CptS260: Introduction to Computer Architecture

Homework 1 Solution

School of Electrical Engineering and Computer Science
Spring 2022

Part 1: Basic (10 pts)

You can find all details regarding this part, in your textbook and lecture notes.

Part 2: Performance Evaluation

1. Three different processors P1, P2, P3 executing the same set of instruction. P1 has a 3 GHz clock rate and a CPI of 1.5. P2 has 2 GHz clock rate and CPI of 1.2. P3 has 4 GHz and CPI of 2.4.

   a. [5 pt] Based on the following calculations, P1 has the highest performance, and the performance of P2 and P3 are the same in terms of the number of instructions per second.

$$\text{Performance of P1 (instructions/sec)} = 3 \times 10^9/1.5 = 2.00 \times 10^9$$
$$\text{performance of P2 (instructions/sec)} = 2 \times 10^9/1.2 = 1.67 \times 10^9$$
$$\text{performance of P3 (instructions/sec)} = 4 \times 10^9/2.4 = 1.67 \times 10^9$$

   b. [5pts] No. of Instructions and Cycles for each processor:

$$\text{No. Cycles (P1)} = 5 \times 3 \times 10^9 = 15 \times 10^9$$

$$\text{No. Cycles (P2)} = 5 \times 2 \times 10^9 = 10 \times 10^9$$

$$\text{No. Cycles (P3)} = 5 \times 4 \times 10^9 = 20 \times 10^9$$

$$\text{No. Instructions (P1)} = \frac{No.\ Cycles(P1)}{CPI(P1)} = \frac{(15 \times 10^9)}{1.5} = 10 \times 10^9$$

$$\text{No. Instructions (P2)} = \frac{(10 \times 10^9)}{1.2} = 8.34 \times 10^9$$

$$\text{No. Instructions (P2)} = \frac{(20 \times 10^9)}{2.4} = 8.34 \times 10^9$$

c. [5pts] Increase of 25% in the CPI ➔ $CPI_{new} = CPI_{old} \times 1.25,$ therefore:

$$CPI(P1) = 1.875, \quad CPI(P2) = 1.5, \quad CPI(P3) = 3$$

$$Clock\ rate = \frac{No.Instructions \times CPI}{CPU\ time}$$

20 % reduction, means that goal time is 80%, therefore:

$$CPU\ time_{new} = 0.8 \times CPU\ time_{old}$$

$$Clock\ rate_{\ new} = \frac{No.Instructions \times CPI_{new}}{CPU\ time_{new}}$$

$$\rightarrow Clock\ rate_{\ new} = \frac{No.Instructions \times CPI_{new}}{0.8 \times CPU\ time_{old}}$$

$$\rightarrow Clock\ rate_{\ new} = \frac{No.Instructions \times CPI_{new}}{0.8 \times (\frac{No.Instructions \times CPI_{old}}{Clock\ rate_{\ old}})}$$

$$\rightarrow Clock\ rate_{\ new} = \frac{Clock\ rate_{\ old} \times CPI_{new}}{0.8 \times (CPI_{old})}$$

$$Clock\ rate_{\ new}\ (P1) = \frac{3 \times 10^9 \times 1.875}{0.8 \times 1.5} = 4.68 \times 10^9 = 4.68\ GHz$$

You can calculate the new clock rate for P2 and P3 using the same method.

[5 pts for each part] Consider two processors with for types of instructions as listed in Table 1. Given a program with a dynamic instruction count of 10^6 instructions divided into classes as follows 10% class A, 20% class B, 50% class C, and 20% class D.

a. Firstly, we should calculate the time of program execution using each one of the processors

$$Time = \frac{\sum_{i=1}^{class\ of\ instructions} No.\ Instruction_i \times CPI_i}{clockrate}$$

$$Time(p1) = \frac{(0.1 \times 1 + 0.2 \times 2 + 0.5 \times 3 + 0.2 \times 3) \times 10^6}{2.5 \times 10^9} = 1.04 \times 10^{-3}\ s$$

$$Time(p2) = \frac{(0.1 \times 2 + 0.2 \times 4 + 0.5 \times 2 + 0.2 \times 3) \times 10^6}{3 \times 10^9} = 0.87 \times 10^{-3}$$

Therefore, P2 is faster.

b. Average CPI for each processor is the average of CPIs for all the instruction classes, therefore

$$CPI_{avg} = \sum_{i=1}^{class\ of\ instructions} Percentage_i \times CPI_i$$

$$CPI_{avg}(P1) = (0.1 \times 1 + 0.2 \times 2 + 0.5 \times 3 + 0.2 \times 3) = 2.6$$

$$CPI_{avg}(P2) = (0.1 \times 2 + 0.2 \times 4 + 0.5 \times 2 + 0.2 \times 3) = 2.6$$

Which is the same for both processors.

c. Total number of cycles is the product of CPI_{avg} and number of instructions. Therefore:

$$CPI_{avg} \times Number\ of\ Instructions$$
$$Cycles(P1) = 2.6 \times 10^6$$
$$Cycles(P2) = 2.6 \times 10^6$$

Part 3: Number Representations

1. (10 pts) In digital computers, for signed number we use the 2's complement representation, which the most significant bit acts as the sign of number, and it has the greatest weight.

For unsigned number, the smallest number is 0, and the biggest number is 2^n-1, which n the number of bits. On the other hand, for signed number the range is from -2^{n-1} to 2^{n-1}-1.

Consider these two 8 bit binary number X = 10010010, Y = 01101100, and try to convert them into decimal numbers for both case:

If we consider unsigned numbers: X = $2^7 + 2^4 + 2^1 = 146$, and Y = $2^6 + 2^5 + 2^3 + 2^2 = 108$

If we consider signed numbers: X = $-2^7 + 2^4 + 2^1 = -110$, and Y = $2^6 + 2^5 + 2^3 + 2^2 = 108$

As you can see, a number with the most significant bit of 0, gives us the same number for both signed and unsigned numbers.

2. (10 pts) Convert $(1249)_{10}$ to binary, hexadecimal, and octal. What is the minimum number of bits that you need? (You can show decimal numbers unsigned: 1249d = $(1249)_{10}$), and h for hexadecimal numbers, and b for binary numbers). Extend the sign of this number, and convert it to a 32 bits digit.

$$(1249)_{10} = 4E1_H = 10011100001_b = 2341_o$$

You need at least 11 bits. But if you use 2's complement representation, you need 12 bits at least: 010011100001_b ➔ because it is positive number.

3. Show $(-1249)_{10}$ as a 2's complement signed number
   a. (5 pts) From $(1249)_{10}$

$$(1249)_{10} = 010011100001_b$$

$$\rightarrow 1's\ complement: \ 101100011110$$

$$1's + 1 \ \rightarrow 2's\ complement: \ 101100011111 = \ (-1249)_{10}$$

   b. (5 pts) Using the $x = -x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \cdots + x_1 2^1 + x_0 2^0$ formula

$$-2^{11} + 2^9 + 2^8 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = (-1249)_{10} = 101100011111$$

4. (5 pts) For sign-magnitude format: sign is negative, so the sign bit is 1. The rest would be the binary code for 1249.

5. There is no overflow for part a, b, or c.
   a. (5 pts)

   -10 = 11111111111111111111111111110110
   +12 = 00000000000000000000000000001100
   Result = 1 00000000000000000000000000000010 : There is no overflow, we should ignore 32nd bit!
   b. (5 pts)
   -20 = 11111111111111111111111111101100
   +19 = 00000000000000000000000000010011
   Result = 11111111111111111111111111111111

   c. (5 pts)

   N1 = 00010000….000
   + N2 = 00010000….000

   Result = 0010000...00 = $20000000_h = +536870912_d$