CPSC 3300 - 002

Homework 1

**Reagan Leonard**

1. (30pt) A processor P has a 4.2 GHz clock rate and has a CPI of 1.5.
2. If the processor executes a program in 8 seconds, find the number of cycles and the number of instructions.

Rate \* execution time = 4.2 \* 8 =

**Cycles = 3.36 \* 10^10**

Cycles / CPI = 3.36 / 1.5 =

**IC = 2.24 \* 10^10 instructions**

1. What is the MIPS rate for the processor?

Rate / CPI = 4.2 \*10^9 seconds / 1.5 \* 10^6 =

**2.8 \* 10^3 MIPS**

1. We are trying to reduce the execution time by 20% but this leads to an increase of 10% in the CPI. What clock rate should we have to get this time reduction?

0.8 \* 8 = new execution time: 6.4 sec

1.1 \* 1.5 = new CPI: 1.65

Proportion:

(New CPI/new clock rate) = (old CPI/old clock rate) 🡪

(1.1/X) = (0.8/4.2)

X = **new clock rate = 5.775 GHz**

1. (20pt) Consider two different implementation of the same instruction set architecture. The instructions can be divided into four classes according to their CPI (class A, B, C, and D). P1 has a clock rate of 2.5 GHz and CPIs of 1 (class A), 2 (class B), 3 (class C), and 5 (class D).

Given a program with a dynamic instruction count of 1.0E6 instructions divided into classes as follows: 15% class A, 50% class B, 25% class C, and 10% class D.

1. What is the global CPI?

Since we’re dealing with GHz and 1 million instructions, these will cancel out, so we can simply do (1\*0.15)+(2\*0.5)+(3\*0.25)+(5\*0.1)=

**2.4**

1. Find the clock cycles required to run the program on P1.

CPI \* instruction count=

**2.4 \* 10^6** (2.4 million)

1. (20pt) Assume for a given processor the CPI of arithmetic instructions is 1, the CPI of load/store instructions is 8, and the CPI of branch instructions is 3. Assume a program has the following instruction breakdowns: 600 million arithmetic instructions, 80 million load/store instructions, 120 million branch instructions.
2. Suppose we find a way to double the performance of the arithmetic instructions. What is the overall speedup of our machine? **We will have sped up the machine by 18.75%**

(600\*1)+(80\*8)+(120\*3)=1600 (600\***0.5**)+(80\*8)+(120\*3)=1300

1600-1300=300 million cycle decrease

300/1600=0.1875

1. If we find a way to double the performance of the load/store instructions, what is the overall speedup of our machine? **We will have sped up the machine by 20%**

(600\*1)+(80\*8)+(120\*3)=1600 (600\*1)+(80\***4**)+(120\*3)=1300

1600-1280=320 million cycle decrease

320/1600=0.2

1. (30pt) On machine newton using the **perf** tool, examine how compiler optimization levels and options change the number of instructions for the program **whetstone** and the number of CPU cycles to execute the program. Use gcc to compile your program. Refer to the following web page for information on how to use **perf** to count the number of instructions and cycles among other statistics: https://perf.wiki.kernel.org/index.php/Tutorial#Counting\_with\_perf\_stat

I: Download the **whetstone** benchmark to your home directory:

http://www.netlib.org/benchmark/whetstone.c

Compile whetstone. You may need to explicitly specify the

math lib folder and link to it, e.g.,

gcc -o whetstone whetstone.c -lm

#link the math with -lm

II: Examine the performance of **whetstone** looping 200,000 times

(**./whetstone 200000**) compiled with the following levels/options:

* 1. -O0
  2. -O1
  3. -O2
  4. -O3
  5. -O3 -funroll-loops

Use a table to show the instruction count, #cycles, IPC, and time for each of the experiments, and calculate the speedup based on the execution time with -O0. Paste your screen shot at the end.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | IC | #Cycles | IPC | Time (ms) | Speedup |
| -O0 | 22,785,365,480 | 17,185,519,509 | 1.33 | 6649.45 ms |  |
| -O1 | 10,959,024,831 | 8,861,773,824 | 1.24 | 3480.89 ms | 1.91 |
| -O2 | 6,054,235,563 | 6,145,505,999 | 0.99 | 2505.78 ms | 2.65 |
| -O3 | 6,024,215,645 | 6,109,872,998 | 0.99 | 2493.67 ms | 2.66 |
| -O3 -funroll-loops | 5,013,675,533 | 5,450,369,417 | 0.92 | 2246.38 ms | 2.96 |

