COMP1411 (Spring 2023)   Introduction to Computer Systems

Individual Assignment 2   Duration: 00:00, 11-Mar-2023 ~ 23:59, 12-Mar-2023

Name	**WANG Ruijie**
Student number	**22103808d**

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 -5(%rdi), %r8`".

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

Answer:

The destination operand is the register %r8, and the source operand is the memory -5(%rdi), thus the instruction code is 5, the function code is 0, and the encoded representation for the operands is 87.

Then convert -5 into its hexadecimal format,

FF FF FF FF FF FF FF FB.

According to the little-endian ordering, the encoded representation should be

FB FF FF FF FF FF FF FF.

Therefore, the encoded representation for the whole instruction should be:

50 87 FB FF FF FF FF FF FF FF.

1(b) [2 marks]

Consider the execution of the instruction "`mrmovq -5(%rdi), %r8`". Assume that the data in register `%rdi` is 0x10A. The value of PC is 0x300. Suppose the data reading from the main memory is -22.

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

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 -5(%rdi), %r8`" by replacing the generic variables of the second column with their corresponding 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:

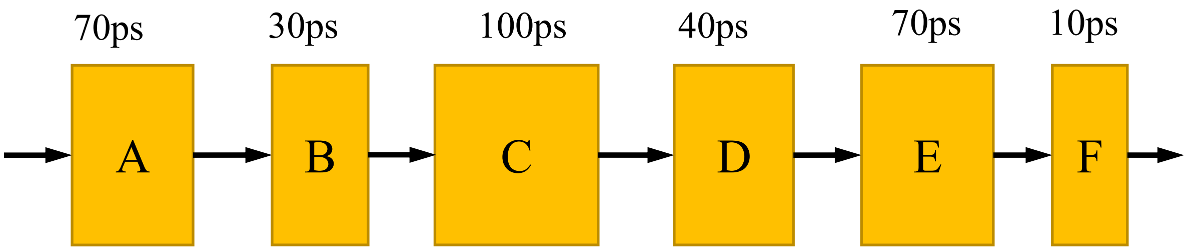
Stages	**mrmovq D(rB), rA**	**mrmovq -5(%rdi), %r8**

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[0x300 + 1] = 8:7
		valC ← M_8[0x300 + 2] = 0xFB
		valP ← 0x300 + 10 = 0x30A

Decode	valB ← R[rB]	valB ← R[%rdi] = 0x10A
Execute	valE ← valB + valC	valE ← 0x10A + 0xFB = 0x105
Memory	valM ← M_8[valE]	valM ← M_8[0x105]
Write back	R[rA] ← valM	R[%r8] ← valM = M_8[0x105] = -27
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.

The first register operation should be inserted between B and C, and the second register operation should be inserted between C and D.

Therefore, the time consuming of each stage is 100ps, 100ps and 120ps separately. Hence the latency is

$$(120ps + 20ps) * 3 = 420ps,$$

and the throughput is

$$(140ps)^{-1} = 7.14*10^{9}IPS.$$

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.

The registers should be inserted between B and C, C and D, and D and E separately.

Therefore, the time consuming of each stage is 100ps, 100ps, 40ps and 80ps separately.

Hence the latency is

$$(100ps + 20ps) * 4 = 480ps,$$

and the throughput is

$$(120ps)^{-1} = 8.33 * 10^{9} \text{ IPS}.$$

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.

630030F3030000000000000030F130000000000000007023030000000000006010613162117 61F03000000000000090

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:

The function is computing the sum of the number sequence

3,6,9,12,15,18,21,24,27,30,33,36,39,42,45,48.

According to the Y96-64 Instruction Set, the instructions should be (In parentheses are the address changes stored in PC):

63 00 → xorq %rax, %rax (0x300 → 0x302)

30 F3 0300000000000000 → irmovq $3, %rbx (0x302 → 0x30C)

30 F1 3000000000000000 → irmovq $48, %rcx (0x30C → 0x316)

70 2303000000000000 → jmp 0x323 (0x316 → 0x31F)

60 10 → addq %rcx, %rax (0x31F → 0x321)

61 31 → subq %rbx, %rcx (0x321 → 0x323)

62 11 → andq %rcx, %rcx (0x323 → 0x325)

76 1F03000000000000 → jg 0x31F (0x325 → 0x32E)

90 → ret (0x32E → 0x32F)

The assembly codes should be:

```
func:
        xorq %rax, %rax
        irmovq $3, %rbx
        irmovq $48, %rcx
        jmp L2
L2:
        andq %rcx, %rcx
        jg L3
        ret
L3:
        addq %rcx, %rax
        subq %rbx, %rcx
```