FLORIDA INTERNATIONAL UNIVERSITY

EEL 5741 – Advanced Microprocessor Systems

Project 1 Benchmark Profiling And Processor Architecture

Date Assigned: 9/8/2016
Due Date: 9/28/2016 (11:59 pm)

Introduction

The goal of this project is to use simulator to profile the executions of several benchmark programs, to analyze the performance of different computer systems based on these benchmark programs, and to examine the roles of the compiler.

An overview of SimpleScalar

In this project, you need to use *SimpleScalar* simulation tools to profile the commercial benchmark programs as well as other C-source programs. The SimpleScalar tool set, originally developed at the University of Wisconsin-Madison, is a set of tools---including compiler, assembler, linker, simulation, and visualization tools---for the SimpleScalar architecture to simulate the programs on a variety of modern processor systems. Figure 1[1] gives an overview of the SimpleScalar tool set. More information on the Simplescalar tool set can be found at the *SimpleScalar Tools* web page *(*http://www.simplescalar.com*)*.

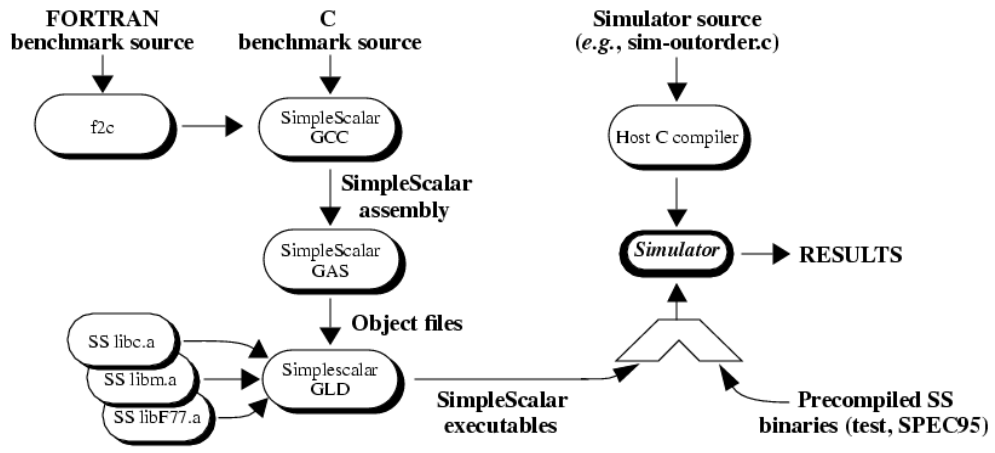

Figure 1. SimpleScalar tool set overview

Assume that the SimpleScalar tool packages are installed in the Linux machines under directory */usr/local/3rdparty/simplescalar/*. The following example shows how you can run one of the SimpleScalar tools, i.e., the *sim-profile*.

[1] Adopt from http://www.simplescalar.com/docs/users_guide_v2.pdf

Example: */usr/local/3rdparty/simplescalar/simplesim-3.0/sim-profile -iclass go.ss 2 8 go.in*
where

- *go.ss* is the SimpleScalar pre-compiled binary for the *go* benchmark.
- *2* and *8* are the two required input parameters for the *go* simulation.
- *go.in* is a file that specifies the starting board position (in this case an empty file.)

For more detailed information on how to use SimpleScalar simulation tools, you can refer to the Simplescalar Too Set Document on the web.

Project Details

1. Read the Simplescalar Tool Set Document and understand the Simplescalar architecture. Download and install Simplescalar 3.0 software package on your computer. Study the usage of SimpleScalar simulation commands ***sim-fast*** and ***sim-profile***.

2. For each of the five precompiled benchmarks, namely, ***go.ss, applu.ss, apsi.ss, gcc95.ss, compress95.ss***. The command lines to use these files are listed as follows :

   - */usr/local/3rdparty/simplescalar/simplesim-3.0/sim-profile -iclass go.ss 4 6 go.in*
   - */usr/local/simplescalar/simplesim-3.0/sim-profile –iclass applu.ss < applu.in*
   - */usr/local/simplescalar/simplesim-3.0/sim-profile –iclass apsi.ss*
   - */usr/local/simplescalar/simplesim-3.0/sim-profile –iclass gcc95.ss –O2 cc1.in –o test.s (test.s will be automatically generated)*
   - */usr/local/simplescalar/simplesim-3.0/sim-profile –iclass compress95.ss < compress95.in*

   Check the outputs. You need to save them for later use. To avoid using the full path all the times, you can set up the path and other environment variables by running the following command:
   source /usr/local/etc/simplescalar.csh
3. Assuming you have the following two machines with the same cycle time but different cycles-per-instruction (CPI) for different instruction groups as follows:

Instruction Types	Load/Store	Integer	Floating Point	Control Flow and others
System 1	2	1	6	2
System 2	1	2	3	3

Table-1: CPIs for different types of instructions

First, let system 1 be the reference machine. With the data you can collect from step 2, compare the performance of these two systems using weighted-sum, normalized arithmetic mean, and normalized geometric mean. Use system 2 as the reference machine and redo the comparison. Discuss your results.

4. Here is the C program for the matrix multiplication, i.e., matmul.c, that computes the production of two matrices A_{mxn} x B_{nxk}. It takes three input parameters, i.e., m, n, and k, randomly generates two matrices, and then computes their product. Compile it with SimpleScalar compiler (i.e., */usr/local/simplescalar/bin/sslittle-na-sstrix-gcc)* ***with no optimization and with level 2 optimization***. Here is an example:

   /usr/local/simplescalar/bin/sslittle-na-sstrix-gcc –O2 –o xgccedfile.ss Csoursefile.c

   The above command compiles C source code *Csoursefile.c* to the SimpleScalar binary file *xgccedfile.ss* with the level 2 optimization option. *Note that* the SimpleScalar compiler has the similar input format as that of gcc. (If you are not familiar with gcc, you should check the details using the UNIX/Linux *manual* command, i.e., *man gcc*.)
   - Let m=n=k=60, simulate the binary files and compare the total instruction counts.
   - What are the overall CPIs of the computer when running the optimized and non-optimized code? Discuss your results.

What/how to hand in

Your report should present your experimental results (with tables and/or figures) and show your understanding of the contents we discuss in the class by clearly interpreting these results and/or answering the questions. ***A summary section is mandatory*** that summaries the general observations, experiences, and conclusions you obtained from this project.

Reference

1. SimpleScalar LLC, http://www.simplescalar.com