COMPUTER ARCHITECTURE & ASSEMBLY LANGUAGE
14:332:331
Rutgers University
Maria Striki Spring 2021

Homework 1: 53 pts = 6 + 9 + 9 + 6 + 8 + 3 + 12 pts

Issued on: Mon 02-01-2021 Due on: Thu Feb. 02-11-21 at 1.00 pm

Your names and contributions of group members: Ashwin Anand , John F. Crespo , Jacques Tege (We all contributed to each problem)

Problem 1: (6 pts, 3 pts each sub-part)
Consider two different implementations of the same instruction set architecture. The instruction can be divided into four classes according to their CPI (class A, B, C and D). P1 with a clock rate of 2.5 GHz and CPIs of 1, 4, 3 and 2, and P2 with a clock 3 GHz and CPIs of 2, 3, 2 and 3. Given a program with a dynamic instruction count of 1.0E6 instructions divided into classes as follows: 20% class A, 30% class B, 20% class C and 30% class D, which implementation is faster?

a) What is the global CPI for each implementation?
b) Find the clock cycle required in both cases.

Solution:

1) P_1 with clock rate of 2.5GHz and CPIs of 1, 4, 3 and 2.

P_2 with a clock rate of 3.0GHz and CPIs of 2, 3, 2 and 2.

Intruction count 1.0E6 divided into 4 classes:

A ⇒ 20%

B ⇒ 30% Which implementation is faster?

C ⇒ 20%

D ⇒ 30%

$$\text{CPUtime} = \frac{\text{CPU clock cycle}}{\text{Clock rate}} \qquad \text{Clock cycle} = \sum CPI_i \cdot C_i$$

$$\text{CPUtime}_{P_1} = \frac{(2\cdot1\cdot10^5)+(3\cdot4\cdot10^5)+(4\cdot2\cdot10^5)+(2\cdot3\cdot10^5)}{2.5\text{ GHz}} = \frac{2.6\times10^6}{2.5\text{GHz}}$$

CPUtime P_1 = 1.015 ms

$$\text{CPU time}_{P_2} = \frac{(2\cdot2\cdot10^5)+(3\cdot3\cdot10^5)+(2\cdot2\cdot10^5)+(2\cdot3\cdot10^5)}{3\text{ GHz}} = \frac{2.3\times10^6}{3.0\text{ GHz}}$$

CPUtime P_2 = 766.67 ms

Processor 2 is faster.

(a) What is the global CPI for each implementation.

$$CPI_{P1} = \frac{\text{CPU clock cycles } P_1}{\text{number of instruction}} = \frac{2.6\cdot10^6}{10^6} = 2.6$$

$$CPI_{P2} = \frac{\text{CPU clock cycles } P_2}{\text{number of instructions}} = \frac{2.3\times10^6}{10^6} = 2.3$$

(b) find the clock cycles required in both cases.

$$\text{Clock cycles}_{P_1} = \#\text{ instructions} \cdot \sum CPI$$

$$10^6\left((1\cdot0.2)+(4\cdot0.3)+(3\cdot0.2)+(2\cdot0.3)\right) = 2.6\times10^6$$ clock cycles

$$\text{Clock cycles}_{P2} = \#\text{ instructions} \cdot \sum CPI$$

$$10^6\left((2\cdot0.2)+(3\cdot0.3)+(2\cdot0.2)+(2\cdot0.3)\right)$$

$$= 2.3\times10^6$$ clock cycles

Problem 2: (9 pts, 3pts each sub-part)
Assume a program requires the execution of 80 x 10^6 FP instructions, 120 x 10^6 INT instructions, 60 x 10^6 Load/Store (L/S) instructions and 25 x 10^6 branch instructions. The CPI for each type of instruction is 2, 1, 4 and 2, respectively. Assume that the processor has a 4 GHz clock rate.

a) By how much must we improve the CPI of FP instructions if we want the program to run two times faster?
b) By how much must we improve the CPI of L/S instruction if we want the program to run two times faster?
c) By how much is the execution time of the program improved if the CPI of INT and FP instructions are reduced by 30% and the CPI of L/S and Branch is reduced by 20%?.

Clarification: you are asked to improve the CPI of one category of commands every time with the goal to make the WHOLE program run two times faster. You may find that one or more of these cases is impossible to implement, or you may be able to identify improvement for all the above cases.

Solution:

2) 80×10^6 FP executions, 120×10^6 INT instructions, 60×10^6 Load/Store Instructions, 25×10^6 branch instructions, CPI is 2, 1, 4 and 2 respectively.
clock rate = 4 GHz

$$\text{Execution time} = \sum_{i=1}^{4} \frac{\text{number of instructions}_i \times CPI_i}{\text{clock rate}}$$

$$\text{Execution time} = \frac{(80 \cdot 10^6 \cdot 2) + (120 \cdot 10^6 \cdot 1) + (60 \cdot 10^6 \cdot 4) + (25 \cdot 10^6 \cdot 2)}{4 \cdot 10^9 \text{ Hz}}$$

Execution time = 0.1425 s

By how much must we improve the CPI of FP instructions if we want the program to run two times faster?

$$\frac{0.1425}{2} = \frac{(80)(10^6)(FP_{CPI}) + (120 \cdot 10^6) + (60 \cdot 10^6 \cdot 4) + (25 \cdot 10^6 \cdot 2)}{4 \cdot 10^9}$$

$$\frac{4 \cdot 10^9 \cdot 0.1425}{2} = (80)(10)^6 \cdot FP_{CPI} + 4.1 \cdot 10^8$$

$$\frac{285000000 - 410000000}{80 \cdot 10^6} = \frac{80 \cdot 10^6 \, FP_{CPI}}{80 \cdot 10^6}$$

$$FP_{CPI} = -1.5625$$

We cannot improve the CPI instruction as it would be negative

By how much must we improve the CPI of L/S instruction if we want the program to rum two times faste?

$$\frac{0.1425}{2} = \frac{(80 \cdot 10^6 \cdot 2) + (120 \cdot 10^6 \cdot 1) + (60 \cdot 10^6) L/S_{CPI} + (25 \cdot 10^6 \cdot 2)}{4 \cdot 10^9 \text{ Hz}}$$

$$\frac{4 \cdot 10^9 \cdot 0.1425}{2} = 60 \cdot 10^6 L/S_{CPI} + 330000000$$

$$\frac{285000000 - 330000000}{60 \cdot 10^6} = L/S_{CPI}$$

$$L/S_{CPI} = -0.75$$

We cannot improve the CPI instruction as it would be negative.

c) By how much is the execution time of the program improved if the CPI of Int and FP instruction's are reduced by 30% and the CPI of L/S and Branch is reduced by 20%.

new $CPI_{FP} = 0.7 \cdot 2 = 1.4$
new $CPI_{INT} = 0.7 \cdot 1 = 0.7$
new $CPI_{L/S} = 0.6 \cdot 4 = 2.4$
new $CPI_{Branch} = 0.6 \cdot 2 = 1.2$

$$\text{Execution time} = \sum_{i=1}^{4} \frac{\text{number of instruc} \times \text{new } CPI_i}{\text{clock rate}}$$

$$\frac{(80 \cdot 10^6 \cdot 1.4) + (120 \cdot 10^6 \cdot 0.7) + (60 \cdot 10^6 \cdot 2.4) + (25 \cdot 10^6 \cdot 1.2)}{4 \cdot 10^9}$$

$= 0.0925 \qquad \frac{0.0925}{0.1425} = 0.65$

this is a speed-up of 35.0%

Problem 3: (9 pts, 3pts each sub-part)

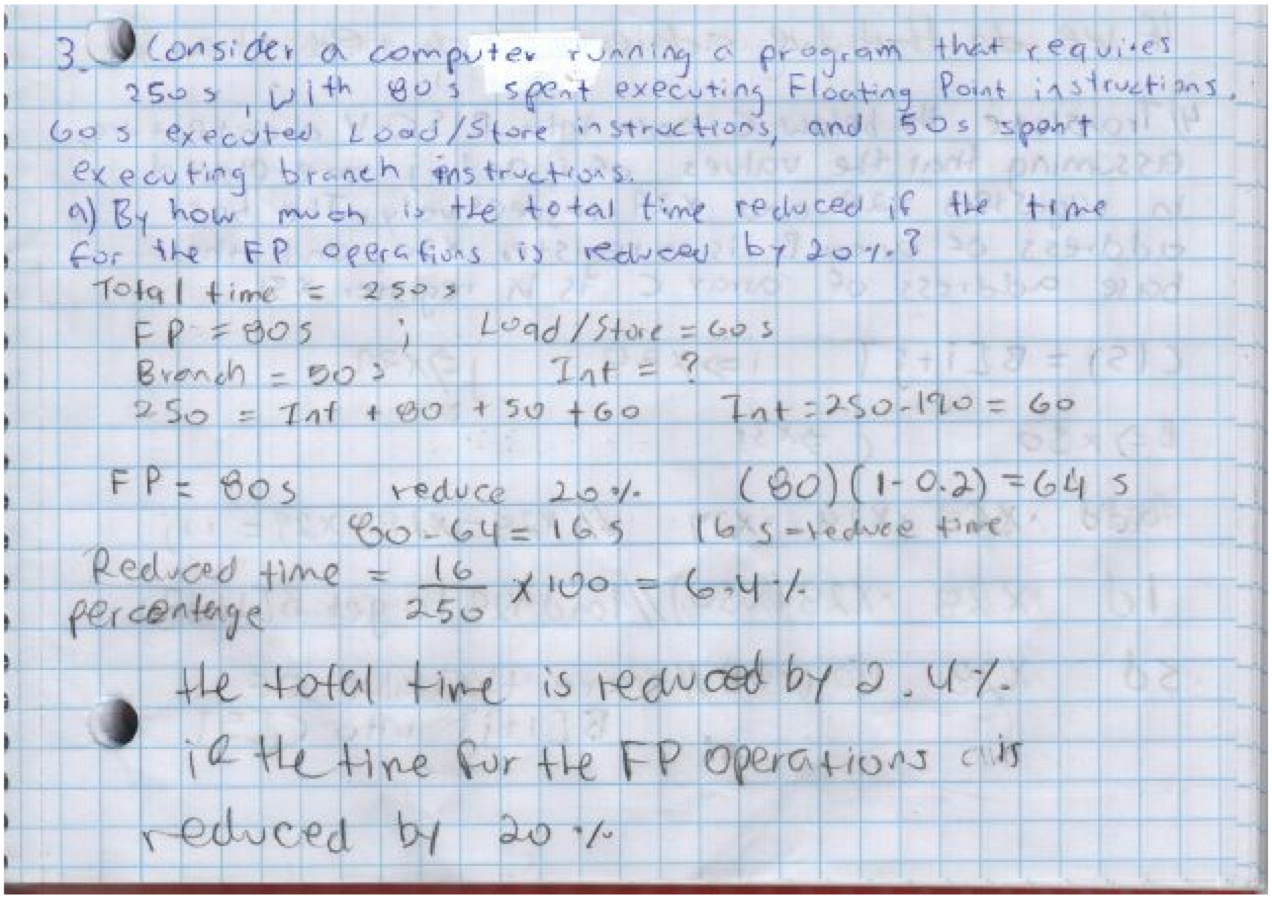
One fallacy students and researchers often make is expecting to improve the overall performance of a computer just by improving only one aspect of the computer. Instructions are classified as: Integer, Floating Point, Load/Store, Branch, in this fictitious processor. Consider a computer running a program that requires 250s, with 80s spent executing Floating Point (FP) instructions, 60s executed Load/Store instructions, and 50s spent executing branch instructions.

a. By how much is the total time reduced if the time for the FP operations is reduced by 20%?

b. By how much is the time for integer (INT) operations reduced if the total time is reduced by 20%?

c. Can the total time be reduced by 20% by reducing only the time for the branch instructions?

Solution:



3. Consider a computer running a program that requires 250s, with 80s spent executing Floating Point instructions, 60s executed Load/Store instructions, and 50s spent executing branch instructions.

a) By how much is the total time reduced if the time for the FP operations is reduced by 20%?

Total time = 250s

FP = 80s ; Load/Store = 60s

Branch = 50s   Int = ?

250 = Int + 80 + 50 + 60   Int = 250 - 190 = 60

FP = 80s   reduce 20% $(80)(1-0.2) = 64$ s

$80 - 64 = 16$ s   16 s = reduce time

Reduced time percentage $= \frac{16}{250} \times 100 = 6.4\%$

the total time is reduced by 2.4% if the time for the FP operations is reduced by 20%

b) By how much is the time for integer (INT) operations reduced if the total time is reduced by 20%?

INT = 60   reduce 20% $60(1-0.2) = 48$ s
reduced time = $60 - 48 = 12$ s

Reduced time percentage $= \frac{12}{250} \times 100\% = 4.8\%$

the total time is reduced by 4.8% if the time fo INT instructions is reduced by 20%.

c) Can the total time be reduced by 20% by reducing only the time for the branch instructions?

let's set time for branch instructions to 0.

$\frac{200}{250} = 0.80 \times 100\% = 80$ or $\frac{50}{250} \times 100\% = \boxed{20\%}$

If we do that we achieve 20% reduction

Problem 4: (6 pts)

Translate the below C code into RISC-V architecture, assuming that the values of *i* and *j* are placed in registers x28 and x29 respectively. The base address of array B is in register x30, and the base address of array C is in register x31.

```
C[5] = B[i+j];
```

Solution:

```
x28 = i

x29 = j

x30 = *B[0]

x31 = *C[0]

add x5, x28, x29    // x5 = x28 + x29 = i+j

slli x6, x5, 3      // using shift logic left to make x6 = x5*(2^3)

add x6, x6, x30     // x6 is address for B[i+j]

ld x7,  0 (x6)      // load  B[i+j] to  x7

sd x7 , 40(x31)     // store the value of x7 into C[5]
```

We are using x5 , x6 , x7 as temporary registers to hold values to make it organized and neat.

Problem 5: (8 pts)
Let register x5 hold the hex number $01a74cd0_{hex}$ and let x6 hold the hex number $00467b44_{hex}$. If this information is enough, then answer the following three questions, else explain what extra information you may need to proceed.

a) (2 pts) What are the contents of register x31 after the following instruction is executed? addi x31, x5, 32

b) (2 pts) What would be the contents of register x28 after the following instruction is executed? add x28, x5, x6

c) (2 pts) What would be the contents of register x29 after the following instruction is executed? ld x29, $1056_{decimal}$ (x30)

d) (2 pts) What would be the contents of register x29 after the following instruction is executed? sd x28, 16 (x29)

Solution:

a) The contents of register x31 are 01a74cf0 in hex or 27741424 in decimal after the instruction is executed
b) The contents of register x28 are 1edc814 in hex or 32360468 in decimal, if converted, after the instruction is executed
c) The extra information needed to answer the question is what are the contents of register x30 specifically at offset 1056
d) When using the results from the previous part of question 5, the contents of register x29 offset 16 will be 01edc814

Problem 6: (3 pts)
Let's consider the RISC-V code below. Indicate what the value in register x20 after the code below is executed, and explain why:

```
        ori x18, x0, 16
        addi x20, x18, 0
        beq x20, x18, HERE
        add x20, x20, x20

  HERE
```

Solution:

ori x18, x0,16: compare and add values, so 16 will be stored in x18

addi x20, x18,0: we add integer 0 with the values stored in x18 which is 16

beq x20,x18,HERE: comparison between x20 and x18, we can conclude that they are equal, so the next instructions in the program will not be executed

add x20,x20,x20: can not be executed (if it was executed the value stored in x20 will be 16+16=32 , x20=x20+x20) because we know that x20=x18 and x18 is given an initial value of 16

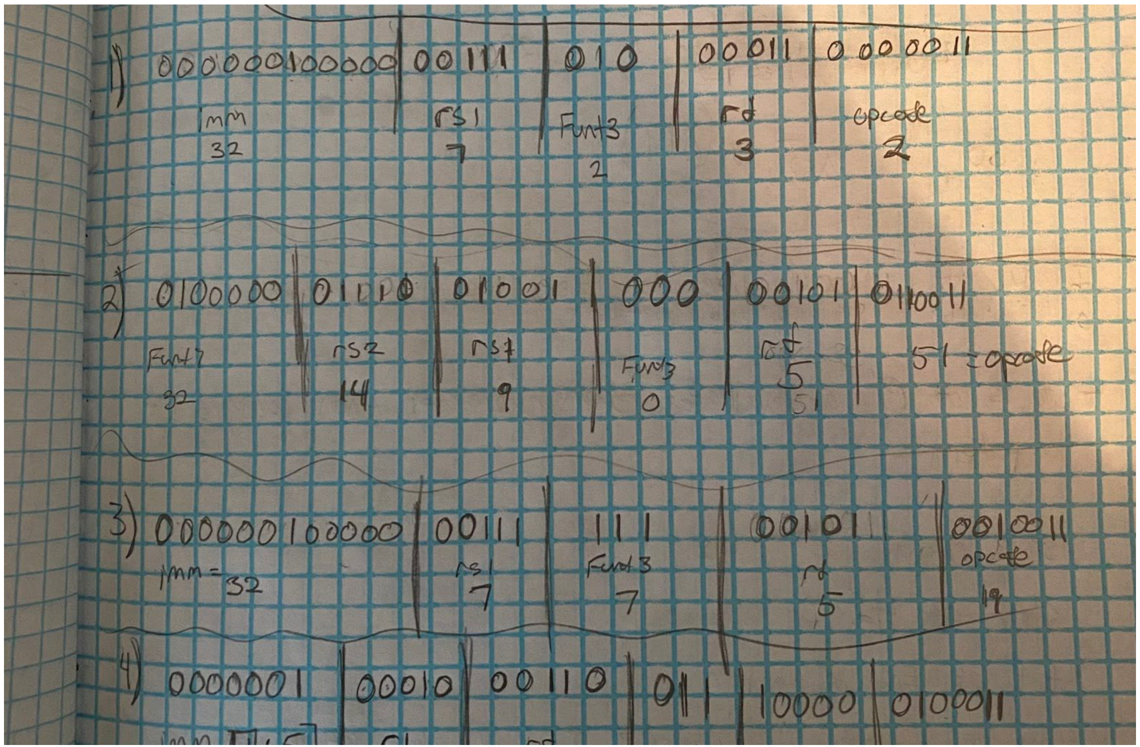
Problem 7: (12 pts)

Convert RISC-V assembly instructions to machine language or vice versa:

1) lw x3, 32(x7)
2) sub x5, x9, x14
3) andi x5, x7, 32
4) sd x2, 48(x6)
5) 00011000010111100000010000010011
6) 01110110010110011011111010100011

Solution:

1) 00000010000000111010000110000011
2) 01000000111001001000001010110011
3) 00000010000000111111001010010011
4) 00000010001000110011100000100011
5) addi x8, x28, 389
6) sd x5, 1917(x19)



1) 000000100000 | 00111 | 010 | 00011 | 0000011

imm 32 | rs1 7 | Funt3 2 | rd 3 | opcode 2

2) 0100000 | 01110 | 01001 | 000 | 00101 | 0110011

Funt7 32 | rs2 14 | rs1 9 | Funt3 0 | rd 5 51 | 51 = opcode

3) 000000100000 | 00111 | 111 | 00101 | 0010011

imm = 32 | rs1 7 | Funt3 7 | rd 5 | opcode 19

4) 0000001 | 00010 | 00110 | 011 | 10000 | 0100011