Motivation

MergeFunctions is an optimization pass of LLVM which merges identical functions. The Comparison of functions boils down to the comparison of basic blocks and function attributes. The comparison of functions is the result of comparison of each instruction. MergeFunction uses FunctionComparator in LLVM utils to introduce total ordering among functions which hash to the same bucket and then implements a logarithmical search on the ordered set.

However, as new types are added to the IR or if the semantics of the IR changes, MergeFunctions can cause miscompiles (e.g.,https://reviews.llvm.org/D79261). The problem lies in the comparison of IR instructions. When the semantics of IR operations change and MergeFunctions is not updated with abilities to compare the new ones, the optimization happily merges functions which may be different in ways unknown to the optimization. So, the primary goal would be to clearly specify the equivalence of each IR operation.

MergeSimilarFunctions is an optimization which can merge not just identical functions but also functions with differing instructions. It achieves this by partial comparators which account for the differing instructions between two functions. We propose to integrate precise and partial comparators of MergeSimilarFunctions to tablegen so that the optimization can call into the auto-generated C++ code for comparing IR instructions. This will help maintain MergeFunctions/MergeSimilarFunctions in sync with the IR. Then, it would be ideal to integrate the features of MergeSimilarFunctions into MergeFunctions. A decision regarding when to schedule the MergeSimilarFunctions pass so that the compile-time is improved is to be determined. Any other optimizations(like function-outlining) which may relate with MergeSimilarFunctions need to be taken care of. In light of these factors, this plan has been designed. Also, documentation is done continuously as that is a benchmark of GSoC.

Tentative Timeline

- **Community Bonding Period**
    - Read the code of MergeSimilarFunctions and MergeFunctions
    - Read about tblgen.
    - Send out the proposal to the llvm-dev mailing list for review.
    - Read publications about *MergeSimilarFunctions* in more detail.

- **Week 1 - 3**
    - Add precise equality to each IR type. Currently, MergeFunctions uses the FunctionComparator defined FunctionComparator.cpp in LLVM Utils. cmpOperations() method from this module is being used to compare two instructions. This function doesn't support equality of every IR type, when presented with a new IR type can lead to a decision driving mis-compiles. So, the precise equality of every IR operation needs to be defined and integrated to be able to compare functions effectively.
    - Figure out a benchmark that can be used to measure compile-time and code size. Build a GitHub repository which checks some of the good large C++ codebases like Chromium, Webkit, MySQL, etc and figure out the benchmark which can be used.

- **Week 4 - 6**
    - The precise and partial comparators are used to compare whether functions are exactly identical or whether functions are similar to a certain extent. Integrating these partial and precise comparators of MergeSimilarFunctions to use the equality operators defined for comparison to maintain sync of IR with MergeSimilarFunctions is vital.
    - Figure out when to run MergeFunctions, that is when to do merging of exact similar functions. If exact similar functions are merged at the beginning, the amount of optimization work decreases.

- **Week 7 - 9**
    - Integrate features of MergeSimilarFunctions into MergeFunctions.
        - Port MergeSimilarFunctions to MergeFunctions and add flags to selectively enable whether to do precise merging or partial merging.
        - Some notable high-level functions / ideas from MergeSimilarFunctions which may need to be moved into MergeFunctions :
            - doDiffMege() : To perform merging of different functions.
            - Setting up a Registry class to handle buckets of similar functions.
            - Adding FunctionComparators with the ability to account for similar functions and differing instructions.
            - Calculation of Similarity metric, a metric used to account for the level of similarity between two functions.

    - Figure out when it is ideal to schedule MergeSimilarFunctions pass. The MegreSimilarFunctions pass merges functions which are similar. If run at the wrong stage, this can lead to an increase in the code-size by merging functions which might have been dumped entirely.
    - Testing with the benchmark for code size, bootstrapping the compiler for correctness, running llvm test suite for correctness.

- **Week 10 - 11**
    - Investigate interactions with other optimizations like inliner, function outliner.
    - Workaround any other optimizations which might conflict with MergeSimilarFunctions.
    - Testing with the benchmark for code size, bootstrapping the compiler for correctness, running llvm test suite for correctness.

- **Week 12**
    - Investigate the debuggability of the functions merged using the improved MergeFunctions pass and how it can be improved.
    - Review the work done.