CS320 Fall 2014

Homework#7
Quiz in Lecture Thurs Nov 20, 2014
There are no make-up quizzes!

Problem 1: Consider two different implementations, M1 and M2, of the same instruction set. There are three classes of instructions (A, B, and C) in the instruction set. M1 has a clock rate of 1.2 GHz and M2 has a clock rate of 3 GHz. The average number of cycles for each instruction class and their frequencies (for a typical program) are as follows:

Instruction Class	M1	M2	Frequency
A	1	2	50%
B	2	4	30%
C	5	4	20%

(a) Calculate the average CPI for each machine, M1, and M2.
ANSWER: For Machine M1: Clocks per Instruction = (50/100)* 1 + (30/100)*2 + (20/100)*5 = 2.1
For Machine M2: Clocks per Instruction = (50/100)*2 + (30/100)*4 + (20/100)*4 = 3

(b) Calculate the average MIPS ratings for each machine, M1 and M2.
ANSWER: For Machine M1: Average MIPS rating = Clock Rate/(CPI * 10^6) = (1.2 * 10^9) / (2.1*10^6) = 570
For Machine M2: Average MIPS rating = Clock Rate/(CPI * 10^6) = (3 * 10^9) / (3*10^6) = 1000

(c) Which machine has a smaller MIPS rating? Which individual instruction class CPI do you need to change, and by how much, to have this machine have the same or better performance as the machine with the higher MIPS rating (you can only change the CPI for one of the instruction classes on the slower machine)?

ANSWER: Machine M2 has higher performance because its MIPS rating is higher Machine M1 has a smaller MIPS rating.

The CPI of instruction class can not be improved past 1 clock cycle.
If we change the CPI of instruction class B for Machine M1 to 1:
Clocks per Instruction = (50/100)*1 + (30/100)*1 + (20/100)*5 = 1.8
Average MIPS rating = Clock Rate/(CPI * 10^6) = (1.2 * 10^9) / (1.8*10^6) = 666.6666

If we change the CPI of instruction class C for Machine M1 to 1:
Clocks per Instruction = (50/100)*1 + (30/100)*2 + (20/100)*1 = 1.3
Average MIPS rating = Clock Rate/(CPI * 10^6) = (1.2 * 10^9) / (1.3*10^6) = 923.0769

You can not make M1 outperform M2 by changing only a singe instruction class. The clock rate of M2 is more than 2x faster than M1 and a single instruction change will not cover the disparity.

Problem 2: Suppose we have two implementations (M1 and M2) of the same instruction set architecture. Machine 1 has a clock rate of 500MHz and takes an average of 2.0 clock cycles per instruction (CPI) for a program. Machine 2 has a clock rate of 400MHz and a CPI of 1.2 for a program.
(a) Which machine is faster for this program, and by how much?
ANSWER: Using I to represent the number of instructions in the program:

Execution Time$_{M1}$ = I x 2.0 x 1/(500x10^6) = I/(250 x 10^6) = 4I x 10^{-9} seconds
Execution Time$_{M2}$ = I x 1.2 x 1/(400x10^6) = 0.3I/(100x10^6) = 3I x 10^{-9} seconds

Since both M1 and M2 are running the same instructions (same program, same instruction set), we can use I in both equations. Comparing the execution times for M1 and M2, I divides out, and we see that the Execution TimeM1 / Execution TimeM2 = 4/3. From this we can conclude that the program runs 33% slower on M1.
(b) If a test program executes in 10 million instructions (on both machines), how long will it take to run the program on Machine 1? Machine 2?
ANSWER: Same formulas from (a), but now we know I = 10^7:

Execution Time$_{M1}$ = I x 2.0 x 1/(500x10^6) = 10^7/(250 x 10^6) = 1/25 = 0.04 seconds
Execution Time$_{M2}$ = I x 1.2 x 1/(400x10^6) = 0.3x10^7/(100x10^6) = 3/100 = 0.03 seconds

(c) Suppose that the 10 million instructions implemented on M1 fall into two performance classes, the first (Class A) takes one clock cycle to execute, while the second (Class B) takes 5 clock cycles to execute. How many Class B instructions are executed when our test program is run?

ANSWER:
Let A be the number of Class A instructions, and B the number of Class B instructions. So, A + B = 10^7
The execution time for M1 (calculated in clock cycles) is: (1A + 5B)(clock cycle time)
The CPI for the entire program is 2.0. That means that, on average, every instruction executed takes 2 clock cycles to run. In terms of CPI, the total execution time is therefore:
CPI x (clock cycle time) (number of instructions) = 2.0 I (clock cycle time)
Thus, A + 5B = 2.0I = 2 x 10^7
(A + 5B) - (A + B) = 4B
4B = (2 x 10^7) - 10^7 = 1 x 10^7

Therefore, 4B = 1 x 10^7
And B = 2.5 x 10^6 = 2.5 million instructions.
Problem 3: (Amdahl's law question) Suppose you have a machine which executes a program consisting of 50% floating point multiply, 20% floating point divide, and the remaining 30% are from other instructions.

(a) Management wants the machine to run 4 times faster. You can make the divide run at most 3 times faster and the multiply run at most 8 times faster. Can you meet management's goal by making only one improvement, and which one?

ANSWER: Amdahl's Law states:
Execution time after improvement = (Execution time affected by improvement/Amount of Improvement) + Execution time unaffected
Speed up = 1/Execution time after improvement
Assuming that the floating point multiply, floating point divide and the other instructions had the same CPI. This is the best case scenario.

Execution time after Improvement with Divide = (.20/3) + (.50 + .30) = 0.8667
Speedup = 1/.5625 = 1.77
Execution time after Improvement with Multiply = (.50/8) + (.20 + .30) = 0.5625
Speedup = 1/.8667 = 1.15

The management's goal can not be met by making the improvement with Multiply or divide alone (Speedup is not 4 times).
(b) Dogbert has now taken over the company removing all the previous managers. If you make both the multiply and divide improvements, what is the speed of the improved machine relative to the original machine?

ANSWER: If we make both the improvements, this means that 50% floating point multiply is improved by 8 times and the 20% floating point divide is improved by 3 times, and the remaining 30% for other instructions is unchanged.

Execution time after Improvement = (Execution time affected by improvement of divide/Amount of Improvement for divide) + (Execution time affected by improvement multiply/Amount of Improvement for multiply) + Unchanged portion

Execution time after Improvement = (.50/8) + (.20/3) + (.30) = .4291
The speedup relative to the original machine = 1/.4291 = 2.33

Problem 4: Suppose that we can improve the floating point instruction performance of machine by a factor of 15 (the same floating point instructions run 15 times faster on this new machine). What percent of the instructions must be floating point to achieve a Speedup of at least 8?

ANSWER: We will use Amdahl's Law again for this question. Let x be percentage of floating point instructions. Since the speedup is 8, if the original program executed in 100 cycles, the new program runs in 100/8 = 12.5 cycles.
(100)/8 = (x)/15 + (100 - x)
Solving for x, we get: x = 93.75. The percent of floating point instructions need to be 93.75.

Problem 5: Multi-cycle performance
Two important parameters control the performance of a processor: cycle time and cycles per instruction. There is an enduring trade-off between these two parameters in the design process of microprocessors. While some designers prefer to increase the processor frequency at the expense of large CPI, other designers follow a different school of thought in which reducing the CPI comes at the expense of lower processor frequency. Consider the following machines, and compare their performance using the following instruction mix: 25% loads, 13% stores, 47% ALU instructions, and 15% branches/jumps. Assume the unmodified multi-cycle datapath and finite state machine.

M1: The multicycle datapath is designed with a 1 GHz clock

M2: A machine like M1 except that register updates are done in the same clock cycle as a memory read of ALU operation. Thus in the finite state machine, states 6 and 7 and states 3 and 4 are combined. This machine has an 3.2 GHz clock, since the register update increases the length of the critical path.

M3: A machine like M2 except that effective address calculations are done in the same clock cycle as a memory access. Thus states 2, 3, and 4 can be combined, as can 2 and 5, as well as 6 and 7. This machine has a 2.8 GHz clock because of the long cycle created by combining address calculation and memory access.

(a) Which of the machines has the shortest cycle time?

(b) Are there instruction mixes that would make another machine have a shorter cycle time, and if so, what are they?

In the original multi-cycle data path the CPI for each instruction is as follows:
Loads: 5 cycles
Stores: 4 cycles
ALU: 4 cycles
Branch/Jumps: 3 cycles

Performance M1:
Average CPI = .25*5 + .13*4 + .47*4 + .15*3 = 4.1
Cycle Time = (CPI*num_instructions)/ clock rate = 4.1I/1GHz = 4.1I*10-9 seconds

Performance M2:
Loads shorten to 4 cycles
ALUs shorten to 3 cycles
Average CPI = .25*4 + .13*4 + .47*3 + .15*3 = 3.38
Cycle Time = (CPI*num_instructions)/ clock rate = 3.38I/3.2GHz = 1.06I*10-9 seconds

Performance M3:
Loads shorten to 3 cycles
Stores shorten to 3 cycles
ALUs shorten to 3 cycles
Average CPI = .25*3 + .13*3 + .47*3 + .15*3 = 3
Cycle Time = (CPI*num_instructions)/ clock rate = 3I/2.8GHz = 1.07I*10-9 seconds

M2 is fastest.
M1 can never be faster than M2, even if all the instructions are branch instructions, the CPI will be 3 for all 3 cases, and the clock rate is faster on the other 2 processors.
M3 can be faster than M2, if all instruction loads or all stores then

Ex: M2: Average CPI = 1*4 + 0*4 + 0*3 + 0*3 = 4
M3: Average CPI = 1*3 + 0*3 + 0*3 + 0*3 = 3
M2 Cycle Time = (CPI*num_instructions)/ clock rate = 4I/3.2GHz = 1.25I*10-9 seconds
M3 Cycle Time = (CPI*num_instructions)/ clock rate = 3I/3.2GHz = 1.07I*10-9 seconds

Problem 6: A pipelined processor has a clock rate of 5.5 GHz and executes a program with 5 million instructions. The pipeline has 5 stages, and instructions are issued at a rate of one per clock cycle. Ignore penalties due to branch instructions, and filling and emptying the pipelines.
(a) What is the speedup of the processor for this program compared to a non-pipelined processor?
(b) what is the throughput (in MIPS) of the pipelined processor?

ANSWER: (a) Since we are ignoring penalties due to branch instructions and filling and emptying the pipelines, we can assume that $CPI_{pipelined} = 1$. Under such perfect assumptions, the speedup of a pipelined processor compared to a non-pipelined processor is equal to the number of pipeline stages. In this case, the number of pipeline stages is 5. So, the speedup is 5.
Assume the non-pipelined processor is a multi-cycle processor where each instruction takes 5 clock cycles. So,$CPI_{non-pipelined} = 5$. Now, we have

$$speedup = \frac{time_{non-pipelined}}{time_{pipelined}} = \frac{\frac{CPI_{non-pipelined} \times 1}{clockrate}}{\frac{CPI_{pipelined} \times 1}{clockrate}} = \frac{CPI_{non-pipelined}}{CPI_{pipelined}} = \frac{5}{1} = 5 \quad (1)$$

So, the speedup is 5.

(b) Since we are ignoring penalties due to branch instructions and filling and emptying the pipelines, we can assume that $CPI_{pipelined} = 1$.

$$MIPS_{pipelined} = \frac{InstructionCount}{ExecutionTime \times 10^6} = \frac{I}{\frac{CPI_{pipelined} \times I}{5.5GHz} \times 10^6} = \frac{5.5GHz}{CPI_{pipelined} \times 10^6} = \frac{5.5 \times 10^9}{1 \times 10^6} = 5.5 \times 10^3 \quad (2)$$

The throughput of the pipelined processor is 5.5×10^3 MIPS.

Problem 7: Computer A has an overall CPI of 1.3 and can be run at a clock rate of 3 GHz. Computer B has a CPI of 2.5 and can be run at a clock rate of 2.2 Ghz. We have a particular program we wish to run. When compiled for computer A, this program has exactly 100,000 instructions. How many instructions would the program need to have when compiled for Computer B, in order for the two computers to have exactly the same execution time for this program?

ANSWER: $(\text{CPUTime})_A = (\text{Instruction count})_A * (\text{CPI})_A * (\text{Clock cycle Time})_A = (100{,}000)*(1.3)/(3*10^9) = 4.33 \ \ 10^{-5}$ sec

$(\text{CPUTime})_B = (\text{Instruction count})_B * (\text{CPI})_B * (\text{Clock cycle Time})_B = (\text{I})_B*(2.5)/(2.2*10^9) = 1.136 \ 10^{-9}$ sec

Since $(\text{CPUTime})_A = (\text{CPUTime})_B$, we have to solve for $(\text{I})_B$ and get 38,116.1972

Problem 8: (a) Multiple 8-bit numbers 20010 * 3510 using the multiplier in Figure 3.3 (Page 184 of your book). What is the value in the Product, Multiplicand and Multiplier after 5 clock cycles? How many total times did you use the ALU to Add?

(b) Repeat part (a) using the multiplier in Figure 3.5 (Page 186 of your book). What is the value in the Product register after 4 clock cycles?

ANSWER: See next page

Problem 9: Dividing integers: Consider 4-bit binary unsigned division. Divide 6 by 4 using the hardware in Figure 3.8 (Page 190). When calculating the result, how many times did you reset the Remainder back to its previous value.

ANSWER: See next page