**Chapter 4 The Von Neumann Model**

**4.3 What is misleading about the name program counter? Why is the name instruction pointer more insightful?**

Even though PC is called as Program Counter, it actually does not maintain a count of any certain sort. In addition, the contents of this register are, in some sense, “pointing” to the next instruction to be processed.

**4.5 The following table represents a small memory. Refer to this table for the following questions.**

**Address Data**

**0000 0001 11100100 0011**

**0001 1111 0000 0010 0101**

**0010 0110 11110000 0001**

**0011 0000 0000 0000 0000**

**0100 0000 0000 0110 0101**

**0101 0000 0000 0000 0110**

**0110 1111 1110 1101 0011**

**0111 0000 0110 1101 1001**

1. **What binary value does location 3 contain? Location 6?**

Location 3: 0000 0000 0000 0000

Location 6: 1111 1110 1101 0011

**b The binary value within each location can be interpreted in many ways. We have seen that binary values can represent unsigned numbers, 2's complement signed numbers, floating point numbers, and so forth.**

1. **Interpret location 0 and location 1 as 2's complement integers.**

Location 0: 0001 1110 0100 0011 = 7747

Location 1: 1111 0000 0010 0101 = -4059

1. **Interpret location 4 as an ASCII value.**

101 = ‘e’

1. **Interpret locations 6 and 7 as an IEEE floating point number. Location 6 contains number [15:0]. Location 7 contains number [31:16].**

0 0000 1101 1011 0011 1111 1101 1010 011

IEEE Floating: 1.10110011111111011010011 \* 2^(-114)

1. **Interpret location 0 and location 1 as unsigned integers.**

Location 0: 7747

Location 1: 61477

1. **In the von Neumann model, the contents of a memory location can also be an instruction. If the binary pattern in location 0 were interpreted as an instruction, what instruction would it represent?**

Add R7 R1 R3: add two numbers, which are from R1 and R3, and store the result in R7.

1. **A binary value can also be interpreted as a memory address. Say the value stored in location 5 is a memory address. To which location does it refer? What binary value does that location contain?**

Location 6: 1111 1110 1101 0011

**4.8 Suppose a 32-bit instruction takes the following format: OPCODE DR SRI SR2 UNUSED If there are 225 opcodes and 120 registers,**

a. What is the minimum number of bits required to represent the OPCODE?

8

*b.* What is the minimum number of bits required to represent the Destination Register (DR)?

7

1. What is maximum number of UNUSED bits in the instruction encoding?

31

**4.9 The FETCH phase of the instruction cycle does two important things. One is that it loads the instruction to be processed next into the IR. What is the other important thing?**

The other important thing which the FETCH phase of the instruction cycle does is to load other address of the next instruction into the program counter.

**Chapter 5 The LC-3**

**5.2 A memory's addressability is 64 bits. What does that tell you about the size of the MAR and MDR?**

Addressability is 64 bits means the value stored in MDR is a 64-bit size of information. However, it does not clarify the size of the MAR.

**5.7 What is the largest positive number we can represent literally (i.e., as an immediate value) within an LC-3 ADD instruction?**

Binary: 01111 decimal: 15

**5.25 Write an LC-3 program that compares two numbers in R2 and R3 and puts the larger number in Rl. If the numbers are equal, then Rl is set equal to 0.**

Logic: 1) R4🡨 -R3

2) R1🡨R2 ADD R4

3) n: R1🡨R3

z: R1🡨0

p: R1🡨R2

Program:

1001 100 011 111111 ; NOT R4,R3

0001 100 100 1 00001 ; ADD R4, R4, #1

0001 001 010 0 00 100 ; ADD R1, R4, R2

0000 0 1 0 000000010 ; BRz

0000 1 0 0 000000001 ; BRn

0000 0 0 1 000000010 ; BRp

0001 001 011 1 00000 ; ADD R1, R3, #0

0000 1 1 1 000000001 ; BRnzp

0001 001 010 1 00000 ; ADD R1, R2, #0

1111 0000 0010 0101 ; TRAP 0x25

**5.33 If the value stored in R0 is 5 at the end of the execution of the following instructions, what can be inferred about R5?**

**x3000 0101 111 111 1 00000 ; R7🡨0**

**X3001 0001 110 111 1 00001 ; R6🡨1**

**x3002 0101 100 101 0 00 110 ; R4🡨 (R5 AND R6)**

**x3003 0000 0 1 0 000000001 ; BRz x3005**

**x3004 0001 000 000 1 00001 ; R0🡨R0+1**

**x3005 0001 110 110 0 00 110 ; R6🡨R6\*2**

**x3006 0001 111 111 1 00001 ; R7🡨R7+1**

**x3007 0001 001 111 1 11000 ; R1🡨R7-8**

**x3008 0000 1 0 0 111111001 ; BRn x3002**

**x3009 0101 111 111 1 00000**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| X3000 | R7:0000 0000 0000 0000 |  |  |  |  |  |  |  |  |
| X3001 | R6:0000 0000 0000 0001 |  |  |  |  |  |  |  |  |
| X3002 | R4: Not Zero | R4: |  |  |  |  |  |  |  |
| X3003 | Not Execute |  |  |  |  |  |  |  |  |
| X3004 | R0:0000 0000 0000 0001 |  |  |  |  |  |  |  |  |
| X3005 | R6:0000 0000 0000 0010 |  |  |  |  |  |  |  |  |
| X3006 | R7:0000 0000 0000 0001 |  |  |  |  |  |  |  |  |
| X3007 | R1 is negative |  |  |  |  |  |  |  |  |
| X3008 | BRn x3002 |  |  |  |  |  |  |  |  |
| X3009 | Not Important |  |  |  |  |  |  |  |  |

Because the value stored in R0 is 5, the program need to perform the instruction of x3004 for 5 times. In addition, since the original value of R7 is 0 and the value will be added one every time the loop performed. Therefore, the total times the loop will perform is 8 until the value of R1 is zero and the instruction of x3008 will not be executed.

R6: 0000 0000 0000 0001 0000 0000 0000 0010 0000 0000 0000 0100

0000 0000 0000 1000 0000 0000 0001 0000 0000 0000 0010 0000

0000 0000 0010 0000

Therefore, it can be inferred that R5 is xxxx xxxx 0001 1111(x means the value of the bit is not decided, it can be both 0 or 1).

**5.40 The logic diagram below shows part of the control structure of the LC-3 machine. What is the purpose of the signal labeled A?**

The signal labeled A is actually performed as a conditional branch instruction to determine whether to change the instruction flow; that is, whether to depart from the usual sequential execution of instructions that we get as a result of incrementing PC during the FETCH phase of each instruction.