FLORIDA INTERNATIONAL UNIVERSITY

EEL 5741 – Advanced Microprocessor Systems

Project 1: Benchmark Profiling and Processor Architecture

Introduction

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

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***. Please check the references at the end of this document.

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

```
a)  /usr/local/3rdparty/simplescalar/simplesim-3.0/sim-profile -iclass go.ss 4 6 go.in

b)  /usr/local/simplescalar/simplesim-3.0/sim-profile -iclass applu.ss < applu.in

c)  /usr/local/simplescalar/simplesim-3.0/sim-profile -iclass apsi.ss

d)  /usr/local/simplescalar/simplesim-3.0/sim-profile -iclass cc1.ss -O2 cc1.in -o test.s

e)  /usr/local/simplescalar/simplesim-3.0/sim-profile -iclass compress95.ss < compress95.in
```

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:

Table 1. CPIs for different types of instructions

Instruction Types	Load/Store	Integer	Floating Point	Control Flow and others
System 1	2	1	5	3
System 2	1	2	4	3

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

4. For this part use the C program for 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

$IDIR/bin/sslittle-na-sstrix-gcc -o hello hello.c
$IDIR/simplesim-3.0/sim-safe hello
```

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*.)

a) Let m=n=k=50, simulate the binary files and compare the total instruction counts.
b) What are the overall CPIs of the computer when running the optimized and non-optimized code? Discuss your results.

Optimization Level	**Parameters**	**CPI**
2	100-100-100	
None	100-100-100	
2	50-50-50	
None	50-50-50	

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
2. How-to install SimpleScalar on Ubuntu GitHub repository. Here you can find a script that installs simplescalar in Ubuntu 16.04 LTS. (https://github.com/sdenel/How-to-install-SimpleScalar-on-Ubuntu)
3. Simplescalar GitHub repository. Here you can find instructions and sources for SPEC95 bechmarks to work with simplescalar. (https://github.com/priyankarroychowdhury3/Simplescalar)