5.2 5.4 5.9 5.15 5.16

5.2

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

Addressability can indicate ONLY the Size of Data is 64 bit,and MDR should be 64 bit

But We can't know how many , which means, we can't know any about MAR

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?
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?
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.

a.$log_2256 = 8$,need 8 bit

b.±20bit need 6bit literal number, because it can make -32 to 31 location away PC

c.PC + 1 + x = 10 ,and PC now = 3,so x = 6

5.9

We would like to have an instruction that does nothing. Many ISAs actually have an opcode devoted to doing nothing. It is usually called NOP, for NO OPERATION. The instruction is fetched, decoded, and executed. The execution phase is to do nothing! Which of the following three instructions could be used for NOP and have the program still work correctly?
a. 0001 001 001 1 00000
b. 0000 111 000000001
c. 0000 000 000000000
What does the ADD instruction do that the others do not do?

a. ADD R1, R1 ,#0 NO ,because it set condition code 0

b. BRnzp #1 NO,because then it will change PC = PC + 2 not PC +1

c. BR but all mask = 0,so it will never happen and don't change condition code

ADD instruction will change Condition code and others can't

5.15

State the contents of R1, 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 = 0x3100 + 1 + 20 = 0x3121

0010 010 000100000

LD R2, 0x20

0011 0001 0010 0010 0100 0101 0110 0110

R2 = MEM[0x3101+1+20] = MEM[0x3122] = 0x4566

1010 011 000100001

LDI R3, 0x20

0011 0001 0010 0011 0100 0101 0110 0111

0100 0101 0110 0111 1010 1011 1100 1101

R3 = MEM[Mem[0x3103 + 20]] = MEM[MEM[3123]]=MEM[0x4567]=0xABCD

0110 100 010 000001

LDR R4, R2, 0x1

R4 = MEM[R2 + 1] = MEM[4566+1]=0xABCD

1111 0000 0010 0101

TRAP 0x25

5.16

Which LC-3 addressing mode makes the most sense to use under the following conditions? (There may be more than one correct answer to each of these; therefore, justify your answers with some explanation.)
a. You want to load one value from an address that is less than ±2^8 locations away.
b. You want to load one value from an address that is more than 2^8 locations away.
c. You want to load an array of sequential addresses.

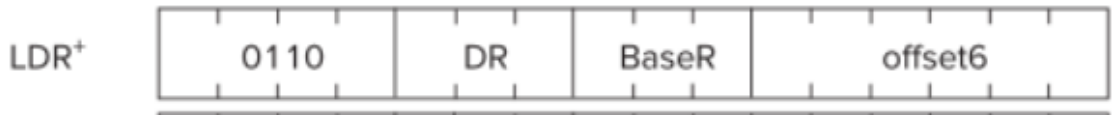
a.LD PC-relative Mode, because LD has 9 bit offset ,which can just cover ±2^8

b.LDI or LDR

if we take LDI , it's Indirect Mode , because max offset in LC3 ISA is LD,more we need MEM or REG to store the PCoffset.

if we take LDR,it's Base + offset Mode , use addres that BaseR contains to fetch data in MEM[Reg+offset]

LDR fast than LDI because LEA + LDR use register and LDI use MEM



c. LDR Base +offset Mode,because array is sequential,we can store array's head address in a register ,eg as follow

LEA + LDR. first we calculate the PC of Array[i] by LEA ,then use the DR of LEA as the Base R of LDR

,then we change the Base R by add 1 each time.