LSU EE 4720 Homework 1 Solution Due: 11 February 2005

Problem 1: POWER is an IBM ISA developed for engineering workstations, PowerPC is an ISA developed by IBM, Apple, and Motorola for personal computers and is based on POWER. POWER and PowerPC have instructions in common but each has instructions the other lacks (and some of the common instructions behave differently). Therefore a POWER implementation could not run every PowerPC program and vice versa.

(*a*) Show the gcc 3.4.3 compiler switches used to compile code for a POWER implementation. *Hint: Google is your friend, look for gcc documentation.*

Either of the following switches compiles for a generic POWER implementation -mpower, -mcpu=power. The compiler can also be told to target a particular implementation, for example, -mcpu=rios2.

(*b*) Show the gcc 3.4.3 compiler switches used to compile code for a PowerPC implementation.

Either of the following switches compiles for a generic PowerPC implementation -mpowerpc, -mcpu=powerpc. The compiler can also be told to target a particular implementation, for example, -mcpu=620

(*c*) Is it possible to use gcc 3.4.3 to compile a program that will run on both? If yes, show the switches.

Yes, one way is to specify two switches: -mno-power -mnopowerpc, the other uses the single switch -mcpu=common.

Problem 2: From the SPEC Web site, `http://www.spec.org`, find the fastest result on the SPECFP2000 (that's FP, not INT) benchmark for each of the following implementations: IBM POWER5, Intel Itanium2, Intel Pentium 4, Fujitsu SPARC64 v, and AMD FX-55. (Use the configurable search form and have it display the processor name.)

(*a*) The non-IA-32 implementations (POWER5, Itanium2, and SPARC64 V) blow away the IA-32 implementations on one benchmark. Which one? Which company (of those listed above) would want that benchmark removed?

The Art benchmark runs much faster on non-IA 32 systems: SPARC64 12.3. POWER5, 23.1; Itanium 2, 21.0 Pentium 4, 52.6, Opteron, 78. AMD would most want it removed.

(*b*) The POWER5 can decode five instructions per clock, the Itanium 2 can decode six instructions per clock, the Pentium 4 and FX-55 each can decode three (what are essentially) instructions per clock, and the SPARC64 V can decode four per clock. Based on the SPECFP2000 results used in the first part, which processor is making best use of these decode opportunities? In other words, if one processor could decode 10^{12} instructions during execution of the suite and another could decode 5×10^{12} instructions during execution of the suite, the first would be more efficient since it ran the suite using fewer instructions. (See last semester's Homework 1 for a similar problem.)

To solve this one needs to multiply the instructions-per-second potential of the processor by the run time for the suite. The instructions-per-second potential is the product of the decode rate given above (say, 5 per second for POWER5) and the clock frequency. The run times are given in the disclosure and can be added, but a less time consuming method would be to use the reciprocal of the score. So for the POWER5 the result would be $\frac{5\times 1900\,\mathrm{MHz}}{2796} = 3.40$, where 2796 is the SPECfp2000 result. For the Itanium 2, $\frac{6\times1600}{2712} = 3.54$; Athlon, $\frac{3\times2600}{2012} = 3.88$; Pentium 4, $\frac{3\times3733}{2016} = 5.56$; and SPARC64 V, $\frac{4\times1870}{1973} = 3.79$. The POWER5 is the "winner" here because it uses the fewest decode slots to execute the SPECfp2000 benchmarks. This can be because POWER5 programs have fewer instructions or because the POWER5 implementation wastes the fewest decode slots (or a combination of the two). In this case the POWER5 is both the most frugal and the fastest. Note that the Athlon and Pentium are almost tied in performance but that the Pentium uses alot more decode slots to attain that performance.

Problem 3: As pointed out in class a processor's CPI varies depending on the program being executed. For the questions below write a program in MIPS assembler (see `http://www.ece.lsu.edu/ee4720/mips32v2.pdf` for a list of instructions), some other assembly language, or assembly pseudocode, as requested below.

(*a*) Write a program that might be used to determine the minimum possible CPI. Suppose you actually used the program to determine the minimum CPI on processor X. How would the CPI be computed? Show an example using made up numbers based on your program an hypothetical processor X. Explain why the result would be the minimum CPI (or close to it).

Many processors do not attain their peak CPI because of program characteristics. Two causes are using instructions that take a long time and having nearby instructions depend on each other, forcing the processor to delay the start of the later instructions. When writing a program to determine peak cpi avoid long-executing instructions and close dependencies. The program below uses integer add instructions, which are fast and which lacks dependencies.

```
LOOP:
 add r1, r2, r3
 add r4, r5, r6
 add r7, r8, r9
 # ...
 j LOOP
 add r28, r29, r30
```

(*b*) Write a program that might be used to determine the maximum possible CPIand as with the previous part, show how CPI is computed. Your answer should include information about instructions in processor X used in your program. Explain why the result would be the maximum CPI (or close to it).

Include long-executing instructions that depend on each other.

```
LOOP:
 div.d f0, f2, f4
 div.d f0, f0, f4
 div.d f0, f0, f4
 div.d f0, f0, f4
 # ...
 j loop
 div.d f0, f0, f4
```