COMP1411 (Spring 2022) Introduction to Computer Systems

Individual Assignment 2 Duration: 00:00, 19-Mar-2022 ~ 23:59, 20-Mar-2022

Name	
Student number	

Question 1. [3 marks]

In this question, we use the Y86-64 instruction set (please refer to Lecture 4-6).

1(a) [1 mark]

Write the machine code encoding of the assembly instruction:

"`mrmovq 0x230(%rax), %rcx`".

Please write the bytes of the machine code in hex-decimal form, i.e., using two hex-decimal digits to represent one byte. The machine has a little-endian byte ordering. You are allowed to leave spaces between adjacent bytes for better readability.

Show your steps. Only giving the final result will NOT get a full mark of this question.

Answer:

mrmovq D(rB), rA → 50 rA rB D

rA: %rcx = 1

rB: %rax = 0

D: 0x230 = 30 02 00 00 00 00 00 00

The bytes of the machine code:

50 10 30 02 00 00 00 00 00 00

1(b) [2 marks]

Consider the execution of the instruction "`mrmovq 0x230(%rax), %rcx`". Assume that for now, the data in register `%rax` is 0x100, just before executing this instruction, the value of PC is 0x300. Please use "**vm**" to represent the actual data reading from the main memory.

Describe the steps done in the following stages: Fetch, Decode, Execute, Memory, Write Back, PC update, by filling in the blanks in the table below.

Note that you are required to fill in the generic form of each step in the second column; and in the third column, fill in the steps for the instruction "`mrmovq 0x230(%rax), %rcx`" with the above given values. If you think there should not be a step in some stage, just leave the blanks unfilled.

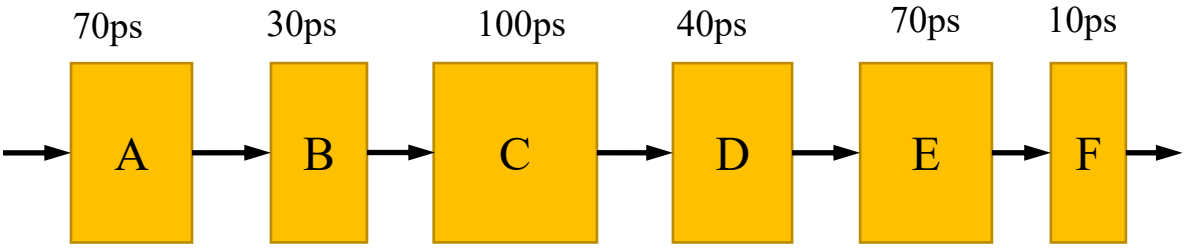
The symbol "←" means reading something from the right side and assign the value to the left side. X:Y means assign the highest 4 bits of a byte to X, and assign the lowest 4 bits of the byte to Y.

Answer:

Stages	**mrmovq D(rB), rA**	**mrmovq 0x230(%rax), %rcx**
Fetch	icode: ifun ← **M_1[PC]** rA:rB ← **M_1[PC+1]** valC ← **M_8[PC+2]** valP ← **PC + 10**	icode: ifun ←**M_1[0x300] = 5:0** rA:rB ← **M_1[0x301] = 1:0** valC ← **M_8[0x302] = 0x230** valP ← **0x300 + 10 = 0x30a**
Decode	valB ← **R[rB]**	valB ← **R[%rax] = 0x100**
Execute	valE ← **valB + valC**	valE ← **0x100 + 0x230 = 0x330**
Memory	valM ← M_8[**valE**]	valM ← M_8[**0x330**] = **vm**
Write back	R[**rA**] ← **valM**	R[**%rcx**] ← **valM = vm**
PC update	PC ← **valP**	PC ← **0x30a**

Question 2. [3 marks]

Suppose a combinational logic is implemented by 6 serially connected components named from A to F. The whole computation logic can be viewed as an instruction. The number on each component is the time delay spent on this component, in time unit ps, where 1ps = 10^{-12} second. Operating each register will take 20ps.



Throughput is defined as how many instructions can be executed on average in one second for a pipeline, and the unit of throughput is IPS, instructions per second.

Latency refers to the time duration starting from the very first component and ending with the last register operation finished, the time unit for latency is ps.

For throughput, please write the result in the form X.XX * 10^{Y} IPS, where X.XX means one digit before the dot and two fractional digits after the dot, and Y is the exponent.

2(a) Making the computation logic a 3-stage pipeline design that has the maximal throughput. Note that a register shall be inserted after each stage to separate their combinational logics. [1.5 marks]

- Please answer how to partition the stages.
- Please compute the throughput and latency for your pipeline design, with steps.

Answer:

Stage1: A and B; (70ps + 30ps = 100ps)

Stage2: C; (100ps)

Stage3: D, E and F (40ps + 70ps +10ps = 120ps)

Cycle time = 120ps + 20ps = 140ps

Throughput = $\frac{1}{140ps} \times 10^{12}\ IPS = 7.14 * 10^{9}\ IPS$

Latency = 140ps * 3 = 420ps

2(b) Making the computation logic a 4-stage pipeline design that has the maximal throughput. Note that a register shall be inserted after each stage to separate their combinational logics. [1.5 marks]

- Please answer how to partition the stages.
- Please compute the throughput and latency for your pipeline design, with steps.

Answer:

Stage1: A and B; (70ps + 30ps = 100ps)

Stage2: C; (100ps)

Stage3: D (40ps)

Stage4: E and F (70ps + 10ps = 80ps)

Cycle time = 100ps + 20ps = 120ps

Throughput = $\frac{1}{120ps} \times 10^{12}\ IPS = \ 8.33 * 10^{9}\ IPS$

Latency = 120ps * 4 = 480ps

Question 3. [4 marks]

The following byte sequence is the machine code of a program function compiled with the Y86-64 instruction set (refer to Lecture 6). The memory address of the first byte is 0x300. Note that the byte sequence is written in hex-decimal form, i.e., each number/letter is one hex-decimal number representing 4 binary bits, and two numbers/letters represent one byte.

630030F3020000000000000030F11E00000000000000702303000000000006010613162117 61F0300000000000090

Please write out the assembly instructions (in Y86-64 instruction set) corresponding to the machine codes given by the above bytes sequence, and explain what this program function is computing. The machine has a little-endian byte ordering.

Answer:

0x300: 63 00

0x302: 30 F3 02 00 00 00 00 00 00

0x30c: 30 F1 1E 00 00 00 00 00 00

0x316: 70 23 03 00 00 00 00 00 00

0x31F: 60 10

0x321: 61 31

0x323: 62 11

0x325: 76 1F 03 00 00 00 00 00 00

0x32E: 90

```
0x300:              xorq %rax, %rax         # 63 00
0x302:              irmovq $2, %rbx         # 30 F3 02 00 00 00 00 00 00
0x30c:              irmovq $30, %rcx        # 30 F1 1E 00 00 00 00 00 00
0x316:              jmp test                # 70 23 03 00 00 00 00 00 00
0x31F:      loop:
0x31F:              addq %rcx, %rax         # 60 10
0x321:              subq %rbx, %rcx         # 61 31
0x323:      test:
0x323:              andq %rcx, %rcx         # 62 11
0x325:              jg loop                 # 76 1F 03 00 00 00 00 00 00
0x32E:              ret                     # 90
```

30+28+26+24+22+20+18+16+14+12+10+8+6+4+2

This program returns the sum of all positive even numbers <= 30.
