Introduction

This project concerns the implementation of a novel programming language feature — Multi-Entry Function — inside of LLVM project's compiler backend. Multi-entry functions are a generalization of the "function" feature of programming languages; instead of restricting the program to jump into the function body from only a single entry point, a multi-entry function allows the program to jump into multiple pre-defined entry points into its body.

```
def _add:
    increment(x, y):
        x, y = x, y + 1
    plus(x, y):
        pass
    return x + y

increment()
plus(0, 0)
```

Example: Multi-entry function _add with entry points increment and plus

Motivation

A Multi-entry function is a zero-cost abstraction of the "function" concept. There is zero overhead in the generated binary if a multi-entry function is defined with a single entry point. But having multiple entry points into a block of code can allow for greater programmer flexibility (like setting variable defaults) and optimization opportunities for the compiler (like register optimizations and call pattern optimization).

Background

Focus of my work

Source languages

Target architectures

Changes to LLVM IR

New IR class hierarchy

```
Class MEFBody:
    multi-entry body

Class MEFEntry:
    entry point
```

New IR code hierarchy

```
Class Function:
    entry point

Class BasicBlock:
    body

Class Argument:
    value
```

Changes to code generation

Key Challenges:

1. Dominator tree → Dominator forest
2. Prologue insertion
   - Passing parameters to registers
   - Exposing all function entry points to outer scope
3. Harmonious integration with the rest of LLVM source code

Assumptions:

- C calling convention
- No special function attributes
- No special parameter attributes

Dominator forest1. Dominator tree
2. Prologue insertion
   - Passing parameters to registers
   - Exposing all function entry points to outer scope
3. Harmonious integration with the rest of LLVM source code

Expected x86 binary

Result

Expected x86 binary

References