CPSC 3300

Homework 2

Due before class 3:30 PM on Thursday, February 1

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. (Unlike the exams, the values in the homework questions are not necessarily chosen for ease of hand calculation.)

1. (30pt)A processor P has a 4.0 GHz clock rate and has a CPI of 2.2.
2. If the processor executes a program in 20 seconds, find the number of cycles and the number of instructions.

# of cycles = time \* clock rate

= 20 \* 4 \* 10^9

= 8 \* 10^10

# of instructions = # of cycles / CPI

= 8 \* 10^10 / 2.2

= 3.64 \* 10^10

1. What is the MIPS rate for the processor?

MIPS = # of instructions / 10^6 / time

= 1.82 \* 10^3

1. we are trying to reduce the execution time by 30% but this leads to an increase of 20% in the CPI. What clock rate should we have to get this time reduction?

Be careful only the # of instructions won’t change

Clock rate = # of cycles / time

= (# of instructions \* CPI) / time

= (3.64\*10^10 \* 2.2 \* (1+20%))/(1-30%)\*20

= 6.857 \* 10^9

=6.857 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, 2, 3, and 3.

Given a program with a dynamic instruction count of 1.0E6 instructions divided into classes as follows: 20% class A, 10% class B, 50% class C, and 20% class D.

1. What is the global CPI?

global CPI = 1 \* 20% + 2\* 10% + 3\* 50% + 3\* 20%

= 2.5

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

# of cycles = # of instructions \* CPI

= 1 \* 10^6 \* 2.5

= 2.5 \* 10^6

1. (20pt)Assume for a given processor the CPI of arithmetic instructions is 1, the CPI of load/store instructions is 10, and the CPI of branch instructions is 3. Assume a program has the following instruction breakdowns: 600 million arithmetic instructions, 250 million load/store instructions, 150 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?

old # of cycles = 1\*600 + 10 \* 250 + 3\*150 million

= 3550 million

After double the performance of the arithmetic instructions:

# of cycles = 0.5\*600 + 10 \* 250 + 3\*150 million

= 3250 million

The speedup = = 3550/ 3250 = 1.0923

1. If we find a way to improve the performance of the arithmetic instructions by 10 times, what is the overall speedup of our machine?

old # of cycles = 1\*600 + 10 \* 250 + 3\*150 million

= 3550 million

After improving the performance of the arithmetic instructions:

# of cycles = 0.1\*600 + 10 \* 250 + 3\*150 million

= 3010 million

The speedup = = 3550/ 3010 = 1.179

1. (30pt)On machine newton, 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.

Examine 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.

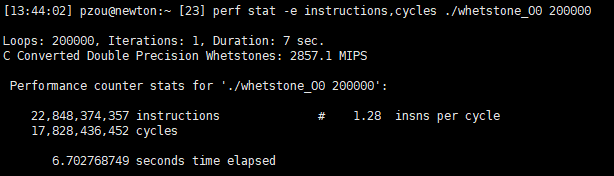
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | IC | #Cycles | IPC | Time | Speedup |
| -O0 | 22,848,374,357 | 17,828,436,452 | 1.28 | 6.702768749s | 1 |
| -O1 | 11,006,291,594 | 9,268,808,696 | 1.19 | 3.333630423s | 2.01 |
| -O2 | 7,241,820,739 | 6,924,628,485 | 1.05 | 2.765889000s | 2.42 |
| -O3 | 7,242,441,442 | 6,909,480,426 | 1.05 | 2.481142099s | 2.70 |
| -O3 -funroll-loops | 6,257,427,496 | 6,062,058,340 | 1.03 | 2.316589712s | 2.89 |

If using percent increase for speedup instead of factor (no reduction in score as long as consistent):

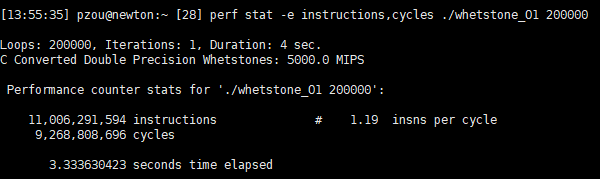
-O0: 0% -O1: ~100% -O2: ~140% -O3: ~170% -O3 -funroll-loops: ~190%

Formula: ((O0\_time / Time) - 1) \* 100

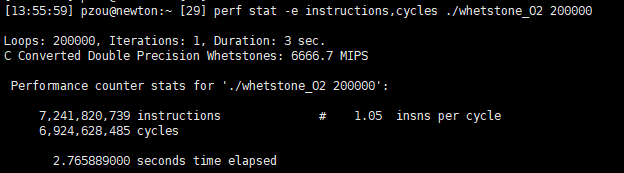
O0:



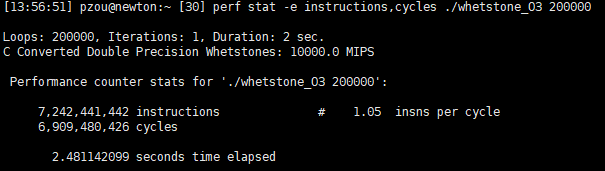
O1:



O2:



O3:



O3 funroll-loops:

