EECS 583 – Homework 1

Fall 2011
Assigned: Mon, September 12, 2011
Due: Fri, September 23, 2011 (midnight)

Preliminaries: LLVM Installation

Setup and install the LLVM compiler system (http://www.llvm.org) under Linux (preferable) or MacOS (if you want to be adventurous). Please use **LLVM 2.9.** If you do not have a good machine available for your own use, we have 3 machines available for class [andrew, hugo, wilma].eecs.umich.edu. Login to one of these, create a directory for yourself on /y/students (everyone should have write permission), and install it there. Try to spread out across the 3 machines so 1 disk does not become the bottleneck. Note, you do not have to build llvm-gcc as that is already done for you. After you have installed LLVM, test your installation by compiling a simple test program and verifying that it runs correctly.

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, branch, and other. Every operation should be placed in one of those categories, so all are counted. Print this information out in a text file named benchmark.opcstats in the following format (tab separated):

FuncName DynOpCount %IALU %FALU %MEM %BRANCH %OTHER

Second, a branch bias histogram that specifies the number of dynamic branches that are taken/fallthrough should be computed. The branch bias is the MAX(taken_frequency, fallthrough _frequency) / execution_frequency. For multiway branches (from switch statements), the most frequent target is used to compute the bias. So, branch bias is the MAX(taken frequency across each target)/execution_frequency. The bias histogram should be broken down into groups of 10 percentage points: biases of 50-59%, 60-69%, 70-79%, 80-89%, 90-100%. Print this information out in a separate text file named benchmark.brstats in the following format (tab separated):

FuncName DynBrCount %50-59 %60-69 %70-79 %80-89 %90-100

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. Once that is run, modules can be invoked to load the profile data into the LLVM IR. Then, you can use the functions: getEdgeWeight(Edge e) and getExecutionCount(const BasicBlock *BB) found in the llvm/Analysis/ProfileInfo.h to access the profile data.

Memory Profiling Pass

In the previous part of the homework, an edge-profiler is used to get the execution counts of functions, basic blocks and edges. In this final part of the assignment, a memory profiler will be used to obtain load/store alias information in a program. A memory profiler gives information about the dependences that manifest through memory and memory profiler toolset which we will be using is called LAMP (Loop Aware Memory Profiling). The LAMP pass is not a part of standard LLVM infrastructure and LAMP itself, along with the details of using LAMP will be

provided on course website or class phorum. Lamp assigns a unique ID to each load/store instruction in the program. Print the following information for each load instruction in a separate text file named benchmark.ldstats in the following format (tab separated):
LoadID DependenceFraction

where LoadID is the id assigned to a load by LAMP and DependenceFraction is:
(Sum of alias counts of all aliased stores) / (load execution count).

Submission

You should submit a single .tgz (gzipped tar file) file into the directory /y/submit on andrew.eecs.umich.edu via scp. Please name your tar file uniquename_hw1.tgz. To scp from your machine, you do the following command: scp uniquename_hw1.tgz uniquename@andrew.eecs.umich.edu:/y/submit. Then type in your EECS password. Your tar file should contain (note the benchmarks will be provided on the 583 course website):

1. Source code for your LLVM pass
2. 3 text files containing the statistics (opcode, branch, and load) for all functions in the benchmark 583simple. These 3 files should be 583simple.opcstats, 583simple.brstats, and 583simple.ldstats.
3. 3 text files containing the statistics (opcode, branch, and load) for all functions in the benchmark 583wc.
4. 3 text files containing the statistics (opcode, branch, and load) for all functions in the benchmark 583compress.