CPSC 3300

Homework 1

Due 11:59PM Thursday, January 28

Submit your answers to Canvas

Please provide sufficient space on your homework solutions so that your calculations and answers are easily readable and so that grading will be easier. Furthermore, except for the simplest questions, giving only the answer without showing your work is not acceptable. For the best chance at partial credit, show the generic equation you are starting with and any derivations needed to handle the information as given in the question, then plug in the values from the question. You may, of course, use a calculator for the homework.

1. (30pt) A processor P has a 2.7 GHz clock rate and has a CPI of 1.2.
2. If the processor executes a program in 6 seconds, find the number of cycles and the number of instructions.

**Clock Cycles = CPU Time x Clock Rate**

**= 6s x 2.7GHZ**

**= 16.2 x 10^9 cycles**

**To find instruction count (IC) use:**

**IC = (CPU Time x Clock Cycle Time) / CPI**

**IC = (6s x 2.7x10^9) / 1.2**

**IC = (1.62x10^10) / 1.2**

**IC = 1.35x10^10**

**or IC = 13,500,000,000**

1. What is the MIPS rate for the processor?

**MIPS = Clock rate / CPIx10^6**

**MIPS = 2.7x10^9 / 1.2x10^6**

**MIPS = 2250**

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

**Execution time = (instructions x CPI) / clock rate**

**Execution time(new) = 0.8 x Execution time(old)**

**Therefore**

**CPI(new)/clock rate(new) = 0.8 x CPI(old)/clock rate(old)**

**CPI(new) = 1.2, so**

**1.2/clock rate(new) = 0.8/clock rate(old)**

**Clock rate(new) = 1.2/0.8 x clock rate(old)**

**Clock rate(new) = 1.5 x clock rate(old)**

**Therefore the old clock rate must increase by 50%**

**Increasing 2.7GHZ by 50% yields 4.05GHZ**

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 25 million 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?

**A = 1.5x10^5, B = 5x10^5, C = 2.5x10^5, D = 10^5**

**Time(P1) = (1.5x10^5 + 5x10^5 x 2 + 2.5x10^5 x 3 + 10^5 x 4) / (2.5x10^9)**

**Time(P1) = 9.2x10^-4**

**CPI(P1) = (9.2x10^-4) x (2.5x10^9) / 10^6**

**CPI(P1) = 2.3**

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

**Clock Cycles utilize part of the equation above.**

**A = 1.5x10^5, B = 5x10^5, C = 2.5x10^5, D = 10^5**

**Clock Cycles(P1) = (1.5x10^5 + 5x10^5 x 2 + 2.5x10^5 x 3 + 10^5 x 4)**

**Clock Cycles = 2.3x10^6**

1. (20pt) Assume for a given processor the CPI of arithmetic instructions is 2, the CPI of load/store instructions is 6, and the CPI of branch instructions is 3. Assume a program has the following instruction breakdowns: 240 million arithmetic instructions, 70 million load/store instructions, 100 million branch instructions.
2. Suppose we find a way to double the performance of the arithmetic instructions. What is the speedup of our machine?

**The old clock cycles was:**

**(2 x 240) + (6 x 70) + (3 x 100) = 1200**

**To double arithmetic instructions our clock cycles becomes:**

**(1 x 240) + (6 x 70) + (3 x 100) = 960**

**Clock cycles(new)/clock cycles(old) = 960/1200 = 0.8**

**The speedup of the machine after doubling the performance of arithmetic instructions was 20%.**

1. If in addition to the arithmetic instruction optimization we also find a way to double the performance of the load/store instructions, what is the overall speedup of our machine?

**The old clock cycle (with optimization) was:**

**(1 x 240) + (6 x 70) + (3 x 100) = 960**

**To double the load/store instructions:**

**(1 x 240) + (3 x 70) + (3 x 100) = 750**

**Clock cycles(new)/clock cycles(old) = 750/960 = 0.78125**

**The speedup of the machine was a further increase of 21.875%.**

**The overall speedup was 20% + 21.875% = 41.875%.**

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 library folder and link to it, e.g.,

gcc -o whetstone whetstone.c -lm

#link the math library 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 | 22,785,254,374 | 16,920,054,510 | 1.35 | 6363 | 0 |
| -O1 | 10,957,309,469 | 8,819,209,982 | 1.24 | 3466 | 45.6% |
| -O2 | 6,056,123,284 | 6,141,991,202 | 0.99 | 2503 | 27.8% |
| -O3 | 6,021,948,416 | 6,127,308,031 | 0.98 | 2498 | 0.2% |
| -O3 -funroll-loops | 5,015,729,716 | 5,442,507,576 | 0.92 | 2245 | 10.1% |

Screenshots:

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated