413 related to pgmath
  - Workaround patch is provided
  - Need rework of vecFuncs.def file for avx2/avx512 support
  - check with llvm-17
- A set of PRs by Dmitry Mikushin
  - Dmitry will be pinged for these PRs
  - flang#1378: Fix classic flang compilation with in-tree flang
    - This PR contains a lot of patches that are unrelated to each other.
    - This set of patches fixes various issues spotted during compilation of flang-classic with clang and LLVM in-tree flang.
    - For C/C++ it addresses #1374 and #1375 and fixes fatal clang issues, such as mismatching function pointer signature.
    - For Fortran it mostly adds a lot of `intent(in)` attributes.
    - **This PR has been decomposed into the list of PRs below.**
  - flang#1379: Add docker container for building flang
    - Do we really need this? It won't be tested unless we fully integrate it into our CI workflows. **Let's discuss with Dmitry in person next meeting.**
    - Related: Should we consolidate our build scripts?

        - **Bryan will push a PR to remove the .sh scripts and use .py scripts in CI, and update the wiki page**
    - **No  update**
- flang#1383: Use preprocessor in traditional mode
    - Fortran compiler is supposed to use traditional C preprocessor only. Therefore, we can't use modern C preprocessor features, such as catenation or stringizing
    - Modifies just the runtime/flang/mmul_dir.h file
    - Stringizing and concatenation are not "traditional", but they seem to be supported by flang1 (fpp.c, accpp.c) natively? Should we remove functionality from flang1?
        - Pawel: Should we aim for full gfortran compatibility? Reaching out to NVIDIA. Previous conclusion was a "No".
        - Pawel: Making runtime code more portable is good; removing functionality is not necessary and may surprise existing users.
    - LLVM Flang supports stringization. We unanimously agree we shouldn't proceed with this PR.
    - rejected
- flang#1386: Emphasize && operator precedence with extra brackets, as suggested by clang warning
    - Single line patch
    - No reviewers assigned yet, but feel free to review
    - **No update this week (Formatting is needed)**
- flang#1392: Adding intent(in)
    - Planting intent(in) all over the code, per strict modern Fortran compilers requirements.
    - Do not duplicate eq and == , ne and /= operators definitions, as they seem to be already aliases of each other, and compilers are not happy if we try to overdefine this fact.
    - Runtime library issue
    - No reviewers assigned yet, but feel free to review
    - Any thought on potential ill impact?
    - **No update this week (need testing details)**
- flang#1400: Do not dump debug location if debug info is not available

            - Problem only occurs for GPU offload compilation? Worry is that the use of flags guarding debug offload compilation is inconsistent
            - Missing test case
            - Shivarama will review - need a testcase to validate
            - No update this week.
        - flang#1401: Link argument parsing functions from libraries, instead of duplicating their source files
            - Windows build is broken, but the refactoring seems okay.
            - Rejected
- QuadFP PRs
    - flang#1411: Support quad-complex ABS intrinsic
        - Need review
- QuadFP Complex Support Plan
    - Semantic analyses for type conversion and operand type promotion
    - I/O support
    - C_LONG_DOUBLE_COMPLEX
    - Trigonometry intrinsic functions
    - MATMUL and TRANSPOSE
    - A couple of optimizations
    - Huawei's partner will upstream more patches for intrinsics, but not high priority at the moment.
- Windows related PRs
    - Windows GitHub workflow is now completely implemented
        - Currently tests cannot run on Windows due to portability issues in the test suite; PRs have been raised to adapt the test suite
    - flang#1408: Fix the dllimport code generation for opaque type
        - Symbols defined in a module should have dllimport attribute
        - Pawel reviewed; Bryan asked for a test case
        - Approved - to be merged
    - flang#1405: Add Windows support to lit.cfg
        - Bryan and Shivarama approved
        - merged
    - flang#1366: Add Windows support for f90_correct tests
        - Needs review
        - Awful lot of tests touched!
    - llvm#161: Add Flang include directory to MSVC toolchain (release_15x)
        - Needs review

- Other PRs
    - flang#1360: Add ALIGN pragma for derived type and fixed-shape array/character type
        - Kiran raised a question about the design of the pragma and advises adopting the syntax of existing compilers (e.g. Intel's)
        - Large patch; needs more reviews
        - AMD will review this patch
        - Bryan has given review comments
        - The author is trying to find the time to refactor this PR based on our feedback and resolve the conflicts
            - Author is occupied until September
            - No update
    - flang#1314: Automatically deploy Sphinx documentation to GitHub Pages after a successful push to master
        - What happened to this PR and how? The status is unclear…
            - Closed accidentally, will be re-issued again
                - Still being closed
            - **No update**

## August 09, 2023

Attendees
AMD: Shivarama
Arm: Pawel
Huawei:
NVIDIA:
Others:

- A set of PRs by Dmitry Mikushin
    - flang#1378: Fix classic flang compilation with in-tree flang
        - This PR contains a lot of patches that are unrelated to each other.
        - This set of patches fixes various issues spotted during compilation of flang-classic with clang and LLVM in-tree flang.
        - For C/C++ it addresses #1374 and #1375 and fixes fatal clang issues, such as mismatching function pointer signature.
        - For Fortran it mostly adds a lot of `intent(in)` attributes.

        - **This PR has been decomposed into the list of PRs below.**
    - flang#1379: Add docker container for building flang
        - Do we really need this? It won't be tested unless we fully integrate it into our CI workflows. **Let's discuss with Dmitry in person next meeting.**
        - Related: Should we consolidate our build scripts?
            - **Bryan will push a PR to remove the .sh scripts and use .py scripts in CI, and update the wiki page**
            - **No  update**
    - flang#1383: Use preprocessor in traditional mode
        - Fortran compiler is supposed to use traditional C preprocessor only. Therefore, we can't use modern C preprocessor features, such as catenation or stringizing
        - Modifies just the runtime/flang/mmul_dir.h file
        - Stringizing and concatenation are not "traditional", but they seem to be supported by flang1 (fpp.c, accpp.c) natively? Should we remove functionality from flang1?
            - Pawel: Should we aim for full gfortran compatibility? Reaching out to NVIDIA. Previous conclusion was a "No".
            - Pawel: Making runtime code more portable is good; removing functionality is not necessary and may surprise existing users.
        - LLVM Flang supports stringization. We unanimously agree we shouldn't proceed with this PR.
    - flang#1386: Emphasize && operator precedence with extra brackets, as suggested by clang warning
        - Single line patch
        - No reviewers assigned yet, but feel free to review
        - **No update this week**
    - flang#1392: Adding intent(in)
        - Planting intent(in) all over the code, per strict modern Fortran compilers requirements.
        - Do not duplicate eq and == , ne and /= operators definitions, as they seem to be already aliases of each other, and compilers are not happy if we try to overdefine this fact.
        - Runtime library issue

            - No reviewers assigned yet, but feel free to review
            - Any thought on potential ill impact?
            - **No update this week**
        - flang#1400: Do not dump debug location if debug info is not available
            - Problem only occurs for GPU offload compilation? Worry is that the use of flags guarding debug offload compilation is inconsistent
            - Missing test case
            - Shivarama will review - need a testcase to validate
        - flang#1401: Link argument parsing functions from libraries, instead of duplicating their source files
            - Windows build is broken, but the refactoring seems okay.
            - Needs reviews; Pawel can comment on it
- QuadFP Complex Support Plan
    - Semantic analyses for type conversion and operand type promotion
    - I/O support
    - C_LONG_DOUBLE_COMPLEX
    - Trigonometry intrinsic functions
    - MATMUL and TRANSPOSE
    - A couple of optimizations
    - Huawei's partner will upstream more patches for intrinsics, but not high priority at the moment.
- Windows related PRs
    - Windows GitHub workflow is now completely implemented
        - Currently tests cannot run on Windows due to portability issues in the test suite; PRs have been raised to adapt the test suite
    - flang#1408: Fix the dllimport code generation for opaque type
        - Symbols defined in a module should have dllimport attribute
        - Pawel reviewed; Bryan asked for a test case
        - Approved - to be merged
    - flang#1405: Add Windows support to lit.cfg
        - Bryan and Shivarama approved
        - Still need to amend the patch with a comment requested by Pawel
    - flang#1396: Fix llvm_ir_correct tests on Windows
        - On Windows some tests are unable to open iso_fortran_env.mod unless we include it explicitly.
        - Partially fixes flang#1363

        - Bryan responded that the patch was incorrectly abusing %flags.
        - Closed by author.
    - flang#1395: Fix flang2 tests on Windows
        - On Windows '%flang2' is interpreted incorrectly as 'flang.exe2'. This change assumes that flang2 executable is in the user's path.
        - Partially fixes flang#1363
        - Closed by author; superseded by flang#1402
    - flang#1394: fix sema tests on Windows
        - On Windows `%flang1` is interpreted incorrectly as `flang.exe1`. This change assumes that flang1 executable is in the user's path.
        - Partially fixes flang#1363
        - Closed by author; superseded by flang#1402
    - flang#1366: Add Windows support for f90_correct tests
        - Needs review
        - Awful lot of tests touched!
        - No update
- Other PRs
    - flang#1414: Fix failures for imports with LLVM-17
        - This PR contains patch that fixes the compatibility issues with LLVM-17
        - Approved and merged.

    - flang#1360: Add ALIGN pragma for derived type and fixed-shape array/character type
        - Kiran raised a question about the design of the pragma and advises adopting the syntax of existing compilers (e.g. Intel's)
        - Large patch; needs more reviews
        - AMD will review this patch
        - Bryan has given review comments
        - The author is trying to find the time to refactor this PR based on our feedback and resolve the conflicts
            - Author is occupied until September
    - flang#1314: Automatically deploy Sphinx documentation to GitHub Pages after a successful push to master
        - What happened to this PR and how? The status is unclear…
            - Closed accidentally, will be re-issued again
                - Still being closed
            - **No update**

## Jul 26, 2023

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan
NVIDIA:
Others:

- Breaking news 🙂: flang#1376 flang2 code is terrible
  - A set of PRs by Dmitry Mikushin
    - flang#1378: Fix classic flang compilation with in-tree flang
      - This PR contains a lot of patches that are unrelated to each other.
      - This set of patches fixes various issues spotted during compilation of flang-classic with clang and LLVM in-tree flang.
      - For C/C++ it addresses #1374 and #1375 and fixes fatal clang issues, such as mismatching function pointer signature.
      - For Fortran it mostly adds a lot of `intent(in)` attributes.
      - **This PR has been decomposed into the list of PRs below.**
    - flang#1379: Add docker container for building flang
      - Do we really need this? It won't be tested unless we fully integrate it into our CI workflows. **Let's discuss with Dmitry in person next meeting.**
      - Related: Should we consolidate our build scripts?
        - **Bryan will push a PR to remove the .sh scripts and use .py scripts in CI, and update the wiki page**
        - **No  update**
    - flang#1383: Use preprocessor in traditional mode
      - Fortran compiler is supposed to use traditional C preprocessor only. Therefore, we can't use modern C preprocessor features, such as catenation or stringizing
      - Modifies just the runtime/flang/mmul_dir.h file
      - Stringizing and concatenation are not "traditional", but they seem to be supported by flang1 (fpp.c,

        accpp.c) natively? Should we remove functionality from flang1?
            - Pawel: Should we aim for full gfortran compatibility? Reaching out to NVIDIA. Previous conclusion was a "No".
            - Pawel: Making runtime code more portable is good; removing functionality is not necessary and may surprise existing users.
        - LLVM Flang supports stringization. We unanimously agree we shouldn't proceed with this PR.
    - flang#1386: Emphasize && operator precedence with extra brackets, as suggested by clang warning
        - Single line patch
        - No reviewers assigned yet, but feel free to review
    - flang#1390: Adjust the functions signatures to match the expected function pointer signatures
        - `int *` vs. `void *`, and missing params
        - See also flang#1385 (Adjust the functions signatures to match the expected function pointer signatures)
        - **merged**
    - flang#1391: The array dimensions must be defined *before* being used
        - I guess adding IMPLICIT NONE would expose the problem earlier
        - Runtime library issue
        - **merged**
    - flang#1392: Adding intent(in)
        - Planting intent(in) all over the code, per strict modern Fortran compilers requirements.
        - Do not duplicate eq and == , ne and /= operators definitions, as they seem to be already aliases of each other, and compilers are not happy if we try to overdefine this fact.
        - Runtime library issue
        - No reviewers assigned yet, but feel free to review
        - Any thought on potential ill impact?
    - flang#1400: Do not dump debug location if debug info is not available
        - Problem only occurs for GPU offload compilation? Worry is that the use of flags guarding debug offload compilation is inconsistent

            - Missing test case
            - Shivarama will review
        - flang#1401: Link argument parsing functions from libraries, instead of duplicating their source files
            - Windows build is broken, but the refactoring seems okay.
            - Needs reviews; Pawel can comment on it
- QuadFP Complex Support Plan
    - Semantic analyses for type conversion and operand type promotion
    - I/O support
    - C_LONG_DOUBLE_COMPLEX
    - Trigonometry intrinsic functions
    - MATMUL and TRANSPOSE
    - A couple of optimizations
    - Huawei's partner will upstream more patches for intrinsics, but not high priority at the moment.
- Windows related PRs
    - Windows GitHub workflow is now completely implemented
        - Currently tests cannot run on Windows due to portability issues in the test suite; PRs have been raised to adapt the test suite
    - flang#1408: Fix the dllimport code generation for opaque type
        - Symbols defined in a module should have dllimport attribute
        - Pawel reviewed; Bryan asked for a test case
    - flang#1405: Add Windows support to lit.cfg
        - Bryan and Shivarama approved
        - Still need to amend the patch with a comment requested by Pawel
    - flang#1402: Fix executable names in lit.cfg
        - Need one more review
        - **merged**
    - flang#1396: Fix llvm_ir_correct tests on Windows
        - On Windows some tests are unable to open iso_fortran_env.mod unless we include it explicitly.
        - Partially fixes flang#1363
        - Bryan responded that the patch was incorrectly abusing %flags.
        - Closed by author.
    - flang#1395: Fix flang2 tests on Windows

        - On Windows '%flang2' is interpreted incorrectly as 'flang.exe2'. This change assumes that flang2 executable is in the user's path.
        - Partially fixes flang#1363
        - Closed by author; superseded by flang#1402
    - flang#1394: fix sema tests on Windows
        - On Windows `%flang1` is interpreted incorrectly as `flang.exe1`. This change assumes that flang1 executable is in the user's path.
        - Partially fixes flang#1363
        - Closed by author; superseded by flang#1402
    - flang#1370: Add Windows support to openmp_examples tests
        - **merged**
    - flang#1366: Add Windows support for f90_correct tests
        - Needs review
        - Awful lot of tests touched!
        - No update
- Other PRs
    - flang#1409: [runtime] Fix infinite loop in `__fort_bcopysl`
        - **merged**
    - flang#1404: Fix for handling absolute include paths
        - Bryan has a comment; need more reviews
        - **merged**
    - flang#1360: Add ALIGN pragma for derived type and fixed-shape array/character type
        - Kiran raised a question about the design of the pragma and advises adopting the syntax of existing compilers (e.g. Intel's)
        - Large patch; needs more reviews
        - AMD will review this patch
        - Bryan has given review comments
        - The author is trying to find the time to refactor this PR based on our feedback and resolve the conflicts
            - Author is occupied until September
    - flang#1314: Automatically deploy Sphinx documentation to GitHub Pages after a successful push to master
        - What happened to this PR and how? The status is unclear…
            - Closed accidentally, will be re-issued again
                - Still being closed
            - **No update**

## Jul 12, 2023

Attendees
AMD: Shivarama, Kiran TP
Arm: Pawel
Huawei: Bryan
NVIDIA:
Others:

- LLVM 16 support **merged** into classic-flang-llvm-project/release_16x (llvm#159)
  - Required Classic Flang changes:
    - flang#1403: [LLVM16][build] Specify LLVM_MAIN_SRC_DIR when running build-flang.sh; **MERGED**
    - flang#1381: [LLVM16][flang1] Suppress -Wsingle-bit-bitfield-constant-conversion warnings; **MERGED**
    - flang#1380: [LLVM16][llvm_ir_correct] Expect memory(none) as replacements for readnone; **MERGED**
  - Should we change the default branch to release_16x now? **Yes, done.**
- Breaking news 🙂: flang#1376 flang2 code is terrible
  - A set of PRs by Dmitry Mikushin
    - Is Dmitry receiving invitations for our meetings? **Will send invitation.**
    - flang#1378: Fix classic flang compilation with in-tree flang
      - This PR contains a lot of patches that are unrelated to each other.
      - This set of patches fixes various issues spotted during compilation of flang-classic with clang and LLVM in-tree flang.
      - For C/C++ it addresses #1374 and #1375 and fixes fatal clang issues, such as mismatching function pointer signature.
      - For Fortran it mostly adds a lot of `intent(in)` attributes.
      - **This PR has been decomposed into the list of PRs below.**
    - flang#1379: Add docker container for building flang
      - 18 commits!
      - No reviewers assigned

        - Do we really need this? It won't be tested unless we fully integrate it into our CI workflows. **Let's discuss with Dmitry in person next meeting.**
        - Related: Should we consolidate our build scripts?
            - **Bryan will push a PR to remove the .sh scripts and use .py scripts in CI, and update the wiki page**
    - flang#1383: Use preprocessor in traditional mode
        - Fortran compiler is supposed to use traditional C preprocessor only. Therefore, we can't use modern C preprocessor features, such as catenation or stringizing
        - Modifies just the runtime/flang/mmul_dir.h file
        - Stringizing and concatenation are not "traditional", but they seem to be supported by flang1 (fpp.c, accpp.c) natively? Should we remove functionality from flang1?
            - Pawel: Should we aim for full gfortran compatibility? Reaching out to NVIDIA. Previous conclusion was a "No".
            - Pawel: Making runtime code more portable is good; removing functionality is not necessary and may surprise existing users.
    - flang#1384: Fix sprintf overflow
        - Give sprintf one extra byte to write the null character in the end of string and resolve the overflow warning
        - That’s why snprintf() should be always used instead of sprintf()
        - Single line patch
        - **merged**
    - flang#1385: Adjust the functions signatures to match the expected function pointer signatures
        - `comm_sked *` vs. `void *`
        - **merged**
    - flang#1386: Emphasize && operator precedence with extra brackets, as suggested by clang warning
        - Single line patch
        - No reviewers assigned yet, but feel free to review
    - flang#1387: Remove unused array
        - `garg_ilis[nargs]` seems not being used
        - Single line patch
        - **merged**

        - flang#1389: Re-arranging the code such that ili is always initialized and used
            - Would be great if the author could extend the commit message, e.g., when the fixed issue manifests itself?
            - **merged**
        - flang#1390: Adjust the functions signatures to match the expected function pointer signatures
            - `int *` vs. `void *`, and missing params
            - See also flang#1385 (Adjust the functions signatures to match the expected function pointer signatures)
            - No reviewers assigned yet, but feel free to review
                - Bryan will review and amend with possible changes
        - flang#1391: The array dimensions must be defined *before* being used
            - I guess adding IMPLICIT NONE would expose the problem earlier
            - Runtime library issue
            - **merged**
        - flang#1392: Adding intent(in)
            - Planting intent(in) all over the code, per strict modern Fortran compilers requirements.
            - Do not duplicate eq and == , ne and /= operators definitions, as they seem to be already aliases of each other, and compilers are not happy if we try to overdefine this fact.
            - Runtime library issue
            - No reviewers assigned yet, but feel free to review
            - Any thought on potential ill impact?
        - flang#1400: Do not dump debug location if debug info is not available
            - Problem only occurs for GPU offload compilation?
            - Worry is that the use of flags guarding debug offload compilation is inconsistent
            - Shivarama will look
        - flang#1401: Link argument parsing functions from libraries, instead of duplicating their source files
            - Windows build is broken, but the refactoring seems okay.
            - Needs reviews
- QuadFP Complex Support Plan

    - Semantic analyses for type conversion and operand type promotion
    - I/O support
    - C_LONG_DOUBLE_COMPLEX
    - Trigonometry intrinsic functions
    - MATMUL and TRANSPOSE
    - A couple of optimizations
    - Huawei's partner will upstream more patches for intrinsics, but not high priority at the moment.
- Windows related PRs
    - Windows GitHub workflow is now completely implemented
        - Currently tests cannot run on Windows due to portability issues in the test suite; PRs have been raised to adapt the test suite
    - flang#1402: Fix executable names in lit.cfg
        - Need one more review
    - flang#1397: fix mp_correct tests on Windows
        - This change enables mp_correct test suite to run on Windows. (Test suite fails similarly to issue flang#1371)
        - Partially fixes flang#1363
        - **MERGED**
    - flang#1396: Fix llvm_ir_correct tests on Windows
        - On Windows some tests are unable to open iso_fortran_env.mod unless we include it explicitly.
        - Partially fixes flang#1363
        - No reviewers assigned yet, but feel free to review
        - CI failures
    - flang#1395: Fix flang2 tests on Windows
        - On Windows '%flang2' is interpreted incorrectly as 'flang.exe2'. This change assumes that flang2 executable is in the user's path.
        - Partially fixes flang#1363
        - No reviewers assigned yet, but feel free to review
    - flang#1394: fix sema tests on Windows
        - On Windows `%flang1` is interpreted incorrectly as `flang.exe1`. This change assumes that flang1 executable is in the user's path.
        - Partially fixes flang#1363
        - No reviewers assigned yet, but feel free to review
    - flang#1370: Add Windows support to openmp_examples tests
        - Needs review
    - flang#1366: Add Windows support for f90_correct tests
        - Needs review

        - Awful lot of tests touched!
    - llvm#160: Pass Flang libraries to MSVC linker correctly (also llvm#157 for release_15x)
        - Mostly changes in clang/lib/Driver/ToolChains/MSVC.cpp
        - **MERGED**
- Other PRs
    - flang#1404: Fix for handling absolute include paths
        - Bryan has a comment; need more reviews
    - flang#1393: fix missing local lit config file in ili_correct test folder
        - Single line patch
        - No reviewers assigned yet, but feel free to review
        - **merged**
    - flang#1360: Add ALIGN pragma for derived type and fixed-shape array/character type
        - Kiran raised a question about the design of the pragma and advises adopting the syntax of existing compilers (e.g. Intel's)
        - Large patch; needs more reviews
        - AMD will review this patch
        - Bryan has given review comments
        - The author is trying to find the time to refactor this PR based on our feedback and resolve the conflicts
    - flang#1314: Automatically deploy Sphinx documentation to GitHub Pages after a successful push to master
        - What happened to this PR and how? The status is unclear…
            - Closed accidentally, will be re-issued again
                - Still being closed
            - **Still no update**

## Jun 28, 2023

Attendees
AMD: Shivarama
Arm: Pawel,Kiran
Huawei: Bryan
NVIDIA:
Others:

- Breaking news 🙂
    - flang#1376 flang2 code is terrible

            - More a statement than an actual issue
- QuadFP Complex Support Plan
    - Semantic analyses for type conversion and operand type promotion
    - I/O support
    - C_LONG_DOUBLE_COMPLEX
    - Trigonometry intrinsic functions
    - MATMUL and TRANSPOSE
    - A couple of optimizations
    - PRS submitted for review (need reviews)
        - flang#1359: [QuadFP] Support quad-complex SIN intrinsic
            - PRs for other QuadFP intrinsics will be submitted in the future
            - **merged**
        - Bryan will submit one more PR after addressing above PRs
- LLVM16 related PRs
    - The release_16x branch has been created, the release_15x branch is still the default
    - flang#1380: [LLVM16][llvm_ir_correct] Expect memory(none) as replacements for readnone
        - More reviewers needed, but this is so trivial, could it go with just one approval?
    - flang#1381: [LLVM16][flang1] Suppress -Wsingle-bit-bitfield-constant-conversion warnings
        - Fixes flang#1374 (implicit truncation from 'int' to a one-bit wide bit-field changes value from 1 to -1)
        - More reviewers needed
    - llvm#159: Port Classic Flang changes to LLVM 16
        - More reviewers needed
- Windows related PRs
    - Windows GitHub workflow is now completely implemented
        - Currently tests cannot run on Windows due to portability issues in the test suite
            - Are the commits below fixing any of this?
    - flang#1397: fix mp_correct tests on Windows
        - This change enables mp_correct test suite to run on Windows. (Test suite fails similarly to issue flang#1371)
        - Partially fixes flang#1363
        - No reviewers assigned yet, but feel free to review
    - flang#1396: Fix llvm_ir_correct tests on Windows

        - On Windows some tests are unable to open iso_fortran_env.mod unless we include it explicitly.
        - Partially fixes flang#1363
        - No reviewers assigned yet, but feel free to review
    - flang#1395: Fix flang2 tests on Windows
        - On Windows '%flang2' is interpreted incorrectly as 'flang.exe2'. This change assumes that flang2 executable is in the user's path.
        - Partially fixes flang#1363
        - No reviewers assigned yet, but feel free to review
    - flang#1394: fix sema tests on Windows
        - On Windows `%flang1` is interpreted incorrectly as `flang.exe1`. This change assumes that flang1 executable is in the user's path.
        - Partially fixes flang#1363
        - No reviewers assigned yet, but feel free to review
    - flang#1366: Add Windows support for f90_correct tests
        - Needs review
        - Awful lot of tests touched!
- NCAR kernels
    - Are these tests still NOT being run?
        - Yes it shoult be running since flang#1369 was merged
    - Bryan will try to run them offline and make it part of CI if it doesn't take too much time. On X86 two of the tests are failing and need to be looked into.
        - Submitted PR flang#1369 to fix the test failures and enable CI to run NCAR kernels
            - **merged**
- A set of PRs by Dmitry Mikushin
        - Is the author of those patches receiving invitations for our meetings?
    - flang#1378: Fix clasic flang compilation with in-tree flang
        - This PR contains a lot of patches that are unrelated to each other.
        - This set of patches fixes various issues spotted during compilation of flang-classic with clang and LLVM in-tree flang.
        - For C/C++ it addresses #1374 and #1375 and fixes fatal clang issues, such as mismatching function pointer signature.
        - For Fortran it mostly adds a lot of `intent(in)` attributes.
        - No reviewers assigned but Bryan has engaged in some longer discussion

- flang#1379: Add docker container for building flang
    - 18 commits!
    - No reviewers assigned
- flang#1383: Use preprocessor in traditional mode
    - Fortran compiler is supposed to use traditional C preprocessor only. Therefore, we can't use modern C preprocessor features, such as catenation or stringizing
    - Modifies just the runtime/flang/mmul_dir.h file
    - No reviewers assigned yet, but feel free to review
- flang#1384: Fix sprintf overflow
    - Give sprintf one extra byte to write the null character in the end of string and resolve the overflow warning
    - That's why snprintf() should be always used instead of sprintf()
    - Single line patch
    - **merged**
- flang#1385: Adjust the functions signatures to match the expected function pointer signatures
    - `comm_sked *` vs. `void *`
    - No reviewers assigned yet, but feel free to review
- flang#1386: Emphasize && operator precedence with extra brackets, as suggested by clang warning
    - Single line patch
    - No reviewers assigned yet, but feel free to review
- flang#1387: Remove unused array
    - `garg_ilis[nargs]` seems not being used
    - Single line patch
    - **merged**
- flang#1389:
    - Would be great if the author could extend the commit message, e.g., when the fixed issue manifests itself?
    - No reviewers assigned yet, but feel free to review
- flang#1390: Adjust the functions signatures to match the expected function pointer signatures
    - `int *` vs. `void *`, and missing params
    - See also flang#1385 (Adjust the functions signatures to match the expected function pointer signatures)
    - No reviewers assigned yet, but feel free to review
- flang#1391: The array dimensions must be defined *before* being used
    - I guess adding IMPLICIT NONE would expose the problem earlier
    - Runtime library issue

        - No reviewers assigned yet, but feel free to review
    - [flang#1392](): Adding intent(in)
        - Planting intent(in) all over the code, per strict modern Fortran compilers requirements.
        - Do not duplicate eq and == , ne and /= operators definitions, as they seem to be already aliases of each other, and compilers are not happy if we try to overdefine this fact.
        - Runtime library issue
        - No reviewers assigned yet, but feel free to review
- Other PRs
    - [flang#1393](): fix missing local lit config file in ili_correct test folder
        - Single line patch
        - No reviewers assigned yet, but feel free to review
    - [flang#1360](): Add ALIGN pragma for derived type and fixed-shape array/character type
        - Kiran raised a question about the design of the pragma and advises adopting the syntax of existing compilers (e.g. Intel's)
        - Large patch; needs more reviews
        - AMD will review this patch
        - Bryan has given review comments
        - The author is trying to find the time to refactor this PR based on our feedback and resolve the conflicts
    - [flang#1314](): Automatically deploy Sphinx documentation to GitHub Pages after a successful push to master
        - What happened to this PR and how? The status is unclear…
            - Closed accidentally, will be re-issued again
                - Still being closed
            - **No update**

## Jun 14, 2023

Attendees
AMD: Shivarama
Arm: Pawel,Kiran
Huawei: Bryan
NVIDIA:
Others:

- QuadFP Complex Support Plan

    - Semantic analyses for type conversion and operand type promotion
    - I/O support
    - C_LONG_DOUBLE_COMPLEX
    - Trigonometry intrinsic functions
    - MATMUL and TRANSPOSE
    - A couple of optimizations
    - PRS submitted for review (need reviews)
        - flang#1358: [QuadFP][flang1] Support quad complex zero in collapse_assignment
            - merged
        - flang#1359: [QuadFP] Support quad-complex SIN intrinsic
            - PRs for other QuadFP intrinsics will be submitted in the future
            - Needs one more reviewer
        - Bryan will submit one more PR after addressing above PRs
- Current issues
    - flang#1238: -Mbounds reports false positive for array slices passed to DOT_PRODUCT intrinsic
        - No update
- Other PRs
    - flang#1366: Add Windows support for f90_correct tests
        - Needs review
        - Awful lot of tests touched!
    - flang#1360: Add ALIGN pragma for derived type and fixed-shape array/character type
        - Kiran raised a question about the design of the pragma and advises adopting the syntax of existing compilers (e.g. Intel's)
        - Large patch; needs more reviews
        - AMD will review this patch
        - Bryan has given review comments
        - The author is trying to find the time to refactor this PR based on our feedback and resolve the conflicts

    - flang#1314: Automatically deploy Sphinx documentation to GitHub Pages after a successful push to master
        - What happened to this PR and how? The status is unclear…
            - Closed accidentally, will be re-issued again

            - Still being closed
          - **No update**
- Windows GitHub workflow is now completely implemented
  - Currently tests cannot run on Windows due to portability issues in the test suite
- NCAR kernels
  - Currently these tests are NOT running.
  - Bryan will try to run them offline and make it part of CI if it doesn't take too much time. On X86 two of the tests are failing and need to be looked into.
    - Submitted PR [flang#1369]() to fix the test failures and enable CI to run NCAR kernels
      - More reviewers needed
  - Can we use these execution tests for LLVM Flang?
    - The tests need to be moved to lit format. (not in plan)
- Any plans for llvm-16 branch creation?
  - LLVM16 has been released now
  - Bryan has created the release_16x branch
  - Bryan to upstream some of the downstream work for enabling LLVM16
  - AMD will provide the patch to support Classic Flang on llvm-project.
    - There will be no update for the next 2 weeks
  - Huawei has upgraded internally and identified some flang test failures
    - Are these fixable?

## May 31, 2023

Attendees
AMD: Shivarama
Arm: Pawel,Kiran
Huawei: Bryan
NVIDIA:
Others:

- QuadFP Complex Support Plan
  - Semantic analyses for type conversion and operand type promotion
  - I/O support
  - C_LONG_DOUBLE_COMPLEX
  - Trigonometry intrinsic functions
  - MATMUL and TRANSPOSE
  - A couple of optimizations

    - PRS submitted for review (need reviews)
        - flang#1358: [QuadFP][flang1] Support quad complex zero in collapse_assignment
            - Bryan has updated the pull request; reviews needed
        - flang#1359: [QuadFP] Support quad-complex SIN intrinsic
            - PRs for other QuadFP intrinsics will be submitted in the future
            - Need reviews
        - Bryan will submit one more PR after addressing above PRs
- Current issues
    - flang#1238: -Mbounds reports false positive for array slices passed to DOT_PRODUCT intrinsic
        - The same array slices cause no problem when passed to a user function that in turn calls DOT_PRODUCT
        - Fixed in nvfortran but the fix is not upstreamed
        - Bryan and Pawel will try to reproduce the problem
            - EDIT by Pawel: I've managed to reproduce the issue, our downstream compiler is also affected by this.
            - Bryan: Huawei's downstream compiler is also affected.
        - No updates
- Other PRs
    - flang#1366: Add Windows support for f90_correct tests
        - Needs review
    - flang#1360: Add ALIGN pragma for derived type and fixed-shape array/character type
        - Kiran raised a question about the design of the pragma and advises adopting the syntax of existing compilers (e.g. Intel's)
        - Large patch; needs more reviews
        - AMD will review this patch
        - Bryan has given review comments

    - flang#1314: Automatically deploy Sphinx documentation to GitHub Pages after a successful push to master
        - What happened to this PR and how? The status is unclear…
            - Closed accidentally, will be re-issued again

            - **No update**
    - flang#1177: Fix dllimport code generation for module symbols
        - Merged
- Windows GitHub workflow is now completely implemented
    - Currently tests cannot run on Windows due to portability issues in the test suite
    - flang#1349 - Add Windows x64/arm64 GitHub CI workflow to Flang project
        - MERGED
- NAG test exclusion list; **any update from Arm/AMD?**
    - There was a proposal to gate upstreamed patches by the complete passing of NAG tests that all three companies agree on. Is this too much to ask for developers/contributors?
    - Having a shared list is still useful. Bryan will check with Kiran@Arm about his thoughts.
    - No update this time. Nag tests can not run on public CIs. Companies need to run them individually.
- NCAR kernels
    - Currently these tests are NOT running.
    - Can we make it part of CI?
    - Bryan will try to run them offline and make it part of CI if it doesn't take too much of time. On X86 two of the tests are failing and need to be looked into.
    - Can we use these execution tests for llvm-flang?
        - The tests need to be moved to lit format. (not in plan)
    - No updates
- Any plans for llvm-16 branch creation?
    - Llvm-16 is released now
    - Bryan has created the release_16x branch; AMD will provide the patch to support classic flang on llvm-project.
    - AMD - no updates this week.
    - Huawei has upgraded internally and identified some flang test failures

## May 17, 2023

Attendees
AMD: Shivarama
Arm: Pawel, Kiran
Huawei: Bryan
NVIDIA:
Others:

- QuadFP Complex Support Plan
    - Semantic analyses for type conversion and operand type promotion
    - I/O support
    - C_LONG_DOUBLE_COMPLEX
    - Trigonometry intrinsic functions
    - MATMUL and TRANSPOSE
    - A couple of optimizations
    - PRS submitted for review (need reviews)
        - flang#1358: [QuadFP][flang1] Support quad complex zero in collapse_assignment
            - Bryan will update the pull request
        - flang#1359: [QuadFP] Support quad-complex SIN intrinsic
            - Other trigonometry intrinsics will be added in the future
        - Bryan will submit one more PR after addressing above PRs
        - No Updates
- Current issues
    - flang#1336: Unexpected result when the function with BIND(C) attribute has ENTRY statements.
        - AMD submitted a pull request for review
        - flang#1362
        - Merged
    - flang#1361: Flang does not allow expression as argument for NORM2
        - AMD submitted the pull request flang#1365
        - AMD to address review comments.
    - flang#1238: -Mbounds reports false positive for array slices passed to DOT_PRODUCT intrinsic
        - The same array slices cause no problem when passed to a user function that in turn calls DOT_PRODUCT
        - Fixed in nvfortran but the fix is not upstreamed
        - Bryan and Pawel will try to reproduce the problem
            - EDIT by Pawel: I’ve managed to reproduce the issue, our downstream compiler is also affected by this.
        - No updates
- Other PRs

    - flang#1360: Add ALIGN pragma for derived type and fixed-shape array/character type
        - Kiran raised a question about the design of the pragma and advises adopting the syntax of existing compilers (e.g. Intel's)
        - Large patch; needs more reviews
        - AMD will review this patch
        - No updates
    - flang#1314: Automatically deploy Sphinx documentation to GitHub Pages after a successful push to master
        - What happened to this PR and how? The status is unclear…
            - Closed accidentally, will be re-issued again
            - **No update**
    - flang#1352: cmake: stop asking llvm-config for src-root as it is not supported since LLVM16.0
        - Merged
    - flang#1288: Test which tracks default allocatable behavior
        - Needs reviews from Bryan/Shivarama
        - Bryan will create a separate issue for -Mallocatable=95 coverage
        - Merged.
- Windows PRs
    - flang#1177: Fix dllimport code generation for module symbols
        - Bryan has approved; needs one more review
        - AMD approved.
        - Need to be merged.
- Windows GitHub workflow is now completely implemented
    - Currently tests cannot run on Windows due to portability issues in the test suite
    - flang#1349 - Add Windows x64/arm64 GitHub CI workflow to Flang project
        - MERGED
- NAG test exclusion list; **any update from Arm/AMD?**
    - There was a proposal to gate upstreamed patches by the complete passing of NAG tests that all three companies agree on. Is this too much to ask for developers/contributors?
    - Having a shared list is still useful. Bryan will check with Kiran@Arm about his thoughts.
    - No update this time. Nag tests can not run on public CIs. Companies need to run them individually.
- NCAR kernels
    - Currently these tests are NOT running.

    - Can we make it part of CI?
    - Bryan will try to run them offline and make it part of CI if it doesn’t take too much of time. On X86 two of the tests are failing and need to be looked into.
    - Can we use these execution tests for llvm-flang?
        - The tests need to be moved to lit format.
- Any plans for llvm-16 branch creation?
    - Llvm-16 is released now
    - Bryan has created the release_16x branch; AMD will provide the patch to support classic flang on llvm-project.

## May 03, 2023

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan
NVIDIA:
Others:

- QuadFP Complex Support Plan
    - Semantic analyses for type conversion and operand type promotion
    - I/O support
    - C_LONG_DOUBLE_COMPLEX
    - Trigonometry intrinsic functions
    - MATMUL and TRANSPOSE
    - A couple of optimizations
    - PRS submitted for review (need reviews)
        - flang#1358: [QuadFP][flang1] Support quad complex zero in collapse_assignment
            - Bryan needs to add a test case
        - flang#1359: [QuadFP] Support quad-complex SIN intrinsic
            - Other trigonometry intrinsics will be added in the future
        - Bryan will submit one more PR after addressing above PRs
        - No Updates
- Current issues

        - flang#1336: Unexpected result when the function with BIND(C) attribute has ENTRY statements.
            - AMD submitted a pull request for review
            - flang#1362
            - Need reviews
        - flang#1361: Flang does not allow expression as argument for NORM2
            - First reported by Pawel in #893
            - Huawei's downstream compiler doesn't seem to have a problem; need to check history to see how it was fixed (by accident?)
            - AMD will check if this issue can be reproduced in their downstream compiler.
        - flang#1238: -Mbounds reports false positive for array slices passed to DOT_PRODUCT intrinsic
            - The same array slices cause no problem when passed to a user function that in turn calls DOT_PRODUCT
            - Fixed in nvfortran but the fix is not upstreamed
            - Bryan and Pawel will try to reproduce the problem
                - EDIT by Pawel: I've managed to reproduce the issue, our downstream compiler is also affected by this.
            - No updates
- Other PRs
    - flang#1360: Add ALIGN pragma for derived type and fixed-shape array/character type
        - Kiran raised a question about the design of the pragma and advises adopting the syntax of existing compilers (e.g. Intel's)
        - Large patch; needs more reviews
        - AMD will review this patch
    - flang#1314: Automatically deploy Sphinx documentation to GitHub Pages after a successful push to master
        - What happened to this PR and how? The status is unclear…
            - Closed accidentally, will be re-issued again
            - **No update**
    - flang#1352: cmake: stop asking llvm-config for src-root as it is not supported since LLVM16.0
        - Shivarama and Bryan will review and merge this as-is for LLVM 16 compatibility.

        - We have agreement on removing support for in-tree builds; when we have time we can delete that code in a separate PR.
        - Above PR can be merged after approval from AMD.
    - flang#1288: Test which tracks default allocatable behaviour
        - Needs reviews from Bryan/Shivarama
        - Bryan will create a separate issue for -Mallocatable=95 coverage
- Windows PRs
    - flang#1177: Fix dllimport code generation for module symbols
        - Bryan has approved; needs one more review
        - AMD will review
    - flang#1341: [scutil] Add support Windows Complex types for pow folding
        - MERGED
    - llvm#154, llvm#155: script: update llvm python build script
        - Bryan has approved
        - MERGED
- Windows GitHub workflow is now completely implemented
    - Currently tests cannot run on Windows due to portability issues in the test suite
    - flang#1349 - Add Windows x64/arm64 GitHub CI workflow to Flang project
        - MERGED
- NAG test exclusion list; **any update from Arm/AMD?**
    - There was a proposal to gate upstreamed patches by the complete passing of NAG tests that all three companies agree on. Is this too much to ask for developers/contributors?
    - Having a shared list is still useful. Bryan will check with Kiran@Arm about his lists.
- NCAR kernels
    - Currently these tests are NOT running.
    - Can we make it part of CI?
    - Bryan will try to run them offline and make it part of CI if it doesn't take too much of time.
- Any plans for llvm-16 branch creation?
    - Llvm-16 is released now
    - Bryan will create the base branch

## April 19, 2023 (note: April 5 meeting has been canceled)

Attendees

AMD: Shivarama
Arm: Pawel
Huawei: Bryan
NVIDIA:
Others:

- QuadFP Complex Support Plan
    - Semantic analyses for type conversion and operand type promotion
    - I/O support
    - C_LONG_DOUBLE_COMPLEX
    - Trigonometry intrinsic functions
    - MATMUL and TRANSPOSE
    - A couple of optimizations
    - PRS submitted for review (need reviews)
        - flang#1354: [QuadFP][runtime] Add f90io_sc_cq_ldw for writing quad complex numbers
            - MERGED
        - flang#1356: [QuadFP][test] Make comments clear and consistent in style; NFC
            - MERGED
        - flang#1357: [QuadFP][runtime] Define C_LONG_DOUBLE_COMPLEX as correct type kind
            - MERGED
        - flang#1358: [QuadFP][flang1] Support quad complex zero in collapse_assignment
            - Bryan needs to add a test case
        - flang#1359: [QuadFP] Support quad-complex SIN intrinsic
            - Other trigonometry intrinsics will be added in the future
        - Bryan will submit one more PR after addressing above PRs
- Current issues
    - flang#1336: Unexpected result when the function with BIND(C) attribute has ENTRY statements.
        - When the function return type and the entry return type have different ABIs (return-by-reference vs. return-by-value), the frontend does not generate code to return the correct value from the entry.
        - Different error on x86_64 than on AArch64

            - Some concerns regarding ability of compiling the reproducer code
                - Responded by the author of the reproducer code
            - Writing more test cases to understand what is the expected behaviour; even gfortran doesn't behave consistently
        - flang#1353: Add build-flang.sh and build-llvm-project.sh to the repos of flang and classic-flang-llvm-project
            - CLOSED, already done.
        - flang#1361: Flang does not allow expression as argument for NORM2
            - First reported by Pawel in #893
            - Huawei's downstream compiler doesn't seem to have a problem; need to check history to see how it was fixed (by accident?)
        - flang#1238: -Mbounds reports false positive for array slices passed to DOT_PRODUCT intrinsic
            - The same array slices cause no problem when passed to a user function that in turn calls DOT_PRODUCT
            - Fixed in nvfortran but the fix is not upstreamed
            - Bryan and Pawel will try to reproduce the problem
                - EDIT by Pawel: I've managed to reproduce the issue, our downstream compiler is also affected by this.
- Other PRs
    - flang#1360: Add ALIGN pragma for derived type and fixed-shape array/character type
        - Kiran raised a question about the design of the pragma and advises adopting the syntax of existing compilers (e.g. Intel's)
        - Large patch; needs more reviews
    - flang#1314: Automatically deploy Sphinx documentation to GitHub Pages after a successful push to master
        - What happened to this PR and how? The status is unclear…
            - Closed accidentally, will be re-issued again
            - **No update**
    - flang#1352: cmake: stop asking llvm-config for src-root as it is not supported since LLVM16.0
        - Shivarama and Bryan will review and merge this as-is for LLVM 16 compatibility.

        - We have agreement on removing support for in-tree builds; when we have time we can delete that code in a separate PR.
    - flang#1288: Test which tracks default allocatable behaviour
        - Needs reviews from Bryan/Shivarama
        - Bryan will create a separate issue for -Mallocatable=95 coverage
- Windows PRs
    - flang#1177: Fix dllimport code generation for module symbols
        - Bryan has approved; needs one more review
    - flang#1341: [scutil] Add support Windows Complex types for pow folding
        - MERGED
    - llvm#154, llvm#155: script: update llvm python build script
        - Bryan has approved
- Windows GitHub workflow is now completely implemented
    - Currently tests cannot run on Windows due to portability issues in the test suite
    - flang#1349 - Add Windows x64/arm64 GitHub CI workflow to Flang project
        - MERGED
- NAG test exclusion list; **any update from Arm/AMD?**
    - There was a proposal to gate upstreamed patches by the complete passing of NAG tests that all three companies agree on. Is this too much to ask for developers/contributors?
    - Having a shared list is still useful. Bryan will check with Kiran@Arm about his lists.

## March 22, 2023

Attendees
AMD: Shivarama
Arm: Pawel
Huawei: Bryan
NVIDIA:
Others:

- QuadFP Complex Support Plan
    - Semantic analyses for type conversion and operand type promotion
    - I/O support
    - C_LONG_DOUBLE_COMPLEX
    - Trigonometry intrinsic functions

    - MATMUL and TRANSPOSE
    - A couple of optimizations
    - PRS submitted for review (need reviews)
        - flang#1354: [QuadFP][runtime] Add f90io_sc_cq_ldw for writing quad complex numbers
            - Waits for one more approval
        - flang#1356: [QuadFP][test] Make comments clear and consistent in style; NFC
        - flang#1357: [QuadFP][runtime] Define C_LONG_DOUBLE_COMPLEX as correct type kind
        - flang#1358: [QuadFP][flang1] Support quad complex zero in collapse_assignment
        - flang#1359: [QuadFP] Support complex trigonometry intrinsic functions
        - Bryan will submit one more PR after addressing above PRs
- Current issues
    - flang#1336: Unexpected result when the function with BIND(C) attribute has ENTRY statements.
        - When the function return type and the entry return type have different ABIs (return-by-reference vs. return-by-value), the frontend does not generate code to return the correct value from the entry.
        - Different error on x86_64 than on AArch64
        - Some concerns regarding ability of compiling the reproducer code
            - Responded by the author of the reproducer code
    - flang#1353: Add build-flang.sh and build-llvm-project.sh to the repos of flang and classic-flang-llvm-project
        - Is this worth considering?
- Other PRs
    - flang#1314: Automatically deploy Sphinx documentation to GitHub Pages after a successful push to master
        - What happened to this PR and how? The status is unclear…
            - Closed accidentally, will be re-issued again
    - flang#1355: [flang1] Fix a regression about array constructor
        - It has two approvals already, maybe it's time to merge it?
    - llvm#153: [Driver][ClassicFlang] Add options -fno-automatic and -f(no-)implicit…
        - Slightly forgotten, waits for one more approval
- Windows PRs
    - flang#1177: Fix dllimport code generation for module symbols

        - Needs reviews.
    - flang#1341: [scutil] Add support Windows Complex types for pow folding
        - Two approvals gained, should be enough for the merge.
        - There are some Review comments from Neumann-A
            - It is about a lack of knowledge regarding MinGW. Yet since we mostly care about Visual Studio, we can silently continue unless we have any MinGW expert around who could have their say on this.
    - For Windows patches, can we merge with one approval?
        - YES. they can be merged with one approval.
- Windows GitHub workflow
    - Adam is implementing CI jobs for x86_64 and arm64 Windows on GitHub
        - What is the progress here?
    - llvm#149: Add GitHub action to build and upload Windows binaries. Now it runs some tests.
        - Ongoing effort, seems we're closer to a success
    - flang#1349 - Add windows x64 runner support
        - More reviewer's comments to address
        - Bryan's comments need to be addressed
        - One of the checks is failing, is it because the LLVM part has not been merged yet? Yes.

## March 8, 2023

Attendees
AMD: Shivarama, Kiran TP
Arm: Pawel, Kiran
Huawei: Bryan
NVIDIA:
Others: Adam

- Supporting Classic Flang and LLVM Flang at the same time
    - Any update on this?
- QuadFP Complex Support Plan
    - Semantic analyses for type conversion and operand type promotion
    - I/O support
    - C_LONG_DOUBLE_COMPLEX
    - Trigonometry intrinsic functions
    - MATMUL and TRANSPOSE
    - A couple of optimizations

    - PRS submitted for review (need reviews)
        - flang#1344: runtime: use __float128 only in linux
            - Merged
        - flang#1350: [quadfp] semantic analysis for type conversion
            - Approved. Ready for merge.
                - Bryan has amended the PR with comments about the constants in the test cases.
        - flang#1351: [quadfp] semantic check for operand type promotion
            - Approved. Ready for merge.
        - flang#1354: [QuadFP][runtime] Add f90io_sc_cq_ldw for writing quad complex numbers
            - 122 files to review!
                - Most of them are the test cases, fortunately
        - Bryan will submit next set of PRs after addressing above PRs
- Current issues
    - flang#1336: Unexpected result when the function with BIND(C) attribute has ENTRY statements.
        - When the function return type and the entry return type have different ABIs (return-by-reference vs. return-by-value), the frontend does not generate code to return the correct value from the entry.
        - Currently unowned. AMD will look into this.

  Update:  Not yet started. Update will be provided on the next meeting.
- Other PRs
    - flang#1314: Automatically deploy sphinx documentation to github pages
        - AMD and Arm approved.
        - Bryan will take a look at it.
- Windows PRs
    - flang#1177: Fix dllimport code generation for module symbols
        - Needs reviews.
    - flang#1335: Fixed missing prototype error reported by Clang 15
        - Seems to have attracted enough of the reviews
        - There are some Review comments from Neumann-A (addressed already?)
        - Merged.
    - flang#1341: [scutil] Add support Windows Complex types for pow folding
        - Needs more reviews (or, can it be merged with one approval? See below)
        - There are some Review comments from Neumann-A

    - For Windows patches, can we merge with one approval?
        - YES. they can be merged with one approval.
- Windows GitHub workflow
    - Adam is implementing CI jobs for x86_64 and arm64 Windows on GitHub
        - What is the progress here?
    - llvm#149: Add GitHub action to build and upload Windows binaries. Now it runs some tests.
        - Bryan has a question but he may also approve it anyway
    - flang#1349 - Add windows x64 runner support
        - More reviewer's comments to address
        - Bryan's comments need to be addressed
        - One of the checks is failing, is it because the LLVM part has not been merged yet? Yes.
- LLVM16 migration
    - In LLVM 16, llvm-config has dropped the –src-root option; Classic Flang CMakeLists depends on this. ARM is solving this problem.
        - Pawel has pushed the patch flang#1352
            - Alternatively, we could revert Michal Gorny's patch (LLVM repo) which removes the –src-root option and stop worrying about it. Which approach is better?

## February 22, 2023

Attendees
AMD: Shivarama, Kiran TP
Arm: Pawel, Kiran
Huawei: Bryan
NVIDIA:
Others: Adam

- Supporting Classic Flang and LLVM Flang at the same time
    - Sourabh is working on this. He can build Classic Flang and it passes Classic Flang tests, but LLVM Flang tests fail because Driver is constructing jobs incorrectly (getting confused about Classic Flang and LLVM Flang jobs).
    - Main issues: Conflicts in command-line options and file type association. May need to modify TableGen. See discussion in Slack channel.
    - This is currently on hold. **Shivarama will update the plan at the next meeting.**

    - Dan from AMD has plans to work on this as well.
    - **Update:** Sourabh is not getting enough time to work on this. This is not the priority item  for both CPU and GPU teams in AMD and we are not getting resources to dedicatedly work on this.
    - **NO further updates on this**
- QuadFP Complex Support Plan
    - Semantic analyses for type conversion and operand type promotion
    - I/O support
    - C_LONG_DOUBLE_COMPLEX
    - Trigonometry intrinsic functions
    - MATMUL and TRANSPOSE
    - A couple of optimizations
    - PRS submitted for review (need reviews)
        - flang#1344: runtime: use __float128 only in linux
            - Approved by AMD. need Approval from ARM
        - flang#1350: [quadfp] semantic analysis for type conversion
            - Approved
        - flang#1351: [quadfp] semantic check for operand type promotion
            - Approved
        - Pawel had question about origin of constants in the testcases. Is it derived from any test-suites?
            - Bryan will amend the PR with comments about the constants in the test cases.
        - Bryan will submit next set of PRs after addressing above PRs

- Issues
    - flang#1336: Unexpected result when the function with BIND(C) attribute has ENTRY statements.
        - When the function return type and the entry return type have different ABIs (return-by-reference vs. return-by-value), the frontend does not generate code to return the correct value from the entry.
        - Currently unowned. AMD will look into this.

  Update:  Not yet started.
    - flang#1342: Fail to compile on Raspberry Pi 4B
        - Did not include the compiler error message.
        - Update: user shared the error log. Need to look into the error log and check the error and if needed more information will be asked.
        - Issue closed.

        - flang#1338: Build error, resulting from flang2 being placed in an unexpected location when CMake is invoked a certain way. Bryan will push a fix.
            - flang#1345 - approved and merged.
            - Issue closed.
- Other PRs
    - flang#1314: Automatically deploy sphinx documentation to github pages
        - AMD and Arm approved.
        - No progress.
    - flang#912: Fix Ftn_strcmp_klen return value type (from int64_t to int)
        - Pawel approved; can be merged.
    - Windows PRs
        - Python build scripts have been merged.
        - llvm#147, llvm#148: Change Python build script to not install by default.
            - **MERGED**
        - flang#1158: Fix implicit declaration of function warnings reported by clang-cl
            - Recently rebased; has enough approvals and can be merged.
            - **MERGED**
        - flang#1177: Fix dllimport code generation for module symbols
            - Needs a rebase and reviews.
        - flang#1335: Fixed missing prototype error reported by Clang 15
            - Needs more reviews
            - There are some Review comments from Neumann
        - flang#1341: [scutil] Add support Windows Complex types for pow folding
            - Needs more reviews
            - There are some Review comments from Neumann
        - For windows patches can we merge with one approval
            - YES. they can be merged with one approval.

- Windows GitHub workflow
    - Adam is implementing CI jobs for x86_64 and arm64 Windows on GitHub
    - llvm#149: Add GitHub action to build and upload Windows binaries. Now runs some tests.
    - flang#1349 - Add windows x64 runner support

- NAG test exclusion list
    - Bryan will send an initial form for Pawel and Shivarama to fill out and return.
    - Document will be password-protected; password to be communicated over Slack.
    - Update: Bryan shared the first version of the document. The document has
        - Exclusion list
        - Tests with intermittent failures

- Shivarama/KiranTP will send a public message on Slack to advertise this call and invite more interested parties. Shivarama/KiranTP will also send the invitation to calendar@llvm.org and stress that it is for Classic Flang.
    - Kiran(AMD) had questions with slack advertising.
        - Suggestion is to PIN the meeting invite in slack channel.
    - llvm.org is for llvm community meetings - should we add this to their invite?
        - Will be added to meeting invite
    - **UPDATE: done**
- In LLVM 16, llvm-config has dropped the –src-root option; Classic Flang CMakeLists depends on this. ARM is solving this problem.
    - Pawel has pushed the patch flang#1352
- J3 Fortran Standards meeting report (by Mark Leir) about classic flang in last section : https://j3-fortran.org/doc/year/23/23-123.txt

## February 08, 2023

Attendees
AMD: Shivarama, Kiran TP, Jini
Arm: Pawel
Huawei: Bryan
NVIDIA:
Others: Adam

- Opaque pointers: Both patches have been merged
    - AMD/Arm/Huawei have tested downstream; no issue found so far
- Supporting Classic Flang and LLVM Flang at the same time
    - Sourabh is working on this. He can build Classic Flang and it passes Classic Flang tests, but LLVM Flang tests fail because Driver is constructing jobs incorrectly (getting confused about Classic Flang and LLVM Flang jobs).

        - Main issues: Conflicts in command-line options and file type association. May need to modify TableGen. See discussion in Slack channel.
        - This is currently on hold. **Shivarama will update the plan at the next meeting.**
        - Dan from AMD has plans to work on this as well.
        - **Update:** Sourabh is not getting enough time to work on this. This is not the priority item  for both CPU and GPU teams in AMD and we are not getting resources to dedicatedly work on this.
- QuadFP Complex Support Plan
    - Semantic analyses for type conversion and operand type promotion
    - I/O support
    - C_LONG_DOUBLE_COMPLEX
    - Trigonometry intrinsic functions
    - MATMUL and TRANSPOSE
    - A couple of optimizations
    - PRS submitted for review (need reviews)
        - flang#1344: runtime: use __float128 only in linux
            - Approved by AMD. need ARM reviews
        - flang#1350: [quadfp] semantic analysis for type conversion
            - Need approval from AMD
        - flang#1351: [quadfp] semantic check for operand type promotion
            - Need approval from AMD
        - Pawell had question about origin of constants in the testcases. Is it derived from any test-suites?. Bryan to get back.

- Issues
    - flang#1336: Unexpected result when the function with BIND(C) attribute has ENTRY statements.
        - When the function return type and the entry return type have different ABIs (return-by-reference vs. return-by-value), the frontend does not generate code to return the correct value from the entry.
        - Currently unowned. AMD will look into this.
    - flang#1334: Forcing long double to be 128bit fails with assertion on newer VS 2019 version
        - Adam has pushed a urgent workaround in flang#1340: Remove '-mlong-double-128' compiler option; Bryan has approved

        - This is probably okay unless the Windows team wants to enable QuadFP support, in which case more fundamental work is needed to address the lack of 128-bit FP support in the Microsoft compiler. Silently turning all float128_t variables into 64-bit values isn't providing true support; it also makes the code harder to maintain.
        - Reference: Microsoft Learn: Math and floating-point support states: "Previous 16-bit versions of Microsoft C/C++ and Microsoft Visual C++ supported the `long double` type as an 80-bit precision floating-point data type. In later versions of Visual C++, the `long double` data type is a 64-bit precision floating-point data type identical to the `double` type. The compiler treats `long double` and `double` as distinct types, but the `long double` functions are identical to their `double` counterparts."
        - **Update: above code is merged**
    - flang#1342: Fail to compile on Raspberry Pi 4B
        - Did not include the compiler error message.
        - Update: user shared the error log. Need to look into the error log and check the error and if needed more information will be asked.
    - flang#1338: Build error, resulting from flang2 being placed in an unexpected location when CMake is invoked a certain way. Bryan will push a fix.
        - flang#1345 - need reviews
    - flang#1247: Another build error, possibly because the build system is missing libstdc++-dev.
        - Update: works after installing libstdc++-dev
- Other PRs
    - flang#1314: Automatically deploy sphinx documentation to github pages
        - AMD and Arm approved.
        - What is required from Kiran Chandramohan?
        - No progress.
    - flang#912: Fix Ftn_strcmp_klen return value type (from int64_t to int)
        - Paul has approved; Bryan has some questions.
    - Windows PRs
        - Python build scripts have been merged.
        - llvm#147, llvm#148: Change Python build script to not install by default.
            - **MERGED**

            - flang#1158: Fix implicit declaration of function warnings reported by clang-cl
                - Recently rebased; has enough approvals and can be merged.
                - **MERGED**
            - flang#1177: Fix dllimport code generation for module symbols
                - Needs a rebase and reviews.
            - flang#1335: Fixed missing prototype error reported by Clang 15
                - Needs more reviews
            - flang#1341: [scutil] Add support Windows Complex types for pow folding
                - Needs more reviews
- Windows GitHub workflow
    - Adam is implementing CI jobs for x86_64 and arm64 Windows on GitHub
    - llvm#149: Add GitHub action to build and upload Windows binaries. Now runs some tests.
    - flang#1349 - Add windows x64 runner support
- NAG test exclusion list
    - Bryan will send an initial form for Pawel and Shivarama to fill out and return.
    - Document will be password-protected; password to be communicated over Slack.
    - In Progress
- Shivarama/KiranTP will send a public message on Slack to advertise this call and invite more interested parties. Shivarama/KiranTP will also send the invitation to calendar@llvm.org and stress that it is for Classic Flang.
    - Kiran(AMD) had questions with slack advertising.
        - Suggestion is to PIN the meeting invite in slack channel.
    - llvm.org is for llvm community meetings - should we add this to their invite?
        - Will be added to meeting invite
- In LLVM 16, llvm-config has dropped the –src-root option; Classic Flang CMakeLists depends on this. ARM is solving this problem.
    - In Progress.
- LLVM 16 repository creation and CI update
    - After llvm-16.0.0 tag creation.

## Jan 25, 2023

Attendees

AMD: Shivarama, Kiran TP
Arm: Pawel
Huawei: Bryan
NVIDIA:
Others:

- Opaque pointers: Both patches have been merged
    - AMD/Arm/Huawei have tested downstream; no issue found so far
- Supporting Classic Flang and LLVM Flang at the same time
    - Sourabh is working on this. He can build Classic Flang and it passes Classic Flang tests, but LLVM Flang tests fail because Driver is constructing jobs incorrectly (getting confused about Classic Flang and LLVM Flang jobs).
    - Main issues: Conflicts in command-line options and file type association. May need to modify TableGen. See discussion in Slack channel.
    - This is currently on hold. **Shivarama will update the plan at the next meeting.**
    - Dan from AMD has plans to work on this as well.
- QuadFP Complex Support Plan
    - Semantic analyses for type conversion and operand type promotion
        - PRs will be coming soon
    - I/O support
    - C_LONG_DOUBLE_COMPLEX
    - Trigonometry intrinsic functions
    - MATMUL and TRANSPOSE
    - A couple of optimizations
- Issues
    - flang#1336: Unexpected result when the function with BIND(C) attribute has ENTRY statements.
        - When the function return type and the entry return type have different ABIs (return-by-reference vs. return-by-value), the frontend does not generate code to return the correct value from the entry.
        - Currently unowned.
    - flang#1334: Forcing long double to be 128bit fails with assertion on newer VS 2019 version
        - Adam has pushed a urgent workaround in flang#1340: Remove '-mlong-double-128' compiler option; Bryan has approved
        - This is probably okay unless the Windows team wants to enable QuadFP support, in which case more fundamental

work is needed to address the lack of 128-bit FP support in the Microsoft compiler. Silently turning all float128_t variables into 64-bit values isn't providing true support; it also makes the code harder to maintain.
        - Reference: Microsoft Learn: Math and floating-point support states: "Previous 16-bit versions of Microsoft C/C++ and Microsoft Visual C++ supported the `long double` type as an 80-bit precision floating-point data type. In later versions of Visual C++, the `long double` data type is a 64-bit precision floating-point data type identical to the `double` type. The compiler treats `long double` and `double` as distinct types, but the `long double` functions are identical to their `double` counterparts."
    - flang#1342: Fail to compile on Raspberry Pi 4B
        - Did not include the compiler error message.
    - flang#1338: Build error, resulting from flang2 being placed in an unexpected location when CMake is invoked a certain way. Bryan will push a fix.
    - flang#1247: Another build error, possibly because the build system is missing libstdc++-dev.
- Other PRs
    - flang#1314: Automatically deploy sphinx documentation to github pages
        - AMD and Arm approved.
        - What is required from Kiran Chandramohan?
        - No progress.
    - flang#912: Fix Ftn_strcmp_klen return value type (from int64_t to int)
        - Paul has approved; Bryan has some questions.
    - Windows PRs
        - Python build scripts have been merged.
        - llvm#147, llvm#148: Change Python build script to not install by default.
            - Needs more reviews
        - flang#1158: Fix implicit declaration of function warnings reported by clang-cl
            - Recently rebased; has enough approvals and can be merged.
        - flang#1177: Fix dllimport code generation for module symbols
            - Needs a rebase.
        - flang#1335: Fixed missing prototype error reported by Clang 15

                - Needs more reviews
            - flang#1341: [scutil] Add support Windows Complex types for pow folding
                - Needs more reviews
- Windows GitHub workflow
    - Adam is implementing CI jobs for x86_64 and arm64 Windows on GitHub
    - llvm#149: Add GitHub action to build and upload Windows binaries. Doesn't run any test.
- NAG test exclusion list
    - Bryan will send an initial form for Pawel and Shivarama to fill out and return.
    - Document will be password-protected; password to be communicated over Slack.
- Shivarama/KiranTP will send a public message on Slack to advertise this call and invite more interested parties. Shivarama/KiranTP will also send the invitation to calendar@llvm.org and stress that it is for Classic Flang.
- In LLVM 16, llvm-config has dropped the –src-root option; Classic Flang CMakeLists depends on this. Arm is solving this problem.

## Jan 11, 2023

Attendees
AMD: Shivaram, Kiran TP, Bhuvan
Arm: Paul, Kiran
Huawei: Bryan
NVIDIA:
Others: Adam

- Opaque pointers: Both patches have been merged
    - AMD/Arm have tested downstream; no issue found so far
    - Huawei will start this month
- Supporting Classic Flang and LLVM Flang at the same time
    - Sourabh is working on this. He can build Classic Flang and it passes Classic Flang tests, but LLVM Flang tests fail because Driver is constructing jobs incorrectly (getting confused about Classic Flang and LLVM Flang jobs).
    - Main issues: Conflicts in command-line options and file type association. May need to modify TableGen. See discussion in Slack channel.
    - This is currently on hold. Shivarama will update the plan next meeting.
- QuadFP Complex support

    - Bryan will update the plan next meeting.
        - flang#1318 - Merged
        - New PRs are going to be smaller. 2 or 3 patches.
        - Parse-tree something?
        - Intrinsic support
- Issues
    - Unexpected result when the function with BIND(C) attribute has ENTRY statements. #1336
        - Who can own this?
    - Forcing long double to be 128bit fails with assertion on newer VS 2019 version #1334
        - Bryan will look
- Other PRs
    - flang#1314 - automatically deploy sphinx documentation to github pages
        - AMD and Arm approved.
        - What is required from Kiran Chandramohan ?
    - Windows PRs
        - llvm#133, llvm#134: Add platform independent build script for LLVM (release_13x, release_14x, release_15x)
            - Recently updated; Bryan will review and merge.
        - flang#1158: Fix implicit declaration of function warnings reported by clang-cl
            - Dependent on #1163
                - …and then needs a rebase
        - flang#1177: Fix dllimport code generation for module symbols
            - Needs a rebase.
- Windows GitHub workflow
    - Adam is implementing CI jobs for x86_64 and arm64 Windows on GitHub

## Dec 14, 2022

Attendees
AMD: Shivaram
Arm: Kiran
Huawei: Bryan
NVIDIA:
Others:

- shivaram0206@gmail.com are you joining today?
    - Use the link above.

- Bryan has set up the Call for November
    - This meeting Dec 14, 2022 will be the last this year
    - Jan 11, 2022 will be the first next year
    - shivaram0206@gmail.com to set up a Teams meeting.
- Important points to discuss
    - I just updated the existing one.
- LLVM 14 and LLVM 15
    - Nothing here unless we want 15.0.6 or something.
- Opaque pointers
    - flang#1295: Xin Liu from HNC (Huawei partner team) is working on supporting opaque pointer output from flang2. PRs flang#1321 and flang#1322 have been posted.
        - flang#1321 is now merged
        - flang#1322 status?
    - We agreed to use legacy branch for non opaque pointer code (14 and below)
        - Need to maintain two branches
        - Future PRs need to be created for both branches (legacy and master)
    - Downstream compilers to test the patches
        - Approved by ARM, AMD
        - shivaram0206@gmail.com would be great to have some AMD/X86 testing.
- Supporting Classic Flang and LLVM Flang at the same time
    - Sourabh is working on this. He can build Classic Flang and it passes Classic Flang tests, but LLVM Flang tests fail because Driver is constructing jobs incorrectly (getting confused about Classic Flang and LLVM Flang jobs).
    - Main issues: Conflicts in command-line options and file type association. May need to modify TableGen. See discussion in Slack channel.
    - Any update?
- QuadFP support
    - What is next here?
        - flang#1318 - Merged
        - New PRs are going to be smaller. 2 or 3 patches.
        - Parse-tree something?
        - Intrinsic support
- Issues
    - 
- Other PRs
    - flang#1314 - automatically deploy sphinx documentation to github pages

                ■ AMD and Arm approved.
                ■ What is required from Kiran Chandramohan?
            ○ Windows PRs
                ■ llvm#133, llvm#134: Add platform independent build script for LLVM (release_13x, release_14x)
                    ● Need another review to make sure the build script's options match the Flang counterpart.
                    ● No approvals now.
                ■ flang#1163: win: clean up Windows related macros in runtime/flang
                    ● I think this can be merged now.
                ■ flang#1158: Fix implicit declaration of function warnings reported by clang-cl
                    ● Dependent on #1163
                        ○ …and then needs a rebase
                ■ flang#1177: Fix dllimport code generation for module symbols
                    ● Needs a rebase.
- Thread-safety in Classic Flang I/O runtime
    - Current runtime code doesn't have any obvious or complete support for MP or Async I/O.
    - Some dead code/stubs were removed by Paul previously.
    - Downstream compilers have no support either.

## Nov 30, 2022

Attendees
AMD: Shivaram, Bhuvan, KiranTP
Arm: Pawel, Kiran
Huawei: Bryan
NVIDIA:
Others:

- Bryan has set up the Call for November
    - November 30 call time: 9PM IST, 3:30PM GMT, 10:30AM EST
    - Hosted by ARM
    - Next meeting Dec 14, 2022 will be the last this year
    - Jan 11, 2022 will be the first next year
- LLVM 14 and LLVM 15
    - What's the point of the issue flang#1320 "About LLVM15 release date"?
        - Maybe it just needs to be closed?

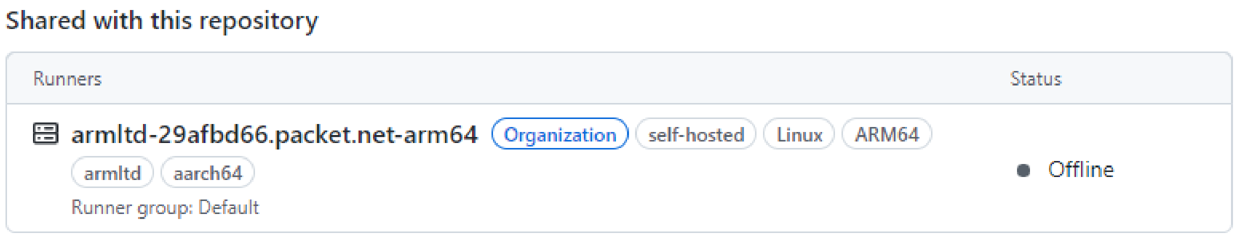
    - Can we make release_15x the default branch now?
        - The ‘legacy’ branch isn’t the default for the flang repo
- Opaque pointers
    - flang#1295: Xin Liu from HNC (Huawei partner team) is working on supporting opaque pointer output from flang2. PRs flang#1321 and flang#1322 have been posted.
        - flang#1321 is now merged
        - flang#1322 need to be rebased as #1321 is now merged
            - Still has review comments to address
    - We agreed to use legacy branch for non opaque pointer code
        - Need to maintain two branches
        - Future PRs need to be created for both branches (legacy and master)
    - Downstream compilers to test the patches
        - Approved by ARM, AMD
- Supporting Classic Flang and LLVM Flang at the same time
    - Sourabh is working on this. He can build Classic Flang and it passes Classic Flang tests, but LLVM Flang tests fail because Driver is constructing jobs incorrectly (getting confused about Classic Flang and LLVM Flang jobs).
    - Main issues: Conflicts in command-line options and file type association. May need to modify TableGen. See discussion in Slack channel.
    - Any update?
- QuadFP support
    - Next patch series will implement quad-precision complex support; fixed all tests, refactoring patch series into smaller patches that can be reviewed more easily.
        - flang#1318 - Minimum functional quad precision complex support.
            - AMD  to review (approved by ARM)
            - Splitting into multiple PRs will help review (future PRs)
- Issues
    - flang#1302: Flang runtime depends on libomp but does not link it with -rpath
        - Downstream compilers probably all have workarounds for this
            - Huawei has submitted flang#1327
                - Reviewed, ready for the merger
    - llvm#11: problematic veclib definition of sincos for x86_64
        - PR llvm#132 is for release_13x branch, should we merge it or drop it?

            - Can we re-run the testing to ensure there is no failure?
        - PR llvm#136 is for release_14x branch
            - Needs to be squashed on merge
            - Cherry-pick to release_15x branch recommended
- Other PRs
    - llvm#137, flang#1319: Make -msve-vector-bits= option behave the same way as Clang's.
        - Both approved and merged
    - flang#1314 - automatically deploy sphinx documentation to github pages
        - Waiting for reviews.
    - Windows PRs
        - llvm#133, llvm#134: Add platform independent build script for LLVM (release_13x, release_14x)
            - Need another review to make sure the build script's options match the Flang counterpart.
        - flang#1163: win: clean up Windows related macros in runtime/flang
            - Reached minimal required number of reviews? Needs a rebase.
        - flang#1158: Fix implicit declaration of function warnings reported by clang-cl
            - Dependent on #1163
                - …and then needs a rebase
        - flang#1177: Fix dllimport code generation for module symbols
            - Needs a rebase.
    - Thread-safety in Classic Flang I/O runtime
        - Current runtime code doesn't have any obvious or complete support for MP or Async I/O.
        - Some dead code/stubs were removed by Paul previously.
        - Downstream compilers have no support either.

## Nov 16, 2022

Attendees
AMD: Shivaram
Arm: Pawel, Kiran
Huawei: Bryan, Joey
NVIDIA:
Others:

- Bryan setup the Call for November
    - November 30 call time: 9PM IST, 3:30PM GMT, 10:30AM EST
    - To be hosted by ARM
- Arm's CI machine was offline, blocking a number of PRs; now fixed.


Shared with this repository

| Runners | Status |
|---|---|
| armltd-29afbd66.packet.net-arm64 Organization self-hosted Linux ARM64 armltd aarch64 Runner group: Default | ● Offline |

- LLVM 14 and LLVM 15
    - release_14x is now the default branch.
    - release_15x branch is ready for patches (llvm#135 has been merged).
        - Branch is based on llvmorg-15.0.3.
        - classic-flang-llvm-project PRs will also be tested on AArch64.
        - **Shivaram: a few debug related lines are commented. Is it part of TODO?**
            - Bryan: A patch that deleted these disabled blocks was not cherry-picked. Fixed in llvm#138, llvm#139, llvm#140.
- Opaque pointers
    - flang#1295: Xin Liu from HNC (Huawei partner team) is working on supporting opaque pointer output from flang2. PRs flang#1321 and flang#1322 have been posted.
        - flang#1321 need to be rebased and merged
        - flang#1322 need to be rebased after #1321 is merged.
        - Use legacy branch for non opaque pointer code?
            - Need to maintain two branches
        - Other options:
            - Use compile time macros
            - Use llvm version checks
    - Future PRs need to be created for both branches (legacy and master)
    - Shivarama has a patch for turning off opaque pointer support in LLVM 16; it is not applicable to LLVM 15. In LLVM 15, CMake flag can be used instead.
    - New legacy branch (without opaque pointer) and CI jobs for LLVM 14 will been created. In progress.
    - Downstream compilers to test the patches

        - Approved by ARM, AMD
- LLVM 12, LLVM 13 release branches
    - AMD and ARM approves
- Supporting Classic Flang and LLVM Flang at the same time
    - Sourabh is working on this. He can build Classic Flang and it passes Classic Flang tests, but LLVM Flang tests fail because Driver is constructing jobs incorrectly (getting confused about Classic Flang and LLVM Flang jobs).
    - Main issues: Conflicts in command-line options and file type association. May need to modify TableGen. See discussion in Slack channel.
        - No update.
- QuadFP support
    - Next patch series will implement quad-precision complex support; fixed all tests, refactoring patch series into smaller patches that can be reviewed more easily.
        - flang#1318 - Minimum functional quad precision complex support.
            - AMD  to review (approved by ARM)
            - Splitting into multiple PRs will help review (future PRs)
- Issues
    - flang#1302: Flang runtime depends on libomp but does not link it with -rpath
        - Downstream compilers probably all have workarounds for this
            - Huawei has submitted flang#1327
        - Has anyone tried removing those libomp dependencies from libflang/libflangrti?
            - Arm had a downstream fix that PGI didn't like; two sets of runtime libraries are shipped, and used depending whether -fopenmp or -fno-openmp is set
            - Previous rejected related PRs: flang#495, flang#487
- Sharing NAG test exemption list: AMD to update on their communication with NAG
    - Sent email to NAG, received approval. **Next steps?**
    - The format of exemption list
        - Text?
        - Spreadsheet?
    - We need to look into the test-suite details (update in next meeting)
    - Should not be disclosed in public

    - Due to differences in downstream compilers the tests may pass/fail differently
    - Exclusion list to be shared (with full path name)
    - Intersection of excluded list will be created.
- CI Build need to change with LLVM assertion enabled (release with debug info)
    - Blocking PR llvm#136
    - Bryan submitted PRs which uncovered CI problems with RelWithDebInfo
        - llvm#141, llvm#142, llvm#143
        - Haven't had time to investigate the build/test crashes
        - Also a disk space shortage in the runner; tried a workaround without success
        - Alternative:: Release build with LLVM_ENABLE_ASSERTIONS=ON
    - Llvm 13 branch support is not required.
- Other PRs
    - llvm#137, flang#1319: Make -msve-vector-bits= option behave the same way as Clang's.
        - Waiting for approvals.
    - llvm#132: Delete pgmath veclib definitions for sincos
        - There are still some sincos definitions for AArch64 after Pawel's recent changes, but they are not actually used since the frontend doesn't generate sincos calls on AArch64 (IIRC).
            - Bryan to confirm whether AArch64 behaviour is guarded statically or with a dynamic option
            - Briefly looked at flang2 code for generating sincos; there doesn't seem to be any exception made for AArch64. Need to continue looking.
        - Author is unresponsive; new PR llvm#136 is in progress.
    - flang#1313 - Build and test flang with release_14x and release_15x
        - Release_14x will be with legacy flang branch
        - Merged.
    - flang#1314 - automatically deploy sphinx documentation to github pages
        - Waiting for reviews.
    - flang#1315 - Fix -Wparenthesis warnings
        - Merged.
    - flang#1316 - Add build script options to allow customization
        - Merged.

    - flang#1317 - Do not zero xflags when registering command line option
        - Merged.
    - Windows PRs
        - flang#1284: Use long double instead of __float128 on Windows platform  windows
            - Merged.
        - flang#1185: Add a platform independent build script for Flang
            - Merged.
        - llvm#133, llvm#134: Add platform independent build script for LLVM (release_13x, release_14x)
            - Need another review to make sure the build script's options match the Flang counterpart.
        - flang#1163: win: clean up Windows related macros in runtime/flang
            - Reached minimal required number of reviews? Needs a rebase.
        - flang#1158: Fix implicit declaration of function warnings reported by clang-cl
            - Dependent on #1163
                - …and then needs a rebase
        - flang#1177: Fix dllimport code generation for module symbols
            - Needs a rebase.

## Nov 02, 2022

Attendees
AMD: Shivaram, KiranTP
Arm: Pawel
Huawei: Bryan, Joey
NVIDIA:
Others:

- Bryan setup the Call for November
    - November 16 call time: 9PM IST, 3:30PM GMT, 10:30AM EST
- LLVM 14 and LLVM 15
    - release_14x is now the default branch.
    - release_15x branch is ready for patches (llvm#135 has been merged).
        - Branch is based on llvmorg-15.0.3.

        - classic-flang-llvm-project PRs will also be tested on AArch64.
        - **Shivaram: a few debug related lines are commented. Is it part of TODO?**
            - Bryan: A patch that deleted these disabled blocks was not cherry-picked. Will fix in a new PR.
- Opaque pointers
    - flang#1295: Xin Liu from HNC (Huawei partner team) is working on supporting opaque pointer output from flang2. PRs flang#1321 and flang#1322 have been posted.
        - Use legacy branch for non opaque pointer code?
            - Need to maintain two branches
        - Other options:
            - Use compile time macros
            - Use llvm version checks
    - Shivarama has a patch for turning off opaque pointer support in LLVM 16; it is not applicable to LLVM 15. In LLVM 15, CMake flag can be used instead.
    - New legacy branch (without opaque pointer) and CI jobs for llvm 14 will be created.
    - Downstream compilers to test the patches (before next meeting)
- Llvm 12, llvm 13 release branches
    - AMD and ARM approves
    - 
- Supporting Classic Flang and LLVM Flang at the same time
    - Sourabh is working on this. He can build Classic Flang and it passes Classic Flang tests, but LLVM Flang tests fail because Driver is constructing jobs incorrectly (getting confused about Classic Flang and LLVM Flang jobs).
    - Main issues: Conflicts in command-line options and file type association. May need to modify TableGen. See discussion in Slack channel.
        - No update.
- QuadFP support
    - Next patch series will implement quad-precision complex support; fixed all tests, refactoring patch series into smaller patches that can be reviewed more easily.
        - flang#1318 - Minimum functional quad precision complex support.
            - AMD and ARM to review
            - Splitting into multiple PRs will help review (future PRs)
- Issues

        - flang#1302: Flang runtime depends on libomp but does not link it with -rpath
            - Downstream compilers probably all have workarounds for this
                - Huawei will submit a workaround PR; no update yet
            - Has anyone tried removing those libomp dependencies from libflang/libflangrti?
                - Arm had a downstream fix that PGI didn't like; two sets of runtime libraries are shipped, and used depending whether -fopenmp or -fno-openmp is set
                - Previous rejected related PRs: flang#495, flang#487
- Sharing NAG test exemption list: AMD to update on their communication with NAG
    - Sent email to NAG, received approval. **Next steps?**
    - The format of exemption list
        - Text?
        - Spreadsheet?
    - We need to look into the test-suite details (update in next meeting)
- CI Build need to change with LLVM assertion enabled (release with debug info)
- 
- LLVM Dev Meeting
- Other PRs
    - llvm#132: Delete pgmath veclib definitions for sincos
        - There are still some sincos definitions for AArch64 after Pawel's recent changes, but they are not actually used since the frontend doesn't generate sincos calls on AArch64 (IIRC).
            - Bryan to confirm whether AArch64 behaviour is guarded statically or with a dynamic option
            - Briefly looked at flang2 code for generating sincos; there doesn't seem to be any exception made for AArch64. Need to continue looking.
        - Author is unresponsive.
    - flang#1313 - Build and test flang with release_14x and release_15x
        - Release_14x will be with legacy flang branch
    - flang#1314 - automatically deploy sphynx documentation to github pages
    - flang#1315 - Fix -Wparenthesis warnings
    - flang#1316 - Add build script options to allow customization

- flang#1317 - Do not zero xflags when registering command line option
- Windows PRs
    - flang#1284: Use long double instead of __float128 on Windows platform  windows
        - Merged
    - flang#1185: Add a platform independent build script for Flang
        - Merged.
    - llvm#133, llvm#134: Add platform independent build script for LLVM (release_13x, release_14x)
        - Need another review to make sure the build script's options match the Flang counterpart.
    - flang#1163: win: clean up Windows related macros in runtime/flang
        - Reached minimal required number of reviews? Needs a rebase.
    - flang#1158: Fix implicit declaration of function warnings reported by clang-cl
        - Dependent on #1163
            - …and then needs a rebase
    - flang#1177: Fix dllimport code generation for module symbols
        - Needs a rebase.

## Oct 19, 2022

Attendees
AMD: Shivaram, Bhuvan, Sourabh
Arm: Kiran, Pawel
Huawei: Bryan
NVIDIA:
Others:

- The current meeting invitation from Arm will expire in November. Bryan set up a test call for today but his system has some caveats (needs a new invitation for every call, needs a new meeting client unless using browser). Is this acceptable to everyone?
    - November 2 call time: 8PM IST, 2:30PM GMT, 10:30AM EDT
    - November 16 call time: 9PM IST, 3:30PM GMT, 10:30AM EST

- FYI: The Flang report submitted to J3 (US Fortran standards meeting body) mentions that Pawel, Shivaram and Bryan runs the call. See second-last paragraph. https://j3-fortran.org/doc/year/22/22-189r1.txt
- LLVM 14 and LLVM 15
    - release_14x is now the default branch.
    - release_15x branch **has been updated to be based on llvmorg-15.0.3.**
        - A new patch is added to release_15x to enable AArch64 CI on PRs.
        - classic-flang-llvm-project PRs will also be tested on AArch64.
    - Huawei has ported Classic Flang patches in PR llvm#135.
        - **Needs reviews!**
        - Merged all NFCI commits in the Flang repo to eliminate new compiler warnings. Merged flang#1307 as a continuation of flang#1292.
        - "check-all" no longer invokes "check-flang"; fixed in flang#1308.
            - Needs reviews.
- Opaque pointers
    - flang#1295: Xin Liu from HNC (Huawei partner team) is working on supporting opaque pointer output from flang2.
    - flang#1306: Pawel submitted an opaque pointer patch. Xin is studying.
    - Shivarama has a patch for turning off opaque pointer support in LLVM 16; it is not applicable to LLVM 15. In LLVM 15, CMake flag can be used instead.
- Supporting Classic Flang and LLVM Flang at the same time
    - Sourabh is working on this. He can build Classic Flang and its passes Classic Flang tests, but LLVM Flang tests fail because Driver is constructing jobs incorrectly (getting confused about Classic Flang and LLVM Flang jobs).
    - Main issues: Conflicts in command-line options and file type association. May need to modify TableGen. See discussion in Slack channel.
- QuadFP support
    - Next patch series will implement quad-precision complex support; fixed all tests, refactoring patch series into smaller patches that can be reviewed more easily.
        - No update.
- Issues
    - flang#1296: Double-free of allocatable members in derived-type elements of temporary arrays

            - Huawei submitted a fix (flang#1304); merged with approvals from AMD and NVIDIA.
        - flang#1302: Flang runtime depends on libomp but does not link it with -rpath
            - Downstream compilers probably all have workarounds for this
                - Huawei will submit a PR
            - Has anyone tried removing those libomp dependencies from libflang/libflangrti?
                - Arm had a downstream fix that PGI didn't like; two sets of runtime libraries are shipped, and used depending whether -fopenmp or -fno-openmp is set
                - Previous rejected related PRs: flang#495, flang#487
- Sharing NAG test exemption list: AMD to update on their communication with NAG
    - Sent email to NAG, waiting for response.
- LLVM Dev Meeting
- Other PRs
    - flang#1091: Add more checkings for derived types in COMMON statements
        - Merged with approvals from AMD and Arm
    - llvm#132: Delete pgmath veclib definitions for sincos
        - There are still some sincos definitions for AArch64 after Pawel's recent changes, but they are not actually used since the frontend doesn't generate sincos calls on AArch64 (IIRC).
            - Bryan to confirm whether AArch64 behaviour is guarded statically or with a dynamic option
            - Briefly looked at flang2 code for generating sincos; there doesn't seem to be any exception made for AArch64. Need to continue looking.
        - Author is unresponsive.
    - Windows PRs
        - flang#1284: Use long double instead of __float128 on Windows platform  windows
            - Merged
        - flang#1185: Add a platform independent build script for Flang
            - Merged.
        - llvm#133, llvm#134: Add platform independent build script for LLVM (release_13x, release_14x)

            - Need another review to make sure the build script's options match the Flang counterpart.
        - flang#1163: win: clean up Windows related macros in runtime/flang
            - Reached minimal required number of reviews? Needs a rebase.
        - flang#1158: Fix implicit declaration of function warnings reported by clang-cl
            - Dependent on #1163
                - …and then needs a rebase
        - flang#1177: Fix dllimport code generation for module symbols
            - Needs a rebase.

## Oct 5, 2022

Attendees
AMD: Shivarama
Arm: Pawel, Kiran
Huawei: Bryan
NVIDIA:
Others:

- The current meeting invitation from Arm will expire in November.
    - In the second meeting of October, a test Zoom-ish meeting invitation will be sent by Bryan. If it works for everyone, we will convert to the new meeting in November.
    - **Test call for October 19:** https://welink.zhumu.com/j/158682177
- QuadFP support
    - Next patch series will implement quad-precision complex support; fixed all tests, refactoring patch series into smaller patches that can be reviewed more easily.
- LLVM14 integration
    - We will update the default branch on classic-flang-llvm-project to release_14x.
        - No objection from anyone.
- LLVM15 integration
    - release_15x branch created; Huawei has ported Classic Flang patches in PR llvm#135
        - Requires a few NFCI commits in the Flang repo to eliminate new compiler warnings. The last PR waiting for review from AMD (will merge anyway if no response this week):

            - flang#1301: [flang1] Clean up duplicate declarations of MAXDIMS; fix is_ordered prototype
    - Handling opaque pointers (Huawei, AMD)
        - flang#1295: Xinliu from HNC (Huawei partner team) is working on supporting opaque pointer output from flang2; pending LLVM 15 integration.
        - Shivarama will provide a patch for turning off opaque pointer support in LLVM 15.
        - What is the possible time scale for this?
- Issues
    - flang#1296: Double-free of allocatable members in derived-type elements of temporary arrays
        - Huawei working on a fix
    - flang#1302: Flang runtime depends on libomp but does not link it with -rpath
        - Downstream compilers probably all have workarounds for this
        - Has anyone tried removing those libomp dependencies from libflang/libflangrti?
            - Arm had a downstream fix that PGI didn't like; two sets of runtime libraries are shipped, and used depending whether -fopenmp or -fno-openmp is set
            - Previous rejected related PRs: flang#495, flang#487
- Other PRs
    - flang#1091: Add more checkings for derived types in COMMON statements
        - Needs review from AMD
    - llvm#132: Delete pgmath veclib definitions for sincos
        - There are still some sincos definitions for AArch64 after Pawel's recent changes, but they are not actually used since the frontend doesn't generate sincos calls on AArch64 (IIRC).
            - Bryan to confirm whether AArch64 behaviour is guarded statically or with a dynamic option
        - Needs review
    - Portable build scripts in Python
        - flang#1185: Add a platform independent build script for Flang
            - Looks fairly good, just needs some more approvals; it doesn't have to replace build-flang.sh and CI doesn't have to change right away

            - llvm#133, llvm#134: Add platform independent build script for LLVM (release_13x, release_14x)
    - Windows PRs
        - flang#1163: win: clean up Windows related macros in runtime/flang
            - Reached minimal required number of reviews? Needs a rebase.
        - flang#1158: Fix implicit declaration of function warnings reported by clang-cl
            - Dependent on #1163
                - …and then needs a rebase
        - flang#1177: Fix dllimport code generation for module symbols
    - Other issues?
- Sharing NAG test exemption list: AMD to update on their communication with NAG

## Sep 21, 2022

Attendees
AMD: Shivarama, Bhuvan, Alok, Themos
Arm: Pawel, Kiran
Huawei: Bryan
NVIDIA:
Others:

- QuadFP support
    - Next steps: I/O support, quad-precision complex
        - Next patch will be quad-precision complex support.
- LLVM14 integration
    - Only X86 tests seem to be performed, is AArch64 CI on release_14x still failing? possibly caused by a bug in LLVM - false alarm
- LLVM15 integration
    - release_15x branch created; Huawei has ported Classic Flang patches in PR #135
        - Requires a few NFCI commits in the Flang repo to eliminate new compiler warnings. Call for reviewers:
            - #1289: [LLVM 15][runtime] Define overloaded function types and improve type checking is merged now
            - #1290: [LLVM 15][runtime] Stop using K&R function declarations is merged now

        - #1291: [LLVM 15][runtime] Remove extern function declarations is merged now
        - #1292: [LLVM 15][runtime] Add C prototypes for external Fortran matmul functions
        - #1293: [LLVM 15][test] Rename lit.site.cfg variable to support LLVM 15 (replaces #1266 which became closed now)
        - #1294: [LLVM 15][libpgmath] Decouple from libflangrti by using errno directly; NFCI is merged now
  - Handling opaque pointers (Huawei, AMD)?
    - Huawei and AMD estimating effort required to make flang2 output opaque pointers
    - Huawei has some proof of concept in IR, now it's time to make changes in the flang2's code
    - What is the possible time scale for this?
  - Honoring of order of options, completion of the subject
    - PR #118 merged now
  - -fsigned-zeros and -fassociative-math and nsz/reassoc attributes:
    - classic-flang-llvm-project repo: PR #118, #119, #120, #121 are all merged now
    - flang repo: PR #1268 is merged now; some tweaking in the test cases were required
  - Some sema check improvements (Arm/Huawei approved); would AMD take a look?
    - #1091: Add more checkings for derived types in COMMON statements
      - Need review from AMD
  - Portable build scripts in Python
    - flang #1185: Add a platform independent build script for Flang
      - Looks fairly good, just needs some more approvals; it doesn't have to replace build-flang.sh and CI doesn't have to change right away
  - Windows PRs
    - #1163: win: clean up Windows related macros in runtime/flang
      - Reached minimal required number of reviews?
    - #1158: Fix implicit declaration of function warnings reported by clang-cl
      - Dependent on #1163
        - …and then needs a rebase
    - #1177: Fix dllimport code generation for module symbols

    - Other issues?

## Sep 07, 2022

Attendees
AMD: Shivarama, Bhuvan, Alok, Themos
Arm: Pawel, Kiran
Huawei: Bryan
NVIDIA:
Others:

- Functional issues (PRs)
    - #1088: merged (Fix a segmentation fault related to assumed length character functions)
    - #1064: merged (Fix the bug where the v_list argument is not provided as a zero-sized array in some cases as required by the standard)
    - #1052: merged (Fix the bug where the name of ENTRY is used as an argument)
    - #1050: merged (Fix improper handling of errors caused by invalid declaration of result variable)
    - #131 : merged (removing -lompstub from the link line)
- Debuginfo issues (PRs)
    - #1198: merged (Debug support for assumed shape array at higher optimizations)
    - #1201: merged (Use enhanced DIImportedEntity to generate better IR for renamed modules)
- QuadFP support
    - Issue #1278: closed (SEVERE ERROR reported by symini.cpp)
    - Next steps: I/O support, quad-precision complex
        - Next patch will be quad-precision complex support.
- LLVM14 integration
    - #1274: merged (Update workflows to build release_13x and release_14x)
    - Is AArch64 CI on release_14x still failing? possibly caused by a bug in LLVM
- LLVM15 integration
    - release_15x branch created; Huawei has ported Classic Flang patches in PR #135
        - Requires a few NFCI commits in the Flang repo to eliminate new compiler warnings; will submit PR today
    - Upgrading to new pass manager
    - Handling opaque pointers (Huawei, AMD)?

        - Huawei and AMD estimating effort required to make flang2 output opaque pointers
        - Huawei has some proof of concept in IR, now it’s time to make changes in the flang2’s code
    - Initiate discussion on slack channel for opaque pointer
- SPEC benchmarks with upstream flang (no progress on that)
    - Cam4 fails with VE error.
    - Issue #11 - addresses sincos issue.
- Honoring of order of options
    - New version of PR #118
        - Are the reviewers happy now?
            - Huawei: yes
            - AMD: waiting for review
- Arm’s efforts towards upstreaming internal changes, reducing downstream delta and associated cost of the technical debt
    - PR# 129
        - Provide separate veclib functions of PGMATH for AARCH64
            - Arm: Need to rework it for release_14x branch
    - PR #118
        - New version (mentioned above)
        - Changes for reassoc attributes
    - -fsigned-zeros and -fassociative-math and nsz/reassoc attributes:
        - classic-flang-llvm-project repo: PR #118, #119, #120, #121 (these are dedicated to various LLVM versions)
            - -fsigned-zeros and -fassociative-math options should be considered together with -Kieee and -ffast-math/-fno-fast-math
        - flang repo: PR #1268, classic-flang-llvm-project PR’s need to be applied first to make all of the tests pass
            - Huawei has approved
            - AMD has approved
            - Classic-flang-llvm-project part needs to be merged first
    - Some sema check improvements (Arm/Huawei approved); would AMD take a look?
        - #1091: Add more checkings for derived types in COMMON statements
            - Need review from AMD
    - Portable build scripts in Python
        - flang #1185: Add a platform independent build script for Flang

            - Looks fairly good, just needs some more approvals; it doesn't have to replace build-flang.sh and CI doesn't have to change right away
        - classic-flang-llvm-project #73: Add platform independent build script for LLVM
            - Requested author to rebase
            - Closed due to two other PRs being opened
    - Windows PRs
        - #1163: win: clean up Windows related macros in runtime/flang
            - Reached minimal required number of reviews?
        - #1158: Fix implicit declaration of function warnings reported by clang-cl
            - Dependent on #1163
                - …and then needs rebase
        - #1177: Fix dllimport code generation for module symbols
    - Other issues?

## Aug 24, 2022

Attendees
AMD: Shivarama, Bhuvan, Kiran Kumar
Arm: Pawel, Kiran
Huawei: Bryan
NVIDIA:
Others:

- Functional Issues
    - #1088: Fix a segmentation fault related to assumed length character functions
        - AMD to approve and requires review from ARM (ARM: but we did approve it…)
        - Approved by AMD and ARM - need to be merged
    - #1064: Fix the bug where the v_list argument is not provided as a zero-sized array in some cases as required by the standard
        - AMD need to approve (will be done by EOD)
    - #1052: Fix the bug where the name of ENTRY is used as an argument
        - Require review from AMD (Approved and merged)
    - #1050: Fix improper handling of errors caused by invalid declaration of result variable
        - Need review from AMD and ARM (approved and merged)
    - #131 : removing -lompstub from the link line

        - Need review from AMD and ARM (Approved)
- GNU isnan() extension from AMD (originally #672 and #773)
    - PR #1269 has been merged as a combination of the above PRs
- QuadFP support
    - PR #1215: QuadFP patch has been approved by Paul;approved by AMD and meged
        - Intermediated commits are merged as individual commits
        - Cleanup of master branch will be done by Bryan
    - Issue #1278: SEVERE ERROR reported by symini.cpp; corrected in PR #1215

  Reviewed and merged. Issue to be closed.
    - Next steps: I/O support, quad-precision complex
        - Next patch will be quad-precision complex support.
- LLVM14 integration
    - #1274: Update workflows to build release_13x and release_14x
        - PR #1274 merged
        - AArch64 CI on release_14x is failing; possibly caused by a bug in LLVM pre-build
        - Lack of a good release_14x CI is blocking the following PRs:
        - #1198: Debug support for assumed shape array at higher optimizations
            - Need review from ARM
        - #1201: Use enhanced DIImportedEntity to generate better IR for renamed modules
            - Bryan to review and waiting for review for ARM
- LLVM15 integration
    - Upgrading to new pass manager
    - Handling opaque pointers
    - Initiate discussion on slack channel for opaque pointer
- SPEC benchmarks with upstream flang
    - Cam4 fails with VE error.
    - Issue #11 - addresses sincos issue.
- Honoring of order of options
    - Regardless of the position of reassoc-math, Kieee overrides it. This needs to be fixed.
- Arm's efforts towards upstreaming internal changes, reducing downstream delta and associated cost of the technical debt
    - PR# 129
        - Provide separate veclib functions of PGMATH for AARCH64
    - PR #118
        - Changes for reassoc attributes

    - Classic Flang shouldn't use fast-math by default; Paul will upstream a patch
        - classic-flang-llvm-project (#122, #123, #124, #125) and flang #1276 have been merged
    - -fp-contract=fast and FMA patches have been merged, but question remains
        - classic-flang-llvm-project repo: PR #113, #115, #116, #117 (these are dedicated to various LLVM versions)
        - flang repo: PR #1267
            - -ffp-contract=fast and -Mfma= options should be considered together with -Kieee and -ffast-math/-fno-fast-math (cf. RenderFloatingPointOptions in clang/lib/Driver/ToolChains/Clang.cpp)
    - -fsigned-zeros and -fassociative-math and nsz/reassoc attributes:
        - classic-flang-llvm-project repo: PR #118, #119, #120, #121 (these are dedicated to various LLVM versions)
            - -fsigned-zeros and -fassociative-math options should be considered together with -Kieee and -ffast-math/-fno-fast-math
        - flang repo: PR #1268, classic-flang-llvm-project PR's need to be applied first to make all of the tests pass
            - Huawei has approved
            - Need review from AMD
    - Loop metadata test cases: PR #1259
        - Needs reviews (AMD and Bryan)
    - Improve formatted input performance by using optimized libc functions on ARM64: PR #642
        - Needs reviews
    - Use FLANG_VENDOR: PR #150
- __WORDSIZE is only required by kmp_critical_name; why do we need to typedef kmp_critical_name two different ways? openmp/runtime/src/kmp.h does not.
    - #1271: portability: __WORDSIZE isn't defined in MinGW
        - Merged
    - #1280: Use OpenMP definition of kmp_critical_name (CLOSED)
        - This causes a mp_correct test to fail on AArch64, showing that the alignment is significant (need some debugging to find the root cause)
    - #1272: portability: Fix alignment and size of kmp_critical_name for LLP64
        - Arm and Huawei approved; AMD approved and merged

- Some sema check improvements (Arm/Huawei approved); would AMD take a look?
    - #1051: Add reporting for two grammar errors pertaining to PROTECTED attribute
        - Merged
    - #1091: Add more checkings for derived types in COMMON statements
        - Need review from AMD
- Fallout of LLVM's switch to opaque pointers? How are we affected?
    - Clang will continue to consume IR with typed pointers until LLVM 16.
    - Huawei and AMD estimating effort required to make flang2 output opaque pointers; any update?
- LLVM 15 branch initialized; Arm has done early testing and found only one problem
    - #1277: test/directives/vector_directive3.f90 may vectorize by larger factors than 2
        - Merged
    - Investigating problems related to LTO and opaque pointers (mixing C and Fortran code); Driver changes may be required
- CP2K and challenges it creates
    - Building OpenBLAS for CP2K: setting INTERFACE64=1 and BINARY=64 turned out to be a bad idea
    - A new intermittent issue in QS benchmark: `natom` local variable gots malformed in an OpenMP parallel region in the src/core_ppnl.F file. It is used to compute a hash for indexing a lock passed to the explicitly called omp_set_lock() function (of the OpenMP runtime), in effect it results in a segfault. A workaround is to stop using the `natom` variable, and compute the value it holds (namely, `SIZE(particle_set)`) in every place the `natom` variable was read.
        - It is still very suspicious that the value held in this variable got malformed; possible regression?
        - We were avoiding using OpenMP in CP2K previously (with a small patch helping to do so), but as it got harder and harder to avoid it, we now always build CP2K with OpenMP
    - `pdpotrf` subroutine issue: Cholesky decompose failure
    - No update since last time: Shivarama's team will be working on it
- Portable build scripts in Python
    - flang #1185: Add a platform independent build script for Flang

        - Looks fairly good, just needs some more approvals; it doesn't have to replace build-flang.sh and CI doesn't have to change right away
    - classic-flang-llvm-project #73: Add platform independent build script for LLVM
        - Requested author to rebase
- Windows PRs
    - #1163: win: clean up Windows related macros in runtime/flang
    - #1158: Fix implicit declaration of function warnings reported by clang-cl
        - Dependent on #1163
    - #1177: Fix dllimport code generation for module symbols
- Other issues?

## Aug 10, 2022

Attendees
AMD: Shivarama,
Arm: Rich
Huawei: Bryan
NVIDIA:
Others:

- Functional Issues
    - Issue #1253 (post PR #1212) now has a fix under review (thanks!)
        - PR #1261 has been merged
    - Paul found a NAG regression post PR #1101
        - PR #1279 fixes this and has been merged
    - #1088: Fix a segmentation fault related to assumed length character functions
        - AMD to approve and requires review from ARM
    - #1064: Fix the bug where the v_list argument is not provided as a zero-sized array in some cases as required by the standard
        - AMD need to approve
    - #1052: Fix the bug where the name of ENTRY is used as an argument
        - Require review from AMD
    - #1050: Fix improper handling of errors caused by invalid declaration of result variable
        - Need review from AMD and ARM
    - #131 : removing -lompstub from the link line
        - Reviewed and merged
- GNU isnan() extension from AMD (originally #672 and #773)

        - PR #1269 has been merged as a combination of the above PRs
- QuadFP support
    - PR #1215: QuadFP patch has been approved by Paul;approved by AMD and meged
        - Intermediated commits are merged as individual commits
        - Cleanup of master branch will be done by Bryan
    - Issue #1278: SEVERE ERROR reported by symini.cpp; corrected in PR #1215
    - Next steps: I/O support, quad-precision complex
        - Next patch will be quad-precision complex support.
- LLVM14 integration
    - #1274: Update workflows to build release_13x and release_14x
        - PR #1274 merged
        - AArch64 CI on release_14x is failing; possibly caused by a bug in LLVM pre-build
        - Lack of a good release_14x CI is blocking the following PRs:
        - #1198: Debug support for assumed shape array at higher optimizations
            - Need review from ARM
        - #1201: Use enhanced DIImportedEntity to generate better IR for renamed modules
            - Bryan to review and waiting for review for ARM
- LLVM15 integration
    - Upgrading to new pass manager
    - Handling opaque pointers
    - Initiat discussion on slack channel for opaque pointer
- Arm’s efforts towards upstreaming internal changes, reducing downstream delta and associated cost of the technical debt
    - Classic Flang shouldn't use fast-math by default; Paul will upstream a patch
        - classic-flang-llvm-project (#122, #123, #124, #125) and flang #1276 have been merged
    - -fp-contract=fast and FMA patches have been merged, but question remains
        - classic-flang-llvm-project repo: PR #113, #115, #116, #117 (these are dedicated to various LLVM versions)
        - flang repo: PR #1267
            - -ffp-contract=fast and -Mfma= options should be considered together with -Kieee and -ffast-math/-fno-fast-math (cf. RenderFloatingPointOptions in clang/lib/Driver/ToolChains/Clang.cpp)

    - -fsigned-zeros and -fassociative-math and nsz/reassoc attributes:
        - classic-flang-llvm-project repo: PR #118, #119, #120, #121 (these are dedicated to various LLVM versions)
            - -fsigned-zeros and -fassociative-math options should be considered together with -Kieee and -ffast-math/-fno-fast-math
        - flang repo: PR #1268, classic-flang-llvm-project PR's need to be applied first to make all of the tests pass
            - Huawei has approved
            - Need review from AMD
    - Loop metadata test cases: PR #1259
        - Needs reviews (AMD and Bryan)
    - Improve formatted input performance by using optimized libc functions on ARM64: PR #642
        - Needs reviews
    - Use FLANG_VENDOR: PR #150
- __WORDSIZE is only required by kmp_critical_name; why do we need to typedef kmp_critical_name two different ways? openmp/runtime/src/kmp.h does not.
    - #1271: portability: __WORDSIZE isn't defined in MinGW
        - Merged
    - #1280: Use OpenMP definition of kmp_critical_name (CLOSED)
        - This causes a mp_correct test to fail on AArch64, showing that the alignment is significant (need some debugging to find the root cause)
    - #1272: portability: Fix alignment and size of kmp_critical_name for LLP64
        - Arm and Huawei approved; AMD approved and merged
- Some sema check improvements (Arm/Huawei approved); would AMD take a look?
    - #1051: Add reporting for two grammar errors pertaining to PROTECTED attribute
        - Merged
    - #1091: Add more checkings for derived types in COMMON statements
        - Need review from AMD
- Fallout of LLVM's switch to opaque pointers? How are we affected?
    - Clang will continue to consume IR with typed pointers until LLVM 16.
    - Huawei and AMD estimating effort required to make flang2 output opaque pointers; any update?
- LLVM 15 branch initialized; Arm has done early testing and found only one problem

        - #1277: test/directives/vector_directive3.f90 may vectorize by larger factors than 2
            - Merged
        - Investigating problems related to LTO and opaque pointers (mixing C and Fortran code); Driver changes may be required
- CP2K and challenges it creates
    - Building OpenBLAS for CP2K: setting INTERFACE64=1 and BINARY=64 turned out to be a bad idea
    - A new intermittent issue in QS benchmark: `natom` local variable gots malformed in an OpenMP parallel region in the src/core_ppnl.F file. It is used to compute a hash for indexing a lock passed to the explicitly called omp_set_lock() function (of the OpenMP runtime), in effect it results in a segfault. A workaround is to stop using the `natom` variable, and compute the value it holds (namely, `SIZE(particle_set)`) in every place the `natom` variable was read.
        - It is still very suspicious that the value held in this variable got malformed; possible regression?
        - We were avoiding using OpenMP in CP2K previously (with a small patch helping to do so), but as it got harder and harder to avoid it, we now always build CP2K with OpenMP
    - `pdpotrf` subroutine issue: Cholesky decompose failure
    - No update since last time: Shivarama's team will be working on it
- Portable build scripts in Python
    - flang #1185: Add a platform independent build script for Flang
        - Looks fairly good, just needs some more approvals; it doesn't have to replace build-flang.sh and CI doesn't have to change right away
    - classic-flang-llvm-project #73: Add platform independent build script for LLVM
        - Requested author to rebase
- Windows PRs
    - #1163: win: clean up Windows related macros in runtime/flang
    - #1158: Fix implicit declaration of function warnings reported by clang-cl
        - Dependent on #1163
    - #1177: Fix dllimport code generation for module symbols
- Other issues?

## Jul 27, 2022

Attendees

AMD: Shivarama, Bhuvanendra, Kiran TP
Arm: Pawel
Huawei: Bryan
NVIDIA:
Others:

- Functional Issues
  - Issue #1253 (post PR #1212) now has a fix under review (thanks!)
    - PR #1261 has been merged
  - Paul found a NAG regression post PR #1101
    - PR #1279 fixes this and has been merged
  - #1088: Fix a segmentation fault related to assumed length character functions
  - #1064: Fix the bug where the v_list argument is not provided as a zero-sized array in some cases as required by the standard
  - #1052: Fix the bug where the name of ENTRY is used as an argument
  - #1050: Fix improper handling of errors caused by invalid declaration of result variable
- GNU isnan() extension from AMD (originally #672 and #773)
  - PR #1269 has been merged as a combination of the above PRs
- QuadFP support
  - PR #1215: QuadFP patch has been approved by Paul; need reviews from AMD
  - Issue #1278: SEVERE ERROR reported by symini.cpp; corrected in PR #1215
  - Next steps: I/O support, quad-precision complex
- LLVM14 integration
  - #1274: Update workflows to build release_13x and release_14x
    - AArch64 CI on release_14x is failing; possibly caused by a bug in LLVM pre-build
    - Lack of a good release_14x CI is blocking the following PRs:
    - #1198: Debug support for assumed shape array at higher optimizations
    - #1201: Use enhanced DIImportedEntity to generate better IR for renamed modules
- Arm's efforts towards upstreaming internal changes, reducing downstream delta and associated cost of the technical debt
  - Classic Flang shouldn't use fast-math by default; Paul will upstream a patch
    - classic-flang-llvm-project (#122, #123, #124, #125) and flang #1276 have been merged
  - -fp-contract=fast and FMA patches have been merged, but question remains
    - classic-flang-llvm-project repo: PR #113, #115, #116, #117 (these are dedicated to various LLVM versions)
    - flang repo: PR #1267
      - -ffp-contract=fast and -Mfma= options should be considered together with -Kieee and -ffast-math/-fno-fast-math (cf.

            RenderFloatingPointOptions in clang/lib/Driver/ToolChains/Clang.cpp)
    - -fsigned-zeros and -fassociative-math and nsz/reassoc attributes:
        - classic-flang-llvm-project repo: PR #118, #119, #120, #121 (these are dedicated to various LLVM versions)
            - -fsigned-zeros and -fassociative-math options should be considered together with -Kieee and -ffast-math/-fno-fast-math
        - flang repo: PR #1268, classic-flang-llvm-project PR's need to be applied first to make all of the tests pass
            - Huawei has approved
    - Loop metadata test cases: PR #1259
        - Needs reviews
    - Improve formatted input performance by using optimized libc functions on ARM64: PR #642
        - Needs reviews
    - Use FLANG_VENDOR: PR #150
- __WORDSIZE is only required by kmp_critical_name; why do we need to typedef kmp_critical_name two different ways? openmp/runtime/src/kmp.h does not.
    - #1271: portability: __WORDSIZE isn't defined in MinGW
        - Merged
    - #1280: Use OpenMP definition of kmp_critical_name (CLOSED)
        - This causes a mp_correct test to fail on AArch64, showing that the alignment is significant (need some debugging to find the root cause)
    - #1272: portability: Fix alignment and size of kmp_critical_name for LLP64
        - Arm and Huawei approved; AMD?
- Some sema check improvements (Arm/Huawei approved); would AMD take a look?
    - #1051: Add reporting for two grammar errors pertaining to PROTECTED attribute
        - Merged
    - #1091: Add more checkings for derived types in COMMON statements
- Fallout of LLVM's switch to opaque pointers? How are we affected?
    - Clang will continue to consume IR with typed pointers until LLVM 16.
    - Huawei and AMD estimating effort required to make flang2 output opaque pointers; any update?
- LLVM 15 branch initialized; Arm has done early testing and found only one problem
    - #1277: test/directives/vector_directive3.f90 may vectorize by larger factors than 2
        - Merged
    - Investigating problems related to LTO and opaque pointers (mixing C and Fortran code); Driver changes may be required
- CP2K and challenges it creates
    - Building OpenBLAS for CP2K: setting INTERFACE64=1 and BINARY=64 turned out to be a bad idea
    - A new intermittent issue in QS benchmark: `natom` local variable gots malformed in an OpenMP parallel region in the src/core_ppnl.F file. It is used to compute a

hash for indexing a lock passed to the explicitly called omp_set_lock() function (of the OpenMP runtime), in effect it results in a segfault. A workaround is to stop using the `natom` variable, and compute the value it holds (namely, `SIZE(particle_set)`) in every place the `natom` variable was read.
        - It is still very suspicious that the value held in this variable got malformed; possible regression?
        - We were avoiding using OpenMP in CP2K previously (with a small patch helping to do so), but as it got harder and harder to avoid it, we now always build CP2K with OpenMP
    - `pdpotrf` subroutine issue: Cholesky decompose failure
    - No update since last time: Shivarama's team will be working on it
- Portable build scripts in Python
    - flang #1185: Add a platform independent build script for Flang
        - Looks fairly good, just needs some more approvals; it doesn't have to replace build-flang.sh and CI doesn't have to change right away
    - classic-flang-llvm-project #73: Add platform independent build script for LLVM
        - Requested author to rebase
- Windows PRs
    - #1163: win: clean up Windows related macros in runtime/flang
    - #1158: Fix implicit declaration of function warnings reported by clang-cl
        - Dependent on #1163
    - #1177: Fix dllimport code generation for module symbols
- Other issues?

## Jul 13, 2022

Attendees
AMD: Shivarama, Bhuvanendra, Kiran TP
Arm: Paul
Huawei: Bryan
Nvidia:
Others:

- Issues and Pull Requests
    - Issue #1253 (post PR #1212) now has a fix under review (thanks!)
        - https://github.com/flang-compiler/flang/pull/1261
        - Author is working on addressing the review comments
    - Paul found a NAG regression post PR #1101
        - Huawei team will follow up
    - A question to AMD: can PR’s #672 and #773 be reissued as one new PR (taking ownership effectively); we are using this code downstream for a couple of years and it passes in our CI
        - PR #1269 has been submitted and needs reviews

    - PR #1215: QuadFP patch being updated to address Paul's comments; will be ready by EOD
- How far are we from LLVM14 integration?
    - PR #1266 replaces TARGET_TRIPLE with LLVM_TARGET_TRIPLE: CI failed
        - Leave it for now
    - AArch64 CI is missing: any update?
      Kiran: No update. What additional permissions will you need to have a look? I suspect even if you have permissions there is still something on the Arm side that might block progress.
- Arm's efforts towards upstreaming internal changes, reducing downstream delta and associated cost of the technical debt
    - -fp-contract=fast and FMA:
        - classic-flang-llvm-project repo: PR #113, #115, #116, #117 (these are dedicated to various LLVM versions)
            - Bryan has reviewed
            - Go bindings test has been disabled in release_13x and release_14x
        - flang repo: PR #1267, classic-flang-llvm-project PR's need to be applied first to make all of the tests pass
    - -fsigned-zeros and -fassociative-math and nsz/reassoc attributes:
        - classic-flang-llvm-project repo: PR #118, #119, #120, #121 (these are dedicated to various LLVM versions)
            - Bryan has reviewed
        - flang repo: PR #1268, classic-flang-llvm-project PR's need to be applied first to make all of the tests pass
    - Classic Flang shouldn't use fast-math by default; Paul will upstream a patch
    - Handling longer paths and filenames: PR #1262
        - Shivarama: Increasing the buffer size may still be insufficient in some cases
    - Initialize cached_access_group_metadata: PR #1257
    - Loop metadata test cases: PR #1259
    - Improve formatted input performance by using optimized libc functions on ARM64: PR #642
    - Use FLANG_VENDOR: PR #150
- __WORDSIZE is only required by kmp_critical_name; why do we need to typedef kmp_critical_name two different ways? openmp/runtime/src/kmp.h does not.
    - #1271: portability: __WORDSIZE isn't defined in MinGW
    - #1272: portability: Fix alignment and size for LLP64
- Some sema check improvements already received two approvals (Paul, Bryan); would AMD take a look?
    - #1051: Add reporting for two grammar errors pertaining to PROTECTED attribute
    - #1091: Add more checkings for derived types in COMMON statements
- #1185: Add a platform independent build script for Flang

    - Looks fairly good, just needs some more approvals; it doesn't have to replace build-flang.sh and CI doesn't have to change right away
- #1201: Use enhanced DIImportedEntity to generate better IR for renamed modules
    - Blocked by AArch64 CI job which is still building release_11x and release_12x only; need a change in .github/workflows/build_flang_arm64.yml
- Fallout of LLVM's switch to opaque pointers? How are we affected?
    - Clang will continue to consume IR with typed pointers until LLVM 16.
    - Huawei and AMD estimating effort required to make flang2 output opaque pointers; any update?
- Anyone working on building Classic Flang and LLVM Flang together? Dan (from AMD GPU team) mentioned that they were looking into it.
    - AMD will not be pursuing runtime selection; providing two sets of binaries
- CP2K and challenges it creates
    - Building OpenBLAS for CP2K: setting INTERFACE64=1 and BINARY=64 turned out to be a bad idea
    - A new intermittent issue in QS benchmark: `natom` local variable gots malformed in an OpenMP parallel region in the src/core_ppnl.F file. It is used to compute a hash for indexing a lock passed to the explicitly called omp_set_lock() function (of the OpenMP runtime), in effect it results in a segfault. A workaround is to stop using the `natom` variable, and compute the value it holds (namely, `SIZE(particle_set)`) in every place the `natom` variable was read.
        - It is still very suspicious that the value held in this variable got malformed; possible regression?
        - We were avoiding using OpenMP in CP2K previously (with a small patch helping to do so), but as it got harder and harder to avoid it, we now always build CP2K with OpenMP
    - `pdpotrf` subroutine issue: Cholesky decompose failure
    - No update since last time: Shivarama's team will be working on it
- Other issues?

## Jun 29, 2022

Attendees
AMD: Shivaram, Chirag, Raghu
Arm: Kiran, Paul
Huawei: Bryan
Nvidia:
Others:

- Remaining from last week:
    - Issue #1253 (post PR #1212) now has a fix under review (thanks!), PR #1261

        - A question to AMD: can PR’s #672 and #773 be reissued as one new PR (taking ownership effectively); we are using this code downstream for a couple of years and it passes in our CI
- How far are we from LLVM14 integration?
    - PR #1266 (flang repo) may be needed!
    - AArch64 CI is missing
- Arm’s efforts towards upstreaming internal changes, reducing downstream delta and associated cost of the technical debt
    - fp-contract=fast and FMA:
        - classic-flang-llvm-project repo: PR #113, #114, #115, #116, #117 (these are dedicated to various LLVM versions)
            - In case of the release_13x branch, also a problem with the Go bindings tests had to be sorted
        - flang repo: PR #1267, classic-flang-llvm-project PR’s need to be applied first to make all of the tests pass
    - Handling longer paths and filenames: PR #1262
    - Other smaller bits: PR #1257, #1259, #1260, #1265
- CP2K and challenges it creates
    - Building OpenBLAS for CP2K: setting INTERFACE64=1 and BINARY=64 turned out to be a bad idea
    - A new intermittent issue: `natom` local variable gots malformed in an OpenMP parallel region in the src/core_ppnl.F file. It is used to compute a hash for indexing a lock passed to the explicitly called omp_set_lock() function (of the OpenMP runtime), in effect it results in a segfault. A workaround is to stop using the `natom` variable, and compute the value it holds (namely, `SIZE(particle_set)`) in every place the `natom` variable was read.
        - It is still very suspicious that the value held in this variable got malformed; possible regression?
        - We were avoiding using OpenMP in CP2K previously (with a small patch helping to do so), but as it got harder and harder to avoid it, we now always build CP2K with OpenMP
    - `pdpotrf` subroutine issue: Cholesky decompose failure:

```
Number of electrons:                                    512
Number of occupied orbitals:                            256
Number of molecular orbitals:                           256

Number of orbital functions:                            2560
Number of independent orbital functions:                       2560

Extrapolation method: initial_guess

*******************************************************************************
*   ___                                                   *
```

```
 *  /   \                                                             *
 * [ABORT]                                                             *
 *  \___/          Cholesky decompose failed: the matrix is not positive definite or
*
 *        |                              ill-conditioned.              *
 *  O/|                                                                *
 * /| |                                                                *
 * / \                                           fm/cp_fm_cholesky.F:95 *
 ***************************************************************************



 ===== Routine Calling Stack =====

            9 cp_fm_cholesky_decompose
            8 make_basis_sm
            7 calculate_first_density_matrix
            6 scf_env_initial_rho_setup
            5 init_scf_run
            4 qs_energies
            3 qs_forces
            2 qs_mol_dyn_low
            1 CP2K
```

How was it called?:

```
#if defined(__SCALAPACK)
            desca(:) = matrix%matrix_struct%descriptor(:)

            IF (matrix%use_sp) THEN
            CALL pspotrf('U', my_n, a_sp(1, 1), 1, 1, desca, info)
            ELSE
            CALL pdpotrf('U', my_n, a(1, 1), 1, 1, desca, info) ← this was called!
            END IF

#else

....

#endif

            IF (PRESENT(info_out)) THEN
            info_out = info
            ELSE
            IF (info /= 0) &
            CALL cp_abort(__LOCATION__, &
                        "Cholesky decompose failed: the matrix is not positive definite
or ill-conditioned.")
```

```
        END IF
```

- Fall out of llvm’s switch to opaque pointers? How are we affected? Anyone working on this?
- Other issues?
    - https://github.com/flang-compiler/classic-flang-llvm-project/pull/112 requires Alok's review; Flang portion #1263 already merged

## Jun 15, 2022

Attendees
AMD: Shivaram, Chirag, Raghu
Arm: Kiran, Paul
Huawei: Bryan
Nvidia:
Others:

- Remaining from last week:
    - PR #1244 still waits for a second review
    - Issue #1253 (post PR #1212) seems to be taken care of; please don’t rush it, take as much time as you need to recognize the problem thoroughly
    - Issue #1249 we decided to just close it (Bryan, please do proceed)
- How far are we from LLVM14 integration? (PR #110 on classic-flang-llvm-project repo)
- Arm’s efforts towards upstreaming internal changes, reducing downstream delta and associated cost of the technical debt
    - E.g. PRs #1259, #1257, PR #659 on classic flang repo
        - PR #111 on classic-flang-llvm-project repo
    - A question to AMD: can PR’s #672 and #773 be reissued as one new PR (taking ownership effectively); we are using this code downstream for a couple of years and it passes in our CI
- CP2K and challenges it creates
    - Do others build it with libint (and -D__LIBINT flag)?
        - Dependencies we build: fftw (with Fortran module), OpenBLAS, ScaLAPACK, libint
    - Do others encounter issues with ScaLAPACK?
        - Results produced by pdpotrf subroutine seem incorrect
            - Could not prepare minimal reproducer, smaller codes and ScaLAPACK test cases seem unaffected
        - How do the others configure OpenBLAS (ScaLAPACK and CP2K dependency)?
            - INTERFACE64=1? BINARY=64? OpenMP vs pthread?

- Files that currently need modifications:
    - src/motion/space_groups.F - NORM2() function call
    - src/qs_kpp1_env_methods.F - NULL() function that takes a parameter
    - src/rpa_gw_ic.F - the ‘n’ variable needs to be privatized in the OpenMP parallel region
- Sharing NAG test failure list

## Jun 1, 2022

Attendees
AMD: Shivaram, Chirag, Raghu
Arm: Kiran, Pawel
Huawei: Bryan
Nvidia:
Others:

- Any important points to add to the agenda?
- How to conduct this call in future?
    - I (Kiran Arm) have been transitioning to work on llvm/flang. My daily work is mostly llvm/flang now.
    - It is the consortium of companies that is maintaining the project. The calls and maintenance of the community should ideally reflect that. Is rotation across the participating companies a possibility for conducting the calls and some other activities (responding to messages in slack, bugs etc)?
    - Rotation is fine with AMD (Shivaram). Pawel (Arm) is fine. Rotation is fine with Bryan (Huawei).
    - How to do this rotation? One company per month or rotate every call or something else?
      Once per month.
- Motion to merge PRs with single approval (will merge by next meeting if no update):
    - [Previous Weeks]
        - QuadFP runtime: large patch, taking time to review and reply to review comments. Smaller patches might introduce failures. Bryan will talk to team.
          Pawel : Switching off QuadFP is not precise. Failures in NAG tests. Slack discussion planned.
          Bryan : Remote team unavailable due to holidays. Will reply to all comments.
          Pawel : Using limits.h
          Bryan : Need to figure a standard type for all platforms/compilers
    - [This Week]
        - https://github.com/flang-compiler/flang/pull/1244

- Is the NAG test (zero length array with bindc) important for standard conformance? z2a

- Share NAG exclusion/failure list
  Check and update during next call.

- Regressions after downstream merge:
  https://github.com/flang-compiler/flang/issues/1253 : Afer merging implied do/array constructor fix (https://github.com/flang-compiler/flang/pull/1212).
  Huawei engineers/contractors will have a look and reply.

- Standardising coding style
  Bryan: Has config for clang-format (see below). Can check how big or small that would be for flang1 and flang2 (not runtime).

```
BasedOnStyle: LLVM
AlwaysBreakAfterDefinitionReturnType: TopLevel
BreakBeforeBraces: WebKit
SortIncludes: false
```

- Pawel (Arm) working on Tealeaf fixes

- Release_13 is now the default in classic-flang-llvm-project
  - We also need to have CI on release_13. **Not done**

- Release_14 work (Bryan)
  - PR open. Rich gave some comments and were addressed. Also changed workflows. Another PR in Flang reviewed and merged. There is already a 14 branch and will probably backport to 14.0.2 as part of this work itself.

- Release_15 work (Pawel)
  - Only one bug found, accepted and merged into flang. PIE is default with llvm-15. The test was failing due to this issue. SpecOmp2k12 uses a specific memory model and collides with PIE. If info about pIE was passed through driver to IR then it should have worked fine.
    Shivaram: ld was required for some benchmarks (lld failed). Arm uses gold.
  - Should wait for the llvm release to happen.

- ABI handling for MacOS port

- Quad-precision FP support**:**
  - Additional QuadFP support has been pushed to the following PR: https://github.com/flang-compiler/flang/pull/1215
  - Status after Pawel's comments?
  - **05/18:** Rebased, reformatted, addressed some review comments, analyzed some reported test failures and submitted PRs, still working on other suggestions (__BIGREAL_T vs. __REAL16_T, __float128).

- Debug PRs:
  - https://github.com/flang-compiler/flang/pull/1198 : Debug support for assumed shape array at higher optimizations (Arm CI failed)
  - https://github.com/flang-compiler/flang/pull/1201 : Use enhanced DIImportEntity to generate better IR for renamed modules (Arm CI failed)
- Implied do and array constructor bug(s)
  - https://github.com/flang-compiler/flang/issues/1200
    - Fix by Huawei: https://github.com/flang-compiler/flang/pull/1212 (merged)
  - https://github.com/flang-compiler/flang/issues/1227
    - New issue; AMD will take a look (updated: Huawei team has already taken a look)
    - https://github.com/flang-compiler/flang/pull/1232 (Pawel approved; merged)
- Windows
  - https://github.com/flang-compiler/flang/pull/1158 : Fix implicit function declaration warnings (approved but needs #1163 to be merged first)
  - https://github.com/flang-compiler/flang/pull/1163 : Clean up Windows-related macros in runtime/flang (no approval)
  - https://github.com/flang-compiler/flang/pull/1210 : Disable QuadFP on Windows (Pawel and Bryan approved; waiting for Isuru's approval)
  - https://github.com/flang-compiler/flang/pull/135 : Towards architectural neutrality (Bryan approved; waiting for xoviat's approval)
  - Platform-independent build scripts
    - https://github.com/flang-compiler/classic-flang-llvm-project/pull/73 (no approval)
    - https://github.com/flang-compiler/flang/pull/1185 (Michal approved)
- macOS
  - https://github.com/flang-compiler/flang/pull/1220 : libpgmath: add support for generic OSX builds (Bryan reviewed)
- Miscellaneous
  - https://github.com/flang-compiler/flang/pull/1032 : Test cases for #966 (need update from Shivaram)

## May 18, 2022

Attendees
AMD:
Arm:
Huawei:
Nvidia:
Others:

- Any important points to add to the agenda?
- How to conduct this call in future?

    - I (Kiran Arm) have been transitioning to work on llvm/flang. My daily work is mostly llvm/flang now. I am not sure whether I will be able to participate in the future, particularly later (from Aug) this year.
    - It is the consortium of companies that is maintaining the project.
    - The calls and maintenance of the community should ideally reflect that.
    - Is rotation across the participating companies a possibility for conducting the calls and some other activities (responding to messages in slack, bugs etc)?
    - I would like to spend some time to finish the documentation, windows and macos compilation support.
    - Rotation is fine with AMD (Shivaram). Pawel (Arm) is fine. Rotation is fine with Bryan (Huawei).
- Motion to merge PRs with single approval (will merge by next meeting if no update):
    - [Previous Weeks]
        - QuadFP runtime: large patch, taking time to review and reply to review comments. Smaller patches might introduce failures. Bryan will talk to team.
          Pawel : Switching off QuadFP is not precise. Failures in NAG tests. Slack discussion planned.
          Bryan : Remote team unavailable due to holidays. Will reply to all comments.
          Pawel : Using limits.h
          Bryan : Need to figure a standard type for all platforms/compilers
          Pawel: Should we relax requirements for coding style?
          Bryan: Has config for clang-format (see below). Can check how big or small that would be for flang1 and flang2 (not runtime).

```
BasedOnStyle: LLVM
AlwaysBreakAfterDefinitionReturnType: TopLevel
BreakBeforeBraces: WebKit
SortIncludes: false
```

    - [This Week]
        - https://github.com/flang-compiler/flang/pull/1244
        - https://github.com/flang-compiler/flang/pull/1250
        - https://github.com/flang-compiler/flang/pull/1101
        - https://github.com/flang-compiler/flang/pull/1062

- Pawel (Arm) working on Tealeaf fixes

- Release_13 is now the default in classic-flang-llvm-project
    - We also need to have CI on release_13. **Not done**

- Release_14 work (Bryan)
    - PR open. Rich gave some comments and were addressed. Also changed workflows. Another PR in Flang reviewed and merged. There is already a 14 branch and will probably backport to 14.0.2 as part of this work itself.
- Release_15 work (Pawel)

    - Only one bug found, accepted and merged into flang. PIE is default with llvm-15. The test was failing due to this issue. SpecOmp2k12 uses a specific memory model and collides with PIE. If info about pIE was passed through driver to IR then it should have worked fine.
      Shivaram: ld was required for some benchmarks (lld failed). Arm uses gold.
    - Should wait for the llvm release to happen.
- ABI handling for MacOS port
- Quad-precision FP support**:**
    - Additional QuadFP support has been pushed to the following PR: https://github.com/flang-compiler/flang/pull/1215
    - Status after Pawel's comments?
    - **05/18:** Rebased, reformatted, addressed some review comments, analyzed some reported test failures and submitted PRs, still working on other suggestions (__BIGREAL_T vs. __REAL16_T, __float128).
- Debug PRs:
    - https://github.com/flang-compiler/flang/pull/1233 : Common block variables in subroutines (Bryan, Kiran approved, merged)
    - https://github.com/flang-compiler/flang/pull/1198 : Debug support for assumed shape array at higher optimizations (Arm CI failed)
    - https://github.com/flang-compiler/flang/pull/1201 : Use enhanced DIImportEntity to generate better IR for renamed modules (Arm CI failed)
    - https://github.com/flang-compiler/flang/pull/1234 : Support debuginfo for deferred array in module (Pawel gave review comments)
- Implied do and array constructor bug(s)
    - https://github.com/flang-compiler/flang/issues/1200
        - Fix by Huawei: https://github.com/flang-compiler/flang/pull/1212 (merged)
    - https://github.com/flang-compiler/flang/issues/1227
        - New issue; AMD will take a look (updated: Huawei team has already taken a look)
        - https://github.com/flang-compiler/flang/pull/1232 (Pawel approved; merged)
- Windows
    - https://github.com/flang-compiler/flang/pull/1158 : Fix implicit function declaration warnings (approved but needs #1163 to be merged first)
    - https://github.com/flang-compiler/flang/pull/1163 : Clean up Windows-related macros in runtime/flang (no approval)
    - https://github.com/flang-compiler/flang/pull/1210 : Disable QuadFP on Windows (Pawel and Bryan approved; waiting for Isuru's approval)
    - https://github.com/flang-compiler/flang/pull/135 : Towards architectural neutrality (Bryan approved; waiting for xoviat's approval)
    - Platform-independent build scripts
        - https://github.com/flang-compiler/classic-flang-llvm-project/pull/73 (no approval)
        - https://github.com/flang-compiler/flang/pull/1185 (Michal approved)
- macOS
    - https://github.com/flang-compiler/flang/pull/1220 : libpgmath: add support for generic OSX builds (Bryan reviewed)

- Miscellaneous
  - https://github.com/flang-compiler/flang/pull/1032 : Test cases for #966 (need update from Shivaram)

## May 4, 2022

Attendees
AMD: Shivaram, Kiran Kumar, Bhuvan
Arm: Kiran, Pawel
Huawei: Bryan
Nvidia:
Others: Marcus (Apple Port)

- Any important points to add to the agenda?
- Motion to merge PRs with single approval (will merge by next meeting if no update):
  - [Previous Weeks]
    - Fix lowering of move_alloc (Approved and merged)
      - https://github.com/flang-compiler/flang/pull/1209: Requested review. https://github.com/flang-compiler/flang/pull/1209
    - QuadFP runtime: large patch, taking time to review and reply to review comments. Smaller patches might introduce failures. Bryan will talk to team.
      Pawel : Switching off QuadFP is not precise. Failures in NAG tests. Slack discussion planned.
      Bryan : Remote team unavailable due to holidays. Will reply to all comments.
      Pawel : Using limits.h
      Bryan : Need to figure a standard type for all platforms/compilers
      Pawel: Should we relax requirements for coding style?
      Bryan: Has config for clang-format (see below). Can check how big or small that would be for flang1 and flang2 (not runtime).
      ```
      BasedOnStyle: LLVM
      AlwaysBreakAfterDefinitionReturnType: TopLevel
      BreakBeforeBraces: WebKit
      SortIncludes: false
      ```
  - [This Week]
    - 
- Release_13 is now the default in classic-flang-llvm-project
  - We also need to have CI on release_13. **Not done**
  - Backport stable branch patches to classic-flang-llvm-project
    - We will waive the review. @bryanpkc will merge this. Will add tags and cherry-pick changes.
    - Update 04/20: Tagged flang-12.0.0-20220325 and flang-13.0.0-20220325; replaced release_12x and release_13x with the two branches.
- Release_14 work (Bryan)

    - PR open. Rich gave some comments and were addressed. Also changed workflows. Another PR in Flang reviewed and merged. There is already a 14 branch and will probably backport to 14.0.2 as part of this work itself.
- Release_15 work (Pawel)
    - Only one bug found, accepted and merged into flang. PIE is default with llvm-15. The test was failing due to this issue. SpecOmp2k12 uses a specific memory model and collides with PIE. If info about pIE was passed through driver to IR then it should have worked fine.
    - Should wait for the llvm release to happen.
- ABI handling question in slack
    - **meow464** 🥕 1:50 AM
    - With respect to the Apple Silicon port: given that I need to change how runtime functions (implemented in C) are called, would it make sense for the changes to be to ILM or ILI instead of the LLVM ABI lowering step?
    - I'm officially out of ideas on how to make flang put all var_args on the stack on apple silicon. `ll_abi_classify_va_arg_dtype` is never called for any architecture, I can't find where it should probably be called and I can't find anywhere else I could plug it.
- Quad-precision FP support**:**
    - Additional QuadFP support has been pushed to the following PR: https://github.com/flang-compiler/flang/pull/1215
    - Status after Pawel's comments?
- Debug PRs:
    - https://github.com/flang-compiler/flang/pull/1233 : Common block variables in subroutines (Bryan, Kiran approved, can be merged)
    - https://github.com/flang-compiler/flang/pull/1198 : Debug support for assumed shape array at higher optimizations (Arm CI failed)
    - https://github.com/flang-compiler/flang/pull/1201 : Use enhanced DIImportEntity to generate better IR for renamed modules (Arm CI failed)
    - https://github.com/flang-compiler/flang/pull/1234 : Support debuginfo for deferred array in module (Pawel gave review comments)
- Fix lowering of move_alloc by Pawel Osmialowski
    - https://github.com/flang-compiler/flang/pull/1209 (no approval, comments from Kiran and Bryan. Bryan wanted to have a deeper look.)
    - Member of struct/derived type used in move_alloc
- Implied do and array constructor bug(s)
    - https://github.com/flang-compiler/flang/issues/1200
        - Fix by Huawei: https://github.com/flang-compiler/flang/pull/1212 (merged)
    - https://github.com/flang-compiler/flang/issues/1227
        - New issue; AMD will take a look (updated: Huawei team has already taken a look)
        - https://github.com/flang-compiler/flang/pull/1232 (Pawel approved)
- Windows

        - https://github.com/flang-compiler/flang/pull/1158 : Fix implicit function declaration warnings (approved but needs #1163 to be merged first)
        - https://github.com/flang-compiler/flang/pull/1163 : Clean up Windows-related macros in runtime/flang (no approval)
        - https://github.com/flang-compiler/flang/pull/1210 : Disable QuadFP on Windows (Pawel and Bryan approved; waiting for Isuru's approval)
        - https://github.com/flang-compiler/flang/pull/135 : Towards architectural neutrality (Bryan approved; waiting for xoviat's approval)
        - Platform-independent build scripts
            - https://github.com/flang-compiler/classic-flang-llvm-project/pull/73 (no approval)
            - https://github.com/flang-compiler/flang/pull/1185 (Michal approved)
- macOS
    - https://github.com/flang-compiler/flang/pull/1220 : libpgmath: add support for generic OSX builds (Bryan reviewed)
- Miscellaneous
    - https://github.com/flang-compiler/flang/pull/1032 : Test cases for #966 (need update from Shivaram)

## Apr 20, 2022

Attendees
AMD:
Arm:
Huawei:
Nvidia:
Others:

- Any important points to add to the agenda?
- Release_13 is now the default in classic-flang-llvm-project
    - We also need to have CI on release_13. Not done
    - Backport stable branch patches to classic-flang-llvm-project
        - https://github.com/flang-compiler/classic-flang-llvm-project/pull/96 : release_12x
        - https://github.com/flang-compiler/classic-flang-llvm-project/pull/97 : release_13x
        - We will waive the review. @bryanpkc will merge this. Will add tags and cherry-pick changes.
        - Update 04/20: Tagged flang-12.0.0-20220325 and flang-13.0.0-20220325; replaced release_12x and release_13x with the two branches.
- Bryan updated Pull-Request review process. **https://github.com/flang-compiler/flang/wiki/Community**
    - **shivarama.rao@amd.com to update the AMD reviewer list**
- Motion to merge PRs with single approval (will merge by next meeting if no update):
    - [Previous Weeks]

        - Fix lowering of move_alloc
            - https://github.com/flang-compiler/flang/pull/1209: Requested review.
        - QuadFP runtime: large patch, taking time to review and reply to review comments. Smaller patches might introduce failures. Bryan will talk to team.
          Pawel: Should we relax requirements for coding style?
          Bryan: Has config for clang-format (see below). Can check how big or small that would be for flang1 and flang2 (not runtime).

```
BasedOnStyle: LLVM
AlwaysBreakAfterDefinitionReturnType: TopLevel
BreakBeforeBraces: WebKit
SortIncludes: false
```

    - [This Week]
        -
- ABI handling question in slack
    - **meow464**🥕 1:50 AM
    - With respect to the Apple Silicon port: given that I need to change how runtime functions (implemented in C) are called, would it make sense for the changes to be to ILM or ILI instead of the LLVM ABI lowering step?
    - I'm officially out of ideas on how to make flang put all var_args on the stack on apple silicon. `ll_abi_classify_va_arg_dtype` is never called for any architecture, I can't find where it should probably be called and I can't find anywhere else I could plug it.
    -
- What is the plan for math function vectorisation in classic-flang and classic-flang-llvm-project?
    - Are the vec-lib tables sufficient for purpose?
    - llvm/lib/Analysis/TargetLibraryInfo.cpp, llvm/include/llvm/Analysis/VecFuncs.def
- Volunteers for moving to new version of LLVM?
    - Bryan has plans for LLVM 14.
    - Arm needs 15 port.
- Quad-precision FP support**:**
    - Additional QuadFP support has been pushed to the following PR: https://github.com/flang-compiler/flang/pull/1215
    - Huawei going through the list of tests that were provided by Pawel. Working on two fixes. Try to have summary for next time.
    - Not trivial to switch off Quad FP.
    - Pawel has provided some comments.
- Debug PRs:

        - https://github.com/flang-compiler/flang/pull/1233 : Common block variables in subroutines (Bryan, Kiran approved, can be merged)
        - https://github.com/flang-compiler/flang/pull/1198 : Debug support for assumed shape array at higher optimizations (CI failed)
        - https://github.com/flang-compiler/flang/pull/1201 : Use enhanced DIImportEntity to generate better IR for renamed modules (No reviewers)
- Fix lowering of move_alloc by Pawel Osmialowski
    - https://github.com/flang-compiler/flang/pull/1209 (no approval, comments from Kiran and Bryan. Bryan wanted to have a deeper look.)
    - Member of struct/derived type used in move_alloc
- Implied do and array constructor bug(s)
    - https://github.com/flang-compiler/flang/issues/1200
        - Fix by Huawei: https://github.com/flang-compiler/flang/pull/1212 (merged)
    - https://github.com/flang-compiler/flang/issues/1227
        - New issue; AMD will take a look (updated: Huawei team has already taken a look)
        - https://github.com/flang-compiler/flang/pull/1232 (Pawel approved)
- Windows
    - https://github.com/flang-compiler/flang/pull/1158 : Fix implicit function declaration warnings (approved but needs #1163 to be merged first)
    - https://github.com/flang-compiler/flang/pull/1163 : Clean up Windows-related macros in runtime/flang (no approval)
    - https://github.com/flang-compiler/flang/pull/1210 : Disable QuadFP on Windows (Pawel and Bryan approved; waiting for Isuru's approval)
    - https://github.com/flang-compiler/flang/pull/135 : Towards architectural neutrality (Bryan approved; waiting for xoviat's approval)
    - Platform-independent build scripts
        - https://github.com/flang-compiler/classic-flang-llvm-project/pull/73 (no approval)
        - https://github.com/flang-compiler/flang/pull/1185 (Michal approved)
- macOS
    - https://github.com/flang-compiler/flang/pull/1220 : libpgmath: add support for generic OSX builds (Bryan reviewed)
- Miscellaneous
    - https://github.com/flang-compiler/flang/pull/1032 : Test cases for #966 (need update from Shivaram)

## Apr 6, 2022

Attendees
AMD:
Arm:
Huawei:
Nvidia:
Others:

- Any important points to add to the agenda?
    - Release_13 not the default.
    - Merges to Release_12 but not to Release_13. Arm is odd number release basis. AMD is for every release. Huawei is even release. AMD is OK with maintaining on odd release, Huawei is also OK. Change the default branch to release_13: Kiran. We also need to have CI on release_13.
- Windows Flang update (Niyas)
- Bryan updated Pull-Request review process. **https://github.com/flang-compiler/flang/wiki/Community**
    - **shivarama.rao@amd.comto update the AMD reviewer list**
- Motion to merge PRs with single approval (will merge by next meeting if no update):
    - [Previous Weeks] Bug fixes for torture test failures from Huawei/HNC:
        - https://github.com/flang-compiler/flang/pull/1053 (Mark, Bryan, Kiran approved, Waiting for Shivaram. Shivaram will have a look).
        - Fix lowering of move_alloc
            - https://github.com/flang-compiler/flang/pull/1209: Requested review. Kiran Chandramohan to check which application is this from. Internal test-suite.
    - [This Week]
        - 
- Test system revamp proposal
    - f90_correct/mp_correct tests have extremely overweight boilerplate. We want to rewrite a few examples in the new way (more like llvm_ir_direct tests), then mandate that all newly added run-time tests be written in the new way. Conversion of the old tests can happen more gradually.
    - Chirag wrote some run-time tests in the new way: provided below
        - Example: RUN: %flang %s -o /tmp/xyz && /tmp/xyz | xargs | FileCheck %s || rm -f /tmp/xyz
    - Encourage people to use the new style of testing. Lot of engineering work to move existing tests.
- Quad-precision FP support**:**
    - Additional QuadFP support has been pushed to the following PR: https://github.com/flang-compiler/flang/pull/1215
    - shivarama.rao@amd.com has promised to look into it by next meeting. Arm will try next week.
    - Huawei going through the list of tests that were provided by Pawel. Working on two fixes. Try to have summary for next time.
    - Not trivial to switch off Quad FP.
- Remove diffs in llvm debuginfo related to GlobalVariables
    - AMD engineers had a look and have https://github.com/flang-compiler/flang/pull/1228 : Stop generating artificial DebugInfo for common block (Failing CI)
- Debug PRs:
    - https://github.com/flang-compiler/flang/pull/1233 : Common block variables in subroutines (Bryan, Kiran approved, can be merged)

    - https://github.com/flang-compiler/flang/pull/1198 : Debug support for assumed shape array at higher optimizations (CI failed)
    - https://github.com/flang-compiler/flang/pull/1228 : Stop generating artificial DebugInfo for common block
    - https://github.com/flang-compiler/flang/pull/1201 : Use enhanced DIImportEntity to generate better IR for renamed modules (No reviewers)
- Fix lowering of move_alloc by Pawel Osmialowski
    - https://github.com/flang-compiler/flang/pull/1209 (no approval)
    - Member of struct/derived type used in move_alloc
- Implied do and array constructor bug(s)
    - https://github.com/flang-compiler/flang/issues/1200
        - Fix by Huawei: https://github.com/flang-compiler/flang/pull/1212 (merged)
    - https://github.com/flang-compiler/flang/issues/1227
        - New issue; AMD will take a look (updated: Huawei team has already taken a look)
        - https://github.com/flang-compiler/flang/pull/1232 (Pawel approved)
- Backport stable branch patches to classic-flang-llvm-project
    - https://github.com/flang-compiler/classic-flang-llvm-project/pull/96 : release_12x
    - https://github.com/flang-compiler/classic-flang-llvm-project/pull/97 : release_13x
    - We will wave the review. @bryanpkc will merge this. Will add tags and cherry-pick changes.
- Windows
    - https://github.com/flang-compiler/flang/pull/1158 : Fix implicit function declaration warnings (approved but needs #1163 to be merged first)
    - https://github.com/flang-compiler/flang/pull/1163 : Clean up Windows-related macros in runtime/flang (no approval)
    - https://github.com/flang-compiler/flang/pull/1210 : Disable QuadFP on Windows (Pawel and Bryan approved; waiting for Isuru's approval)
    - https://github.com/flang-compiler/flang/pull/135 : Towards architectural neutrality (Bryan approved; waiting for xoviat's approval)
    - Platform-independent build scripts
        - https://github.com/flang-compiler/classic-flang-llvm-project/pull/73 (no approval)
        - https://github.com/flang-compiler/flang/pull/1185 (Michal approved)
- macOS
    - https://github.com/flang-compiler/flang/pull/1220 : libpgmath: add support for generic OSX builds (Bryan reviewed)
- Miscellaneous
    - https://github.com/flang-compiler/flang/pull/1032 : Test cases for #966 (need update from Shivaram)

## Mar 23, 2022

Attendees

AMD:

Arm:

Huawei:
Nvidia:
Others:

- The review process got blocked too easily. Maybe it's time to change some of our policies?
  - **Consensus from last time:** Two approvals are required; but patch author can move for acceptance with one approval after two weeks of inactivity. Gatekeeper from patch author's organization is responsible for merging. When appropriate, reviewers should remove themselves from the list of reviewers, or delegate to someone more knowledgeable, to prevent stalling the process.
  - **Shall we add the above to the wiki page? Needs an edit anyway, https://github.com/flang-compiler/flang/wiki/Community**
  - **Bryan to update the PR process in wiki and Shivaram to update the AMD reviewers list.**
- Motion to merge PRs with single approval (will merge by next meeting if no update):
  - [Previous Week] Bug fixes for torture test failures from Huawei/HNC:
    - https://github.com/flang-compiler/flang/pull/1053 (Mark, Bryan approved, Kiran has comments) : Will wait to address review comments.
  - [This Week]
    - Vectorize_width: https://github.com/flang-compiler/flang/pull/1225 (Rich, Pawel, AMD approved) : Mobica will update this PR addressing review comments. (Note: they are leaving next month)
    - Allow /dev/null to be an input file:
      - https://github.com/flang-compiler/classic-flang-llvm-project/pull/94 (All approved, waiting for windows check) : Will wait for one more day and merge if no response from windows reviewers.
    - Fix lowering of move_alloc
      - https://github.com/flang-compiler/flang/pull/1209: Requested review. Kiran Chandramohan to check which application is this from. Internal test-suite.
- Test system revamp proposal
  - f90_correct/mp_correct tests have extremely overweight boilerplate. We want to rewrite a few examples in the new way (more like llvm_ir_direct tests), then mandate that all newly added run-time tests be written in the new way. Conversion of the old tests can happen more gradually.
  - Chirag wrote some run-time tests in the new way: provided below
    - Example: RUN: %flang %s -o /tmp/xyz && /tmp/xyz | xargs | FileCheck %s || rm -f /tmp/xyz
- **Quad-precision FP support:**
  - **Bug fix for OpenMPI build waiting has been merged: https://github.com/flang-compiler/flang/pull/1222**
  - **Arm's internal tests are failing due to incomplete support for QuadFP; Huawei is verifying those tests internally and making sure they will pass with the complete patch set. (looking at NAG)**

    - **Additional QuadFP support has been pushed to the following PR: https://github.com/flang-compiler/flang/pull/1215**
    - **shivarama.rao@amd.com has promised to look into it by next meeting. Arm will try.**
- Remove diffs in llvm debuginfo related to GlobalVariables
    - AMD engineers had a look and have https://github.com/flang-compiler/flang/pull/1228 : Stop generating artificial DebugInfo for common block (Failing CI)
- Debug PRs:
    - https://github.com/flang-compiler/flang/pull/1233 (in place of https://github.com/flang-compiler/flang/pull/1169) Common block variables in subroutines (Bryan has comments)
    - https://github.com/flang-compiler/flang/pull/1198 : Debug support for assumed shape array at higher optimizations (CI failed)
    - https://github.com/flang-compiler/flang/pull/1228 : Stop generating artificial DebugInfo for common block (Failing CI)
    - https://github.com/flang-compiler/flang/pull/1201 : Use enhanced DIImportEntity to generate better IR for renamed modules (CI failed)
- Implement vectorize_width directive
    - https://github.com/flang-compiler/flang/pull/1225 (Rich approved; Huawei/AMD?)
- Fix lowering of move_alloc by Pawel Osmialowski
    - https://github.com/flang-compiler/flang/pull/1209 (no approval)
    - Member of struct/derived type used in move_alloc
- Implied do and array constructor bug(s)
    - https://github.com/flang-compiler/flang/issues/1200
        - Fix by Huawei: https://github.com/flang-compiler/flang/pull/1212 (merged)
    - https://github.com/flang-compiler/flang/issues/1227
        - New issue; AMD will take a look (updated: Huawei team has already taken a look)
        - https://github.com/flang-compiler/flang/pull/1232 (Pawel approved)
- Backport stable branch patches to classic-flang-llvm-project
    - https://github.com/flang-compiler/classic-flang-llvm-project/pull/96 : release_12x
    - https://github.com/flang-compiler/classic-flang-llvm-project/pull/97 : release_13x
    - We will wave the review. @bryanpkc will merge this. Will add tags and cherry-pick changes.
- Windows
    - https://github.com/flang-compiler/flang/pull/1158 : Fix implicit function declaration warnings (approved but needs #1163 to be merged first)
    - https://github.com/flang-compiler/flang/pull/1163 : Clean up Windows-related macros in runtime/flang (no approval)
    - https://github.com/flang-compiler/flang/pull/1210 : Disable QuadFP on Windows (Pawel and Bryan approved; waiting for Isuru's approval)
    - https://github.com/flang-compiler/flang/pull/135 : Towards architectural neutrality (Bryan approved; waiting for xoviat's approval)
    - Platform-independent build scripts
        - https://github.com/flang-compiler/classic-flang-llvm-project/pull/73 (no approval)

        - https://github.com/flang-compiler/flang/pull/1185 (Michal approved)
    - Niyas (Linaro) using Classic Flang on Windows on Arm. Can help with CI/CD for windows on Arm.
- macOS
    - https://github.com/flang-compiler/flang/pull/1220 : libpgmath: add support for generic OSX builds (Bryan reviewed)
- Miscellaneous
    - https://github.com/flang-compiler/flang/pull/1032 : Test cases for #966 (need update from Shivaram)

## Mar 9, 2022

Attendees
AMD: Alok, Bhuvan, Chirag, Jini, KiranTP, Raghu, Shivaram
Arm:
Huawei: Bryan
Nvidia:
Others: Przemek

- The review process got blocked too easily. Maybe it's time to change some of our policies?
    - **Consensus from last time:** Two approvals are required; but patch author can move for acceptance with one approval after two weeks of inactivity. Gatekeeper from patch author's organization is responsible for merging. When appropriate, reviewers should remove themselves from the list of reviewers, or delegate to someone more knowledgeable, to prevent stalling the process.
- Motion to merge PRs with single approval (will merge by next meeting if no update):
    - Bug fixes for torture test failures from Huawei/HNC:
        - https://github.com/flang-compiler/flang/pull/1055 (Pawel approved)
        - https://github.com/flang-compiler/flang/pull/1053 (Mark approved)
- Test system revamp proposal
    - f90_correct/mp_correct tests have extremely overweight boilerplate. We want to rewrite a few examples in the new way (more like llvm_ir_direct tests), then mandate that all newly added run-time tests be written in the new way. Conversion of the old tests can happen more gradually.
    - Chirag wrote some run-time tests in the new way: provided below
        - Example: RUN: %flang %s -o /tmp/xyz && /tmp/xyz | xargs | FileCheck %s || rm -f /tmp/xyz
- Quad-precision FP support:
    - Bug fix for OpenMPI build waiting for AMD approval: https://github.com/flang-compiler/flang/pull/1222
    - Arm's internal tests are failing due to incomplete support for QuadFP; Huawei is verifying those tests internally and making sure they will pass with the complete patch set.

    - Additional QuadFP support has been pushed to the following PR: https://github.com/flang-compiler/flang/pull/1215
- Debug PRs:
    - https://github.com/flang-compiler/flang/pull/1157 : Deferred length string type (Bryan/Pawel approved; two formatting remarks remain to be addressed)
    - https://github.com/flang-compiler/flang/pull/1169 : Common block variables in subroutines (Still needs a rebase due to merge conflicts, no approval)
    - https://github.com/flang-compiler/flang/pull/1198 : Debug support for assumed shape array at higher optimizations (CI failed)
    - https://github.com/flang-compiler/flang/pull/1228 : Stop generating artificial DebugInfo for common block (Failing CI)
    - https://github.com/flang-compiler/flang/pull/1231 : Corrected DebugInfo for modules; refresh of #898 (Failing CI)
    - https://github.com/flang-compiler/flang/pull/1201 : Use enhanced DIImportEntity to generate better IR for renamed modules (CI failed)
    - https://github.com/flang-compiler/classic-flang-llvm-project/pull/93 Backport: Enhance DIImportedEntity to accept children entities (Pawel and Bryan approved)
    - https://github.com/flang-compiler/classic-flang-llvm-project/pull/76 Same thing for dev_11x (Bryan approved; macOS CI failing)
    - https://github.com/flang-compiler/classic-flang-llvm-project/pull/75 Backport: Upgrade DISubrange::count to accept DIExpression also (Bryan approved; macOS CI failing)
        - Bryan submitted a macOS CI fix for release_11x https://github.com/flang-compiler/classic-flang-llvm-project/pull/95 (Merged)
    - https://github.com/flang-compiler/flang/pull/1141 : Procedure pointer (Merged)
    - https://github.com/flang-compiler/flang/pull/1147 : Assumed length string type (Merged)
- Emit vscale_range attribute when sve target option is set.
    - https://github.com/flang-compiler/classic-flang-llvm-project/pull/86 (Merged)
    - https://github.com/flang-compiler/flang/pull/1214 (Merged)
- Implement vectorize_width directive
    - https://github.com/flang-compiler/flang/pull/1225 (Rich approved; Huawei/AMD?)
- Fix lowering of move_alloc by Pawel Osmialowski
    - https://github.com/flang-compiler/flang/pull/1209 (no approval)
    - Member of struct/derived type used in move_alloc
- Implied do and array constructor bug(s)
    - https://github.com/flang-compiler/flang/issues/1200
        - Fix by Huawei: https://github.com/flang-compiler/flang/pull/1212 (merged)
    - https://github.com/flang-compiler/flang/issues/1227
        - New issue; AMD will take a look (updated: Huawei team has already taken a look)
- Adding support for -dM (display all Macros):
    - https://github.com/flang-compiler/classic-flang-llvm-project/pull/90 (Bryan approved; Pawel requested a change which has been made; AMD?)
    - https://github.com/flang-compiler/flang/pull/1226 (Merged)
- Allow /dev/null to be an input file:

    - https://github.com/flang-compiler/classic-flang-llvm-project/pull/94 (Pawel approved)
- Windows
    - https://github.com/flang-compiler/flang/pull/1158 : Fix implicit function declaration warnings (approved but needs #1163 to be merged first)
    - https://github.com/flang-compiler/flang/pull/1163 : Clean up Windows-related macros in runtime/flang (no approval)
    - https://github.com/flang-compiler/flang/pull/1210 : Disable QuadFP on Windows (Pawel and Bryan approved; waiting for Isuru's approval)
    - https://github.com/flang-compiler/flang/pull/135 : Towards architectural neutrality (Bryan approved; waiting for xoviat's approval)
    - Platform-independent build scripts
        - https://github.com/flang-compiler/classic-flang-llvm-project/pull/73 (no approval)
        - https://github.com/flang-compiler/flang/pull/1185 (Michal approved)
    - Niyas (Linaro) using Classic Flang on Windows on Arm. Can help with CI/CD for windows on Arm.
- macOS
    - https://github.com/flang-compiler/flang/pull/1220 : libpgmath: add support for generic OSX builds (Bryan reviewed)
- Miscellaneous
    - https://github.com/flang-compiler/flang/pull/1032 : Test cases for #966 (need update from Shivaram)

## Feb 23, 2022

Attendees
AMD: Shivaram
Arm: Pawel
Huawei: Bryan
Nvidia:
Others: Przemek

- Kiran and Pawel are away for the next couple of calls. Pawel has agreed to host on the 23rd of February.
    - Bryan has kindly agreed to host the one on March 9th.
- Huawei has patches for quad-precision.
    - Current state of classic flang prevents it from building OpenMPI, this needs to be addressed somehow
        - Revert current changes and re-apply complete quad-precision implementation?
- The review process got blocked too easily. Maybe it's time to change some of our policies?
    - Single approval should suffice, but the reviewer takes full responsibility and must participate in corrective actions (e.g. review a commit) if submitted PR causes disturbance.
    - If an assigned reviewer does not recognize themselves as an expert in an area (e.g. Windows, MacOS or DebugInfo), or is at that particular moment too busy to do the review, they should remove their assignment (eventually assigning someone else); this is to prevent stalling of the review process.
- Debug PRs
    - https://github.com/flang-compiler/flang/pull/1147 : Assumed length string type (Bryan and Pawel approved)
    - https://github.com/flang-compiler/flang/pull/1157 : Deferred length string type (Bryan approved, Pawel approved with small formatting remark)
    - https://github.com/flang-compiler/flang/pull/1169 : Commonblock in subroutines (Still needs a rebase due to merge conflicts, no approval)
    - https://github.com/flang-compiler/flang/pull/1141 : Procedure pointer (Merged)

- Addition of the -emit-flang-llvm option (PR #88 on classic-flang-llvm-project repo), this is particularly useful feature, still waiting for reviewers to look at it
- Adding support for -dM (display all Macros). Any update?
- Vscale_range attribute (alternative implementation completed). Emit vscale_range when sve target option is set.

Reimplemented by Pawel. Includes CI changes as well to test the flang driver. A separate CI only patch exists (PR #87, added for a reference only; will be closed after #86 is merged)
https://github.com/flang-compiler/classic-flang-llvm-project/pull/86
https://github.com/flang-compiler/flang/pull/1214

- Fix lowering of move_alloc by Pawel Osmialowski
    - https://github.com/flang-compiler/flang/pull/1209
    - Member of struct/derived type used in move_alloc

- Implied do and array constructor bug.
    - https://github.com/flang-compiler/flang/issues/1200
    - Fix by Huawei: https://github.com/flang-compiler/flang/pull/1212

- Windows
    - Windows PR list. https://github.com/flang-compiler/flang/issues/1154. Only one more required to get Windows to compile.
      -> static libs and export patches. -> CMake change to allow windows required.
      -> Adam can share the build recipe
      -> 320 : Looking into this. Colleagues working on tests. Shell scripts not Running on windows.
    - PR to enable the Windows build in CMakefiles here:
      https://github.com/flang-compiler/flang/pull/1184
      https://github.com/flang-compiler/flang/pull/1181
    - We did a python build script to supports the Flang build on Windows from the scratch: https://github.com/flang-compiler/flang/pull/1185
    - Niyas (Linaro) using Classic Flang on Windows on Arm. Can help with CI/CD for windows on Arm.

## Feb 9, 2022

Attendees
AMD: Shivaram
Arm: Kiran, Pawel
Huawei: Bryan
Nvidia:
Others: Przemek

flang-compiler/flang

- I am away for the next couple of calls. Pawel has agreed to host on the 23rd of February. We would need someone else to host on March 9th. Volunteers?
    - Bryan will host on March 9th (double check the date).

- Anything important to discuss today? Any plans to share regarding flang work upstream from companies. Also what are the patches that need review soon?
    - Arm : Has release work going on. Bit busy. MacOS M1 work, patches : generic pgmath build, architectural neutrality. Pull request containing new flang tests (vscale_range attributes patch), has changes from others, removed not needed tests, patch for va_list. Reducing diff with upstream.
    - Huawei: Implied do fix works. Pawel reported issues (OpenMPI, Nag) after merging quad1 precision support. It is the splitting of patches that caused these issues. Other bug fixes not reviewed yet. In future would like some changes we can try both flang and classic flang.
    - AMD: Continue to upstream debug. F2008 features. Offloading difficult to upstream this year due to dependency on Clang, Changes are being upstreamed to Clang now.

- Differences in llvm debug, upstream and classic flang llvm project. AMD engineers will have a look. Bryan thinks that we already emit names for common blocks so this change might not be needed. https://github.com/flang-compiler/classic-flang-llvm-project/issues/2

- Alok and Bhuvan (AMD) gave a talk at FOSDEM LLVM devroom on improvements to llvm debug for Fortran. https://fosdem.org/2022/schedule/event/llvm_fortran_debug/

- Debug PRs
    - https://github.com/flang-compiler/flang/pull/1147 : Assumed length string type (bryan approved)
    - https://github.com/flang-compiler/flang/pull/1157 : Deferred length string type (bryan approved)

- https://github.com/flang-compiler/flang/pull/1169 : Commonblock in subroutines (Needs a rebase, no approval)
- https://github.com/flang-compiler/flang/pull/1141 : Procedure pointer (Merged)

- Adding support for -dM (display all Macros). Currently not working. Gfortran supports these Macros. Currently lists Macros from Clang.

- Vscale_range attribute (?). Emit vscale_range when sve target option is set. Reimplemented by Pawel. Includes CI changes as well to test the flang driver. A separate CI only patch exists.
https://github.com/flang-compiler/classic-flang-llvm-project/pull/86
https://github.com/flang-compiler/flang/pull/1214

- GCC 11 : CI changes
  - https://github.com/flang-compiler/classic-flang-llvm-project/pull/79
  - Also #80 and #81
  - Status: Finished working on this and waiting for review.

- New directive: vectorize_width (https://github.com/flang-compiler/flang/pull/1172)
  - What is the status?
  - Resolving comments from Rich.
  - Waiting for further review

- Fix lowering of move_alloc by Pawel Osmialowski
  - https://github.com/flang-compiler/flang/pull/1209
  - Member of struct/derived type used in move_alloc

- Huawei has patches for quad-precision.
  - New patch for quad-precision I/O support https://github.com/flang-compiler/flang/pull/1215
  - Pawel saw some regressions after https://github.com/flang-compiler/flang/pull/1129 , A new fix was raised in https://github.com/flang-compiler/flang/pull/1222. Seems to be a couple more issues.

- Implied do and array constructor bug.
  - https://github.com/flang-compiler/flang/issues/1200
  - Fix by Huawei: https://github.com/flang-compiler/flang/pull/1212

- Windows
  - Windows PR list. https://github.com/flang-compiler/flang/issues/1154. Only one more required to get Windows to compile.

-> static libs and export patches. -> CMake change to allow windows required.
-> Adam can share the build recipe
-> 320 : Looking into this. Colleagues working on tests. Shell scripts not Running on windows.
    - PR to enable the Windows build in CMakefiles here:
      https://github.com/flang-compiler/flang/pull/1184
      https://github.com/flang-compiler/flang/pull/1181
    - We did a python build script to supports the Flang build on Windows from the scratch: https://github.com/flang-compiler/flang/pull/1185
    - Niyas (Linaro) using Classic Flang on Windows on Arm. Can help with CI/CD for windows on Arm.

- Would like to send a status mail. Please add any topics that we should cover https://docs.google.com/document/d/1n-TpNzNbAmHZIP4ziFykyH-GeGe2xP7yd8KDnDvzJsA/edit?usp=sharing
    - No progress here. I hope to spend some time on this while I am away.

- Bryan created several PRs for fixing warnings with llvm-12. There are possibly a few more PRs.

## Jan 26, 2022

Attendees
AMD: Shivaram, Jini, Bhuvan, Raghu
Arm: Pawel, Kiran
Huawei:
Nvidia:
Others:

flang-compiler/flang

- CI failing
    - Only failing for release_11. X86 github actions. Removing release_11 worked. Huawei using 12 and AMD using 13. OK with everyone. We have 13 release branch, and CI in classic-flang-llvm-project but no release 13 CI in flang.
    - Przemek to remove 11 and later add 13.

- Would like to send a status mail. Please add any topics that we should cover https://docs.google.com/document/d/1n-TpNzNbAmHZlP4ziFykyH-GeGe2xP7yd8KDnDvzJsA/edit?usp=sharing
  - No progress here

- Bryan created several PRs for fixing warnings with llvm-12. There are possibly a few more PRs.

- llvm-13 : Bryan mentioned that some string debug changes were removed. AMD engineers confirmed it is OK to remove.

- Vscale_range attribute (?). Emit vscale_range when sve target option is set. Reimplemented by Pawel. Includes CI changes as well to test the flang driver. A separate CI only patch exists.
https://github.com/flang-compiler/classic-flang-llvm-project/pull/86
https://github.com/flang-compiler/flang/pull/1214

- Huawei has patches for quad-precision. New patch for quad-precision I/O support https://github.com/flang-compiler/flang/pull/1215

- GCC 11 : CI changes
  - https://github.com/flang-compiler/classic-flang-llvm-project/pull/79
  - Also #80 and #81
  - Status: Finished working on this and waiting for review.

- CP2K DILocation fix: https://github.com/flang-compiler/flang/pull/1188 (Bryan approved, AMD approved)

- New directive: vectorize_width (https://github.com/flang-compiler/flang/pull/1172)
  - What is the status?
  - Resolving comments from Rich.

- Implied do and array constructor bug.
  - https://github.com/flang-compiler/flang/issues/1200
  - Fix by Huawei: https://github.com/flang-compiler/flang/pull/1212

- Fix lowering of move_alloc by Pawel Osmialowski
  - https://github.com/flang-compiler/flang/pull/1209
  - Member of struct/derived type used in move_alloc

- Windows
  - Windows PR list. https://github.com/flang-compiler/flang/issues/1154. Only one more required to get Windows to compile.

    -> static libs and export patches. -> CMake change to allow windows required.
    -> Adam can share the build recipe
    -> 320 : Looking into this. Colleagues working on tests. Shell scripts not Running on windows.
  - PR to enable the Windows build in CMakefiles here:
    https://github.com/flang-compiler/flang/pull/1184
    https://github.com/flang-compiler/flang/pull/1181
  - We did a python build script to supports the Flang build on Windows from the scratch: https://github.com/flang-compiler/flang/pull/1185
  - Niyas (Linaro) using Classic Flang on Windows on Arm. Can help with CI/CD for windows on Arm.
- Debug PRs
  - 1147, 1157, 1169, 1141 (procedure pointer)

- Status of PRs
  AMD:
  https://github.com/flang-compiler/flang/pull/898: [DebugInfo] Corrected debuginfo for modules. Dependency on LLVM11. [Bryan approved. Can proceed after removing llvm-10 CI]
  DebugInfo: 1140 (can skip, can close after 1141),**1141,1142,1143 (module variable different file issue) bryanpkc@gmail.com Kiran Chandramohan**
  https://github.com/flang-compiler/flang/pull/1147
  String debug with llvm-11. Will try to raise PRs this week.

  Arm:
  **https://github.com/flang-compiler/flang/pull/1094** : Add loop start, end to llvm.loop metadata. [Arm accepted]. Useful to have it in the beginning of July. [AMD, Huawei approved. One question from Kiran]
  https://github.com/flang-compiler/flang/pull/146 : Machreg Refactor. [Bryan approved. Paul provided info that AMD requested].
  https://github.com/flang-compiler/flang/pull/659 : Changes to support transpose intrinsic in initialization. [Bryan has comments]

  Huawei: (PRs are sorted in order)
  **(high priority; would like to enable -Werror sooner rather than later)**
  https://github.com/flang-compiler/flang/issues/1065 : LLVM 12 : All PRs mentioned.
  https://github.com/flang-compiler/flang/pull/1100 : Incorrect behavior when passing a polymorphic array as an argument to a function. [Ready to merge. Pawel has a request.]
  https://github.com/flang-compiler/flang/pull/1123: Use const char* for commandline args.

[Waiting for approval from Shivaram]
https://github.com/flang-compiler/flang/pull/1132 : Format issues?
+++++++++++++++++++++++++++++++++++++ ^Target for next weeks call
https://github.com/flang-compiler/flang/pull/1063 : Correct the type of unlimited polymorphic pointers
https://github.com/flang-compiler/flang/pull/1102 : Fix bugs in array constructor with an implied-do containing an expression that is a function call returning an array
https://github.com/flang-compiler/flang/pull/1055 : Fix bugs in PACK intrinsic. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1064 : Provide v_list argument as a zero-sized array in some I/O cases as required by the standard.
https://github.com/flang-compiler/flang/pull/1053 : Fix bugs in getarg. [Huawei started review, and requested inputs from Mark.] Should we be consistent with gfortran, ifort?
https://github.com/flang-compiler/flang/pull/1104 : Fix incorrect code generation for a polymorphic derived type member that is given a runtime type of CHARACTER(*). **(medium priority)**
https://github.com/flang-compiler/flang/pull/1088 : Fix a segmentation fault related to assumed length character functions.
https://github.com/flang-compiler/flang/pull/1052 : Fix a segmentation fault when the name of an ENTRY is used to return a value instead of the correct RESULT variable.
https://github.com/flang-compiler/flang/pull/1036 : Fix error of processing ENTRY statement. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1050 : Correct the type of unlimited polymorphic pointers.
https://github.com/flang-compiler/flang/pull/1089 : Prevent illegal argument accessing during procedure exporting.
https://github.com/flang-compiler/flang/pull/1091 : Add more checks for derived types in COMMON statements (to avoid compile-time segmentation faults).
https://github.com/flang-compiler/flang/pull/1062 : Fix an ICE when a named constant array is declared deferred shape (and then used in RESHAPE for another array).
https://github.com/flang-compiler/flang/pull/1101 : Fix compile-time errors caused by empty derived type constructors nested in array constructors.
https://github.com/flang-compiler/flang/pull/1051 : Report two grammar errors pertaining to PROTECTED attribute.
https://github.com/flang-compiler/flang/pull/1105 : Suppress debugloc for actual argument bitcast. [AMD (jini) could not reproduce the issue]

Windows:
Other patches? **991** (Adam) needs a rebase.

MacOS:
https://github.com/flang-compiler/flang/pull/1120 [Paul/Arm approved]

## Jan 12, 2022

Attendees
AMD: Alok
Arm: Pawel, Kiran
Huawei:
Nvidia:
Others: Przemek (Mobica/Arm)

flang-compiler/flang

- Would like to send a status mail. Please add any topics that we should cover https://docs.google.com/document/d/1n-TpNzNbAmHZlP4ziFykyH-GeGe2xP7yd8KDnDvzJsA/edit?usp=sharing

- Bryan created several PRs for fixing warnings with llvm-12. https://github.com/flang-compiler/flang/pull/1191 can be merged.
  Bryan: New ones might be there.
  Format and format-pedantic being handled by Bryan.
  There is an issue filed regarding Werror=format-security. This seems to be the default in ArchLinux. Will we be able to fix the issues here? Is it in plan? https://github.com/flang-compiler/flang/issues/1204

- llvm-13 :
  - Bryan mentioned that some string debug changes were removed. Need AMD guys to confirm whether we need to port the llvm/lib/CodeGen/AsmPrinter/DwarfDebug.cpp changes in the following commit to release_13x or not (dependent type tracking, DebugLocEntry::finalize for DIStringType, etc.): https://github.com/flang-compiler/classic-flang-llvm-project/commit/59f78e87b76e ; the code in question originated from this commit in release_50: https://github.com/flang-compiler/llvm/commit/80698caba789e44da476094d493b0eeb9a558be3

- Vscale_range attribute (?). Emit vscale_range when sve target option is set. https://github.com/flang-compiler/classic-flang-llvm-project/pull/84 https://github.com/flang-compiler/flang/pull/1187
  - Status
  - Przemek finishing work on classic-flang-llvm-project
  - Pawel: Flang portion not working properly. Upstream clang/llvm is also not working. Pawel working on it downstream and will share. Suggest Przemek not work on this item this week.

- GCC 11
  - https://github.com/flang-compiler/classic-flang-llvm-project/pull/79
  - Also #80 and #81
  - Status: Finished working on this and waiting for review.

- DILocation fix: https://github.com/flang-compiler/flang/pull/1188 (Bryan approved, AMD)
  - Pawel's patch. AMD will have a look.

- New directive: vectorize_width (https://github.com/flang-compiler/flang/pull/1172)
  - What is the status? (Parked for now.)

- Quad-precision FP support in Flang. Huawei has now submitted two patches. Third patch? Will be ready in a couple of days.

- Implied do and array constructor bug. https://github.com/flang-compiler/flang/issues/1200

- Cross-compilation issues https://github.com/flang-compiler/flang/issues/1202#issuecomment-1010263623

- Windows
  - Windows PR list. https://github.com/flang-compiler/flang/issues/1154. Only one more required to get Windows to compile.
    -> static libs and export patches. -> CMake change to allow windows required.
    -> Adam can share the build recipe
    -> 320 : Looking into this. Colleagues working on tests. Shell scripts not Running on windows.
  - PR to enable the Windows build in CMakefiles here:
    https://github.com/flang-compiler/flang/pull/1184
    https://github.com/flang-compiler/flang/pull/1181
  - We did a python build script to supports the Flang build on Windows from the scratch: https://github.com/flang-compiler/flang/pull/1185
- Add links from chat (from Alok, Bhuvan).

- Status of PRs
  AMD:
  https://github.com/flang-compiler/flang/pull/898: [DebugInfo] Corrected debuginfo for modules. Dependency on LLVM11. [Bryan approved. Can proceed after removing llvm-10 CI]

DebugInfo: 1140 (can skip, can close after 1141),**1141,1142,1143 (module variable different file issue) bryanpkc@gmail.com Kiran Chandramohan**
https://github.com/flang-compiler/flang/pull/1147
String debug with llvm-11. Will try to raise PRs this week.

Arm:
**https://github.com/flang-compiler/flang/pull/1094** : Add loop start, end to llvm.loop metadata. [Arm accepted]. Useful to have it in the beginning of July. [AMD, Huawei approved. One question from Kiran]
https://github.com/flang-compiler/flang/pull/146 : Machreg Refactor. [Bryan approved. Paul provided info that AMD requested].
https://github.com/flang-compiler/flang/pull/659 : Changes to support transpose intrinsic in initialization. [Bryan has comments]

Huawei: (PRs are sorted in order)
**(high priority; would like to enable -Werror sooner rather than later)**
https://github.com/flang-compiler/flang/issues/1065 : LLVM 12 : All PRs mentioned.
https://github.com/flang-compiler/flang/pull/1100 : Incorrect behavior when passing a polymorphic array as an argument to a function. [Ready to merge. Pawel has a request.]
https://github.com/flang-compiler/flang/pull/1123: Use const char* for commandline args. [Waiting for approval from Shivaram]
https://github.com/flang-compiler/flang/pull/1132 : Format issues?
++++++++++++++++++++++++++++++++++++++ ^Target for next weeks call
https://github.com/flang-compiler/flang/pull/1063 : Correct the type of unlimited polymorphic pointers
https://github.com/flang-compiler/flang/pull/1102 : Fix bugs in array constructor with an implied-do containing an expression that is a function call returning an array
https://github.com/flang-compiler/flang/pull/1055 : Fix bugs in PACK intrinsic. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1064 : Provide v_list argument as a zero-sized array in some I/O cases as required by the standard.
https://github.com/flang-compiler/flang/pull/1053 : Fix bugs in getarg. [Huawei started review, and requested inputs from Mark.] Should we be consistent with gfortran, ifort?
https://github.com/flang-compiler/flang/pull/1104 : Fix incorrect code generation for a polymorphic derived type member that is given a runtime type of CHARACTER(*).
**(medium priority)**
https://github.com/flang-compiler/flang/pull/1088 : Fix a segmentation fault related to assumed length character functions.
https://github.com/flang-compiler/flang/pull/1052 : Fix a segmentation fault when the name of an ENTRY is used to return a value instead of the correct RESULT variable.
https://github.com/flang-compiler/flang/pull/1036 : Fix error of processing ENTRY statement. [Arm accepted]

https://github.com/flang-compiler/flang/pull/1050 : Correct the type of unlimited polymorphic pointers.
https://github.com/flang-compiler/flang/pull/1089 : Prevent illegal argument accessing during procedure exporting.
https://github.com/flang-compiler/flang/pull/1091 : Add more checks for derived types in COMMON statements (to avoid compile-time segmentation faults).
https://github.com/flang-compiler/flang/pull/1062 : Fix an ICE when a named constant array is declared deferred shape (and then used in RESHAPE for another array).
https://github.com/flang-compiler/flang/pull/1101 : Fix compile-time errors caused by empty derived type constructors nested in array constructors.
https://github.com/flang-compiler/flang/pull/1051 : Report two grammar errors pertaining to PROTECTED attribute.
https://github.com/flang-compiler/flang/pull/1105 : Suppress debugloc for actual argument bitcast. [AMD (jini) could not reproduce the issue]

Windows:
Other patches? **991** (Adam) needs a rebase.

MacOS:
https://github.com/flang-compiler/flang/pull/1120 [Paul/Arm approved]

## Dec 15, 2021

Attendees
AMD:
Arm: Pawel, Kiran
Huawei: Bryan
Nvidia:
Others:

flang-compiler/flang

- FOSDEM'22 LLVM Devroom. Please submit if you have something interesting.
  https://llvm.discourse.group/t/cfp-fosdem-22-llvm-devroom/4944
  -> Debug changes to LLVM for supporting Fortran

- Would like to send an end of year mail. Please add any topics that we should cover
  https://docs.google.com/document/d/1n-TpNzNbAmHZIP4ziFykyH-GeGe2xP7yd8KDnDvzJsA/edit?usp=sharing

- llvm-12 : Bryan has now created several PRs. List available in https://github.com/flang-compiler/flang/issues/1065.

bryanpkc@gmail.com looks like all the PRs are merged now.
One more patch. Sign changes, type casts. Minor fixes.

- llvm-13 : Abraham (Arm/Mobica) is working on this.
    - What is the status?
        - Looks like patches are all submitted. Is anything remaining?
        - Bryan mentioned that some string debug changes were removed. Need AMD guys to confirm whether we need to port the llvm/lib/CodeGen/AsmPrinter/DwarfDebug.cpp changes in the following commit to release_13x or not (dependent type tracking, DebugLocEntry::finalize for DIStringType, etc.): https://github.com/flang-compiler/classic-flang-llvm-project/commit/59f78e87b76e ; the code in question originated from this commit in release_50: https://github.com/flang-compiler/llvm/commit/80698caba789e44da476094d493b0eeb9a558be3

- New directive: vectorize_width (https://github.com/flang-compiler/flang/pull/1172)
    - What is the status? (Parked for now. Focus on another ticket : BigDFT. Waiting for review from Shivaram (AMD), 1179. Also working on GCC 11.)

- Vscale_range attribute (?). Emit vscale_range when sve target option is set.
    - Status
    - Bryan suggested putting it in #0
    - Need reviewers. Arm only change.

- Quad-precision FP support in Flang. Huawei has now submitted two patches.

- Cam4 benchmark : Memory allocation and stat handling : https://github.com/flang-compiler/flang/issues/884. Anyone has a fix?
  AMD and Huawei Reverted patch. Kiran Chandramohan will check with Nvidia and if they are not planning to upstream, Arm will upstream a patch.

- Windows
    - Windows PR list. https://github.com/flang-compiler/flang/issues/1154. Only one more required to get Windows to compile.
      -> static libs and export patches. -> CMake change to allow windows required.
      -> Adam can share the build recipe
      -> 320 : Looking into this. Colleagues working on tests. Shell scripts not Running on windows.
    - PR to enable the Windows build in CMakefiles here:

      https://github.com/flang-compiler/flang/pull/1184

    - We did a python build script to supports the Flang build on Windows from the scratch: https://github.com/flang-compiler/flang/pull/1185

- MacOS
    - https://github.com/flang-compiler/flang/pull/1120 [Paul, Bryan approved] shivarama.rao@amd.com ? Kiran Chandramohan will look into this and merge.
    - Driver patches merged

- Gdb community flang/fortran patches. Any progress here?

- Offloading code : Only support SIMD kernels.

- Status of PRs
  AMD:
  https://github.com/flang-compiler/flang/pull/898: [DebugInfo] Corrected debuginfo for modules. Dependency on LLVM11. [Bryan approved. Can proceed after removing llvm-10 CI]
  DebugInfo: 1140 (can skip, can close after 1141),**1141,1142,1143 (module variable different file issue) bryanpkc@gmail.com Kiran Chandramohan**
  https://github.com/flang-compiler/flang/pull/1147
  String debug with llvm-11. Will try to raise PRs this week.

  Arm:
  **https://github.com/flang-compiler/flang/pull/1094** : Add loop start, end to llvm.loop metadata. [Arm accepted]. Useful to have it in the beginning of July. [AMD, Huawei approved. One question from Kiran]
  https://github.com/flang-compiler/flang/pull/146 : Machreg Refactor. [Bryan approved. Paul provided info that AMD requested].
  https://github.com/flang-compiler/flang/pull/659 : Changes to support transpose intrinsic in initialization. [Bryan has comments]

  Huawei: (PRs are sorted in order)
  **(high priority; would like to enable -Werror sooner rather than later)**
  https://github.com/flang-compiler/flang/issues/1065 : LLVM 12 : All PRs mentioned.
  https://github.com/flang-compiler/flang/pull/1100 : Incorrect behavior when passing a polymorphic array as an argument to a function. [Ready to merge. Pawel has a request.]
  https://github.com/flang-compiler/flang/pull/1123: Use const char* for commandline args. [Waiting for approval from Shivaram]
  https://github.com/flang-compiler/flang/pull/1132 : Format issues?
  +++++++++++++++++++++++++++++++++++++++ ^Target for next weeks call

https://github.com/flang-compiler/flang/pull/1063 : Correct the type of unlimited polymorphic pointers
https://github.com/flang-compiler/flang/pull/1102 : Fix bugs in array constructor with an implied-do containing an expression that is a function call returning an array
https://github.com/flang-compiler/flang/pull/1055 : Fix bugs in PACK intrinsic. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1064 : Provide v_list argument as a zero-sized array in some I/O cases as required by the standard.
https://github.com/flang-compiler/flang/pull/1053 : Fix bugs in getarg. [Huawei started review, and requested inputs from Mark.] Should we be consistent with gfortran, ifort?
https://github.com/flang-compiler/flang/pull/1104 : Fix incorrect code generation for a polymorphic derived type member that is given a runtime type of CHARACTER(*). **(medium priority)**
https://github.com/flang-compiler/flang/pull/1088 : Fix a segmentation fault related to assumed length character functions.
https://github.com/flang-compiler/flang/pull/1052 : Fix a segmentation fault when the name of an ENTRY is used to return a value instead of the correct RESULT variable.
https://github.com/flang-compiler/flang/pull/1036 : Fix error of processing ENTRY statement. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1050 : Correct the type of unlimited polymorphic pointers.
https://github.com/flang-compiler/flang/pull/1089 : Prevent illegal argument accessing during procedure exporting.
https://github.com/flang-compiler/flang/pull/1091 : Add more checks for derived types in COMMON statements (to avoid compile-time segmentation faults).
https://github.com/flang-compiler/flang/pull/1062 : Fix an ICE when a named constant array is declared deferred shape (and then used in RESHAPE for another array).
https://github.com/flang-compiler/flang/pull/1101 : Fix compile-time errors caused by empty derived type constructors nested in array constructors.
https://github.com/flang-compiler/flang/pull/1051 : Report two grammar errors pertaining to PROTECTED attribute.
https://github.com/flang-compiler/flang/pull/1105 : Suppress debugloc for actual argument bitcast. [AMD (jini) could not reproduce the issue]

Windows:
Other patches? **991** (Adam) needs a rebase.

MacOS:
https://github.com/flang-compiler/flang/pull/1120 [Paul/Arm approved]

## Dec 1, 2021

Attendees
AMD:
Arm: Pawel, Kiran
Huawei: Bryan
Nvidia:
Others:

flang-compiler/flang

- llvm-12 : Bryan has now created several PRs. List available in https://github.com/flang-compiler/flang/issues/1065.
  PRs under review.
  Uninitialized variable warnings: https://github.com/flang-compiler/flang/pull/1174
  Fix implicit function decl warnings: https://github.com/flang-compiler/flang/pull/1180

- llvm-13 : Abraham (Arm/Mobica) is working on this.
  - What is the status?
    - Merged classic-flang-llvm-project and flang PR
      LLVM13 CI changes:
      https://github.com/flang-compiler/classic-flang-llvm-project/pull/67
    - Exempt Wunused-but-set-var warnings:
      https://github.com/flang-compiler/flang/pull/1183
    - Bryan: String debug changes removed.
      Abraham: Got a segmentation fault. Gcc-10 crashed with these changes.
      Bryan: Will put up a patch.

- New directive: vectorize_width (https://github.com/flang-compiler/flang/pull/1172)
  - What is the status?
  - Is the following syntax OK?
    - `!dir$ vector vectorlength(n1[, n2,..,fixed|scalable])`
    - `!dir$ vector vectorlength(fixed|scalable)`
    - `Bryan: Can vectorize_width also be supported? Can decide once we decide for llvm/flang.`
- Vscale_range attribute (?). Emit vscale_range when sve target option is set.
  - Status
  - Bryan: Put it in #0
    Here is an example:
    attributes #0 = { noinline nounwind optnone uwtable "disable-tail-calls"="false" "frame-pointer"="all" "less-precise-fpmad"="false" "min-legal-vector-width"="0" "no-infs-fp-math"="false" "no-jumalse" "no-signed-zeros-fp-math"="false" "no-trapping-math"="true" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87"
    "tune-cpu"="generic"oat"="false" }

you could add "vscale-range" = "..." in this attribute.
That is, if Clang always does it this way.

- Good place to add vscale flags?
  LLVM target attributes to LLVM IR through flang2. Need it for LTO. (https://github.com/flang-compiler/flang/pull/1173). Huawei did something similar, Bryan will have a look. classic-flang-llvm-project (flang driver) PR is https://github.com/flang-compiler/classic-flang-llvm-project/pull/66.

- Quad-precision FP support in Flang. Huawei has submitted the first patch. The second patch is approved.
  - https://github.com/flang-compiler/flang/pull/1129 : Ready to merge

- Windows
  - A label was created (Do we need labels for anything else?)
  - Windows PR list. https://github.com/flang-compiler/flang/issues/1154. Only two more required to get Windows to compile.
    -> static libs and export patches. -> CMake change to allow windows required.
    -> Adam can share the build recipe
    -> 320 : Looking into this. Colleagues working on tests. Shell scripts not Running on windows.
  - https://github.com/flang-compiler/flang/pull/1156: **Fix the inconsistent usage of dllexport/dllimport** : Can merge
  - https://github.com/flang-compiler/flang/pull/1150: **#1150 win: 'ffast-math' option is not recognized by clang-c**l : Can merge
  - PR to enable the Windows build in CMakefiles here: https://github.com/flang-compiler/flang/pull/1184
  - We did a python build script to supports the Flang build on Windows from the scratch: https://github.com/flang-compiler/flang/pull/1185

- MacOS
  - https://github.com/flang-compiler/flang/pull/1120 [Paul, Bryan approved] shivarama.rao@amd.com ?
  - Driver patches merged

- Gdb community flang/fortran patches. Any progress here?

- Would like to send an end of year mail. Please add any topics that we should cover https://docs.google.com/document/d/1n-TpNzNbAmHZlP4ziFykyH-GeGe2xP7yd8KDnDvzJsA/edit?usp=sharing

- Status of PRs
  AMD:
  https://github.com/flang-compiler/flang/pull/898: [DebugInfo] Corrected debuginfo for modules. Dependency on LLVM11. [Bryan approved. Can proceed after removing llvm-10 CI]
  DebugInfo: 1140 (can skip, can close after 1141),**1141,1142,1143 (module variable different file issue) bryanpkc@gmail.com Kiran Chandramohan**
  https://github.com/flang-compiler/flang/pull/1147
  String debug with llvm-11. Will try to raise PRs this week.

  Arm:
  **https://github.com/flang-compiler/flang/pull/1094** : Add loop start, end to llvm.loop metadata. [Arm accepted]. Useful to have it in the beginning of July. [AMD, Huawei approved. One question from Kiran]
  https://github.com/flang-compiler/flang/pull/146 : Machreg Refactor. [Bryan approved. Paul provided info that AMD requested].
  https://github.com/flang-compiler/flang/pull/659 : Changes to support transpose intrinsic in initialization. [Bryan has comments]

  Huawei: (PRs are sorted in order)
  **(high priority; would like to enable -Werror sooner rather than later)**
  https://github.com/flang-compiler/flang/issues/1065 : LLVM 12 : All PRs mentioned.
  https://github.com/flang-compiler/flang/pull/1100 : Incorrect behavior when passing a polymorphic array as an argument to a function. [Ready to merge. Pawel has a request.]
  https://github.com/flang-compiler/flang/pull/1123: Use const char* for commandline args. [Waiting for approval from Shivaram]
  https://github.com/flang-compiler/flang/pull/1129 : quad precision for intrinsics
  https://github.com/flang-compiler/flang/pull/1132 : Format issues?
  +++++++++++++++++++++++++++++++++++++++ ^Target for next weeks call
  https://github.com/flang-compiler/flang/pull/1063 : Correct the type of unlimited polymorphic pointers
  https://github.com/flang-compiler/flang/pull/1102 : Fix bugs in array constructor with an implied-do containing an expression that is a function call returning an array
  https://github.com/flang-compiler/flang/pull/1055 : Fix bugs in PACK intrinsic. [Arm accepted]
  https://github.com/flang-compiler/flang/pull/1064 : Provide v_list argument as a zero-sized array in some I/O cases as required by the standard.
  https://github.com/flang-compiler/flang/pull/1053 : Fix bugs in getarg. [Huawei started review, and requested inputs from Mark.] Should we be consistent with gfortran, ifort?
  https://github.com/flang-compiler/flang/pull/1104 : Fix incorrect code generation for a polymorphic derived type member that is given a runtime type of CHARACTER(*).

**(medium priority)**
https://github.com/flang-compiler/flang/pull/1088 : Fix a segmentation fault related to assumed length character functions.
https://github.com/flang-compiler/flang/pull/1052 : Fix a segmentation fault when the name of an ENTRY is used to return a value instead of the correct RESULT variable.
https://github.com/flang-compiler/flang/pull/1036 : Fix error of processing ENTRY statement. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1050 : Correct the type of unlimited polymorphic pointers.
https://github.com/flang-compiler/flang/pull/1089 : Prevent illegal argument accessing during procedure exporting.
https://github.com/flang-compiler/flang/pull/1091 : Add more checks for derived types in COMMON statements (to avoid compile-time segmentation faults).
https://github.com/flang-compiler/flang/pull/1062 : Fix an ICE when a named constant array is declared deferred shape (and then used in RESHAPE for another array).
https://github.com/flang-compiler/flang/pull/1101 : Fix compile-time errors caused by empty derived type constructors nested in array constructors.
https://github.com/flang-compiler/flang/pull/1051 : Report two grammar errors pertaining to PROTECTED attribute.
https://github.com/flang-compiler/flang/pull/1105 : Suppress debugloc for actual argument bitcast. [AMD (jini) could not reproduce the issue]

Windows:
Other patches? **991** (Adam) needs a rebase.

MacOS:
https://github.com/flang-compiler/flang/pull/1120 [Paul/Arm approved]

## Nov 17, 2021

Attendees
AMD:
Arm:
Huawei:
Nvidia:
Others:

flang-compiler/flang

- Timeslot and zoom/teams. Is it OK with everyone?
  Slot works fine.

- llvm-12 : Bryan has now created several PRs. List available in https://github.com/flang-compiler/flang/issues/1065.
  PRs under review.
  Const pointer related warnings: https://github.com/flang-compiler/flang/pull/1164
  Missing field initializer/braces warnings: https://github.com/flang-compiler/flang/pull/1162
  Uninitialized variable warnings: https://github.com/flang-compiler/flang/pull/1174
  Couple more warnings related patches.

- llvm-13 : Abraham (Arm/Mobica) is working on this.
  - What is the status?
    - Bryan has review comments.
      https://github.com/flang-compiler/classic-flang-llvm-project/pull/62
      Addressed comments, patches are in order now.
      Pawel: Please mark comments as resolved.
      Bryan: Shivaram review?
    - Prefetch test issue.
      Is this fixed?
      Pawel looking into this. Will issue a PR.

- New directive: vectorize_width (https://github.com/flang-compiler/flang/pull/1172)
  - Currently follows clang style
  - Should it be accommodated into !dir$ vector that we already have?

  - Status
    Metadata in the wrong place. Need time. Looking at other vectorize pragmas.
    Pawel: Use flang -### filename.f90 to know the individual stages (flang1, flang2, IR, linker)
    Bryan: You can debug the "flang" driver with gdb, and use "set follow-fork-mode child" inside gdb to tell the debugger to step into flang1/flang2
    Rich: Trying to upstream emit-flang-llvm. Rich will add another comment with the final option name.

- Vscale_range attribute (?)
  - Status
    LLVM target attributes to LLVM IR through flang2. Need it for LTO. (https://github.com/flang-compiler/flang/pull/1173). Huawei did something similar, Bryan will have a look. classic-flang-llvm-project (flang driver) PR is https://github.com/flang-compiler/classic-flang-llvm-project/pull/66.
  - Emit vscale_range when sve target option is set.

- Quad-precision FP support in Flang. Huawei has submitted the first patch. The second patch was available.

    - https://github.com/flang-compiler/flang/pull/1129 : Author Will fix merge conflicts in 2 days.
    - shivarama.rao@amd.com will have a look this week.

- Windows
    - A label was created (Do we need labels for anything else?)
    - Windows PR list. https://github.com/flang-compiler/flang/issues/1154. Only two more required to get Windows to compile.
      -> static libs and export patches. -> CMake change to allow windows required.
      -> Adam can share the build recipe
      -> 320 : Looking into this. Colleagues working on tests. Shell scripts not Running on windows.

- MacOS
    - https://github.com/flang-compiler/flang/pull/1120 [Paul/Arm approved]
    - Driver patches merged
    -
- Gdb community flang/fortran patches.

- Status of PRs
  AMD:
  https://github.com/flang-compiler/flang/pull/898: [DebugInfo] Corrected debuginfo for modules. Dependency on LLVM11. [Bryan approved. Can proceed after removing llvm-10 CI]
  DebugInfo: 1140 (can skip, can close after 1141),**1141,1142,1143 (module variable different file issue) bryanpkc@gmail.com Kiran Chandramohan**
  https://github.com/flang-compiler/flang/pull/1147
  String debug with llvm-11. Will try to raise PRs this week.

  Arm:
  **https://github.com/flang-compiler/flang/pull/1094** : Add loop start, end to llvm.loop metadata. [Arm accepted]. Useful to have it in the beginning of July. [AMD, Huawei approved. One question from Kiran]
  https://github.com/flang-compiler/flang/pull/146 : Machreg Refactor. [Bryan approved. Paul provided info that AMD requested].
  https://github.com/flang-compiler/flang/pull/659 : Changes to support transpose intrinsic in initialization. [Bryan has comments]

  Huawei: (PRs are sorted in order)
  **(high priority; would like to enable -Werror sooner rather than later)**
  https://github.com/flang-compiler/flang/issues/1065 : LLVM 12 : All PRs mentioned.

https://github.com/flang-compiler/flang/pull/1100 : Incorrect behavior when passing a polymorphic array as an argument to a function. [Ready to merge. Pawel has a request.]
https://github.com/flang-compiler/flang/pull/1123: Use const char* for commandline args. [Waiting for approval from Shivaram]
https://github.com/flang-compiler/flang/pull/1129 : quad precision for intrinsics
https://github.com/flang-compiler/flang/pull/1132 : Format issues?
++++++++++++++++++++++++++++++++++++ ^Target for next weeks call
https://github.com/flang-compiler/flang/pull/1063 : Correct the type of unlimited polymorphic pointers
https://github.com/flang-compiler/flang/pull/1102 : Fix bugs in array constructor with an implied-do containing an expression that is a function call returning an array
https://github.com/flang-compiler/flang/pull/1055 : Fix bugs in PACK intrinsic. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1064 : Provide v_list argument as a zero-sized array in some I/O cases as required by the standard.
https://github.com/flang-compiler/flang/pull/1053 : Fix bugs in getarg. [Huawei started review, and requested inputs from Mark.] Should we be consistent with gfortran, ifort?
https://github.com/flang-compiler/flang/pull/1104 : Fix incorrect code generation for a polymorphic derived type member that is given a runtime type of CHARACTER(*).
**(medium priority)**
https://github.com/flang-compiler/flang/pull/1088 : Fix a segmentation fault related to assumed length character functions.
https://github.com/flang-compiler/flang/pull/1052 : Fix a segmentation fault when the name of an ENTRY is used to return a value instead of the correct RESULT variable.
https://github.com/flang-compiler/flang/pull/1036 : Fix error of processing ENTRY statement. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1050 : Correct the type of unlimited polymorphic pointers.
https://github.com/flang-compiler/flang/pull/1089 : Prevent illegal argument accessing during procedure exporting.
https://github.com/flang-compiler/flang/pull/1091 : Add more checks for derived types in COMMON statements (to avoid compile-time segmentation faults).
https://github.com/flang-compiler/flang/pull/1062 : Fix an ICE when a named constant array is declared deferred shape (and then used in RESHAPE for another array).
https://github.com/flang-compiler/flang/pull/1101 : Fix compile-time errors caused by empty derived type constructors nested in array constructors.
https://github.com/flang-compiler/flang/pull/1051 : Report two grammar errors pertaining to PROTECTED attribute.
https://github.com/flang-compiler/flang/pull/1105 : Suppress debugloc for actual argument bitcast. [AMD (jini) could not reproduce the issue]

Windows:

Other patches? **991** (Adam) needs a rebase.

MacOS:
https://github.com/flang-compiler/flang/pull/1120 [Paul/Arm approved]

## Nov 3, 2021

Attendees
AMD:
Arm:
Huawei:
Nvidia:
Others:

flang-compiler/flang

- Timeslot and zoom/teams.
  AMD/India : This time is OK half-an-hour earlier
  3.30pm UK, Time 9pm IST, 10.30 ET, OK for mobica as well.
  Zoom invite.

- llvm-12 : Bryan has now created several PRs. List available in https://github.com/flang-compiler/flang/issues/1065.
  Last call, we agreed to add a few flags in CMake to disable some warnings. Bryan has a few a new PRs which fixes some warnings.
  Const pointer related warnings: https://github.com/flang-compiler/flang/pull/1164
  Missing field initializer/braces warnings: https://github.com/flang-compiler/flang/pull/1162
  Unused var,func,label warnings: https://github.com/flang-compiler/flang/pull/1160 : has flags to disable warnings.
  shivarama.rao@amd.com will have a look in the weekend.

  Stop writing to source dir: https://github.com/flang-compiler/flang/pull/1161 : Merged

  Have follow up patches which fixes separate warnings or bugs.

- llvm-13 : Abraham (Arm/Mobica) is working on this.
  - What is the status?
    - Bryan has review comments.
      https://github.com/flang-compiler/classic-flang-llvm-project/pull/62
    - Prefetch test issue. Last time we suggested the following
      Kiran: Add emit-flang-llvm option upstream
      Rich: Relax test
      Bryan: Test at O0. IR same except for a header with llvm-12 and llvm-13

- Issue #1146 'Get the wrong variable after module USE statement', regression after PR#949
  - Decided to wait for a fix rather than revert.
  - https://github.com/flang-compiler/flang/pull/1149 created by Huawei. What is the status?
  - bryanpkc@gmail.com will follow up on the discussion in https://github.com/flang-compiler/flang/issues/1146 and let others know is slack.
  - bryanpkc@gmail.com shivarama.rao@amd.com please have a look.
- Quad-precision FP support in Flang. Huawei has submitted the first patch. The second patch was available.
  - https://github.com/flang-compiler/flang/pull/1129 : Author Will fix merge conflicts in 2 days.
  - shivarama.rao@amd.com can you have a look?
  - Further PRs expected within 2 weeks
- https://github.com/flang-compiler/flang/pull/1094 : Add loop start, end to llvm.loop metadata. [Approval from Bryan, Shivaram. LGTM from Chirag]
  - On hold

- Windows
  - A label was created (Do we need labels for anything else?)
  - Windows PR list. https://github.com/flang-compiler/flang/issues/1154. Only three more required to get Windows to compile.
- MacOS
  - https://github.com/flang-compiler/flang/pull/1120 [Paul/Arm approved]
  - Driver patches merged
- New directive: vectorize_width mleair@nvidia.com

- Status of PRs

  AMD:

  https://github.com/flang-compiler/flang/pull/898: [DebugInfo] Corrected debuginfo for modules. Dependency on LLVM11. [Bryan approved. Can proceed after removing llvm-10 CI]

  DebugInfo: 1140 (can skip, can close after 1141),**1141,1142,1143 (module variable different file issue) bryanpkc@gmail.com Kiran Chandramohan**

  https://github.com/flang-compiler/flang/pull/1147

  String debug with llvm-11. Will try to raise PRs this week.

  Arm:

  **https://github.com/flang-compiler/flang/pull/1094** : Add loop start, end to llvm.loop metadata. [Arm accepted]. Useful to have it in the beginning of July. [AMD, Huawei

approved. One question from Kiran]
https://github.com/flang-compiler/flang/pull/146 : Machreg Refactor. [Bryan approved. Paul provided info that AMD requested].
https://github.com/flang-compiler/flang/pull/659 : Changes to support transpose intrinsic in initialization. [Bryan has comments]

Huawei: (PRs are sorted in order)
**(high priority; would like to enable -Werror sooner rather than later)**
https://github.com/flang-compiler/flang/issues/1065 : LLVM 12 : All PRs mentioned.
https://github.com/flang-compiler/flang/pull/1100 : Incorrect behavior when passing a polymorphic array as an argument to a function. [Ready to merge. Pawel has a request.]
https://github.com/flang-compiler/flang/pull/1123: Use const char* for commandline args. [Waiting for approval from Shivaram]
https://github.com/flang-compiler/flang/pull/1129 : quad precision for intrinsics
https://github.com/flang-compiler/flang/pull/1132 : Format issues?
+++++++++++++++++++++++++++++++++++++++ ^Target for next weeks call
https://github.com/flang-compiler/flang/pull/1063 : Correct the type of unlimited polymorphic pointers
https://github.com/flang-compiler/flang/pull/1102 : Fix bugs in array constructor with an implied-do containing an expression that is a function call returning an array
https://github.com/flang-compiler/flang/pull/1055 : Fix bugs in PACK intrinsic. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1064 : Provide v_list argument as a zero-sized array in some I/O cases as required by the standard.
https://github.com/flang-compiler/flang/pull/1053 : Fix bugs in getarg. [Huawei started review, and requested inputs from Mark.] Should we be consistent with gfortran, ifort?
https://github.com/flang-compiler/flang/pull/1104 : Fix incorrect code generation for a polymorphic derived type member that is given a runtime type of CHARACTER(*).
**(medium priority)**
https://github.com/flang-compiler/flang/pull/1088 : Fix a segmentation fault related to assumed length character functions.
https://github.com/flang-compiler/flang/pull/1052 : Fix a segmentation fault when the name of an ENTRY is used to return a value instead of the correct RESULT variable.
https://github.com/flang-compiler/flang/pull/1036 : Fix error of processing ENTRY statement. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1050 : Correct the type of unlimited polymorphic pointers.
https://github.com/flang-compiler/flang/pull/1089 : Prevent illegal argument accessing during procedure exporting.
https://github.com/flang-compiler/flang/pull/1091 : Add more checks for derived types in COMMON statements (to avoid compile-time segmentation faults).
https://github.com/flang-compiler/flang/pull/1062 : Fix an ICE when a named constant array is declared deferred shape (and then used in RESHAPE for another array).

https://github.com/flang-compiler/flang/pull/1101 : Fix compile-time errors caused by empty derived type constructors nested in array constructors.
https://github.com/flang-compiler/flang/pull/1051 : Report two grammar errors pertaining to PROTECTED attribute.
https://github.com/flang-compiler/flang/pull/1105 : Suppress debugloc for actual argument bitcast. [AMD (jini) could not reproduce the issue]

Windows:
Other patches? **991** (Adam) needs a rebase.

MacOS:
https://github.com/flang-compiler/flang/pull/1120 [Paul/Arm approved]

## Oct 20, 2021

Attendees
AMD:
Arm:
Huawei:
Nvidia:
Others:

flang-compiler/flang

- llvm-12 : Bryan has now created several PRs. List available in https://github.com/flang-compiler/flang/issues/1065.
  Currently reviewing the following:
  [NFCI] Use const char in CCFF API: https://github.com/flang-compiler/flang/pull/1130 (Pawel/Arm approved). Shivaram can you approve or waive?
  shivarama.rao@amd.com will have a look today.
  Some roll up patches. Bryan is refactoring but blocked on CCFF.
  Paul is OK with adding a warning disable flag to CMake for typecasts.
- llvm-13 : Abraham (Arm/Mobica) is working on this.
  - What is the status?
    - Will pull in PR 62.
    - No build/test issues with clang. One test issue with flang. Issue related to prefetch test. Same test not working with llvm-12 and llvm-13. Changes with instcombine leads to matching llvm-12 and llvm-13 IR.
      Kiran: Add emit-flang-llvm option upstream
      Rich: Relax test
      Bryan: Test at O0
    - CI changes ready, waiting for merging llvm/flang patches.

- Issue #1146 ‘Get the wrong variable after module USE statement’, regression after PR#949
    - Decided to wait for a fix rather than revert.
    - https://github.com/flang-compiler/flang/pull/1149 created by Huawei. What is the status?
    - bryanpkc@gmail.com will follow up on the discussion in https://github.com/flang-compiler/flang/issues/1146 and let others know is slack.
- Quad-precision FP support in Flang. Huawei has submitted the first patch. The second patch was available.
    - https://github.com/flang-compiler/flang/pull/1129 : Author Will fix merge conflicts in 2 days.
    - shivarama.rao@amd.com can you have a look?
    - Further PRs expected within 2 weeks
- https://github.com/flang-compiler/flang/pull/1094 : Add loop start, end to llvm.loop metadata. [Approval from Bryan, Shivaram. LGTM from Chirag]
    - On hold
- Tealeaf issue
    - Arm has shared the list of downstream reverts: https://github.com/flang-compiler/flang/issues/1024
    - Shivaram will provide update
    - Threadlocal workaround. Internal variables created by the compiler and they need to be declared threadlocal.
- Windows
    - A label was created (Do we need labels for anything else?)
    - Windows PR list. https://github.com/flang-compiler/flang/issues/1154. Only three more required to get Windows to compile.
- MacOS
    - https://github.com/flang-compiler/flang/pull/1120 [Paul/Arm approved]
    - Driver patches merged
- Status of PRs
  AMD:
  https://github.com/flang-compiler/flang/pull/898: [DebugInfo] Corrected debuginfo for modules. Dependency on LLVM11. [Bryan approved. Can proceed after removing llvm-10 CI]
  DebugInfo: 1140 (can skip, can close after 1141),1141,1142,1143 (module variable different file issue)
  https://github.com/flang-compiler/flang/pull/1147
  String debug with llvm-11. Will try to raise PRs this week.

  Arm:
  **https://github.com/flang-compiler/flang/pull/1094** : Add loop start, end to llvm.loop metadata. [Arm accepted]. Useful to have it in the beginning of July. [AMD, Huawei approved. One question from Kiran]

https://github.com/flang-compiler/flang/pull/146 : Machreg Refactor. [Bryan approved. Paul provided info that AMD requested].
https://github.com/flang-compiler/flang/pull/659 : Changes to support transpose intrinsic in initialization. [Bryan has comments]

Huawei: (PRs are sorted in order)
**(high priority; would like to enable -Werror sooner rather than later)**
https://github.com/flang-compiler/flang/issues/1065 : LLVM 12 : All PRs mentioned.
https://github.com/flang-compiler/flang/pull/1100 : Incorrect behavior when passing a polymorphic array as an argument to a function. [Ready to merge. Pawel has a request.]
https://github.com/flang-compiler/flang/pull/1123: Use const char* for commandline args. [Waiting for approval from Shivaram]
https://github.com/flang-compiler/flang/pull/1129 : quad precision for intrinsics
https://github.com/flang-compiler/flang/pull/1132 : Format issues?
++++++++++++++++++++++++++++++++++++++ ^Target for next weeks call
https://github.com/flang-compiler/flang/pull/1063 : Correct the type of unlimited polymorphic pointers
https://github.com/flang-compiler/flang/pull/1102 : Fix bugs in array constructor with an implied-do containing an expression that is a function call returning an array
https://github.com/flang-compiler/flang/pull/1055 : Fix bugs in PACK intrinsic. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1064 : Provide v_list argument as a zero-sized array in some I/O cases as required by the standard.
https://github.com/flang-compiler/flang/pull/1053 : Fix bugs in getarg. [Huawei started review, and requested inputs from Mark.] Should we be consistent with gfortran, ifort?
https://github.com/flang-compiler/flang/pull/1104 : Fix incorrect code generation for a polymorphic derived type member that is given a runtime type of CHARACTER(*).
**(medium priority)**
https://github.com/flang-compiler/flang/pull/1088 : Fix a segmentation fault related to assumed length character functions.
https://github.com/flang-compiler/flang/pull/1052 : Fix a segmentation fault when the name of an ENTRY is used to return a value instead of the correct RESULT variable.
https://github.com/flang-compiler/flang/pull/1036 : Fix error of processing ENTRY statement. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1050 : Correct the type of unlimited polymorphic pointers.
https://github.com/flang-compiler/flang/pull/1089 : Prevent illegal argument accessing during procedure exporting.
https://github.com/flang-compiler/flang/pull/1091 : Add more checks for derived types in COMMON statements (to avoid compile-time segmentation faults).
https://github.com/flang-compiler/flang/pull/1062 : Fix an ICE when a named constant array is declared deferred shape (and then used in RESHAPE for another array).

https://github.com/flang-compiler/flang/pull/1101 : Fix compile-time errors caused by empty derived type constructors nested in array constructors.
https://github.com/flang-compiler/flang/pull/1051 : Report two grammar errors pertaining to PROTECTED attribute.
https://github.com/flang-compiler/flang/pull/1105 : Suppress debugloc for actual argument bitcast. [AMD (jini) could not reproduce the issue]

Windows:
Other patches? **991** (Adam) needs a rebase.

MacOS:
https://github.com/flang-compiler/flang/pull/1120 [Paul/Arm approved]

## Oct 6, 2021

Attendees
AMD:
Arm:
Huawei:
Nvidia:
Others:

flang-compiler/flang

- Issue #1146 'Get the wrong variable after module USE statement', regression after PR#949
    - Collective decision needed: should the commit introduced by PR#949 be reverted? We decided to leave it as it is and wait for a fix.
- llvm-13 : Abraham (Arm/Mobica) is working on this.
    - The in26 test case failure (leak in SmallVector usage?)
    - x86_64 compilation issues
        - LLVM's init-aarch64.c test case failing on x86_64 (due to missing __SIZE_MAX__ define in the preprocessor output?)
        - segfault in X86 Assembly Printer when compiling flang's assumed_shaped_noopt.f90 test case
    - Any other failures?
- Quad-precision FP support in Flang. Huawei has submitted the first patch. The second patch was available.
    - Further PRs expected within 2 weeks
- https://github.com/flang-compiler/flang/pull/1094 : Add loop start, end to llvm.loop metadata. [Approval from Bryan, Shivaram. LGTM from Chirag]
    - On hold

- Tealeaf issue.
    - Arm has shared the list of downstream reverts: https://github.com/flang-compiler/flang/issues/1024
    - Shivaram will provide update

Windows: No update
MacOS: No update

## Sep 22, 2021

Attendees
AMD:
Arm:
Huawei:
Nvidia:
Others:

flang-compiler/flang

- LLVM 12 work for flang, classic-flang/classic-flang-llvm-project
  Bryan has now created several PRs. List available in https://github.com/flang-compiler/flang/issues/1065.
  Roll up patches:
  https://github.com/flang-compiler/flang/pull/1085 [In Draft mode]
  https://github.com/flang-compiler/flang/pull/1086 [Needs a rebase]
  https://github.com/flang-compiler/flang/pull/1087 [Review not requested] This is the runtime specific patch.
  Split into 3 patches ?
  https://github.com/flang-compiler/flang/pull/1130

- llvm-13 : Abraham (Arm/Mobica) is working on this. Updates?
  Progress: 11 -> 5 failing tests. Fixed issues with Options.td.
  Paul: Compare output of llvm-12 and llvm-13.
  Tests inside Flang.
  Shivaram: remaining 5 tests are similar?
  Paul: -J fix?
  Abraham: Took definition from release-13 and used the group (Fortran->gfortran_group) information from release-12.
  Kiran: Will create a branch in classic-flang-llvm-project and then Abraham can make a PR to that branch with his changes.

  Abraham: I'm still doing the debugging of the issues. I need your insight in the following, I have this snipped from a test that is passing:
  ```
  ! CHECK:  {{.*}} add nsw <[[VF:[0-9]+]] x i32>{{.*}}
  ```

```
! METADATA: load {{.*}}, !llvm.mem.parallel_loop_access ![[TAG1:[0-9]+]]
```

And this one is from a failing test

```
! CHECK-NOT:  {{.*}} add nsw <[[VF:[0-9]+]] x i32>{{.*}}
! METADATA-NOT: load {{.*}}, !llvm.mem.parallel_loop_access ![[TAG1:[0-9]+]]
```

The test gives error saying that "TAG1" is not being declared.
The negation on CHECK and METADATA, is it like the TAG variable takes a negative value? as the inverse?

- llvm-10 CI removal.
  Abraham: Already done and merged.

- Bryan:  Quad-precision FP support in Flang. Huawei has submitted the first patch. The second patch is now available.
  https://github.com/flang-compiler/flang/pull/1129 : quad precision for intrinsics
  https://github.com/flang-compiler/flang/pull/1132 : Format issues?
  Kiran Chandramohan suggests going ahead (as in https://github.com/flang-compiler/flang/pull/1125 ) without AMD approval since the feature is not enabled on  x86_64.
  AMD is OK.
- https://github.com/flang-compiler/flang/pull/1094 : Add loop start, end to llvm.loop metadata. [Approval from Bryan, Shivaram. LGTM from Chirag]
  Kiran Chandramohan I would like to understand what fix_nodepcheck_flag does before completely removing it from the flow. Alternatively, introduce a fix_access_group_flag function in place of fix_nodepcheck_flag.
  Kiran Chandramohan to have a look and then approve.

- Procedure pointer issues. Can patches be upstreamed to fix issues here. (AMD to Nvidia)
  Shivaram: Working to create a minimum reproducer
  Kiran: Try to share list of reverts for Tealeaf.

- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/898: [DebugInfo] Corrected debuginfo for modules. Dependency on LLVM11. [Bryan approved. Can proceed after removing llvm-10 CI]
  DebugInfo: 1140 (can skip, can close after 1141),1141,1142,1143 (module variable different file issue)
  String debug with llvm-11. Will try to raise PRs this week.

Arm:
**https://github.com/flang-compiler/flang/pull/1094** : Add loop start, end to llvm.loop metadata. [Arm accepted]. Useful to have it in the beginning of July. [AMD, Huawei approved. One question from Kiran]
https://github.com/flang-compiler/flang/pull/146 : Machreg Refactor. [Bryan approved. Paul provided info that AMD requested].
https://github.com/flang-compiler/flang/pull/659 : Changes to support transpose intrinsic in initialization. [Bryan has comments]

Huawei: (PRs are sorted in order)
**(high priority; would like to enable -Werror sooner rather than later)**
https://github.com/flang-compiler/flang/issues/1065 : LLVM 12 : All PRs mentioned.
https://github.com/flang-compiler/flang/pull/1100 : Incorrect behavior when passing a polymorphic array as an argument to a function. [Ready to merge. Pawel has a request.]
https://github.com/flang-compiler/flang/pull/1123: Use const char* for commandline args. [Waiting for approval from Shivaram]
https://github.com/flang-compiler/flang/pull/1129 : quad precision for intrinsics
https://github.com/flang-compiler/flang/pull/1132 : Format issues?
+++++++++++++++++++++++++++++++++++++ ^Target for next weeks call
https://github.com/flang-compiler/flang/pull/1063 : Correct the type of unlimited polymorphic pointers
https://github.com/flang-compiler/flang/pull/1102 : Fix bugs in array constructor with an implied-do containing an expression that is a function call returning an array
https://github.com/flang-compiler/flang/pull/1055 : Fix bugs in PACK intrinsic. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1064 : Provide v_list argument as a zero-sized array in some I/O cases as required by the standard.
https://github.com/flang-compiler/flang/pull/1053 : Fix bugs in getarg. [Huawei started review, and requested inputs from Mark.] Should we be consistent with gfortran, ifort?
https://github.com/flang-compiler/flang/pull/1104 : Fix incorrect code generation for a polymorphic derived type member that is given a runtime type of CHARACTER(*).
**(medium priority)**
https://github.com/flang-compiler/flang/pull/1088 : Fix a segmentation fault related to assumed length character functions.
https://github.com/flang-compiler/flang/pull/1052 : Fix a segmentation fault when the name of an ENTRY is used to return a value instead of the correct RESULT variable.
https://github.com/flang-compiler/flang/pull/1036 : Fix error of processing ENTRY statement. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1050 : Correct the type of unlimited polymorphic pointers.
https://github.com/flang-compiler/flang/pull/1089 : Prevent illegal argument accessing during procedure exporting.

https://github.com/flang-compiler/flang/pull/1091 : Add more checks for derived types in COMMON statements (to avoid compile-time segmentation faults).
https://github.com/flang-compiler/flang/pull/1062 : Fix an ICE when a named constant array is declared deferred shape (and then used in RESHAPE for another array).
https://github.com/flang-compiler/flang/pull/1101 : Fix compile-time errors caused by empty derived type constructors nested in array constructors.
https://github.com/flang-compiler/flang/pull/1051 : Report two grammar errors pertaining to PROTECTED attribute.
https://github.com/flang-compiler/flang/pull/1105 : Suppress debugloc for actual argument bitcast. [AMD (jini) could not reproduce the issue]

Windows:
Other patches? **991** (Adam) needs a rebase.

MacOS:
https://github.com/flang-compiler/flang/pull/1120 [Paul/Arm approved]

## Sep 8, 2021

Attendees
AMD: Shivaram, Kiran, Jini
Arm: Kiran, Pawel
Huawei: Joey, Bryan
Nvidia:
Others: Abraham (Mobica/Arm)

flang-compiler/flang

- LLVM 12 work for flang, classic-flang/classic-flang-llvm-project
  Bryan has now created several PRs. List available in https://github.com/flang-compiler/flang/issues/1065.
  Roll up patches:
  https://github.com/flang-compiler/flang/pull/1085 [In Draft mode]
  https://github.com/flang-compiler/flang/pull/1086 [Needs a rebase]
  https://github.com/flang-compiler/flang/pull/1087 [Review not requested] This is the runtime specific patch.
  Split into 3 patches ?
  https://github.com/flang-compiler/flang/pull/1123: Use const char* for commandline args. [Waiting for approval from Shivaram. Internal testing at AMD]
- llvm-13 : Abraham (Arm/Mobica) is working on this. Updates? Can michal.paszta@mobica.com switch OFF the llvm-10 bot and use it for llvm-13?

Finished rebase with llvm-13. Pushed code to private repo. Has an error with "-J" (option in gfortran) in clang-13. Look into integration.
Driver work in f18 removed -J. Backport some code from llvm-12 branch. Some issue with flang being ON in release-13.
Kiran Chandramohan look into -J and the default flang (script/binary).
There is a driver channel.

- Bryan: Quad-precision FP support in Flang. Huawei has submitted the first patch. The second patch is now available.
  https://github.com/flang-compiler/flang/pull/1129 : quad precision for intrinsics
  https://github.com/flang-compiler/flang/pull/1132 : Format issues?
  Kiran Chandramohan suggests going ahead (as in https://github.com/flang-compiler/flang/pull/1125 ) without AMD approval since the feature is not enabled on x86_64.
  AMD is OK.
- https://github.com/flang-compiler/flang/pull/1094 : Add loop start, end to llvm.loop metadata. [Approval from Bryan, Shivaram. LGTM from Chirag]
  Kiran Chandramohan I would like to understand what fix_nodepcheck_flag does before completely removing it from the flow. Alternatively, introduce a fix_access_group_flag function in place of fix_nodepcheck_flag.
  Kiran Chandramohan to have a look and then approve.
- CCFF
  Used in Minfo. Previously in profiler, but not anymore. User hints (vectorisation etc).
- Procedure pointer issues. Can patches be upstreamed to fix issues here.
  Shivaram please file an issue and tag Mark.
- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/898: [DebugInfo] Corrected debuginfo for modules. Dependency on LLVM11. LLVM10 bots failing, need to reach consensus on CI builds for PR having version specific dependencies. [Bryan approved. Can proceed after removing llvm-10 CI]
  String debug with llvm-11.

  Arm:
  **https://github.com/flang-compiler/flang/pull/1094** : Add loop start, end to llvm.loop metadata. [Arm accepted]. Useful to have it in the beginning of July. [AMD, Huawei approved. One question from Kiran]
  https://github.com/flang-compiler/flang/pull/146 : Machreg Refactor. [Bryan approved. Paul provided info that AMD requested].
  https://github.com/flang-compiler/flang/pull/655 : Add missing conversion in Data statement. [Bryan has comments]
  https://github.com/flang-compiler/flang/pull/659 : Changes to support transpose intrinsic in initialization. [Bryan has comments]

Huawei: (PRs are sorted in order)
**(high priority; would like to enable -Werror sooner rather than later)**
https://github.com/flang-compiler/flang/issues/1065 : LLVM 12 : All PRs mentioned.
https://github.com/flang-compiler/flang/pull/1100 : Incorrect behavior when passing a polymorphic array as an argument to a function. [Ready to merge. Pawel has a request.]
https://github.com/flang-compiler/flang/pull/1123: Use const char* for commandline args. [Waiting for approval from Shivaram]
https://github.com/flang-compiler/flang/pull/1129 : quad precision for intrinsics
https://github.com/flang-compiler/flang/pull/1132 : Format issues?
++++++++++++++++++++++++++++++++++++ ^Target for next weeks call
https://github.com/flang-compiler/flang/pull/1063 : Correct the type of unlimited polymorphic pointers
https://github.com/flang-compiler/flang/pull/1102 : Fix bugs in array constructor with an implied-do containing an expression that is a function call returning an array
https://github.com/flang-compiler/flang/pull/1055 : Fix bugs in PACK intrinsic. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1064 : Provide v_list argument as a zero-sized array in some I/O cases as required by the standard.
https://github.com/flang-compiler/flang/pull/1053 : Fix bugs in getarg. [Huawei started review, and requested inputs from Mark.] Should we be consistent with gfortran, ifort?
https://github.com/flang-compiler/flang/pull/1104 : Fix incorrect code generation for a polymorphic derived type member that is given a runtime type of CHARACTER(*).
**(medium priority)**
https://github.com/flang-compiler/flang/pull/1088 : Fix a segmentation fault related to assumed length character functions.
https://github.com/flang-compiler/flang/pull/1052 : Fix a segmentation fault when the name of an ENTRY is used to return a value instead of the correct RESULT variable.
https://github.com/flang-compiler/flang/pull/1036 : Fix error of processing ENTRY statement. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1050 : Correct the type of unlimited polymorphic pointers.
https://github.com/flang-compiler/flang/pull/1089 : Prevent illegal argument accessing during procedure exporting.
https://github.com/flang-compiler/flang/pull/1091 : Add more checks for derived types in COMMON statements (to avoid compile-time segmentation faults).
https://github.com/flang-compiler/flang/pull/1062 : Fix an ICE when a named constant array is declared deferred shape (and then used in RESHAPE for another array).
https://github.com/flang-compiler/flang/pull/1101 : Fix compile-time errors caused by empty derived type constructors nested in array constructors.
https://github.com/flang-compiler/flang/pull/1051 : Report two grammar errors pertaining to PROTECTED attribute.

https://github.com/flang-compiler/flang/pull/1105 : Suppress debugloc for actual argument bitcast. [AMD (jini) could not reproduce the issue]

Windows:
Other patches? **991** (Adam) needs a rebase.

MacOS:
https://github.com/flang-compiler/flang/pull/1120 [Paul/Arm approved]

## Aug 25, 2021

Attendees
AMD: Shivaram, Alok, Chirag, Kiran Kumar, Bhuvan
Arm: Pawel, Kiran
Huawei: Bryan
Nvidia: Mark
Others:

flang-compiler/flang

- LLVM 12 work for flang, classic-flang/classic-flang-llvm-project
  Bryan has now created several PRs. List available in https://github.com/flang-compiler/flang/issues/1065.
  Roll up patches:
  https://github.com/flang-compiler/flang/pull/1085 [In Draft mode]
  https://github.com/flang-compiler/flang/pull/1086 [Needs a rebase]
  https://github.com/flang-compiler/flang/pull/1087 [Review not requested] This is the runtime specific patch.
  Split into 3 patches ?
  https://github.com/flang-compiler/flang/pull/1123: Use const char* for commandline args. [Waiting for approval from Shivaram]
  https://github.com/flang-compiler/flang/pull/1124: Use const char* in file generation tools.
  https://github.com/flang-compiler/flang/pull/1122: Fix member_with_offset in outliner.cpp
  Six or more PRs to review. Mechanical changes. static_cast might need some more work.
- Bryan: Quad-precision FP support in Flang. Huawei has refactored the first patch. The first patch is now available.
  https://github.com/flang-compiler/flang/pull/1125  [Waiting for approval from Shivaram]
  AMD uses quadmath so do not have changes in pgmath.
  Will need help from AMD to enable support on x86_64.
- https://github.com/flang-compiler/flang/pull/1094 : Add loop start, end to llvm.loop metadata. [Approval from Bryan. LGTM from Chirag, can Shivaram approve?]

- Implied Shape Arrays. https://github.com/flang-compiler/flang/issues/893 Older versions of cp2k does not use this feature.
- llvm-13 : Abraham (Arm/Mobica) is working on this. Till when will we need llvm-10? Kiran to ask in classic-flang/pull request channel to huawei. OK to remove support for llvm-10 next month.
- Clang-format all files : potential conflicts, history change etc.
- Any other points?

- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/898: [DebugInfo] Corrected debuginfo for modules. Dependency on LLVM11. LLVM10 bots failing, need to reach consensus on CI builds for PR having version specific dependencies. [AMD/Bryan how to proceed?]
  https://github.com/flang-compiler/flang/pull/1111: Fix inline code for size intrinsic for negative strides. [Application issue reduced to unit test.]

  Arm:
  **https://github.com/flang-compiler/flang/pull/1094** : Add loop start, end to llvm.loop metadata. [Arm accepted]. Useful to have it in the beginning of July. Appreciate reviews.
  https://github.com/flang-compiler/flang/pull/146 : Machreg Refactor. [No reviews. Paul provided info that AMD requested].
  https://github.com/flang-compiler/flang/pull/655 : Add missing conversion in Data statement. [No reviews]
  https://github.com/flang-compiler/flang/pull/659 : Changes to support transpose intrinsic in initialization. [No reviews]

  Huawei: (PRs are sorted in order)
  **(high priority; would like to enable -Werror sooner rather than later)**
  https://github.com/flang-compiler/flang/issues/1065 : LLVM 12 : All PRs mentioned.
  https://github.com/flang-compiler/flang/pull/1037. Fix issue of processing character variables via the DATA statement. [Ready for merge]
  https://github.com/flang-compiler/flang/pull/1049 : NEAREST returns wrong value for +/- infinity. [Ready for merge]
  https://github.com/flang-compiler/flang/pull/1100 : Incorrect behavior when passing a polymorphic array as an argument to a function. [Waiting for AMD approval]
  https://github.com/flang-compiler/flang/pull/1123: Use const char* for commandline args. [Waiting for approval from Shivaram]
  https://github.com/flang-compiler/flang/pull/1124: Use const char* in file generation tools.
  https://github.com/flang-compiler/flang/pull/1122: Fix member_with_offset in outliner.cpp

++++++++++++++++++++++++++++++++++++ ^Target for next weeks call
https://github.com/flang-compiler/flang/pull/1063 : Correct the type of unlimited polymorphic pointers
https://github.com/flang-compiler/flang/pull/1102 : Fix bugs in array constructor with an implied-do containing an expression that is a function call returning an array
https://github.com/flang-compiler/flang/pull/1055 : Fix bugs in PACK intrinsic. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1064 : Provide v_list argument as a zero-sized array in some I/O cases as required by the standard.
https://github.com/flang-compiler/flang/pull/1053 : Fix bugs in getarg. [Huawei started review, and requested inputs from Mark.] Should we be consistent with gfortran, ifort?
https://github.com/flang-compiler/flang/pull/1104 : Fix incorrect code generation for a polymorphic derived type member that is given a runtime type of CHARACTER(*).
**(medium priority)**
https://github.com/flang-compiler/flang/pull/1088 : Fix a segmentation fault related to assumed length character functions.
https://github.com/flang-compiler/flang/pull/1052 : Fix a segmentation fault when the name of an ENTRY is used to return a value instead of the correct RESULT variable.
https://github.com/flang-compiler/flang/pull/1036 : Fix error of processing ENTRY statement. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1050 : Correct the type of unlimited polymorphic pointers.
https://github.com/flang-compiler/flang/pull/1089 : Prevent illegal argument accessing during procedure exporting.
https://github.com/flang-compiler/flang/pull/1091 : Add more checks for derived types in COMMON statements (to avoid compile-time segmentation faults).
https://github.com/flang-compiler/flang/pull/1062 : Fix an ICE when a named constant array is declared deferred shape (and then used in RESHAPE for another array).
https://github.com/flang-compiler/flang/pull/1101 : Fix compile-time errors caused by empty derived type constructors nested in array constructors.
https://github.com/flang-compiler/flang/pull/1051 : Report two grammar errors pertaining to PROTECTED attribute.
https://github.com/flang-compiler/flang/pull/1105 : Suppress debugloc for actual argument bitcast. [AMD (jini) could not reproduce the issue]

Windows:
Other patches? **991** (Adam) needs a rebase. Hello world, some tests working. All these in test branch. Now upstreaming everything. After that will look into tests.

MacOS:
https://github.com/flang-compiler/flang/pull/1120

## Aug 11, 2021

Attendees
AMD: Shivaram, Kiran Kumar, Jini, Chirag
Arm: Kiran
Huawei: Bryan
Nvidia: Mark
Others: Abraham (Mobica/Arm)

flang-compiler/flang

- LLVM 12 work for flang, classic-flang/classic-flang-llvm-project
  Bryan has now created several PRs. List available in
  https://github.com/flang-compiler/flang/issues/1065.
  Roll up patches:
  https://github.com/flang-compiler/flang/pull/1085 [Changes requested by Arm]
  Bryan will put up a fix which addresses all comments in a day or so.
  Others might be independent.
  https://github.com/flang-compiler/flang/pull/1086 [Needs a rebase]
  https://github.com/flang-compiler/flang/pull/1087 [Review not requested] This is the runtime specific patch.
- Any plans for LLVM 13? AMD not working on it. Huawei uses even number releases. Arm will look into this.
- AMD reviewer update. Shivaram will continue reviewing.
- Bryan: Quad-precision FP support in Flang. Any updates here.
  Huawei has refactored the first patch. Rebasing on master has some x86 build issues. Will push a patch after fixing these issues.
- **https://github.com/flang-compiler/flang/pull/1094** : Add loop start, end to llvm.loop metadata. Need help here from both AMD and Huawei.
  Is the patch too big? Should it be split into
  i) updating all directives to new metadata
  ii) Adding loop start, end to llvm.loop metadata and refactoring
  Chirag had a look and it looked OK. Bryan had a look and will hopefully have some feedback before next call.
  Intermittent failure of mp_correct/lit/pv01.sh (possibly not related to this patch. Error occurred when allocating in parallel region.)
- Omp simd is supported without clauses. Usage of if, private clause etc could lead to errors. We should put out warnings and disable omp simd behaviour if these clauses are present.
  Was any work done by AMD for these clauses?
  Arm might put up a patch. (Check behaviour)
- Any other points?
- Status of PRs being reviewed (flang, flang-driver, llvm)

AMD:
https://github.com/flang-compiler/flang/pull/898: [DebugInfo] Corrected debuginfo for modules. Dependency on LLVM11. LLVM10 bots failing, need to reach consensus on CI builds for PR having version specific dependencies. [AMD/Bryan how to proceed?]
**https://github.com/flang-compiler/flang/pull/1057: Fix debuginfo in derived type having scalar pointer member. [AMD, Huawei accepted] Arm? Kiran will look into this today.**

Arm:
**https://github.com/flang-compiler/flang/pull/1094** : Add loop start, end to llvm.loop metadata. [Arm accepted]. Useful to have it in the beginning of July. Appreciate reviews.
**https://github.com/flang-compiler/flang/pull/1113** : Handle atomic loads of logic variables of different size. [Arm approved, others?]
https://github.com/flang-compiler/flang/pull/146 : Machreg Refactor. [No reviews. Paul provided info that AMD requested].
https://github.com/flang-compiler/flang/pull/655 : Add missing conversion in Data statement. [No reviews]
https://github.com/flang-compiler/flang/pull/659 : Changes to support transpose intrinsic in initialization. [No reviews]

Huawei: (PRs are sorted in order)
**(high priority; would like to enable -Werror sooner rather than later)**
https://github.com/flang-compiler/flang/issues/1065 : LLVM 12 : All PRs mentioned.
**https://github.com/flang-compiler/flang/pull/1114** : CP2K issue (Bug in exporting iso_fortran_env). Shivaram please have a look today.
https://github.com/flang-compiler/flang/pull/1040. Do not optimize common block variable that is potentially aliased [Huawei reviewed]
https://github.com/flang-compiler/flang/pull/1037. Fix issue of processing character variables via the DATA statement. [Arm, Huawei accepted] Waiting for AMD's acceptance.
https://github.com/flang-compiler/flang/pull/1049 : NEAREST returns wrong value for +/- infinity
https://github.com/flang-compiler/flang/pull/1100 : Incorrect behavior when passing a polymorphic array as an argument to a function
++++++++++++++++++++++++++++++++++++++ ^Target for next weeks call
https://github.com/flang-compiler/flang/pull/1063 : Correct the type of unlimited polymorphic pointers
https://github.com/flang-compiler/flang/pull/1102 : Fix bugs in array constructor with an implied-do containing an expression that is a function call returning an array
https://github.com/flang-compiler/flang/pull/1055 : Fix bugs in PACK intrinsic. [Arm accepted]

https://github.com/flang-compiler/flang/pull/1064 : Provide v_list argument as a zero-sized array in some I/O cases as required by the standard.
https://github.com/flang-compiler/flang/pull/1053 : Fix bugs in getarg. [Huawei started review, and requested inputs from Mark.] Should we be consistent with gfortran, ifort?
https://github.com/flang-compiler/flang/pull/1104 : Fix incorrect code generation for a polymorphic derived type member that is given a runtime type of CHARACTER(*).
**(medium priority)**
https://github.com/flang-compiler/flang/pull/1088 : Fix a segmentation fault related to assumed length character functions.
https://github.com/flang-compiler/flang/pull/1052 : Fix a segmentation fault when the name of an ENTRY is used to return a value instead of the correct RESULT variable.
https://github.com/flang-compiler/flang/pull/1036 : Fix error of processing ENTRY statement. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1050 : Correct the type of unlimited polymorphic pointers.
https://github.com/flang-compiler/flang/pull/1089 : Prevent illegal argument accessing during procedure exporting.
https://github.com/flang-compiler/flang/pull/1091 : Add more checks for derived types in COMMON statements (to avoid compile-time segmentation faults).
https://github.com/flang-compiler/flang/pull/1062 : Fix an ICE when a named constant array is declared deferred shape (and then used in RESHAPE for another array).
https://github.com/flang-compiler/flang/pull/1101 : Fix compile-time errors caused by empty derived type constructors nested in array constructors.
https://github.com/flang-compiler/flang/pull/1051 : Report two grammar errors pertaining to PROTECTED attribute.
https://github.com/flang-compiler/flang/pull/1105 : Suppress debugloc for actual argument bitcast. [AMD (jini) could not reproduce the issue]

Windows:
Other patches? **991** (Adam) needs a rebase. Hello world, some tests working. All these in test branch. Now upstreaming everything. After that will look into tests.

## Jul 28, 2021

Attendees
AMD: Shivaram,Alok
Arm: Kiran, Paul, Rich
Huawei: Bryan
Nvidia: Mark
Others:

flang-compiler/flang

- LLVM 12 work for flang, classic-flang/classic-flang-llvm-project
  Bryan has now created several PRs. List available in
  https://github.com/flang-compiler/flang/issues/1065.
  https://github.com/flang-compiler/flang/pull/1092 (Ping Shivaram)
  Roll up patches (1085, 1086, 1087) dependent on the above PR.
  Shivaram will look at today.
- Sourabh has left AMD. Not available for Flang activities. AMD will try to find someone soon. Alok will take up the PRs and requests. 8 people at AMD working on Flang.
- Bryan: Quad-precision FP support in Flang. Any updates here.
  PR will be based on warning patches. Currently refactoring patches.
- **https://github.com/flang-compiler/flang/pull/1094** : Add loop start, end to llvm.loop metadata. Need help here from both AMD and Huawei.
  Bryan and Chirag will have a look.
  Backwards compatibility llvm-10. We can also make a statement on the oldest version supported. Michal: Checked that it (metadata) is present in llvm-10.
- General discussion on switching to llvm/flang. Fortran 95 support will be complete soon. Fortran 2003 (in the worst-case) will take one more year. Since classic flang is mostly Fortran 2003, would the end of next year be a good time to switch? Does anyone have a timeline to switch?
  Arm : no current plans.
  Huawei has 2008 features (more intrinsics, 15 dimensions). Switch after new-flang supports 2008. Performance parity also important. OpenMP 4.x support with llvm/flang. For GPU support we need target. Same considerations for AMD.
  Will be a gradual change in focus.
- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/898: [DebugInfo] Corrected debuginfo for modules. Dependency on LLVM11. LLVM10 bots failing, need to reach consensus on CI builds for PR having version specific dependencies. [AMD/Bryan how to proceed?]
  **https://github.com/flang-compiler/flang/pull/1083**: Include compiler_options in DW_AT_producer. [Huawei, AMD accepted] Arm? Low hanging fruit.
  **https://github.com/flang-compiler/flang/pull/1057**: Fix debuginfo in derived type having scalar pointer member. [AMD, Huawei accepted] Arm?
  https://github.com/flang-compiler/flang/pull/950: Do not emit function return type as pointer always. [A merge patch is present in the commit, please correct.]
  Any other patches?

  Arm:
  **https://github.com/flang-compiler/flang/pull/1094** : Add loop start, end to llvm.loop metadata. [Arm accepted]. Useful to have it in the beginning of July. Appreciate reviews.
  **https://github.com/flang-compiler/flang/pull/1110** : Fix for flang misses pure declaration in the contains section (https://github.com/flang-compiler/flang/issues/1016).

CP2K/dbscr issue.
https://github.com/flang-compiler/flang/pull/146 : Machreg Refactor. [No reviews. Paul provided info that AMD requested].
https://github.com/flang-compiler/flang/pull/655 : Add missing conversion in Data statement. [No reviews]
https://github.com/flang-compiler/flang/pull/659 : Changes to support transpose intrinsic in initialization. [No reviews]

Huawei: (PRs are sorted in order)
**(high priority; would like to enable -Werror sooner rather than later)**
https://github.com/flang-compiler/flang/issues/1065 : LLVM 12 : All PRs mentioned.
https://github.com/flang-compiler/flang/pull/1040. Do not optimize common block variable that is potentially aliased [Huawei reviewed]
https://github.com/flang-compiler/flang/pull/1037. Fix issue of processing character variables via the DATA statement. [Arm, Huawei accepted] Waiting for AMD's acceptance.
https://github.com/flang-compiler/flang/pull/1049 : NEAREST returns wrong value for +/- infinity
https://github.com/flang-compiler/flang/pull/1100 : Incorrect behavior when passing a polymorphic array as an argument to a function
https://github.com/flang-compiler/flang/pull/1063 : Correct the type of unlimited polymorphic pointers
https://github.com/flang-compiler/flang/pull/1102 : Fix bugs in array constructor with an implied-do containing an expression that is a function call returning an array
https://github.com/flang-compiler/flang/pull/1055 : Fix bugs in PACK intrinsic. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1064 : Provide v_list argument as a zero-sized array in some I/O cases as required by the standard.
https://github.com/flang-compiler/flang/pull/1053 : Fix bugs in getarg. [Huawei started review, and requested inputs from Mark.] Should we be consistent with gfortran, ifort?
https://github.com/flang-compiler/flang/pull/1104 : Fix incorrect code generation for a polymorphic derived type member that is given a runtime type of CHARACTER(*).
**(medium priority)**
https://github.com/flang-compiler/flang/pull/1088 : Fix a segmentation fault related to assumed length character functions.
https://github.com/flang-compiler/flang/pull/1052 : Fix a segmentation fault when the name of an ENTRY is used to return a value instead of the correct RESULT variable.
https://github.com/flang-compiler/flang/pull/1036 : Fix error of processing ENTRY statement. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1050 : Correct the type of unlimited polymorphic pointers.
https://github.com/flang-compiler/flang/pull/1089 : Prevent illegal argument accessing during procedure exporting.

https://github.com/flang-compiler/flang/pull/1091 : Add more checks for derived types in COMMON statements (to avoid compile-time segmentation faults).
https://github.com/flang-compiler/flang/pull/1062 : Fix an ICE when a named constant array is declared deferred shape (and then used in RESHAPE for another array).
https://github.com/flang-compiler/flang/pull/1101 : Fix compile-time errors caused by empty derived type constructors nested in array constructors.
https://github.com/flang-compiler/flang/pull/1051 : Report two grammar errors pertaining to PROTECTED attribute.
https://github.com/flang-compiler/flang/pull/1105 : Suppress debugloc for actual argument bitcast. [AMD (jini) could not reproduce the issue]

Windows:
https://github.com/flang-compiler/flang/pull/1084 : Windows fixes. [Arm accepted, Windows reviewer accepted, Bryan requested changes]
Other patches? **464, 991** (Adam)

## Jul 14, 2021

Attendees
AMD:
Arm:
Huawei:
Nvidia:
Others:

flang-compiler/flang

- LLVM 12 work for flang, classic-flang/classic-flang-llvm-project
  Bryan has now created several PRs. List available in https://github.com/flang-compiler/flang/issues/1065. Pawel (Arm) and Shivaram (AMD) reviewed several of these. What is the status?
  Bryan: 6 PRs remaining. 3 easy, 3 roll up.
  Shivaram: next couple of days. Issue created for hash_table.
- Bryan: Quad-precision FP support in Flang?
- Question in slack channel.
  Justs Zarins : is there a way to avoid removal of all-zero GEPs when generating IR? It seems to replace those with just loading the base pointer. This makes sense, but would require a special case to handle.
- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/898: [DebugInfo] Corrected debuginfo for modules. Dependency on LLVM11. LLVM10 bots failing, need to reach consensus on CI builds for PR having version specific dependencies. [AMD/Bryan how to proceed?]

**https://github.com/flang-compiler/flang/pull/1083**: Include compiler_options in DW_AT_producer. [Huawei, AMD accepted] Arm? Low hanging fruit.
https://github.com/flang-compiler/flang/pull/1057: Fix debuginfo in derived type having scalar pointer member. [AMD, Huawei accepted] Arm?
https://github.com/flang-compiler/flang/pull/950: Do not emit function return type as pointer always. [A merge patch is present in the commit, please correct.]
Any other patches?

Arm:
https://github.com/flang-compiler/flang/pull/146 : Machreg Refactor. [No reviews. Paul provided info that AMD requested].
**https://github.com/flang-compiler/flang/pull/1094** : Add loop start, end to llvm.loop metadata. [Arm accepted]. Useful to have it in the beginning of July. Appreciate reviews.
https://github.com/flang-compiler/flang/pull/655 : Add missing conversion in Data statement. [No reviews]
https://github.com/flang-compiler/flang/pull/659 : Changes to support transpose intrinsic in initialization. [No reviews]

Huawei:
**(high priority; would like to enable -Werror sooner rather than later)**
https://github.com/flang-compiler/flang/issues/1065 : LLVM 12 : All PRs mentioned.
https://github.com/flang-compiler/flang/pull/1040. Do not optimize common block variable that is potentially aliased [Huawei reviewed]
https://github.com/flang-compiler/flang/pull/1037. Fix issue of processing character variables via the DATA statement. [Arm, Huawei accepted] Waiting for AMD's acceptance.
https://github.com/flang-compiler/flang/pull/1049 : NEAREST returns wrong value for +/- infinity
https://github.com/flang-compiler/flang/pull/1100 : Incorrect behavior when passing a polymorphic array as an argument to a function
https://github.com/flang-compiler/flang/pull/1063 : Correct the type of unlimited polymorphic pointers
https://github.com/flang-compiler/flang/pull/1102 : Fix bugs in array constructor with an implied-do containing an expression that is a function call returning an array
https://github.com/flang-compiler/flang/pull/1055 : Fix bugs in PACK intrinsic. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1064 : Provide v_list argument as a zero-sized array in some I/O cases as required by the standard.
https://github.com/flang-compiler/flang/pull/1053 : Fix bugs in getarg. [Huawei started review, and requested inputs from Mark.] Should we be consistent with gfortran, ifort?
https://github.com/flang-compiler/flang/pull/1104 : Fix incorrect code generation for a polymorphic derived type member that is given a runtime type of CHARACTER(*).

**(medium priority)**
https://github.com/flang-compiler/flang/pull/1088 : Fix a segmentation fault related to assumed length character functions.
https://github.com/flang-compiler/flang/pull/1052 : Fix a segmentation fault when the name of an ENTRY is used to return a value instead of the correct RESULT variable.
https://github.com/flang-compiler/flang/pull/1036 : Fix error of processing ENTRY statement. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1050 : Correct the type of unlimited polymorphic pointers.
https://github.com/flang-compiler/flang/pull/1089 : Prevent illegal argument accessing during procedure exporting.
https://github.com/flang-compiler/flang/pull/1091 : Add more checks for derived types in COMMON statements (to avoid compile-time segmentation faults).
https://github.com/flang-compiler/flang/pull/1062 : Fix an ICE when a named constant array is declared deferred shape (and then used in RESHAPE for another array).
https://github.com/flang-compiler/flang/pull/1101 : Fix compile-time errors caused by empty derived type constructors nested in array constructors.
https://github.com/flang-compiler/flang/pull/1051 : Report two grammar errors pertaining to PROTECTED attribute.
https://github.com/flang-compiler/flang/pull/1105 : Suppress debugloc for actual argument bitcast. [AMD (jini) could not reproduce the issue]

Windows:
https://github.com/flang-compiler/flang/pull/1084 : Windows fixes. [Arm accepted, Windows reviewer accepted, Bryan requested changes]
Other patches? **464, 991** (Adam)

## Jun 30, 2021

Attendees
AMD: Shivaram, Jini, Sourabh, Chirag, Kiran Kumar, Raghu
Arm: Pawel, Kiran, Michal
Huawei: Bryan
Nvidia: Mark LeAir
Others: Adam (Szeged)

flang-compiler/flang

- LLVM 12 work for flang, classic-flang/classic-flang-llvm-project
  Bryan has now created several PRs. List available in https://github.com/flang-compiler/flang/issues/1065. Pawel (Arm) and Shivaram (AMD) reviewed several of these. What is the status?

Bryan: 8 PRs remaining. 5 easy, 3 roll up. Paul has requested changes. 1 of them requires reviewers. After merging Bryan will enable builds with Werror. Working around with pragmas : hashmap (ptr and const ptr), STB tables (sign compare warnings). Will create issues and someone can take it up if they are interested.
Paul: PR 1098 review.

- Huawei contractors joining for review. Will bring it up in next meeting with them.
  Peixin from Bryan's team is now reviewing. Contractors might join later.
- Bryan: Quad-precision FP support in Flang?
  Customer requirement on Arm servers. Implemented it and works OK. Can upstream it. Lot of code can upstream in stages. Compiler support first and intrinsics later.
  Shivaram: AMD has their own implementation. Libquadmath.
  Bryan: Libm already support quad precision. Generic implementation.
  Paul: PPC has some support.
  Shivaram: AMD can review changes.
  Can share.
- Question in slack channel.
  Justs Zarins : is there a way to avoid removal of all-zero GEPs when generating IR? It seems to replace those with just loading the base pointer. This makes sense, but would require a special case to handle.
  Request for more information, tests etc.
- In generated IR: Lot of bitcasts, void * in a function call and then again cast back inside functions.
  Originally written for C. Lot of mods to support fortran.
  Sourabh : Posted some links about opaque pointers. RTL calls are in i8*.
- Any other questions?
  Paul: BigDFT: 1013 bug. Michal has provided a comment.
- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/1033 : Support for -gpubnames in flang. (uses a reserved bit, important for AMD). [Arm accepted, driver changes for two branches pending].
  https://github.com/flang-compiler/flang/pull/898: [DebugInfo] Corrected debuginfo for modules. Dependency on LLVM11. LLVM10 bots failing, need to reach consensus on CI builds for PR having version specific dependencies. [AMD/Bryan how to proceed?]
  **https://github.com/flang-compiler/flang/pull/1083**: Include compiler_options in DW_AT_producer. [Huawei, AMD accepted] Arm? Low hanging fruit.
  https://github.com/flang-compiler/flang/pull/1057: Fix debuginfo in derived type having scalar pointer member. [AMD accepted] Hawei, Arm?
  https://github.com/flang-compiler/flang/pull/950: Do not emit function return type as pointer always. [A merge patch is present in the commit, please correct.]
  Any other patches?

Arm:
https://github.com/flang-compiler/flang/pull/146 : Machreg Refactor. [No reviews. Paul provided info that AMD requested].
**https://github.com/flang-compiler/flang/pull/1094** : Add loop start, end to llvm.loop metadata. [Arm accepted]. Useful to have it in the beginning of July. Appreciate reviews.
https://github.com/flang-compiler/flang/pull/655 : Add missing conversion in Data statement. [No reviews]
https://github.com/flang-compiler/flang/pull/659 : Changes to support transpose intrinsic in initialization. [No reviews]

Huawei:
**(high priority; would like to enable -Werror sooner rather than later)**
https://github.com/flang-compiler/flang/issues/1065 : LLVM 12 : All PRs mentioned.
**(high priority; incorrect run-time behaviour)**
https://github.com/flang-compiler/flang/pull/1098 : Set type length for a derived type member. Fix for https://github.com/flang-compiler/flang/issues/1059. [Arm, Huawei approved] AMD?
https://github.com/flang-compiler/flang/pull/1040. Fix constant propagation of common block variables. [No reviews]
https://github.com/flang-compiler/flang/pull/1037. Fix issue of processing character variables via the DATA statement. [Arm, Huawei accepted] Waiting for AMD's acceptance.
https://github.com/flang-compiler/flang/pull/1049 : NEAREST returns wrong value for +/- infinity
https://github.com/flang-compiler/flang/pull/1100 : Incorrect behavior when passing a polymorphic array as an argument to a function
https://github.com/flang-compiler/flang/pull/1063 : Correct the type of unlimited polymorphic pointers
https://github.com/flang-compiler/flang/pull/1102 : Fix bugs in array constructor with an implied-do containing an expression that is a function call returning an array
https://github.com/flang-compiler/flang/pull/1055 : Fix bugs in PACK intrinsic. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1064 : Provide v_list argument as a zero-sized array in some I/O cases as required by the standard.
https://github.com/flang-compiler/flang/pull/1053 : Fix bugs in getarg. [Huawei started review, and requested inputs from Mark.] Should we be consistent with gfortran, ifort?
https://github.com/flang-compiler/flang/pull/1104 : Fix incorrect code generation for a polymorphic derived type member that is given a runtime type of CHARACTER(*).
**(medium priority)**
https://github.com/flang-compiler/flang/pull/1088 : Fix a segmentation fault related to assumed length character functions.

https://github.com/flang-compiler/flang/pull/1052 : Fix a segmentation fault when the name of an ENTRY is used to return a value instead of the correct RESULT variable.
https://github.com/flang-compiler/flang/pull/1036 : Fix error of processing ENTRY statement. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1050 : Correct the type of unlimited polymorphic pointers.
https://github.com/flang-compiler/flang/pull/1089 : Prevent illegal argument accessing during procedure exporting.
https://github.com/flang-compiler/flang/pull/1091 : Add more checks for derived types in COMMON statements (to avoid compile-time segmentation faults).
https://github.com/flang-compiler/flang/pull/1062 : Fix an ICE when a named constant array is declared deferred shape (and then used in RESHAPE for another array).
https://github.com/flang-compiler/flang/pull/1101 : Fix compile-time errors caused by empty derived type constructors nested in array constructors.
https://github.com/flang-compiler/flang/pull/1051 : Report two grammar errors pertaining to PROTECTED attribute.
https://github.com/flang-compiler/flang/pull/1105 : Suppress debugloc for actual argument bitcast. [AMD (jini) could not reproduce the issue]

Windows:
https://github.com/flang-compiler/flang/pull/1084 : Windows fixes. [Arm accepted, Windows reviewer accepted, Bryan requested changes]
Other patches? **464, 991** (Adam)

## Jun 16, 2021

Attendees
AMD: Shivaram, Jini, Alok, Bhuvan, Chirag, Raghu
Arm: Pawel, Kiran
Huawei: Bryan, Joey
Nvidia:
Others:

flang-compiler/flang

- LLVM 12 work for flang, classic-flang/classic-flang-llvm-project
  Bryan has now created several PRs. List available in https://github.com/flang-compiler/flang/issues/1065. Pawel (Arm) reviewed several of these. How to proceed with these? Will AMD engineers have a look soon?
  Bryan: Around 12 PRs. Some of them are small. Some are big, but changes are mechanical.

Last 3 has CI failures. Depend on earlier changes. Too much rebasing required. Dealing with unused variables.
Pawel has a concern with a patch.
Bryan has some touch up patches coming soon.
https://github.com/flang-compiler/flang/pull/1090 : LLVM 12 porting issue: Make use of unwind, readnone, willreturn LLVM-IR attributes. Has two commits. Patch is for pure functions in pgmath.

- Using new LLVM loop access metadata.
  Bryan will have a look soon.
- Huawei contractors joining for review. Will bring it up in next meeting with them.
- https://github.com/flang-compiler/flang/issues/1059 : Regression (string not displayed) after PR#1009.
- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/1018 : Support for vector always directive with new llvm metadata. [No acceptance, comments addressed.]
  https://github.com/flang-compiler/flang/pull/1033 : Support for -gpubnames in flang. (uses a reserved bit, important for AMD). [Arm accepted, driver changes for two branches pending].
  https://github.com/flang-compiler/flang/pull/898: [DebugInfo] Corrected debuginfo for modules. Dependency on LLVM11. LLVM10 bots failing, need to reach consensus on CI builds for PR having version specific dependencies. [AMD/Bryan how to proceed?]
  https://github.com/flang-compiler/flang/pull/1083: Include compiler_options in DW_AT_producer. [No reviews]
  https://github.com/flang-compiler/flang/pull/1057: Fix debuginfo in derived type having scalar pointer member. [No reviews]
  https://github.com/flang-compiler/flang/pull/950: Do not emit function return type as pointer always. [No reviews]
  Any other patches?

  Arm:
  https://github.com/flang-compiler/flang/pull/146 : Machreg Refactor. [No reviews. Paul provided info that AMD requested].
  https://github.com/flang-compiler/flang/pull/976 : Add loop start, end to llvm.loop metadata. [Only Bryan accepted. Hits an assertion in llvm-11+. Might have to wait till AMD's patch which moves to metadata.] This is on Michal's plate. Looking into it. Ramping up.
  https://github.com/flang-compiler/flang/pull/655 : Add missing conversion in Data statement. [No reviews]
  https://github.com/flang-compiler/flang/pull/659 : Changes to support transpose intrinsic in initialization. [No reviews]
  https://github.com/flang-compiler/flang/pull/1090 : LLVM 12 porting issue: Make use of

unwind, readnone, willreturn LLVM-IR attributes.

Huawei:
https://github.com/flang-compiler/flang/issues/1065 : LLVM 12 : All PRs mentioned.
https://github.com/flang-compiler/flang/pull/1039. Fix power operation optimization when the left operand is -1 and the right is a negative integer. [Arm, Huawei accepted] Waiting for AMD's acceptance.
https://github.com/flang-compiler/flang/pull/1036. Fix error of processing entry statement. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1037. Fix issue of processing character variables via the DATA statement. [Arm, Huawei accepted] Waiting for AMD's acceptance.
https://github.com/flang-compiler/flang/pull/1040. Fix constant propagation optimize of common block variable. [No reviews]
https://github.com/flang-compiler/flang/pull/1055 : Fix bugs in pack intrinsic. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1053 : Fix bugs in getarg. [Huawei started review, and requested inputs from Mark.] Should we be consistent with gfortran, ifort?
Also : 1091, 1089, 1088, 1052,1051,1050,1049
Many from the gfortran-testsuite. Good to be accepted and merged.

Windows:
https://github.com/flang-compiler/flang/pull/1045 : Fix build on Windows. Pthread not available. [Arm accepted. Reviewers have suggestions and alternative patch?]
Cmake update PR. is experimental. With this PR we can build flang on windows. Anyone can build Flang on windows. Existing PR but needs updating. #1003. Need 1045.

## Jun 2, 2021

Attendees
AMD: Sourav, Kiran Kumar, Shivaram
Arm: Pawel, Kiran
Huawei: Bryan
Nvidia: Mark
Others: Adam (Szeged)

flang-compiler/flang

- LLVM 12 work for flang, classic-flang/classic-flang-llvm-project
  Bryan mentioned some bugs, particularly the hash-table.
  Bryan: Pull Request coming this afternoon. Will open issues. The PR might be big.
  AMD considering upstreaming string debug.
  Bhuvan AMD: This week will push the changes. Mostly flang changes. LLVM has

support. Sourav: Will remove old string debug in a subsequent patch.
CI issues for debug. Some new changes will require release_11 and more.
Bryan: Can we port to 10? Jini: Not sure whether the effort is worth it.
Bryan: have some downstream changes based on 10 and was planning to upstream it.
Non-trivial amount of code.
<Sourav, please insert the links here> String, Modules. Array support already backported by Alok.
Sourav and Bhuvan will have a look and then comment.
Bryan: Latest for release_10 is end of this year.

- CI Updates?
  Michal: Regarding the CI updates - the ARM64 CI is running, on Friday I will prepare the PRs for the scripts to be used within CI.
  CI is running AArch64 now. Only building 11, 12 due to availability of machines. Will submit a PR to simplify the scripts so that builds and test failures can be reproduced.
- Moving to new LLVM loop access metadata. Michal is working on it in a branch. Options going forward are,
  1. Merge AMD's 1018 vector always patch after addressing all comments.
  2. Rebase Arm's 976 patch. Check issue with vectorisation before this. Add Michal's changes to this. Review and merge.
  Alternatively Is it OK to take AMD's 1018 patch, Arm's 976 patch and add changes to replace the metadata for all directives (vector always, omp simd etc)?
  https://github.com/michalpasztamobica/flang/commits/loop-line-fix-parallel_loop_access
  Chirag: Will try to work on the pending changes before the next meeting.
- Contractors : Have provided triage access to Yao Liu and Qiao Zhang. Previously provided to Michal. Can they also join reviewing?
  Bryan will talk to yao and qiao for joining the review process. Will not be gate keepers.
- Sourav: Can people provide a description? What is the problem being solved and how is it solved?
- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/1018 : Support for vector always directive with new llvm metadata. [No acceptance, comments from Bryan unaddressed.] Chirag will have a look as soon as he gets time.
  https://github.com/flang-compiler/flang/pull/1033 : Support for -gpubnames in flang. (uses a reserved bit, important for AMD). [Arm accepted, is there a test failure issue?] Some issue with upstream driver. Some downstream changes are missing. On hold from AMD. Update: Revised working fine, CI passed in https://github.com/michalpasztamobica/classic-flang-llvm-project/pull/4 tested and ready for review.
  https://github.com/flang-compiler/classic-flang-llvm-project/pull/32 : Driver patch for above PR. [No acceptance]
  Unblocked. Need reviews to progress.

https://github.com/flang-compiler/flang/pull/898: [DebugInfo] Corrected debuginfo for modules. Dependency on LLVM11. LLVM10 bots failing, need to reach consensus on CI builds for PR having version specific dependencies. [Might need back porting]
Any other patches?

Arm:
https://github.com/flang-compiler/flang/pull/146 : Machreg Refactor. [No reviews. Paul provided info that AMD requested].
https://github.com/flang-compiler/flang/pull/976 : Add loop start, end to llvm.loop metadata. [Only Bryan accepted. Hits an assertion in llvm-11+. Might have to wait till AMD's patch which moves to metadata.] This is on Michal's plate. Looking into it. Ramping up.
https://github.com/flang-compiler/flang/pull/655 : Add missing conversion in Data statement. [No reviews]
https://github.com/flang-compiler/flang/pull/659 : Changes to support transpose intrinsic in initialization. [No reviews]
cp2k issues on a previous merge. If that is OK can make a PR with the suggested fix.

Huawei:
https://github.com/flang-compiler/flang/pull/1039. Fix power operation optimization when the left operand is -1 and the right is a negative integer. [Arm, Huawei accepted] Waiting for AMD's acceptance.
https://github.com/flang-compiler/flang/pull/1036. Fix error of processing entry statement. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1037. Fix issue of processing character variables via the DATA statement. [Arm, Huawei accepted] Waiting for AMD's acceptance.
https://github.com/flang-compiler/flang/pull/1040. Fix constant propagation optimize of common block variable. [No reviews]
https://github.com/flang-compiler/flang/pull/1055 : Fix bugs in pack intrinsic. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1053 : Fix bugs in getarg. [Huawei started review, and requested inputs from Mark.] Should we be consistent with gfortran, ifort?
Also : 1052,1051,1050,1049

Windows:
https://github.com/flang-compiler/flang/pull/1045 : Fix build on Windows. Pthread not available. [Arm accepted]
https://github.com/flang-compiler/flang/pull/1031 : machar : fix assertion failure in Windows. [Windows reviewer accepted]
Cmake update PR. is experimental.
With this PR we can build flang on windows. Anyone can build Flang on windows.
Existing PR but needs updating. #1003. Need 1045.

Headsup:
Making changes to build tests on windows.
A release available. https://github.com/kaadam/flang/releases/tag/v0.1

## May 19, 2021

Attendees
AMD: Shivaram, Kiran, Sourabh, Chirag, Alok, Jini
Arm: Kiran
Huawei: Bryan
Nvidia:
Others: Michal Paszta (Mobica/Arm)

flang-compiler/flang

- LLVM 12 work for classic-flang/classic-flang-llvm-project
    - Some bugs. Some issues long-term. Next week will have PRs and issues.
    - One big issue, Implementation of hash table. Hash key and values are constant pointers and type casts which new compiler does not like.
    - AMD: Planning to submit string debug. Bhuvan is planning to. Using upstream debug in AOCC.
- Any CI updates?
    - Good news + Bad news : Setup PRs in classic-flang-llvm-project for llvm-11, llvm-12. One passes but another fails. Have a PR in flang to show. All for AArch64 CI.
      https://github.com/flang-compiler/classic-flang-llvm-project/pull/41
      https://github.com/flang-compiler/flang/pull/1044
    - Suggestion to change script or use a shell script which will be used by CI and can be used by developers to build flang and reproduce failures.
      Bryan agrees. Have another script in Bryan's fork.
      https://github.com/Huawei-PTLab/classic-flang-llvm-project/wiki

- Bryan: Intern working with offload capability. Trying to use paper and old versions. Any assistance from AMD will be great.
  Shivaram has sent a mail.
- Sourav : Can prebuilt binaries be used? Michal: Github token required to access artifacts. If it will help then we can put up build artifact. Can discuss offline if needed.
- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/1017 : inline directive. Was previously 964. Fixed comments in 964. [Arm accepted] Bryan will have a look and then approve. Sourav, please approve.

https://github.com/flang-compiler/flang/pull/1018 : Support for vector always directive with new llvm metadata. [No acceptance, comments from Bryan unaddressed.] Chirag will have a look as soon as he gets time.
https://github.com/flang-compiler/flang/pull/1033 : Support for -gpubnames in flang. (uses a reserved bit, important for AMD). [Arm accepted, is there a test failure issue?] Some issue with upstream driver. Some downstream changes are missing. On hold from AMD.
https://github.com/flang-compiler/classic-flang-llvm-project/pull/32 : Driver patch for above PR. [No acceptance]
https://github.com/flang-compiler/flang/pull/1042. [Debuginfo] Line table entry missing for statements. [Arm accepted] Can Bryan take a look ? Yes, bryan will have a look.

Arm:
https://github.com/flang-compiler/flang/pull/146 : Machreg Refactor. AMD needs some documentation.
https://github.com/flang-compiler/flang/pull/695 : runtime: remove duplicate files. [Huawei accepted, Has issues with X86_64, Can we close?] Kiran will take it up with Pawel.
https://github.com/flang-compiler/flang/pull/976 : Add loop start, end to llvm.loop metadata. [Only Bryan accepted. Hits an assertion in llvm-11+. Might have to wait till AMD's patch which moves to metadata.] This is on Michal's plate. Looking into it. Ramping up.
https://github.com/flang-compiler/flang/pull/655 : Add missing conversion in Data statement.
https://github.com/flang-compiler/flang/pull/659 : Changes to support transpose intrinsic in initialization.

Huawei:
https://github.com/flang-compiler/flang/pull/1039. Fix power operation optimization when the left operand is -1 and the right is a negative integer. [Arm accepted] Shivaram will have a look soon.
https://github.com/flang-compiler/flang/pull/1036. Fix error of processing entry statement.
https://github.com/flang-compiler/flang/pull/1038. Fix error of outputting derived type. [Everyone accepted, can be merged]
https://github.com/flang-compiler/flang/pull/1037. Fix issue of processing character variables via the DATA statement.
https://github.com/flang-compiler/flang/pull/1040. Fix constant propagation optimize of common block variable

Windows:
What next?

## May 5, 2021

Attendees
AMD:
Arm:
Huawei:
Nvidia:
Others:

flang-compiler/flang

- LLVM 12 work for classic-flang-llvm-project
    - Any further commits to remove warnings?
      Proving to be complicated to remove all warnings. Done with flang1. 80% done with flang2. Found bunch of code quality issues. Will open issues ofr them.
    - Any cherry-picks or rebase?
    - Is AMD planning to upstream string debug changes to Flang?
      AMD will get back on this topic.
- Possible improvements to flang generated TBAA. Presentation by Alban (Arm Ltd).
- Backslash behaviour in Flang: https://github.com/flang-compiler/flang/issues/1028
  Shivaram: Good to make it default (same gfortran). Paul: Not good to change default.
- GPU offload
  Shivaram: Basic support upstream. Bryan: Any pointers will be appreciated.
- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/1017 : inline directive. Was previously 964. Fixed comments in 964
  https://github.com/flang-compiler/flang/pull/1018 : Support for vector always directive with new llvm metadata.
  https://github.com/flang-compiler/flang/pull/1033 : Support for -gpubnames in flang. (uses a reserved bit, important for AMD). [Not accepted by anyone]
  Has issues with symbol lookup with this section. Clang also generates on demand only.
  https://github.com/flang-compiler/flang-driver/pull/95: Driver patch for above PR. [Only Arm accepted]
  https://github.com/flang-compiler/flang/pull/1042. [Debuginfo] Line table entry missing for statements. [Arm accepted]

  Arm:
  https://github.com/flang-compiler/flang/pull/146 : Machreg Refactor
  https://github.com/flang-compiler/flang/pull/695 : runtime: remove duplicate files
  https://github.com/flang-compiler/flang/pull/976 : Add loop start, end to llvm.loop metadata. [Only Bryan accepted] Hits an assertion in llvm-11+. Might have to wait till AMD’s patch which moves to metadata.

Huawei:
https://github.com/flang-compiler/flang/pull/1040. Fix constant propagation optimize of common block variable.
https://github.com/flang-compiler/flang/pull/1039. Fix power operation optimization when the left operand is -1 and the right is a negative integer.
https://github.com/flang-compiler/flang/pull/1038. Fix error of outputting derived type.
https://github.com/flang-compiler/flang/pull/1037. Fix issue of processing character variables via the DATA statement.
https://github.com/flang-compiler/flang/pull/1036. Fix error of processing entry statement.

Windows:
https://github.com/flang-compiler/flang/pull/1020 : patch flang2 to work on Windows. [Bryan has commented, Kadaam and Kiran (Arm) accepted]

## April 21, 2021

Attendees
AMD: Kiran Kumar, Sourabh, Shivaram
Arm: Pawel, Kiran
Huawei: Bryan
Nvidia:Mark
Others:

flang-compiler/flang

- CI update and questions. Michal.
    - ARM would like to share their two 32-core servers to build flang on AArch64 architecture:
        - An “organization owner” needs to add the runners to flang-compiler organization in cooperation with the server’s owner. Who can do this?
        - Due to limited resources the builds would primarily run for release_11x and release_12x with gcc-10 with the possibility of further expansion if more runners are added or if the load proves to be low enough. Arm will share the Docker file and set up guidelines to anyone interested.
        - Separate Github Actions yaml script would take care of the builds in a similar fashion to the existing CI.
    - Can we limit the amount of tested configurations for X86 architecture?
        - Does anyone need llvm-9-based builds?
        - Does anyone need gcc-9-based builds?
          AMD is fine with removing. Huawei also.

    - We currently have no mechanism in CI to prevent a change to classic-flang-llvm-project that would break flang. Should we add a job to build and test flang on PRs to classic-flang-llvm-project?
      OK to have this one Huawei, AMD.

Well established github actions CI. Only x86_64. Arm is offering some machines for github actions CI (AArch64). Improvement to current set up.
Require help from Steve (Nvidia).
AArch64 servers have many cores.
Drone CI effort is on hold. Drone might provide Windows AArch64. Require self-hosted runners.
Our (Arm's) own servers are faster.

- Flang documentation
  Created a document to work together on creating some docs for Flang.
  https://docs.google.com/document/d/1aKoT1i_ptfOWHqpdQw0H9QWavhPPIii6Dsk32XHaDAk/edit?usp=sharing
  There is quite a lot of documentation that is generated. Uploaded generated doc to my github page. Given that flang does not change much is this an acceptable solution.
  Alternative is to have a workflow to upload generated doc whenever there is a change.
  http://kiranchandramohan.github.io/web/html/index.html
- Backslash behaviour in Flang: https://github.com/flang-compiler/flang/issues/1028
- classic-flang-llvm-project/release_12x rebased on LLVM 12.0.0
- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/1017 : inline directive. Was previously 964.
  Fixed comments in 964
  https://github.com/flang-compiler/flang/pull/1018 : Support for vector always directive with new llvm metadata.
  https://github.com/flang-compiler/flang/pull/1033 : Support for -gpubnames in flang. (uses a reserved bit, important for AMD). [Not accepted by anyone]
  Has issues with symbol lookup with this section. Clang also generates on demand only.
  https://github.com/flang-compiler/flang-driver/pull/95: Driver patch for above PR. [Only Arm accepted]

  Arm:
  https://github.com/flang-compiler/flang/pull/976 : Add loop start, end to llvm.loop metadata. [Only Bryan accepted] Hits an assertion in llvm-11+. Might have to wait till AMD's patch which moves to metadata.
  https://github.com/flang-compiler/flang/pull/146 : Machreg Refactor

  Huawei:
  https://github.com/flang-compiler/flang/pull/1009: Set type length of string array pointer. [Paul/Arm, AMD accepted. Bryan is it OK to merge?]

Windows:
https://github.com/flang-compiler/flang/pull/1020 : patch flang2 to work on Windows.
[Bryan has commented]

## April 7, 2021

Attendees
AMD: Sourav, Alok, Kiran Kumar, Jini, Shivaram
Arm: Pawel, Kiran
Huawei: Joey, Bryan
Nvidia: Mark
Others: Adam (Szeged)

flang-compiler/flang

- Is the time OK with everyone?
  Works for Mark (8.30am) and AMD (9pm). Works for Bryan.
- Any important point to discuss?
  Sourav : Textual flags (ili, ilm, ast etc) : PaulO finds it useful. Bryan also fine. Will make a PR.
  Kiran: emit-flang-llvm: arm can upstream.
  New starters guide. SC17 presentation and paper: PawelO. Another one for gpu offloading also gives an overview by Guray, Gary etc. Mark can answer specific questions. Mark will check whether there is anything internal at Nvidia. Everyone has a bits and pieces story but no single big picture. Bryan can have a look.
  Bryan: Can use github.io to put up the docs generated by flang. Have a workflow which can build and put the pages there. Bryan can try this.
  Existing documentation
  -> How the Flang Frontend works
  https://dl.acm.org/citation.cfm?id=3148183
  https://llvm-hpc4-workshop.github.io/slides/Osmialowski.pdf
  -> OpenMP offload in Flang and LLVM
  https://sc18.supercomputing.org/proceedings/workshops/workshop_files/ws_llvmf104s2-file1.pdf
  -> https://thinkingeek.com/2017/06/17/walk-through-flang-part-1/
  -> In-tree docs built with -DFLANG_INCLUDE_DOCS=ON available in rst and html.
- Anyone wants to give a talk/presentation or discuss in detail about anything in Flang?
- Adam trying to cross-compile for windows AArch64. Anyone has experience with this? How can we discuss this?
  Adam : Raise a github issue/community channel.
  Kiran: There is flang-compiler.slack.com #classic-flang
  Joey has experience. Mingw.

Adam: My use case is Host: x86_64-windows, Target: aarch64-windows. Using clang. Will create a github issue.

- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/1017 : inline directive. Was previously 964. Fixed comments in 964
  https://github.com/flang-compiler/flang/pull/1018 : Support for vector always directive with new llvm metadata.
  https://github.com/flang-compiler/flang/pull/934: Fix default emission of debug_names section. (uses a reserved bit, important for AMD). [Not accepted by anyone]
  Has issues with symbol lookup with this section. Clang also generates on demand only.
  https://github.com/flang-compiler/flang-driver/pull/95: Driver patch for above PR. [Only Arm accepted]

  Arm:
  https://github.com/flang-compiler/flang/pull/976 : Add loop start, end to llvm.loop metadata. [Only Bryan accepted] Hits an assertion in llvm-11+. Might have to wait till AMD's patch which moves to metadata.
  https://github.com/flang-compiler/flang/pull/1011 : Do not call streamlined/dgemm-like matmul routines ….
  https://github.com/flang-compiler/flang/pull/146 : Machreg Refactor

  Huawei:
  https://github.com/flang-compiler/flang/pull/1009: Set type length of string array pointer. [Paul/Arm accepted]

  Windows:
  https://github.com/flang-compiler/flang/pull/981 : patch scutil [Arm, Windows team accepted]
  https://github.com/flang-compiler/flang/pull/1020 : patch flang2 to work on Windows. [Bryan has commented]

## March 24, 2021

Attendees
AMD: Shivaram, Kiran Kumar, Chirag, Sourabh
Arm: Kiran Chandramohan, Pawel Osmialowski
Huawei: Joey Ye, Bryan Chan
Nvidia: Mark Leir

Others:

flang-compiler/flang

- [Tracking] Phasing out support for llvm-9, Adding support for llvm-12. Huawei is currently merging llvm-12 to their tool and has volunteered to make the same changes for classic-flang-llvm-project.
  Based on release_12x and in classic-flang-llvm-project. Also merged flang changes required for llvm-12. Require permission to tag. Fine with AMD and Arm. Previous end of October and in March the year before. flang-<date>. Maybe flang-llvm-12-date. Can confuse so better to stick with flang-<date>.
- [Tracking] Currently !llvm.mem.parallel_loop_access !<n> is generated for load and store instructions for ivdep. Here <n> is the loop id. This is deprecated. AMD has implemented the new style directives.
  Equivalent for assume safety in clang. Chirag has a PR with the new loop access metadata. Only for vector always now. In future it can be updated for others. Chirag will re-update by tomorrow evening.
- Timing of call. Have sent out a new invite. Do not have email ids of Joey (Huawei), Jini (AMD). If you did not get an invite then send me an email (kiranDOTchandramohanATarmDOTcom)
- Presentation about Arm Pull Requests. PRs ordered by priority. Please help review.
  https://drive.google.com/file/d/1TqG0dGFPCDC564xnlhfCXetFChHA0Zgb/view?usp=sharing
- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/964 : inline directive. [Bryan has comments]
  https://github.com/flang-compiler/flang/pull/934: Fix default emission of debug_names section. (uses a reserved bit, important for AMD). [Not accepted by anyone]
  Has issues with symbol lookup with this section. Clang also generates on demand only.
  https://github.com/flang-compiler/flang-driver/pull/95: Driver patch for above PR. [Only Arm accepted]

  Arm:
  https://github.com/flang-compiler/flang/pull/976 : Add loop start, end to llvm.loop metadata. [Only Bryan accepted]

  Huawei:
  https://github.com/flang-compiler/flang/pull/962 : Don't reorder non-constant operands. Have questions about short-cut semantics in Fortran.
  https://github.com/flang-compiler/flang/pull/1009: Set type length of string array pointer. [Paul accepted]

## March 10, 2021

Attendees
AMD: Shivaram, Kiran, Alok, Sourav, Bhuvanakumar, Jini
Arm: Pawel, Ashok, Kiran
Huawei: Joey, Bryan
Nvidia: Mark
Others: Adam

flang-compiler/flang

- [Tracking] Phasing out support for llvm-9, Adding support for llvm-12. Huawei is currently merging llvm-12 to their tool and has volunteered to make the same changes for classic-flang-llvm-project.
  Joey to follow up. Bryan says, Completely rebased to 12. Upstream cmake files have changed. Werror. Bunch of pedantic errors. All are fairly simple changes. Timeline is a week.
- [Tracking] Currently !llvm.mem.parallel_loop_access !<n> is generated for load and store instructions for ivdep. Here <n> is the loop id. This is deprecated. AMD has implemented the new style directives.
  Have plans. Might need some time. Can sync next week
- Matrix multiplication question (https://github.com/flang-compiler/flang/issues/1005)
  Paul making progress.
- Temporary array removal question (https://github.com/flang-compiler/flang/issues/924)
  Mark also feels that llvm might be able to remove. Might need alias analysis and IPA.
  Paul says there is similar mechanism in Polly.
- Release_11x. Shivaram has faced some issues in building 11.
- We have to refresh the call soon. Is there any change in time required? 4.30 UK time.
- Spec issue on llvm-11. Arm emits separate calls on x86 it is sincos that is generated and llvm faces this issue on x86_64. Spec2017 benchmarks.
- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/964 : inline directive. [Bryan has comments]
  https://github.com/flang-compiler/flang/pull/934: Fix default emission of debug_names section. (uses a reserved bit, important for AMD). [Not accepted by anyone]
  Has issues with symbol lookup with this section. Clang also generates on demand only.
  https://github.com/flang-compiler/flang-driver/pull/95: Driver patch for above PR. [Only Arm accepted]

  Arm:
  https://github.com/flang-compiler/flang/pull/976 : Add loop start, end to llvm.loop metadata. [Only Bryan accepted]

Huawei:
https://github.com/flang-compiler/flang/pull/962 : Don't reorder non-constant operands. Have questions about short-cut semantics in Fortran.

- New PRs to consider

## Feb 24, 2021

Attendees
AMD:
Arm:
Huawei:
Nvidia:
Others:

flang-compiler/flang

- Phasing out support for llvm-9, Adding support for llvm-12
- Currently !llvm.mem.parallel_loop_access !<n> is generated for load and store instructions for ivdep. Here <n> is the loop id. This is deprecated. Also has issues with llvm IR verification now.
  " llvm.mem.parallel_loop_access metadata on instruction is replaced by llvm.access.group metadata. llvm.access.group points to a distinct MDNode with no operands (avoiding the problem to ever need to add/remove operands), called "access group". Alternatively, it can point to a list of access groups. The LoopID then has an attribute llvm.loop.parallel_accesses with all the access groups that are parallel (no dependencies carries by this loop)."

  AMD has support for access group and can consider upstreaming.
  AMD has changes for directives in llvm. Can upstream to classic-flang-llvm-project.
  Clauses to simd. Will use projects page.
- Couple of llvm patches which does function specialisation and is beneficial for a spec fortran benchmark.
  https://reviews.llvm.org/D93762
  https://reviews.llvm.org/D93838
- Sourabh can be a reviewer from AMD.
- Upstreaming fortran debug changes to llvm. Have added reviewers from Nvidia, Arm. DIConstant and previously DIModule.
- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/964 : inline directive.
  https://github.com/flang-compiler/flang/pull/934: Fix default emission of debug_names section. (uses a reserved bit, important for AMD).
  https://github.com/flang-compiler/flang-driver/pull/95: Driver patch for above PR.

  Arm:

https://github.com/flang-compiler/flang/pull/963 : Errors with implied do array init. Paul-arm and Mark approved. Shivaram/Bryan?
https://github.com/flang-compiler/flang/pull/976 : Add loop start, end to llvm.loop metadata.

Huawei:
https://github.com/flang-compiler/flang/pull/962 : Don't reorder non-constant operands. Have questions about short-cut semantics in Fortran.

- New PRs to consider
  Windows:

## Feb 10, 2021

Attendees
AMD: Shivaram, Kiran Kumar, Jini, Alok, B Kumar, Raghu
Arm: Kiran, Pawel
Huawei: Bryan
Nvidia: Mark
Others: Adam (Szeged), Xoviat

flang-compiler/flang

- Windows Pull requests
  Enable windows cross-compilation with Flang.
  Reviewing xoviat's PRs. Also uploading the PRs where functionality is missing.
  Uploaded draft version of our fork. Making changes for runtime to be available for windows on AArch64.
  WIN32 preprocessor name. Using this for 64 bit architectures is strange.
  Gcc,clang, and ms compilers _WIN32 to determine old/all(?) window platform on various hardware (including modern).
  Xoviat can change to 64. Uses windows. Gfortran uses mingw. Flang is free and can be compiled with visual studio. No 32 bit support but fine world is 64.
  Go to xoviat's repository and there is a working branch.
  Dont use the flang-driver, use a custom driver. Will miss debug support.
- Review of pull request processing
  - Simplifying process for CI changes : Shivaram: CI changes fine
  - Comments/Docs/Readme changes : Bryan: NFC changes fine
  - Simplifying process for cherry-picking from llvm-project : cherry-picks can be botched up. Provide further evidence (like showing that it is the same hash). Otherwise review.
  - Testcases only : CI and one other person.

    - Windows PRs. Bryan started reviewing some patches. Rate of incoming patches high. Will need CI. Shivaram: If linux build on x86 CI passes it should be fine.
    - Debug PRs : Shivaram: Better to have reviews.
    - Having a timeout for PR processing after which PRs can be committed : Bryan motivating others to review. Also trying to increase his availability. Shivaram will try to get one more person. Not in favour of timeout.
- mem.parallel.access
- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/952 : Array debug. Gary and Kiran have approved. What to do about release_11x issue?
  https://github.com/flang-compiler/flang/pull/964 : inline directive. Arm has questions.
  https://github.com/flang-compiler/flang/pull/934: Fix default emission of debug_names section. Kiran-arm has questions.
  https://github.com/flang-compiler/flang-driver/pull/95: Driver patch for above PR. Kiran-arm approved.

  Arm:
  https://github.com/flang-compiler/flang/pull/963 : Errors with implied do array init. Paul-arm and Mark approved. Shivaram/Bryan?
  https://github.com/flang-compiler/flang/pull/976 : Add loop start, end to llvm.loop metadata.

  Huawei:
  https://github.com/flang-compiler/flang/pull/962 : Don't reorder non-constant operands. Have questions about short-cut semantics in Fortran.

- New PRs to consider
  Windows:

## Jan 27, 2021

Attendees
AMD: Shivaram
Arm: Kiran, Ashok
Huawei: Bryan
Nvidia: Mark

flang-compiler/flang

- Review of pull request processing
  https://github.com/flang-compiler/flang/wiki/Community#the-pull-request-process
  Bryan and Paul are not in the gate-keeper list. Should we add more people?
  Bryan and shivaram to continue for now. Bryan's name to be added.
  Have not removed PowerPC.
  Solicits for testing on PowerPC. Leave it for 6 months before deprecation.
  Given that we have not made much headway, should we try a more relaxed processing?
  AMD and Huawei do not merge that often from upstream flang. Huawei intends to set up a CI.
  Both sides have to be active to merge patch.
  Should try to increase reviewers.
  Review and check whhter old patches are stale.
- Short-circuit support in Flang. Bryan says,
  "The `IF (PRESENT(x) .AND. some_expr(x))` pattern works fine when compiled with gfortran, and it is so common that I think Classic Flang ought to recognize the pattern and generate code that doesn't crash. If you agree, I'll work on that and submit a different patch."
  https://github.com/flang-compiler/flang/pull/962
  Bryan: The problematic code came from a closed-source application written by a customer.
- Spec/sincos issue llvm-11. Can we reproduce this issue?
  Bryan: Reproduced on x86. IR is the same. Tuple of doubles is not supported. Bryan is investigating.
- New PR to print llvm.loop metadata with loop start and end location
- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/952 : Array debug. Only Gary has approved. Arm facing a couple of issues in the internal testsuite.
  https://github.com/flang-compiler/flang/pull/966 : Fixes for CP2K. Arm has requested minor changes. Shivaram will update.
  https://github.com/flang-compiler/flang/pull/964 : inline directive. Arm has questions.

  Arm:
  https://github.com/flang-compiler/flang/pull/955 : Issue error when character kind 2 is used. Gary and Shivaram have approved. Aarch64 approval remaining.
  https://github.com/flang-compiler/flang/pull/963 : Errors with implied do array init.
  https://github.com/flang-compiler/flang/pull/968 : Clash between contiguous and save attributes.
  https://github.com/flang-compiler/flang/pull/967 : Testcases for #966 cp2k. Waiting for cp2k patch to be merged.

  Huawei:
  https://github.com/flang-compiler/flang/pull/956 : FileCheck prefixes fix. Gary and Michal approved. Shivaram has to approve.

https://github.com/flang-compiler/flang/pull/962 : Don't reorder non-constant operands. Have questions about short-cut semantics in Fortran.

- New PRs to consider
  AMD:
  https://github.com/flang-compiler/flang/pull/934: Fix default emission of debug_names section.
  https://github.com/flang-compiler/flang-driver/pull/95

  Arm:
  https://github.com/flang-compiler/classic-flang-llvm-project/pull/12 : Github Actions to build release_11x
  https://github.com/flang-compiler/classic-flang-llvm-project/pull/13: Github Actions to build release_11x
  https://github.com/flang-compiler/flang/pull/975: Build release_11x branch of llvm (depends on the two PRs above)
  https://github.com/flang-compiler/flang/pull/974 : Fix hash collision
  https://github.com/flang-compiler/flang/pull/976 : Add loop start, end to llvm.loop metadata
  https://github.com/flang-compiler/flang/pull/978 : Fix readme

  Huawei:

  Nvidia:

## Jan 13, 2021

Attendees
AMD: Shivaram, Jini, Alok, Sourabh
Arm: Kiran, Paul, Rich, Ashok, Michal (Mobica)
Huawei: Bryan, Joey
Nvidia: Mark

flang-compiler/flang
- Happy New Year!
- Branch release_110 ready to be added from Michał's fork repo to flang-compiler/classic-flang-llvm-project:
  - No issues or remarks raised after last week's RFC sent out to flang-dev mailing list - is this sufficient time for review? Would anyone else like to review the branch before pulling?
  - we need a volunteer with access rights in flang-compiler/classic-flang-llvm-project to pull the branch from the fork

    - Bryan mentioned we could remove the DIFortranSubrange code from the branch before pulling, if flang#952 gets merged. What is the plan of merging it? Should we wait for the merge or can we just cherry-pick the relevant changes from release_100?

Michal: Branch for release_110 ready. Have sent an RFC. Please let know of any issues.

Bryan would like to have a look. Wanted to have a patch to remove old functionality. We should try to merge this into 11.

Alok has created a PR.

Michal might not be worth the effort to squash and completely remove old subrange.

- Error when character(kind=2) is used. https://github.com/flang-compiler/flang/pull/955
  Bryan: Support for UTF16 better.
  Mark: Legacy code. Not aware of people using it. OK with error.
  Paul: What does fujitsu compiler do?
  Shivaram: Gfortran gives error.
- Paul would like to discuss Infinite loop in find_funcptr_name issue #960. Hash function, usage etc.
  Paul: would like Mark to have a look at the latest comments posted by Paul. Really hard to create testcase.
  Mark: will have a look.
  Paul: posted today.
  Shivaram: there is an issue with goto continue.
  Mark.Paul; the goto is a bandaid.
- Cp2k issue #966. Discussion on flag value (7 vs 47). Also should these flags have a name?
  Paul: clang and flang patches differ in the value of the flag.
  Shivaram: It should be 47. Will check.
  Only customer is cp2k now. Profile for flang.
  Switches off pointer optimization.
  Shivaram: Above some optimization level and some MPI ranks for high size. Fast matmul is doing something wrong. WIll add this comment to PR.
- Question about new directives to consider.
    - More options for the vector directive (like length)?
    - Clang has “optnone” for disabling optimizations (applied to functions as attribute). Useful for debugging. Intel has !dir$ nooptimize

Kiran Kumar: Change optimization level per function. Might help with issues of optimization.

Bryan: Clang has a similar pragma. Actually only optimization on/off.

- Windows on Arm. Another team in Arm is making good progress.
  Joey: What features are not independent OS? ABI? Runtime, fortran compiler, cmake.
- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/940 : debug location for instruction in prologue. Has approval to merge..

https://github.com/flang-compiler/flang/pull/952 : Array debug. Only Gary has approved. Arm has one test to fix and then codereview.

Arm:
https://github.com/flang-compiler/flang/pull/955 : Issue error when character kind 2 is used. Gary and Shivaram have approved. Aarch64 approval remaining.
https://github.com/flang-compiler/flang/pull/751 : Fix for len intrinsic returning int*8. Gary approved. Arm has to make changes addressing Huawei feedback.

Huawei:
https://github.com/flang-compiler/flang/pull/956 : FileCheck prefixes fix. Gary and Michal approved.

Nvidia:

- New PRs to consider
  AMD:
  https://github.com/flang-compiler/flang/pull/966 : Fixes for CP2K
  https://github.com/flang-compiler/flang/pull/964 : inline directive

  Arm:
  https://github.com/flang-compiler/flang/pull/963 : Errors with implied do array init.
  https://github.com/flang-compiler/flang/pull/968 : Clash between contiguous and save attributes.
  https://github.com/flang-compiler/flang/pull/967 : Testcases for #966 cp2k.

  Huawei:
  https://github.com/flang-compiler/flang/pull/962 : Do not reorder non-const operands.

  Nvidia:

## Dec 16, 2020

Attendees
AMD:
Arm:
Huawei: Bryan Chan
Nvidia: Gary Klimowicz, Mark LeAir

flang-compiler/flang

- Welcome back Pawel Osmialowski (Paul)
- Farewell to Gary who will be retiring

Next week's regular biweekly call (December 23): Alexis Perry-Holby will run the call; she'll probably send out a new calendar invitation after that call.
Mark is probably attending next time onwards. Might need a time change.
For power no decision has been taken. Someone will replace Gary for this or remove the power build/test requirement for PR approvals. Mark is likely to focus on PRs.
Set up OpenPower cloud

- Cancel next meeting (December 30)
  No objection.
- CP2K
  Long time to build. Due to heavy use of oop and fortran 2008, Flang struggles. Either cp2k or dbcsr. Some old fixes at PGI. A few had been fixed by PGI but had to be reverted since it made tealeaf worse. Some CP2k code changes are proposed. Patches from AMD are beneficial. Clang patch was not required if command line options were used. Mar 2020 cp2k. Can AMD go upstream with these patches?
  Shivaram: Next couple of weeks.
  Do we want to aggressively change classic Flang for CP2K?
  What are the intrusive changes downstream and the advanced features used by CP2k?
  Bugs are already reported upstream.
  Can look at patch and then make a decision.
  Bryan: Which are the bugs that are fixed?
- Request for committing NFCs without review:
  1) Some Arm commits went in with the old license header.
  Eg:
  https://github.com/flang-compiler/flang/blob/997feeef4b15e3dbea4be8b6309869ba64991548/runtime/flang/itrailz.c#L2
  Always review. Bryan agrees.
  2) classic-flang-llvm-project: LLVM :: Bindings/Go/go.test has been failing. Since this is not related to our changes, is it OK to blacklist without review?
  Always review. Bryan agrees. A bit strange that llvm-project passes.
- LLVM 11
  When should we move to llvm-11.
  It is on Michał's list for January.
- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/940 : debug location for instruction in prologue. Only Gary has approved.
  https://github.com/flang-compiler/flang/pull/952 : Array debug. Only Gary has approved.

  Arm:
  https://github.com/flang-compiler/flang/pull/947: Fix gcc-10 compilation issues. Only Gary has approved.
  https://github.com/flang-compiler/flang/pull/948 : CI to run with more versions of gcc and llvm. Approval from Shivaram required.

https://github.com/flang-compiler/flang/pull/949 : Fix for imported module variable not available after renaming. Only Gary has approved.
https://github.com/flang-compiler/flang/pull/955 : Issue error when character kind 2 is used. Gary and Shivaram have approved. Aarch64 approval remaining.
https://github.com/flang-compiler/flang/pull/751 : Fix for len intrinsic returning int*8. No approval yet.

Huawei:
https://github.com/flang-compiler/flang/pull/956 : FileCheck prefixes fix. Only Gary has approved.
https://github.com/flang-compiler/classic-flang-llvm-project/pull/9 : DIFortranSubrange getLowerbound assertion failure. Gary has approved. What is the plan for this patch in light of new debug changes?
https://github.com/flang-compiler/flang/pull/957 : Prefetch support. Only Gary has approved.

Nvidia:

- New PRs to consider
  AMD:

  Arm:

  Huawei:

  Nvidia:

## Dec 2, 2020

Attendees
AMD: Shivaram, Kiran
Arm: Kiran
Huawei: Bryan
Nvidia: Gary Klimowicz

flang-compiler/flang
- Windows support
  1) Isuruf is ready to upstream his patches. (Ready for review https://github.com/flang-compiler/flang/pull/725)
  Isuru is using Clang. ABI compatible with Visual Studio. Can also be made compatible with mingw with an option. Lot more patches are needed. Initial patch was large. Complex support has to be through macros. Some changes are needed in flang-driver to make it work for windows, currently only for unix. Link with different runtimes for release

and debug. First Flang has to be fixed, driver will pass option to flang1 and then to flang2 and finally put in LLVM IR.
Updates needed in wiki for windows so that people can get started. Gary might not have time, but there are windows machines.
First cmake instruction changes and then compilation errors.
Windows need different module files for static and shared. Right now an assumption.
AMD,Huawei,Nvidia,Arm OK with accepting windows patches.
People using scipy,numpy.
Gfortran works with mingw and not with others.
2) PGI compiler has support for !DEC$ ATTRIBUTES DLLEXPORT, -Mmakedll. There seems to be some code support for the attribute in flang.
Windows ifdefs were removed. Can check how many places are there for dllexport.
There could be a lot of dependencies missing.

- Minfo support
  PGI compiler has support for Minfo to report various optimisations. In flang this is redirected to report on LLVM optimisations via Rpass. This will miss any optimisation performed by flang like loop fusion. There seems to be support for ccinfo (the infrastructure for Minfo) in the code in Flang but possibly not switched ON.
  Similar to Windows story.There might be vestiges of the code.
- AMD had some questions about directives. What should be syntax and naming?
  !dir$ vector, !dir$ novector vs
  !dir$ vector always, !dir$ vector never
  Is there a standardised way that we can proceed.

  Matching semantics with intel are difficult.
  Many are parsed but ignored. We can make a list and then decide which ones to support and which ones to throw away.
  Nvfortran guide on the web.
  Is there more work for pragmas? Huawei working on inline. AMD worked on unroll but that is already upstream. simd and its clauses.
  Gary: Projects page, github cards available.
- Providing non-merge access for Michal and other engineers.
  OK with everyone.
- Gary has issues with building cmake new version (3.18.*) classic-flang-llvm-project.
  Bryan has updated the build instructions. Building flang wiki page.
- Request for committing NFCs without review:
  1) Some Arm commits went in with the old license header.
  Eg:
  https://github.com/flang-compiler/flang/blob/997feeef4b15e3dbea4be8b6309869ba64991548/runtime/flang/itrailz.c#L2
  2) classic-flang-project: LLVM :: Bindings/Go/go.test has been failing. Since this is not related to our changes, is it OK to blacklist without review?

- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/940: debug location for instruction in prologue. Only Gary has approved.

  Arm:
  https://github.com/flang-compiler/flang/pull/947: Fix gcc-10 compilation issues. Require consensus from everyone.
  https://github.com/flang-compiler/flang/pull/948 : CI to run with more versions of gcc and llvm. Approval from Shivaram required.
  Huawei:

  Nvidia:

- New PRs to consider
  AMD:
  952 patch has all the others.
  Shivaram will get back whether it is OK.
  Is the intention to replace DI_Fortran_subrange with DI_subrange? DI_subrange is available in upstream llvm.

  Arm:
  https://github.com/flang-compiler/flang/pull/949 : Fix for imported module variable not available after renaming.
  https://github.com/flang-compiler/flang/pull/955 : Issue error when character kind 2 is used.
  https://github.com/flang-compiler/flang/pull/751 : Fix for len intrinsic returning int*8.

  Huawei:
  https://github.com/flang-compiler/flang/pull/956 : FileCheck prefixes fix.
  https://github.com/flang-compiler/classic-flang-llvm-project/pull/9 : DIFortranSubrange getLowerbound assertion failure.
  https://github.com/flang-compiler/flang/pull/957 : Prefetch support

  Nvidia:

## Nov 18, 2020

Attendees
AMD: Shivaram
Arm: Kiran, Andrzej, Irina

Huawei: Bryan
Nvidia: Gary Klimowicz

flang-compiler/flang

- Correcting merge to preserve history. Is everyone OK with it?
  https://github.com/flang-compiler/flang/pull/939
  Keep linear history like LLVM. Everyone is OK and Bryan will change after the meeting in the main branch. Done.
- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/883: Fix for showing return variable of functions. OK to submit unless Huawei has concerns.
  https://github.com/flang-compiler/flang/pull/941: Fix for making variable visible in Internal subprogram. OK to submit unless Huawei has concerns.
  Bryan will have a look today and approve.

  ARM:
  https://github.com/flang-compiler/flang/pull/946 : Fix for lowering error with is_contiguous intrinsic. OK to submit unless Huawei has comments.
  https://github.com/flang-compiler/classic-flang-llvm-project/pull/6 : Switching OFF some un-necessary pipelines. Gary has approved. Requires review from AMD and Huawei.
  Bryan will have a look.

  Huawei:
  https://github.com/flang-compiler/flang/pull/942 : Implement unroll(n) directive. Requires review from AMD.
  Shivaram will have a look in the next two days.
  https://github.com/flang-compiler/flang/pull/945 : Stop using pgstdinit.h. Only used in one file in pgmath. Have requested a re-review from PGI. Merged now.

  Nvidia:

- New PRs to consider
  AMD:
  https://github.com/flang-compiler/flang/pull/940: debug location for instruction in prologue.
  https://github.com/flang-compiler/flang/pull/933: xflags which were missing. Gary has provided a list. Flags of a version of a version of the compiler that is not supported now.

  ARM:
  https://github.com/flang-compiler/flang/pull/951: Fix ICE with atomic instruction generation
  https://github.com/flang-compiler/flang/pull/947: Fix gcc-10 compilation issues

Gcc-10 defaults to fno-common, cannot have same global symbols multiple time.
Missing array inits in pgmath. Dave Parks (of Nvidia) is interested in pgmath fix. Will finally push pgmath to llvm-project.
Bryan: Nice to have some documentation.
Arm has shared some documentation.
https://github.com/flang-compiler/classic-flang-llvm-project/pull/7
https://github.com/flang-compiler/flang/pull/948 : CI to run with more versions of gcc and llvm.
Mizchal has sent an RFC. gcc 9/10, llvm 9/10/11
First 947 then classic-flang-project/pull/7, pull/948
release_90 in classic-flang-llvm-project does not have the changes in flang-driver, llvm.
release_100 in classic-flang-llvm-project and flang master.
Arm and AMd will move to new versions of llvm so no need to move to release_90 for classic-flang-project.

Huawei:
Fortran 2008: norm2 listed as not supported. But it is supported. Wiki page needs an update. Gary will have a look. Arm has a page will add here.
https://developer.arm.com/documentation/101380/2010/Standards-support/Fortran-2008?lang=en

Nvidia:

Mobica:
Michał pronounces his name like "Mee-how Pah-shta"

## Nov 4, 2020

Attendees
AMD: Shivaram, Kiran, Alok, Jini
Arm: Kiran, Andrzej
Huawei: Bryan
Nvidia: Gary Klimowicz

flang-compiler/flang

- CI
  -> Github Actions CI PR merged. Now should see a green/red tick on PRs.
  https://github.com/flang-compiler/flang/pulls
  Runs on X86. Run whenever there is a push to master (ie. merge), also when a new PR is created or updated.
  -> Currently llvm-9. llvm-10 not in immediate plan.
  Building llvm and flang-driver with gcc-9.
  Check with gcc-10.

AArch64 and powerpc backends are built separately but not tested or used.
We can run IR tests for different backends.
We can run make check-all for flang-driver, llvm later. Current focus was to enable the flang repo CI. The flang CI runs in 10 minutes.

- Build documentation for llvm-10 not in https://github.com/flang-compiler/flang/wiki/Building-Flang
- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/908 : debug fix for byval. Kiran/Arm has requested changes.
  https://github.com/flang-compiler/flang/pull/907 : dbg.declare, dbg.value for same variable. Gary/Nvidia has requested changes.
  ^ AMD will not be working on the above for the next 2 weeks.
  ARM:
  https://github.com/flang-compiler/flang/pull/939 : Publish documentation. Require review from everyone. Michal will reach out to Gary in slack.
  Huawei:
  https://github.com/flang-compiler/flang/pull/942 : Implement unroll(n) directive. Require review from everyone. Tests fine at Arm.
  Nvidia:

- New PRs to consider
  AMD:
  883 : not printing function result
  941 : not showing a subprogram variable
  ARM:
  https://github.com/flang-compiler/flang/pull/946 : Fix for lowering error with is_contiguous intrinsic. Mark from Nvida provided a suggestion. Works fine. Require review from everyone.
  https://github.com/flang-compiler/classic-flang-llvm-project/pull/6 : Switching OFF some un-necessary pipelines.
  Huawei:
  https://github.com/flang-compiler/flang/pull/945 : Stop using pgstdinit.h. Only used in one file in pgmath. Overwrites some declarations from glibc. Causes some build failures. FORTIFY_SOURCE is a macro supported by glibc, allows compilations with safety checks, checks for buffer overflows. No other issues seen so far.
  Nvidia:

## Oct 21, 2020

Attendees
AMD: Shivaram
Arm: Kiran, Rich, Andrzej

Huawei: Bryan
Nvidia: Gary Klimowicz

flang-compiler/flang

- CI for flang
    - LLVM-9 based flang post-commit CI is working. You might have received mail. LLVM-10 based CI still requires some work. Shivaram to be added.
      http://80735715.packethost.net:8080/
    - Michal's CI work based on github action.
      Got it working with github actions. Changes to make llvm and flang-driver also build so that we get the latest copy. Pre-commit CI. Only the repo for which the PR is made will be built. LLVM-9 based. Default machine with github x86 on ubuntu. 8 minutes. We can discuss whether this can be a substitute for AMD's vote later ON.
- Meeting invites expire with today's meeting. Is this the best time and format going forward? Advance meeting by 30 minutes.
- Anything else other than PRs to discuss?
    - Tagging without testing?
      Requirement internally for Bryan to have a tag. Gary wouldn't say No. Tagged at some date for release. No issues for Arm or AMD. Bryan will tag.
- Status of PRs being reviewed (flang, flang-driver, llvm)
  AMD:
  https://github.com/flang-compiler/flang/pull/908 : debug fix for byval. Kiran/Arm has requested changes.
  https://github.com/flang-compiler/flang/pull/907 : dbg.declare, dbg.value for same variable. Gary/Nvidia has requested changes.
  AMD working on it.

  ARM:
  Fix for BigDFT conflicting use statements.
  https://github.com/flang-compiler/flang/pull/922 : Need approval from Shivaram. Bryan gave comments. Will run tests (Shivaram, Bryan).

  Huawei:
  https://github.com/flang-compiler/classic-flang-llvm-project/pull/5 : Reduce driver diff with upstream. Arm (Andrzej) has approved. Gary is working on it. Initial build on Power failed. Early next week for Shivaram.

  Nvidia:
  https://github.com/flang-compiler/flang/pull/919 : Fix for ICE. Gary investigating the review comments. No update yet.
  Gary would get back to this.

- New PRs to consider
  AMD:
  ARM:
  https://github.com/flang-compiler/flang/pull/939 : Publish Arm's documentation
  Concrete proposal for how the documentation should look like. RST format is OK? File location is OK? Contains: How to add an intrinsic, glossary.
  https://github.com/flang-compiler/flang/pull/660 : Enable support for SIMD directives.
  https://github.com/flang-compiler/flang/pull/742 : Unroll directive
  Two types of unroll for those with and without trip counts. C frontend is probably not important now.
  TODO : Improvements to directive metadata generation

  Huawei:
  938.
  Nvidia:

## Oct 7, 2020

Attendees
Arm: Kiran
Nvidia: Gary Klimowicz
AMD: Shivaram
Huawei: Bryan

flang-compiler/flang

- Status of PRs (flang, flang-driver, llvm)
  **AMD**:
  https://github.com/flang-compiler/flang/pull/908 : debug fix for byval. Not reviewed.
  https://github.com/flang-compiler/flang/pull/907 : dbg.declare, dbg.value for same variable. Not reviewed.
  https://github.com/flang-compiler/flang/pull/878 : Missing optimized debug flag. Approved by Jie, LGTM by Eric.
  https://github.com/flang-compiler/flang/pull/865 : Corrected emission of isoptimized flag. Approved by Jie and Eric.
  Above four are priority for AMD.

  https://github.com/flang-compiler/llvm/pull/86 Ported llvm debug changes for Fortran to llvm-9 #86 (not reviewed). Are they in LLVM-10? Few are in LLVM-11 and later. Need to backport to 10 as well. It is blocking all the below patches.
  Array related debug patches : #901(reviewed assumed shape, currently guarded for llvm 11/12), #902, #913, #925, #926
  https://github.com/flang-compiler/flang/pull/933 : xflags reserved.

Gary plans to have a look.
**Arm**:
https://github.com/flang-compiler/flang/pull/930 : Fix HTML documentation generation issue. Bryan has approved. Require approval from Gary and Shivaram.
**Huawei**:
https://github.com/flang-compiler/flang/pull/932 : Crash on empty derived types. Gary's approval pending.
https://github.com/flang-compiler/flang/pull/931 : Fix some segfaults. No approvals yet.
^ Higher priority.

https://github.com/flang-compiler/flang/pull/927 : Excluding tests on OpenPower for LLVM 10. Gary's approval pending.

**Nvidia**:
https://github.com/flang-compiler/flang/pull/919 : Fix for ICE. Gary investigating the review comments. No update yet.

- Status of PRs (monorepo)
  https://github.com/flang-compiler/classic-flang-llvm-project/pull/3 : Dwarf4/5 debug cherry-pick. Waiting for approval from Gary.
  Peter suggested changes. Bryan can move ahead with Peter's PR.
- Improving documentation for developers
  Putting up the documentation generated by flang as html pages in github.
  Michal: Switch to rst (from nroff) and link to wiki. Not replacing but adding new documentation.
  Bryan: Everything in source code and build github pages.
  Tutorials on how to write a pass, how to add an intrinsic, how to extend the parser, etc.
  How to debug flang.
- Future development
  New Fortran features (intrinsics, f2008/2018 features)
  AMD has some 2008 intrinsics (bit processing), more debug info.
  Nvidia has a lot of fixes. Needs issue list scrubbing to use "nvidia" label for issues that have fixes in nvfortran. Lower priority compared to processing PRs.
  Arm has a lot of fixes.
  General performance/usability improvements

  Nick: Release tagging for Classic Flang.
  More effort on testing tagged releases.
  Difficult with vendors on different llvm releases. Not our days jobs.

## Sep 23, 2020

Attendees
Arm:
Nvidia:
AMD:
Huawei:

flang-compiler/flang

- Status of PRs (flang, flang-driver, llvm)
  Arm: Fix for ICE when using implied do. Need approval Shivaram. Bryan has approved.
  https://github.com/flang-compiler/flang/pull/756
  Arm: Large arrays on stack. AMD is pending.
  https://github.com/flang-compiler/llvm/pull/85
  Huawei: Bryan will fill the list. Fixing reduction issues in maxloc/minloc. Bryan can go through the list and see whether any existing PRs should be prioritised.
  AMD: 921 pending to be merged. flang-driver/93 pending to be merged. Ported llvm debug changes for Fortran to llvm-9 #86 (not reviewed). #901(reviewed assumed shape, currently guarded for llvm 11/12), #902, #913, #925, #926 : all array related.
- Status of PRs (monorepo)
  Everyone has approved. Some issues on the Power Platform with Complex numbers. Two tests fail. Can these be marked unsupported on Power, and issues created and merged? https://github.com/flang-compiler/classic-flang-llvm-project/pull/1
  @bryan will merge after marking as unsupported and raise issues.
- CI
  http://80735715.packethost.net:8080/
  Once Bryan's patch goes in for 10 we can try to activate llvm-10 CI.
- Future development
  New Fortran features
  Debug/Usability improvements
  Bug fixes
  Xflag documentation: Some are reserved and undocumented.
  Rich: No process.
  Easy to make a mistake.
  Need consensus. Huawei documents internally for downstream. Can work on updating the flags. AMD adds new flags. Alok can raise a PR for reserved flag that is used in debug.
  Nick: Spack using old version of flang. Made a PR for master. Being reviewed. No hiccups. Using llvm-9. Couple of bugs encountered (OpenMP target). Will file issue. Some race condition when turning on OpenMP target for nvidia.

## Sep 9, 2020

Attendees
Arm: Kiran, David, Andrzej, Caroline, Ashok
Nvidia: Gary
AMD: Shivaram, Kiran Kumar, Sourabh, Alok
Huawei: Bryan

flang-compiler/flang

- Status of PRs (flang)
  Arm: Flexi app preprocessor fix #658 : Require review from all.
  Gary is testing.
  AMD: Enable debug of parameter variables #888 : Arm has provided feedback.
  Feedback addressed. Now has multiple patches as dependencies. How to go about this? Pending feedback from others.
  921 should go first.
  Dwarf version complaint. Flang: 863 (dwarf 4,5 xflags), driver: 89 , 90, 91
  Shivaram to start with driver patches.
  Nvidia: Fix array constructor for const arrays #919 : Arm has provided feedback, a non-conforming program is passed through the compiler without error. Pending feedback from others.
  Gary will check with nvidia.
  Huawei: Off by one min value fix #916 : Approval from AMD pending.
  Shivaram to look at it.
  Another patch for reduction routines in runtime. Correctness minloc/maxloc.
- Status of PRs (monorepo)
  Enabling classic flang with monorepo and porting changes from release_90
  Bryan has fixed and will push.
  2 debug tests will be skipped for classic flang.
  Combining those macros (CLASSIC_FLANG,FLANG_LLVM_EXTENSIONS).
  Shivaram will run after latest changes go in. Gary has not tried.
  Carol saw some failures

- CI
  - Rich (Arm) proposes to turn off the post-commit AArch64 Jenkins CI on release_70 and release_80. This would free up capacity to add release_100 (and release_110 later on). Does anyone depend on these branches?
    Gary OK.
    Carol will provide link to jenkins CI for flang on AArch64. Kiran to post.

    http://80735715.packethost.net:8080/
    Will add shivaram and Bryan to mailing list of CI.
  - Carol is working on turning on make check-all on the AArch64 CI
    - One regression test fails on release_70 (we would like to de-activate)
    - One regression test fails on release_90 and earlier. Going to backport fix from release_100 to release_90.
  - Arm would like to expand the post-commit CI to cover more targets. Has anyone got any experience with github actions?
    Andrzej: Free for open source projects. Building and testing llvm in a branch in github. Free for 2000 minutes, but using it more.
    Who owns flang-compiler? Steve is paying for it.
    Ashok: Openlab: Huawei is part of it. Is Bryan familiar with it?
    https://github.com/theopenlab/openlab/issues
- Future development
  New Fortran features
  Debug/Usability improvements
  Bug fixes

## Aug 26, 2020

Attendees
Arm: Kiran, Ashok
Nvidia: Gary
AMD: Shivaram, Kiran, Sourabh
Huawei: Bryan, Joey
Debian: Alastair

flang-compiler/flang

- Amendment to Pull request processing
  http://lists.llvm.org/pipermail/flang-dev/2020-August/000513.html
  Gary wants to test on Power.
  What is Nvidia's plan for Power? Gary will work to validate on Power within a week. For foreseeable future Gary will be the Nvidia face for Classic Flang. Gary has a framework now which is set up and easy to test. Gary should be able to process at least 2 per week. Maybe more but has other responsibilities. 895 and 910 before end of week. Gary has already worked through half of Arm's PRs from the queue below. We can revisit in two weeks.
- Public buildbot for all architectures
  Gary says Nvidia has plans for Power but internal for now.
  Alastair says for debian they have buildbots for Power.
  Alastair reports that some headers missing so top of tree not building. Gary corrects it might be llvm/flang. Monorepo for classic flang not yet working.

Debian has a history of publicly testing other projects. Will test on three platforms.
Pushing to Gary's list after arm and amd increases urgency.
AMD cannot have public buildbots. Only internal buildbots can be set up now.
Huawei cannot have public buildbots. Can test in private. Arm is the platform of interest.
Arm has a buildbot. It has some issues now. Will fix.
Can Bryan be a gatekeeper/approver? Approved. Gary will add Bryan to the project.

- Status of PRs
  https://github.com/flang-compiler/flang/pull/886 : stime
  Shivaram to merge this PR soon.
  https://github.com/flang-compiler/flang/pull/573 : pgmath clang friendly
  Arm will merge.
  https://github.com/flang-compiler/flang/pull/895 : Redundant lexical block
  This PR is after 910 on Gary's list.
  https://github.com/flang-compiler/flang/pull/910 : test reorganization
  Shivaram will have a look at this PR soon. This is next on Gary's list.
- https://github.com/flang-compiler/classic-flang-llvm-project/pull/1 : LLVM 10 monorepo
  (Build instruction: https://github.com/Huawei-PTLab/classic-flang-llvm-project/wiki)
  Shivaram had a look. Overall it looked fine. Have to build and check.
  This PR is next on Arm's list.

## Aug 12, 2020

Attendees
Arm: Kiran, Ashok
Nvidia: Gary Klimowicz
AMD: Shivaram
Huawei: Bryan

flang-compiler/flang

- Status of merging PRs
  Gary will approve hopefully by end of day.
  Pgmath clang PR (573):
  Gary will run on power. Bryan and Shivaram will look into this.
  Other pull requests to consider.
  OK to consider 910 after the current two.
  Questions about pull requests.
  About simd directives. Unroll. Working on prefetch directive.
- llvm-10/11 monorepo
  Rebased on 10.x and also rebasing 11.
  Posted an issue regarding debug.
  If there is a commit history to compare with 10.x upstream llvm. Compare button on the fork and then change the branch.
  Huawei has 10.0. Bryan can make PRs for 10.0 in classic-flang-llvm-project

Question about llvm version.
AMD 10. End of year 11.
Multiple versions of flang or llvm-project makes making packages difficult. We can move all changes to the monorepo and that would be ideal.
Harris Austin from ORNL attended to be aware of Fortran compilers for SUmmit.

- llvm_ir_correct tests PR#910 (https://github.com/flang-compiler/flang/pull/910). Can this be approved?

## July 29, 2020

Attendees
Arm: Kiran, Rich, Andrzej, Carol
NVIDIA: Gary Klimowicz, Jie, Varun Jayathirtha
AMD: Alok Kumar Sharma, Sourabh Singh Tomar, Shivaram
Huawei: Bryan

flang-compiler/flang

- AMD debug work presentation
  Engineers from AMD (Alok Kumar Sharma, Sourabh Singh Tomar) presented their debug work.
  http://lists.llvm.org/pipermail/flang-dev/attachments/20200729/43873062/attachment-0001.pdf
- ~~Status of Merging PRs. The flang-compiler/flang wiki has been updated with the new process.~~
  ~~@Kiran merged PR#875.~~
- ~~Debug Infrastructure PR#900 (https://github.com/flang-compiler/flang/pull/900). Can this be approved?~~
  ~~Gary ran ppc64le tests on this and it looked fine. Approved it.~~
  ~~@Kiran merged PR#900.~~
- ~~llvm_ir_correct tests PR#910 (https://github.com/flang-compiler/flang/pull/910). Can this be approved?~~
- ~~STAT= specifier bug #884 (https://github.com/flang-compiler/flang/issues/884). Does NVIDIA have any update?~~
- ~~_CLASSIC_FLANG: Can we define this for the preprocessor? Can we remove the __FLANG preprocessor definition? (I believe LLVM Flang is using it.)~~
- ~~LLVM version numbering.~~
  ~~Flang version numbering started off with 1.1, 1.2, ….~~
  ~~The binary releases were tagged with the data and time of the tag.~~
  ~~LLVM Flang will be using LLVM based major and minor numbers. We should too?~~
- ~~Llvm 10 monorepo patch~~
  ~~AMD CPU team will have a look.~~
- ~~AMD debug patches are pending review~~

## July 15, 2020

Attendees
Arm: Kiran, Carol, Andrzej, Bogdan, Irina
Nvidia: Steve Scalpone
AMD: Shivaram, Sourabh, Kiran Kumar
Huawei: Bryan
ANL: Nick

General

- Welcome (particularly for new members)
  Bryan : At Huawei working on 2 PoC projects with Flang.
  Alastair : Want to include flang in Debian based on LLVM 11.
  Nick : At ANL in the ECP project working with Hal and Pat. Application testing with DoE code.
  Irina and Bogdan: Arm intern/Graduate

flang-compiler/flang

- PR processing restarted. Merged https://github.com/flang-compiler/flang/pull/875
  @Kiran says we have restarted PR processing based on the process that @Gary mailed out to flang-dev. http://lists.llvm.org/pipermail/flang-dev/2020-June/000390.html
  @Bryan suggests that we rebase patches. Previous merges introduced difficulties.
  @Kiran says the new process recommends rebasing by the owner or by the admin at the time of merge.
- Has wiki been updated with the PR merge process?
  @Steve not aware.
  @Gary might get to this after he returns.
- New patches to merge
  Arm suggests: https://github.com/flang-compiler/flang/pull/573 (build pgmath with clang)
  AMD suggests: https://github.com/flang-compiler/flang/pull/900 (dbg test infra patch)
  No disagreements. To continue as per process.
- Reviewing debug patches.
  - Can AMD engineers give a quick presentation on these patches?
  - Can Nvidia and Arm bring the reviewers to this meeting (say next time)?

  @Kiran says AMD has submitted a series of patches for improving debug and they have been requesting reviews for a few months now. Since Jie (who works on debug at Nvidia) has been busy these patches have not yet been reviewed. If AMD can give a quick preso that probably can speedup the review.
  @Steve says Jie from Nvidia should be available next time.
  @Shivaram says AMD can consider presentation.
  @Kiran says there is an additional issue that some of the debug patches work with

debug improvements in LLVM 11. Without having a flang working with LLVM 11 it will be difficult to accept these patches.

- LLVM 10 monorepo
  - Bryan from Huawei has posted a monorepo fork with all the Flang patches (from release_90) https://github.com/Huawei-PTLab/classic-flang-llvm/
  - Has anyone reviewed?

  @Shivaram had a quick look but there are too many changes due to switch to LLVM 10.
  @Bryan says he has 8 patches from flang (llvm/flang-driver) and some bug fixes. There is a wiki page with instructions to build.
  @Steve says Gary/Kiran can make the repo in flang-compiler. @Kiran to follow up, have a look at Bryan's monorepo.
  @Nick says nice to have monorepo.
- Inlining minloc, maxloc
  @Bryan asks about minloc and maxloc inlining. This is an old issue (https://github.com/flang-compiler/flang/issues/387) in github. Code exists in semantics to inline. But that code is not enabled. Enabling it led to some issues.
  @Steve says no work in Nvidia to handle this. Might have come to the conclusion that implementing in runtime is better for performance. But this might have changed.
  @Shivaram says that they tried inlining minloc/maxloc but there are too many cases to handle and hence decided to drop it.
- readonly/readnone attribute not emitted in LLVM IR for pure functions.
  @Kiran says he recently found that readonly/readnone attributes are not emitted for pure Fortran functions. This leads to some missed optimization opportunities, like if there are two calls to the pure function with the same parameter in an expression then no CSE happens. Will create an issue and a PR.
- Forall statements introduced for array assignment. Temporary arrays are added which are sometimes not necessary. Temps created using mallocs causes high overhead when inside a loop.
  ```
  do indx=1,n
      arr2(1:dim1,indx) = arr2(1:dim1,indx)*arr1(indx)
  end do
  ```
  @Steve says that PGI frontend sometimes creates these temps. Are there any passes which removes these?
  @Kiran says there is a check which decides whether to insert these temps and this check can probably be made more precise to avoid it kicking in unnecessary cases.

## July 1, 2020

General

- Would anyone object to making this call completely open? Open call joining instructions advertised somewhere and open minutes? Perhaps rebranding as “classic flang technical call”?

Everyone (AMD, Nvidia, Arm) is OK with making this call open.
@Gary says there are three audiences and we should have calls to address these three.
We will continue with three for now.
@Kiran to inform everyone of this call by mailing to flang-dev.

flang-compiler/flang

- Status of Merging PRs. Can the wiki be updated and PR processing restart?
  Gary has some more test set up. This week there is Fortran con. Gary has a presentation.
  @Kiran will merge. #875. Kiran will own this, rebase and ask for votes.
- Debug Infrastructure PR (https://github.com/flang-compiler/flang/pull/900). Can this be approved?
  Gary will have a look.
- Llvm-10 monorepo patch
  AMD CPU team will have a look.
- AMD debug patches are pending review
  Debug person at Nvidia busy with a higher priority project. Others in the community can help.

llvm-project/flang

- Fortran conference 5 to 1pm thursday. Gary is giving a talk. Overview of the project of both classic and llvm flang and call for participation.
- Rich Bleikamp is volunteered to review FIR upstreaming patches.
- Kiran has a set of patches to the parser for OpenMP 5.0
  Steve suggests to add Bryan.
  Next time we can ask for more details.

## June 17, 2020

Attendees:
Arm: Kiran, Carol, Andrzej
Nvidia: Gary, Steve
AMD: Shivaram, Kiran Kumar, Anchu, Sourabh, Jinu, Sameeran

flang-compiler/flang

- Continue and Finalize discussion on Merging PRs
  @Gary has come up with a proposal to merge PRs and has a version (see below) ready for review by others. This version addresses comments from Steve and Rich. @Gary provided an overview of the process.
  @Kiran asks whether it is implicit in the process that the two reviewers are different from

the vendor who submitted the PR. @Gary says yes.
@Shivaram checks whether there will be further iterations. @Gary says he will mail it to flang-dev and depending on comments there might be. @Gary will post this in the flang-compiler wiki and then it will be final.
The process requires a list of reviewers. @Shivaram, @Kiran, @Gary to provide a list of reviewers who can review and approve for the vendors.

- Monorepo, LLVM 10/11
  @Kiran (Arm) asks about plans for the monorepo and moving to LLVM 10/11. This was discussed in the previous call. @Gary says the immediate plan is to continue with LLVM 9 and the process (for merging PRs) requires testing on at least LLVM 9.
  @Steve asks whether AMD wants to push the changes for the monorepo to the driver for LLVM-10? This will need creation of the monorepo in the flang-compiler github page. Appropriate permissions can be provided. @Shivaram will check and get back whether AMD can take this up.
- Debug testing infrastructure PR (https://github.com/flang-compiler/flang/pull/900)
  @Kiran (Arm) says this PR is required to add unit tests in debug PRs. Kiran has reviewed the PR, can someone from Nvidia have a look and approve?
  @Gary says he will have a look and approve.
- @Steve asks whether anyone has gone through existing PRs and checked for tests and requesting updates.
  @Kiran says he had a look at some of the recent PRs and asked for tests. But not the older ones, these likely have bit-rotten and would need maintenance. And yes, someone should go through the PRs and request to rebase with the master branch.

llvm-project/flang

- Apart from fir-dev, are there other branches to watch for monitoring progress on downstream Flang implementation?
  @Kiran asks this question particularly for non-FIR changes like runtime changes. @Gary says the runtime changes are in a sandbox. @Steve says trying to add the runtime changes to phabricator. Would like it not to go to fir-dev.

Proposed PR merge process (full text of what we would like to put on the Flang wiki):

> *The goal of the Flang Community Pull Request Process is to ensure that only well-tested patches are incorporated into the github.com/flang-compiler/flang master branch. Most Fortran tests, benchmarks, and applications are proprietary, and therefore are not part of the flang public test suite, and these are the tests that need to be run in order to continue to maintain the quality of flang.*

*To that end, a group of individuals and organizations have volunteered to test and discuss pull requests. The process allows just two assenting votes before a gatekeeper is allowed to merge a pull request into the master branch.*

*Current volunteers (reviewers) and their commitments:*

*….*

*If you would like to be a volunteer, please submit your request to flang-dev.*

*Current gatekeepers:*

- *Gary Klimowicz and Steve Scalpone, NVIDIA*
- *Kiran Chandramohan and Rich Barton, Arm*
- *Shivaram Rao, AMD*

*Gatekeepers are chosen because of their involvement in flang development. If you would like to be a gatekeeper, please submit your request to flang-dev.*

*The proposed process:*

1. *Each pull request has an owner. The owner is typically the person who created the pull request; however, anyone may adopt a pull request. The owner is responsible for making sure the PR can be merged with master and for handling feedback during the review and testing process. The owner can signal that a patch is ready for review using flang-dev or the flang slack channel.*
2. *As part of the review process, the reviewers agree to apply the patch and run their internal test suites with it, and then give feedback on the patch. Note that feedback might be limited to simply pass or fail because of restrictions on the test suites.*
3. *Volunteer testers and reviewers offer feedback on the PR in the GitHub discussion on the pull request.*
4. *Even for the most trivial patch, at least two volunteers must apply the patch, run their internal tests, and give feedback on the patch. The patch should be run on at least Arm, x86 and OpenPOWER on LLVM 9.0.*
5. *When two reviewers complete their testing and approve the patch, a gatekeeper can merge the patch. Note that a gatekeeper is not mandated to merge a patch at this point if the discussion and consensus process has not run its course.*

## June 3, 2020

Attendees:
AMD: Shivaram, KiranTP, Anchu, Sourabh, Rich Bleikamp

Arm: Kiran, Rich, Andrzej, David, Caroline
Marvell:
Nvidia: Gary Klimowicz

flang-compiler/flang

- Merging PRs
  @Gary says he can write down the PR process and then it can be iterated on based on feedback from others. Can set up a google doc or email. AMD prefers email. <so put your name here if you want to be included: Gary, Rich (Arm), Kiran (Arm), Shivaram>. @Gary says we can also include Huawei since they seem interested. And since it is an open source project there should be no issue. @Kiran says the feedback he received was that public mailing lists are fine for interacting with Huawei but might have issues with private mail. @Gary suggests flang-dev@flang-compiler. @Rich asks about flang-dev@llvm. @Gary agrees to try LLVM flang-dev if it is not too much of a distraction for the llvm/flang folks.
- LLVM10, monorepo, Huawei request etc
  @Kiran thinks Huawei needs the monorepo for making PRs. @Rich says Huawei is motivated to contribute. @Gary agrees and suggests setting up the monorepo. @Kiran says Arm will only be going to LLVM 11 and hence cannot take this up now. @Shivaram says AMD has moved to LLVM 10 and did not find any significant issues.
  @Gary says he can reply to Huawei's mail, point to this call and minutes. We will have to replicate changes to llvm, flang-driver in monorepo. @Gary can help with the process.
- AMD debug patches : Can unit tests be added to all these? Either at the LLVM IR level or at the object level.
  @Shivaram says adding tests is not an issue. But how will they be tested? @Sourabh says that when they made the initial PR, tests were not recommended. When issues were found in debug code it was fixed at AMD and were manually checked. @Kiran says in Arm's downstream changes they have a llvm directory in tests which contain lit style tests for debug. Another option would be to add checks for debug in the verify section of the makefiles in the execution tests. Having tests will ensure no regression.
  Agreement to add debug tests directory and update existing PRs/create new PR with tests. @sourabh/@shivaram to follow up.
- Status of fix for [https://github.com/flang-compiler/flang/issues/872](https://github.com/flang-compiler/flang/issues/872) and availability in Flang.
  @Gary says the fix is in queue but is blocked by the merge process. @Rich asks whether we can set a date for this. @Gary will mail regarding the process by the end of the week. Try to have the process by next call. @Gary expresses some concerns regarding discussing in flang-dev. @Rich says since it is old Flang, we have lower requirements for addressing opinions of non-users.

llvm-project/flang

- Driver RFC
  http://lists.llvm.org/pipermail/llvm-dev/2020-June/141994.html
  @Andrzej looking for feedback. @Gary will raise in the F18 meeting at Nvidia.
- FIR upstreaming. Thanks for Eric making stubs for the OpenMP lowering. And also upstreaming changes to the PFTBuilder.cpp. Will Eric be upstreaming the lowering bridge code also?
  @Gary says yes Eric will continue to upstream lowering code. But no timelines.
- Co-array work. Do you have info about Rasmussen's work? Is he undertaking a pilot study or is planning to do the full implementation for single image co-arrays?
  @Gary says Rasmussen has approval from LLNL for working on llvm/flang. Previously he worked on using F18's parser for the ROSE tool, now it is Kate who looks into it. Rasmussen's interest is in implementation for co-arrays.

## May 20, 2020

Attendees:
AMD: Kiran TP, Anchu Rajendran, Shivaram Rao, Sameeran, Ron Lieberman
Arm: Kiran Chandramohan, Nathan Sircombe, Caroline, Andrzej
Marvell:
Nvidia: Gary Klimowicz, Steve Scalpone

flang-compiler/flang

- Merging PRs : Have given +1 to PR#875 in slack channel
  @Kiran says it is +1 from Arm for merging PR#875 to master.
  @Gary says it was not clear whether it was a +1 for the idea or a +1 for merging the PR.
  @Kiran says mechanism not understood for voting.
  @Gary suggests may be vote in the original PR. But says there might be differences between the final commits in the staging branch and the original PR.
  @Steve suggests the following:
  1. Bring the PR which is to be tested for merging up to date with master. This has to be the responsibility of the original author or if the author is not available one of the people with commit access can take ownership.
  2. Then this PR can be taken to the staging branch and tests run by Nvidia, Arm and AMD. Gary's role is testing the staging branch with the PR on power/x86.
  3. If OK the original PR can be merged in master and staging branch brought up to date with master. Else report issues to the author of the PR who updates the PR with necessary changes and we go to step 2.
  Gary, Kiran and Shivaram agree on this process for merging PRs.
- Issues (889-896)
- Status of building CP2K with Flang - Arm have reported a bunch of issues related to this.

@Steve says people at Nvidia worked on CP2K. Not sure of engagement. But no current active CP2K issues on PGI compiler.
@Shivram says CP2K page says that it does not compile with PGI. AMD has fixes for the compilation issues, but not sure whether it fixes all the issues. Combination of MPI rank and OpenMP causes issues at runtime. Can make PRs for fixes. @Kiran encourages AMD to make PRs, will review.
@Nathan says similar issues (MPI + OpenMP) seen in Tealeaf. Had to revert procedure pointer fixes because of this.

- Status of fix for https://github.com/flang-compiler/flang/issues/872. Back in Feb a fix was in progress in Nvidia.
  @Gary will check with Mark.
- @Steve asks whether AMD has PRs for debug in Flang and LLVM.
  @Shivaram says AMD has PRs for debug DWARF generation. Another PR in LLVM. There are three engineers working on improving debug in AMD.
  @Ron Lieberman says there is a debug RFC proposal from AMD. @Steve is not referring to this.

llvm-project/flang

- Placeholder for upstreaming from fir-dev to llvm-project/flang
  @Kiran (Arm) will get back to this only end of this week or early next week. Will work in fir-dev to refactor and reduce diff of current PR (https://reviews.llvm.org/D79731).
- @KiranKumar (AMD) says they have reviews in phabricator for Data Statement and Select type. Addressed review comments. Waiting for some time. Will be great if @Gary can follow up with Peter Klausler and Tim for further reviews.
  https://reviews.llvm.org/D78424
  https://reviews.llvm.org/D79851

## May 6, 2020

Attendees:
NVIDIA : Gary Klimowicz, Steve Scalpone
Arm : Caroline Concatto, David Truby, Andrzej Warzynski, Kiran Chandramohan
AMD : Shivaram, Kiran Kumar, Anchu Rajendran
Marvell : Wei Zhao

flang-compiler/flang

- Process for merging PRs.
  @Kiran-arm has created the staging branches in the flang, flang-driver and llvm repos.
  @Gary has the environment set up to do some testing on PRs. Planning to test 3 architecture platforms and llvm 7,8,9. This will be nine set of runs. What is the expectation for testing?
  @kiran-arm is planning to merge the code into their compiler (based on llvm-9) and run some hpc applications and testsuites.

@Gary has to run on power and x86. OK to give up aarch64. Would like to test 2 versions of LLVM.
@shivaram says AMD will run some performance benchmarks and regressions tests on AMD machines. Cannot do all testing in advance. Will there be a process to revert?
@Steve says participants can propose the PRs to move to staging. And test in batches. Assumption is that the proposed PRs are tested. 2 votes to promote from staging to master. Arm can vote for their platform. AMD can also test. This is all precommit testing.
@Gary says have already tested the text file change PR. **@Gary will inform AMD and Arm in slack once it is moved to staging.** Would like to know which patches are candidates to be merged in advance.

llvm-project/flang

- OpenMP Parallel Operation patch is in review. https://reviews.llvm.org/D79410
  @Kiran-arm says @David has created a phabricator review for the OpenMP parallel operation. Currently it has a simple pretty printer and parser. Will improve on it with another one after this is accepted.
- Re-evaluating driver design.
  @Andrzej and @Carol are re-evaluating the driver design. Will share details once a decision is made. @Andrzej thanks @Steve for the feedback.
- Build bots
  @Carol says Arm buildbot is currently up as experimental. @Galina has accepted this and will move to production soon. It builds with gcc 9.3 for Arm platform, trying to get another machine to build with Clang. The buildbot is in zorg.
  No update on buildbots from PGI and AMD.
- CMakefile progress
  - Libraries
    @Steve says @Tim will probably merge the two libraries (FortranEvaluate and Semantics) into a single one and remove circular dependency. This will fix issues for shared builds.
  - Compilation options
    No discussion
  - Pre-commit testing and review process
    @Steve says there has been some back and forth on some patches relating particularly to CMake changes. People could say what platforms they built it on and tested before submission and approval. So that we do not go on merging and reverting patches.
- Outstanding llvmification work. (asserts + post merge work)
  - Are will still tracking these tasks in flang-compiler?
    @Kiran says currently being tracked here. We agreed to do some of the post merge work before the next LLVM release. Arm is currently focused on the driver and OpenMP work, so is looking for help from the community. @RichBarton was planning to discuss this.
  - Any progress on llvm_unreachable?

@Steve would like some one to complete the llvm_unreachable portion. **@David will look into this after some urgent work on the OpenMP parallel patch.**
    - assert() discussion (Steve) -- wants to wait until llvm_unreachable is resolved
- FIR
  @Rich Bleikamp is working on fir-dev branch and has some fixes. Can a PR be made to this branch? Some fixes in runtime and lowering.
  @Steve says for fixes in runtime raise PRs in llvm-flang since the development is happening there. Lowering fixes can be to the fir-dev branch. Fir-dev folks are receptive to changes for F77.

## April 22, 2020

Attendees
Arm Ltd : Kiran, Carol, David, Rich, Nathan
Nvidia : Gary Klimowicz, Steve Scalpone
AMD : Shivarama Rao, Kiran Kumar TP, Anchu Rajendran, Sameeran Joshi
AMD GPU : Rich Bleikamp

flang-compiler/flang

- Is it OK if we go ahead and merge a PR? A unit test or cmake patch?
  Steve says our experience shows that CMake changes are not necessarily simple.
  Gary says we would like to follow the process for all PRs. Always go through the staging branch. Steve agrees.
- Has a staging branch been set up?
  **Steve says Kiran can set up a staging branch**. And then Steve/PGI can set up a buildbot.
  Gary says this can be a permanent branch. All repos (flang-driver/llvm) should have the same branch.
  Kiran asks about how to move PRs to the staging branch and how AMD can be part of this without commit access.
  **Gary can provide commit access to one person from AMD. Shivaram to mail Gary.**
  Steve says first set up branch and buildbot and then get to merging PRs to stage branch.
  Gary says this will need some communication mechanism like a Slack channel. **Gary created slack channel #classic-flang-pull-requests. And will update wiki.**
  Steve says a related issue is LLVM 10. Varun (from Nvidia) did the updates for LLVM 10. Not merged in yet. No problems so far. Have not run big benchmarks. It will be a large merge for flang-driver. Eric usually does it but probably cannot. Looking for volunteers here. The PGI compiler uses LLVM but does not use the flang-driver.
  **Kiran will check internally and update on moving to 10 for clang/llvm.**

- A semantic check is missed for block construct. Also in F18.
  https://github.com/flang-compiler/flang/issues/885
  Steve says he has assigned the issue to Andrzej as requested.

llvm-project/flang

- What's happening with the asserts conversation with David Blakie . Steve/Tim were up for making a proposal to llvm-dev. Update on that? We shouldn't leave this hanging too long.
  Steve says we will get there. There is a question that has to be discussed with the LLVM community. Are assertions allowed in release builds?
  Steve is looking for volunteers to clean up the unreachable portion of the code.
- Newline handling in parser - David and Steve offline conversation
  Steve says the main reason he is concerned is because it shows up in CMake.
  David has a patch for this. David also would like to fix the underlying issue.
  Steve had some comments on the patch to clean up. David says he has updated the patch and has addressed Steve's comments.
- David's OpenMP combined constructs semantics patch - can we commit and then refactor please? Patch: https://reviews.llvm.org/D77812, Steve's comments: http://lists.llvm.org/pipermail/flang-dev/2020-April/000284.html
  Steve says he does not understand the point of making this change and then again making changes to share code with LLVM. Kiran says at the moment the sharing is a proposal, we are not sure when it will land and it might require multiple patches. This patch from David is probably not a big patch. David says it is 300 lines of code and another 600 lines of tests and we would need the tests anyway. Steve says checking in is OK. He will get to this later on.
  Steve adds that with Brian away there will not be much OpenMP work by PGI. Kiran says we are aware.
- Use of Werror by default; should this be removed to not block compilation with new compilers?
  Not discussed.
- Shared library builds.
  Kiran (AMD) asks about shared library builds. Is anyone working on this?
  David will get to it. But not working on this currently.
- @AMD GPU team, now that F18 is in LLVM, could you share what your plans/interests are?
  Rich Bleikamp has spent some time on reading through the F18 code, building and running a few applications. Rich Bleikamp has not determined what to work on. Is open to suggestions, may be LLVM IR gen, Apps to run, OpenMP.
  Rich Bleikamp adds that Ron might be directly interested in OpenMP work.
  Rich Barton would like to set up a call. Bleikamp says Ron on Central time and Rich Bleikamp on Eastern time. **RichBarton to set up a call.**
- Doxygen PR - Need clarification on the expected behaviour with below values:

-DLLVM_ENABLE_DOXYGEN=ON -DLLVM_BUILD_DOCS=OFF
-DFLANG_INCLUDE_DOCS=OFF
David says that he was not initially sure whether this will be a valid combination of flags. Sameeran says this means doxygen llvm and mlir target exists and flang does not. David thinks this set of flags will practically not be used so can go ahead and merge.

- FIR : updates on upcoming patch
  Kiran (AMD) asks about the status of the patch that lowers from AST to FIR. Rich suggests that Kiran (AMD) can send a mail to JeanPerier and Steve. **Kiran (AMD) to send a mail with others in CC.**
- @AMD team: We can take up work related to OpenMP MLIR dialect.
  Kiran (AMD) says AMD would like to take up OpenMP MLIR work. They have people available for this. But would like to know what constructs they can work on.
  Kiran (Arm) says that AMD can work on flush taking inspiration from the parallel operation discussion. Keep variables as of AnyType and during lowering to LLVM dialect or translation to LLVM IR catch the types that do not work for these constructs. Also adds that the constructs that we can work on are limited by what is supported in OpenMPIRBuilder. You can find the status of OpenMPIRBuilder in the following google-doc.
  https://docs.google.com/spreadsheets/d/1FvHPuSkGbl4mQZRAwCIndvQx9dQboffiD-xD0oqxgU0/edit#gid=0
  **Kiran (Arm) is considering setting up a call to discuss OpenMP work.**

## April 8, 2020

Attendees
Arm Ltd : Kiran, Carol, David, Rich, Nathan
Nvidia : Gary Klimowicz, Steve Scalpone
Marvell : Wei Zhao
AMD : Shivarama Rao, Kiran Kumar TP, Anchu Rajendran
-> LLVM submission
Issues (if any) to discuss.
https://github.com/orgs/flang-compiler/projects/8

*@Kiran Chandramohan says that the only remaining issue seems to be the llvm_unreachable issue that David Blaikie raised. @Stephen Scalpone asks whether that will be a blocker. @Richard Barton says David Blaikie will be*

*happy with an agreement. @Stephen Scalpone is happy with DavidTruby's suggestion. Can +1.*

*@Richard Barton says Mehdi reported an issue. @David Truby says he could not reproduce. Could it be a shared library build issue? @Stephen Scalpone could not build a shared library build. Tried with gcc8.3 on ubuntu 18.04 (?). @Richard Barton says Mehdi's issue need not be a blocker. @David Truby says due to a configuration issue on Mehdi's system the module files could not be found. @Richard Barton says can fix or merge and then investigate. @Stephen Scalpone suggests asking Mehdi. @David Truby to continue working with Mehdi.*

*@David Truby has set up bugzilla. An issue was reported there. There was a library ordering issue. Works fine with lld but has issue with bfd. libevaluate needs to have libsemantics.*

*@Stephen Scalpone is not ready with email to list. Will get to it.*

*@T P, Kiran Kumar asks regarding Doxygen PR. @Stephen Scalpone says there were some issues to be fixed. @Sudhakumari, Anchu Rajendran says that Sameeran fixed and was able to build outside the tree.*

*@David Truby says people with existing PRs will have to move them to Phabricator. @Stephen Scalpone says will send a mail having instruction for issues and PRs. Issues can be removed form github page but Pull Requests cannot be. There is a daemon which can be set up to auto-close with a nice message.*

*@Rao, Shivarama says AMD has issues with setting up buildbot outside firewall. Will take more time due to lockdown. No ETA.*

Flang
-> Placeholder for issues with execute_command_line.

*@Caroline Concatto reports issues with cmd_stat and command_msg. For asynchronous and invalid cases.*
*@Gary Klimowicz says have some fixes that are not pushed into.*
*@Caroline Concatto says issues seen in v53/u2b of nag.*

-> @Caroline Concatto shared test failures in nag-testsuite for Block construct tests.
-> Making progress with Flang and accepting PRs. @Stephen Scalpone said "For flang-compiler/flang, you and Gary might consider a two-phase commit & test policy.  For example, create a new branch, call it stage.  Pull request would first be pulled into stage.  The stage branch would be tested, problems addressed, and then declared good.  Once good, the stage branch would be merged into the main branch and declared good."

*@Stephen Scalpone says a staging branch is good for proprietary tests. Many PRs can be tested at the same time. @Richard Barton asks when will someone know that it is ready to merge back to master. @Gary Klimowicz says may be companies can individually accept Pull Requests on the stage branch and when everyone has approved it is ready to merge back.*

AMD
-> General update on Flang/F18 work there.

*Already covered in F18 section.*