**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 0x15F(%rbx), %rax”.

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

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

*Answer*:

(a)

1) mrmovq’s op code: 50

2) rA = %rax, rB = %rbx, so the second byte will be 03

3) displacement = 0x15F, written into 8-byte little-endian encoding is: 5F 01 00 00 00 00 00 00

4) the final machine code: 50 03 5F 01 00 00 00 00 00 00

**1(b)** [2 marks]

Consider the execution of the instruction “mrmovq 0x15F(%rbx), %rax”. Assume that for now, the data in register %rbx is 0x200, just before executing this instruction, the value of PC is 0x420. We use “**vm**” to represent the data read 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 0x15F(%rbx), %rax” 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 0x15F(%rbx), %rax** |
| Fetch | icode: ifun ← \_M1[PC]\_  rA:rB ← \_ M1[PC+1]\_  valC ← \_ M8[PC+2]\_  valP ← \_PC+10\_ | icode: ifun ← \_M1[0x420]=5:0\_  rA:rB ← \_ M1[0x421]=0:3\_  valC ← \_ M8[0x422]=0x15F\_  valP ← \_0x420+10=0x42A\_ |
| Decode | valB ← \_R[rB]\_ | valB ← \_R[%rbx]=0x200\_ |
| Execute | valE ← \_valB + valC\_ | valE ← \_0x200+0x15F=0x35F\_ |
| Memory | valM ← M8[\_valE ] | valM ← M8[0x35F] = vm |
| Write back | R[ rA ] ← \_\_valM\_ | R[ %rax ] ← \_vm\_ |
| PC update | PC ← \_valP\_ | PC ← \_0x42A\_ |

**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.

A

B

C

D

E

F

R

E

G

10ps

55ps

40ps

90ps

20ps

70ps

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 \* 10Y IPS, where X.XX means one digit before the dot and two fractional digits after the dot, and Y is the exponent.

**2(a)** Insert two registers (not including the register in the figure) making the computation logic a 3-stage pipeline design that has the maximal throughput. Note that a register is inserted between any adjacent stages. [1.5 marks]

* Please answer where the two registers are inserted respectively.
* Please compute the throughput and latency for your pipeline design, with steps.

Answer:

The two registers will be inserted between C and D, between D and E.

Throughput: 1 / ((105 + 20) \* 10-12) = 8 \* 109 IPS

Latency: (105 + 20) \* 3 = 375 ps

**2(b)** Insert three registers (not including the register in the figure) making the computation logic a 4-stage pipeline design that has the maximal throughput. Note that a register is inserted between any adjacent stages. [1.5 marks]

* Please answer where the three registers are inserted respectively.
* Please compute the throughput and latency for your pipeline design, with steps.

Answer:

3 registers inserted: between B and C, between C and D, between D and E

Throughput: 1 / ((90 + 20) \* 10-12) = 9.09 \* 109 IPS

Latency: (90 + 20) \* 4 = 440 ps

**Question 3**. [4 marks]

The following byte sequence is the machine code of a function compiled with the Y86-64 instruction set (refer to Lecture 6). The memory address of the first byte is 0x200. 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. Assume the machine is a big-endian byte order machine. Assume that by default the value in register %rax will be returned.

**30F0000000000000002830F3000000000000000030F1000000000000000270000000000000022B600361106200760000000000000227203090**

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 function is computing.

Answer:

0x200: irmovq $40, %rax

0x20A: irmovq $0, %rbx

0x214: irmovq $2, %rcx

0x21E: jmp .L2

L1:

0x227: addq %rax, %rbx

0x229: subq %rcx, %rax

L2:

0x22B: andq %rax, %rax

0x22D: jg .L1

0x236: rrmovq %rbx, %rax

0x238: ret

This program computes: 40 + 38 + 36 + … + 2