EECS 583 – Homework 1

Fall 2019
Assigned: Wed, September 11, 2019
Due: Fri, September 20, 2019 (11:59:59pm)

The goal of this homework is to learn to write your first LLVM pass. As part of this, you will learn to use the profiler which provides dynamic execution frequencies for the compiler to make use of.

Statistics Computation Pass

Write a statistics computation pass in LLVM that computes several dynamic operation counts for each function. First, the total number of dynamic operations should be computed along with the percentages in the following categories: integer ALU, floating-point ALU, memory, biased-branch, unbiased branch, and all other operations. Use the following rules when categorizing the operations:

- Branch: br, switch, indirectbr
- Integer ALU: add, sub, mul, udiv, sdiv, urem, shl, lshr, ashr, and, or, xor, icmp, srem
- Floating-point ALU: fadd, fsub, fmul, fdiv, frem, fcmp
- Memory: alloca, load, store, getelementptr, fence, cmpxchg, atomicrmw
- Others: others

Every operation should be placed in one of the categories, so all are counted. Print this information out in a text file named benchmark.opcstats in the following format (comma separated, one function in each line): FuncName, DynOpCount, %IALU, %FALU, %MEM, %Biased-Branch, %Unbiased-Branch, %Others

For example, if a function has 50% operations being integer ALU operations, its %IALU should be 0.5. Print all zeros if a function has never been executed.

The branch bias is: (frequency_of_most_likely_target / execution_frequency). For branches with only a single target, the branch bias is 100%. For classification, branches are considered biased if the bias > 80% and unbiased otherwise.

To write a pass in LLVM, refer to http://www.llvm.org/docs/Projects.html. You should always keep your code separate from the core compiler code to ease integrating future releases. To get dynamic statistics, you will need to learn how to run the LLVM profiler.

Benchmarks to Run

There are 3 benchmark applications that you should profile and collect statistics: 583simple, 583wc, and 583compress. Each is progressively larger and more complex. To unpack a tgz file, run tar zxvf 583simple.tgz. Each benchmark contains directories for source code (1 .c file), sample input file, sample output file, and a text file with information on how to run the benchmark, exec_info_input1. To run the benchmark, run the SETUP command if it exists, then ARGS contains the command line arguments for the application, ie "a.out cccp.c" for 583wc. The CHECK command tells you how to check your output with the reference output. You can compile each benchmark with gcc to make sure they work before running LLVM.

Submission

To submit your homework, put a single .tgz (gzipped tar file) into the directory /hw1_submissions/ on eecs583a.eecs.umich.edu via scp (later versions will automatically overwrite the earlier versions if you submit multiple times):

```
$ tar cvzf ${uniquename}_hw1.tgz ${uniquename}_hw1
$ scp ${uniquename}_hw1.tgz ${uniquename}@eecs583a.eecs.umich.edu:/hw1_submissions/
```

Please name your tar file ${uniquename}_hw1.tgz, and organize the directory using the following structure:

```
${uniquename}_hw1/
  src/
    operation_statistics.cpp (source code for your pass)
  results/
    583simple.opcstats
    583wc.opcstats
    583compress.opcstats
```