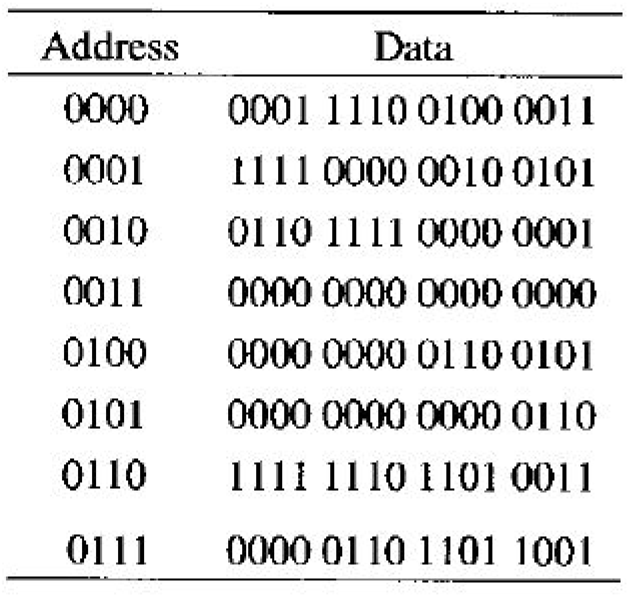
Joseph Hyatt
CSE 2130-01

Homework 2

Chapter (4): 4.5 b; 4.9; 4.11

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



Address	Data
0000	0001 1110 0100 0011
0001	1111 0000 0010 0101
0010	0110 1111 0000 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

b) With each location the binary value 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.
Value at location 0 = 0001 1110 0100 0011 and its 2's complement in binary = 1110 0001 1011 1101 which = 28894. Value at location 1 = 1111 0000 0010 0101 and its 2's complement in binary = 0000 1111 1101 1011 which = -4059.

(2) Interpret location 4 as an ASCII value.
value at location 4 = 0000 0000 0110 0101 which is e in ASCII.

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

We have to interpret what is the Floating point number represented by 00000110110110011111111011010011. Looking at this number, we can identify Sign = 0, Exponent = 00001101,Mantissa = 10110011111111011010011, so Float = 13.7030891180038452148 4.

(4) Interpret location 0 and location 1 as unsigned integers.
Location 0 as an unsigned integer would be = 7747 and Location 1 = 61477.

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?

Another important thing of the FETCH phase is the operation performed during the loading of the address of the next instruction into the program counter.

4.11) State the phases of the instruction cycle and briefly describe what operations occur in each phase.

1. Instruction Fetch
2. Instruction Decode
3. Operand Fetch
4. Instruction Execution
5. Result Write Back to Main Memory

Instruction Fetch:

This is the first phase of the instruction cycle. The PC (program counter) stores the address of the next instruction. The next instruction is fetched from this memory address and stored into the instruction register. The PC(Program Counter) register is updated to store the address of the next instruction.

Instruction Decode:

The decoder interprets or decodes the instruction that is stored in the instruction register.

Operand Fetch:

The required data is fetched from the memory and stored into the data register. The operands are fetched by using the addressing mode. The addressing mode defines a way how the operand will be fetched from the memory.

Instruction Execution:

The instruction that is decoded in the second phase is executed and appropriate action is completed by the instruction.

Result Write Back to Main Memory:

This step is the final step of the instruction cycle. During the execution of the instruction, the intermediate result is stored in the registers. At last, the final result stored in the register is copied into the main memory.

Chapter (5): 5.1; 5.4; 5.15

5.1) Given instructions ADD, JMP, LEA, and NOT, identify whether the instructions are operating instructions, data movement instructions, or control instructions. For each instruction, list the addressing modes that can be used with the instruction.

ADD is a operate instruction. It is used to perform instructions process data. The common operating instructions in LC-3 are ADD, AND, and NOT. The ADD instruction is used to immediate addressing mode and register addressing mode. In the immediate addressing mode, the operation code in the ADD instruction is immediately followed by the value to be referenced. The register addressing mode, a register is used to specify the operand instead of the memory location.

JMP is a control instruction and is used to perform instructions change that is executed. The JMP instruction is used in the register addressing mode. The register addressing mode address field contains a register reference in the case of a register address location.

LEA is a data movement instruction. It is used to transfer data between the memory and general registers and between the registers and the input/output devices. The LEA instruction is used in the immediate addressing mode. The data on which the operation is to be performed contained in the instruction to move.

NOT is a operate instruction, and is used to perform a unary operation. The NOT instruction is used in the register addressing mode, and the mirror address instruction field is needed in the NOT instruction.

5.4) Say we have a memory consisting of 256 locations, and each location contains 16 bits.

a. How many bits are required for the address?
8-bits are required for the address. Memory containing 256 locations, $256=2^8$. Therefore 8-bits required.

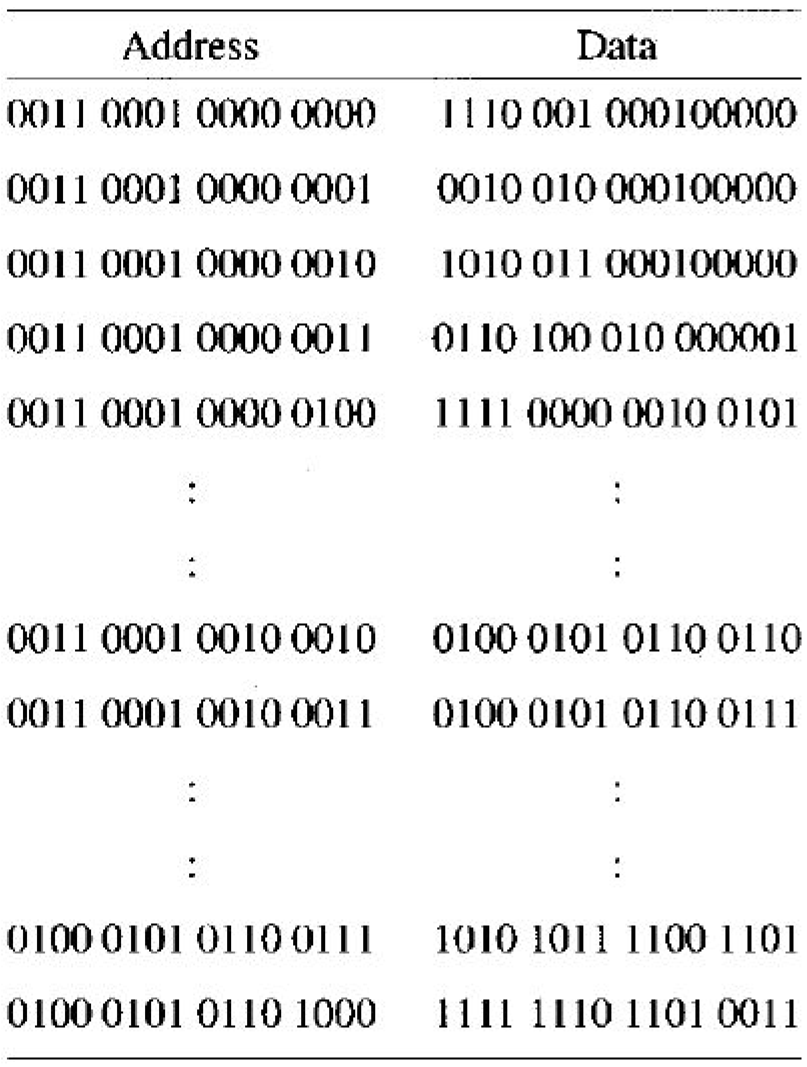
b. If we use the PC-relative addressing mode, and want to allow control transfer between instructions 20 locations away, how many bits of a branch instruction are needed to specify the PC-relative offset?

6. It will give a +/-20 range of 40. Therefore we need 6 bits to specify PC relative offset

c. If a control instruction is in location 3, what is the PC-relative offset of address 10. Assume that the control transfer instructions work the same way as in the LC-3.

6. PC Counter is incremented to 4, which is 10 - 4 = 6. Therefore 6 PC relative offset if the address is 10.

5.15) State the contents of Rl, R2, R3, and R4 after the program starting at location x3100 halts.



Address	Data
0011 0001 0000 0000	1110 001 000100000
0011 0001 0000 0001	0010 010 000100000
0011 0001 0000 0010	1010 011 000100000
0011 0001 0000 0011	0110 100 010 000001
0011 0001 0000 0100	1111 0000 0010 0101
:	:
:	:
0011 0001 0010 0010	0100 0101 0110 0110
0011 0001 0010 0011	0100 0101 0110 0111
:	:
:	:
0100 0101 0110 0111	1010 1011 1100 1101
0100 0101 0110 1000	1111 1110 1101 0011

1110 001 000100000 (LEA R1, 0x20)

R1 <- 0x3121 0010 010 000100000 (LD R2, 0x20

R2 <- Mem[0x3122] = 0x4566 1010 011 000100001 (LDI R3, 0x20)

R3 <- Mem[Mem[0x2123]] = 0xabcd 0110 100 010 000001 (LDR R4, R2, 0x1)

R4 <- Mem[R2 + 0x1] = 0xabcd 1111 0000 0010 0101 (TRAp 0x25)