GSOC 2018 Proposal

Title: Textual Representation of LTO Object Files (Textual LTO dump tool project)
Organization: GNU Compiler Collection (GCC)
Mentor: Martin Liška

Name: Hrishikesh Parag Kulkarni
Country: India
School and University: Pune Vidyarthi Griha's College of Engineering and Technology, Pune
(http://pvgcoet.ac.in/)
Savitribai Phule Pune University (SPPU, Pune)
Email: hrishikeshparag@gmail.com
Website: www.hrishikeshkulkarni.com

Introduction:

As far as I understand, the motivation for LTO framework was to enable cross-file interprocedural optimizations, and for this purpose an ipa pass is divided into following three stages:

1. LGEN: The pass does a local analysis of the function and generates a "summary", i.e. the information relevant to the pass and writes function IL to LTO object file.
2. WPA: The LTO object files are given as input to the linker, which then invokes the lto1 frontend to perform global ipa analysis over the call-graph and write optimized summaries to LTO object files (partitioning). The global ipa analysis is done over summary and not the actual function bodies.
3. LTRANS: The partitions are read back, and the function bodies are reconstructed from summary and are then compiled to produce real object files.

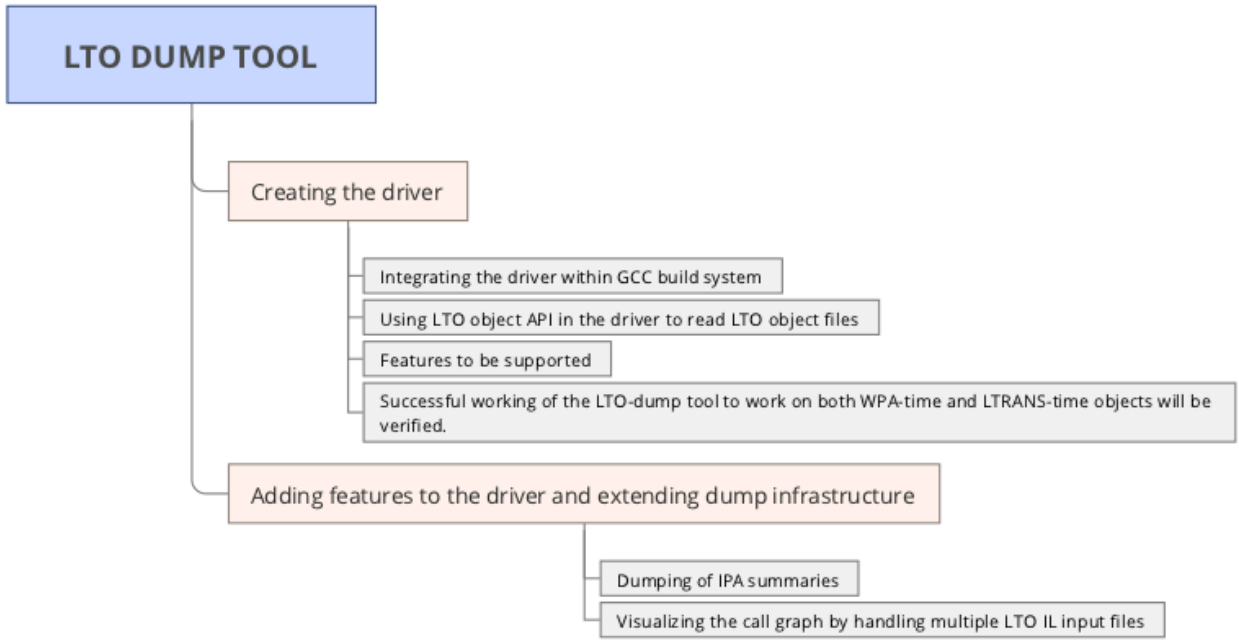
LTO byte code:
The LTO object file is a regular elf file with sections containing LTO byte-code. A LTO object file contains various sections for storing command line options, symbol table, global declarations and types, function bodies in GIMPLE, ipa pass summaries, ipa references, static variable initializers and the call graph.
There are couple of limitations of the byte code format:
1] It is not self descriptive, which makes it harder to debug.
2] The byte code is essentially a "serialized" version of in-memory representations, which makes it prone to break across versions.

The purpose of this project is to create a dump tool for easily analyzing LTO object files similar to readelf or objdump -d for regular ELF object files.

Major activities to be carried out:

1. Creating the driver
    a) Integrating the dump tool within GCC build system.
    b) Using LTO object API in the dump tool to read LTO object files.
    c) Initial list of options that could be possibly added to lto-dump:
    --all: Dump all the info in the LTO object file.
    --symbols: For dumping a list of variables and functions contained in the IL
    --var=<name>: Dump info about the requested varpool_node including it's initializer if present.
    --function=<name>: Dump info about requested cgraph_node.
    --cmd_opts: Dump command line options present in the LTO object file
    d) Successful working of the LTO-dump tool to work on both WPA-time and LTRANS-time objects will be verified.

2. Adding features to the driver and extending dump infrastructure
    a) Dumping of IPA summaries.
    --summary=<passname>: Dump the summary of the requested pass.
    b) Visualizing the call graph by handling multiple LTO IL input files.
    --cgraph: Dump textual call graph of combined LTO object files.
    --cgraph-dot: Create a dot file for visualizing the call graph.



Example of IPA pass summary:

Suppose the user requests to dump summary of ipa-pure-const pass:
lto-dump --summary=ipa-pure-const foo.o bar.o
Assuming that there are a total of 10 functions in .gnu.lto_.pureconst of both the LTO object files, the dump summary could possibly look like:

Count = 10.
Assuming one of the functions is called “foo”, it could dump the function state corresponding to foo:
foo:
pure_const_state = <value>
state_previously_known = <value>
looping_previously_known = <value>
looping = <value>
can_throw = <value>
can_free = <value>
malloc_state = <value>

Sample code for dumping function state of all cgraph_nodes:

```
static void
pure_const_dump_summary (void)
{
  cgraph_node *node;
  FOR_EACH_FUNCTION (node)
  {
          funct_state fs = get_function_state(node);
          fprintf(stderr, "pure_const_state = %d\n”,
                          “state_previously_known = %d\n”,
                          “looping_previously_known = %d\n”,
                          “looping = %d\n”,
                          “can_throw = %d\n”,
                          “can_free = %d\n”,
                          “malloc_state = %d\n",
                          fs->pure_const_state, fs->state_previously_known,
                          fs->looping_previously_known, fs->looping, fs->can_throw,
                          fs->can_free, fs->malloc_state);
  }
}
```

The pure_const_dump_summary could be a hook, say dump and could be called as:

```
ipa_pass *pass = static_cast<ipa_pass *> (get_pass_by_name (passname));
gcc_assert (pass);
if (pass->dump)
  pass->dump ();
```

Tentative Plan for execution of Project:

Timeline	Activities	Deliverables
Up to 23 April:	Understand Project scope, understanding driver requirements	
April 23 to 13 May	Initial Decoding of problem	Clear representation of driver specification from coding perspective
First Half: May 14 to June 24	Create the driver	The LTO dump tool with above mentioned features
Second Half: June 25 to Aug 06	Add more features to driver and any corresponding change to dump infrastructure	Changes to the LTO dump infrastructure: 1) dumping of IPA summaries 2) Visualizing the call graph by handling multiple LTO IL input files
Aug 07 to Aug 14	Buffer period and Final submission	Final submission

Since 14 May to 27 May are my Engineering Exams I will compensate for this period by working extra hours during last week of April and First week of May.

My brief Profile:

I am an undergraduate student in Computer Engineering at Pune University. I am reasonably familiar working with C, C++ and python. I have some rudimentary knowledge of Node JS. My prior experience includes opportunities to work in areas of NLP. Some of my accomplishments in the area include presenting project VicharDhara- A thought Mapper that was selected among top five ideas in Accenture Innovation Challenge among 7000 nationwide entries. My paper on this topic won the best paper award in IEEE Conference ICCUBEA-2017. My previous work was focused on simple parsers, student psychology, thought process detection for team selection. My team is also qualified for Grand Finale of Smart India Hackathon-2018 among top twenty five teams across the nation under Ministry of Urban Development.

Regarding GCC, I have gone through cited documentation, wrote a couple of toy patches, built GCC on quite a few times. I tried to visualize the scope of the project by seeking inputs on my understanding from GCC community. That has given me initial idea about this project and in this document I have tried to build this proposal based on this understanding.

Mailing list discussion on the project:
https://www.mail-archive.com/gcc@gcc.gnu.org/msg84662.html
https://gcc.gnu.org/ml/gcc/2018-03/msg00010.html

My recent Publications:
[1] Hrishikesh Kulkarni, "Contextual Data Representation Using Prime Number Route Mapping Method and Ontology" IEEE Conference, ICCUBEA, 2017
[2] Hrishikesh Kulkarni, "Intent Action Ontology and Tone Matching Algorithm for Organizing News Articles", IEEE Conference, ICECDS, Chennai, 2017
[3] Hrishikesh Kulkarni, "Intelligent Context Based Prediction using Probabilistic Intent-Action Ontology and Tone Matching Algorithm", IEEE Conference, ICACCI, Manipal, (2017)
[4] Hrishikesh Kulkarni, “Multi-Graph based Intent Hierarchy Generation to Determine Action Sequence”, Springer Conference, ICDECT, December 2017, Pune
[5] Hrishikesh Kulkarni, “Thought Process based Team Member Selection Using Contextual Sentiment Closeness,” IEEE Mumbai Chapter Conference International Conference for Convergence in Technology (I2CT), April 2018 (Accepted)

Relevant Experience

Skills:
a) Reasonable familiarity with C, C++ and Python
b) Working knowledge of Node JS and PHP.
Projects:
a) Simple interpreter using C.
b) Using Python, NLTK and Node JS a project to assess thought process

Why GCC?
I find compilers quite interesting as a subject and this interest is converted into this proposal. It was actually very encouraging and engaging to get response and inputs from GCC community. I believe that this opportunity will help me to convert my interest into serious outcome and useful contribution while getting exposure and understanding of real world compilers.

What are the benefits for the project? Why is it important? (Benefits to GCC)
1. Easy access to LTO object files.
2. Useful for debugging LTO streaming issues.

References:

[1] http://www.ucw.cz/~hubicka/slides/labs2013.pdf
[2] https://gcc.gnu.org/wiki/LinkTimeOptimizatio
[3] https://gcc.gnu.org/onlinedocs/gccint/LTO-Overview.html
[4] https://gcc.gnu.org/onlinedocs//gccint/LTO-object-file-layout.html
[5] https://pdfs.semanticscholar.org/cafc/c15a1602c5a8090606333b3bdb42e9e80654.pdf