**COMP1411 (Spring 2023) Introduction to Computer Systems**

Individual Assignment 2 Duration: 12:00, 10-Mar-2023 ~ 23:59, 12-Mar-2023

|  |  |
| --- | --- |
| *Name* |  |
| *Student number* |  |

**Question 1**. [3 marks]

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

**1(a)** [1 mark]

**Write** the machine code encoding of the assembly instruction:

mrmovq 0x1356(%rbp), %rdi

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*:**

1) mrmovq’s op code: 50

2) rA = %rdi, rB = %rbp, so the second byte will be 75

3) displacement = 0x1356, written into 8-byte little-endian encoding is: 56 13 00 00 00 00 00 00

4) the final machine code: 50 75 56 13 00 00 00 00 00 00

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

Consider the execution of the instruction “mrmovq 0x1356(%rbp), %rdi”. Assume that for now, the data in register %rbp is 0x334 just before executing this instruction, the value of PC is 0x540. 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 0x1356(%rbp), %rdi” 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 0x1356(%rbp), %rdi** |
| Fetch | icode: ifun ← \_M1[PC]\_  rA:rB ← \_ M1[PC+1]\_  valC ← \_ M8[PC+2]\_  valP ← \_PC+10\_ | icode: ifun ← \_M1[0x540]=5:0\_  rA:rB ← \_ M1[0x541]=7:5\_  (both 7:5 and 5:7 will receive marks)  valC ← \_ M8[0x542]=0x1356\_  valP ← \_0x540+10=0x54A\_ |
| Decode | valB ← \_R[rB]\_ | valB ← \_R[%rbp]=0x334\_ |
| Execute | valE ← \_valB + valC\_ | valE ← \_0x334+0x1356=0x168A\_ |
| Memory | valM ← M8[\_valE ] | valM ← M8[0x168A] |
| Write back | R[ rA ] ← \_\_valM\_ | R[%rdi] ← \_vm\_/valM |
| PC update | PC ← \_valP\_ | PC ← \_0x54A\_ |

**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

30ps

65ps

50ps

100ps

30ps

80ps

Throughput is defined as how many instructions can be executed on average in one second for a pipeline in the long run, 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)** Make 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. By default, a register will be inserted after the last stage, i.e., after step F. [1.5 marks]

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

Answer:

First Correct: ABC | D | EF

Second correct: ABC | DE | F

Size of stage 1 = 30 +65 + 50 = 145

Size of stage 2 = 100

Size of satage 3 = 30 + 80 = 110

Throughput: 1 / ((145 + 20) \* 10-12) = 6.06 \* 109 IPS

Latency: (145 + 20) \* 3 = 495 ps

**2(b)** Make 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. By default, a register will be inserted after the last stage, i.e., after step F. [1.5 marks]

* Please answer how to partition the stages.
* 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

Size of stage 1 = 30 + 65 = 95

Size of stage 2 = 50

Size of stage 3 = 100

Size of stage 4 = 30 + 80 = 110

Throughput: 1 / ((110 + 20) \* 10-12) = 7.69 \* 109 IPS

Latency: (110 + 20) \* 4 = 520 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 0x1500. 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 little-endian byte order machine.** Assume that by default the value in register %rax will be returned.

**30F3320000000000000030F1000000000000000030F00100000000000000702B15000000000000603161036233762715000000000000201090**

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:

0x1500: irmovq $50, %rbx

0x150A: irmovq $0, %rcx

0x1514: irmovq $1, %rax

0x151E: jmp .L2

L1:

0x1527: addq %rbx, %rcx

0x1529: subq %rax, %rbx

L2:

0x152B: andq %rbx, %rbx

0x152D: jg .L1

0x1536: rrmovq %rcx, %rax

0x1538: ret

This program computes: 50 + 49 + 48 + … + 1