7.2 Suppose that the following instructions are found at the given locations in memory:

a. Show the contents of the IR, the PC, the MAR, the MDR, and A at the conclusion of instruction 20.

b. Show the contents of each register as each step of the fetch–execute cycle is performed for instruction 21.

a.

PC → MAR	
MDR → IR	final value of IR = 150
IR [address] → MAR	final value of MAR = 50
MDR → A	final value of MDR and A = 724
PC+1 → PC	final value of PC = 21

b.

	PC	MAR	MDR	IR	A
PC → MAR	21	21	724	150	724
MDR → IR	21	21	351	351	724
IR [address] → MAR	21	51	351	351	724
A + MDR → A	21	51	006	351	730
PC+1 → PC	21	51	006	351	730

7.4 Why are there two different registers (MAR and MDR) associated with memory? What are the equivalents in the Little Man Computer?

The MAR holds the address in the memory and the MDR connects to every cell in the memory unit. Little Man Computer uses program counter or MAR to get the memory address and he uses some memory cells as MDR.

7.11 Suppose that the instruction format for a modified Little Man Computer requires two consecutive locations for each instruction. The high-order digits of the instruction are located in the first mail slot, followed by the low-order digits. The IR is large enough to hold the entire instruction and can be addressed as IR [high] and IR [low] to load it. You may assume that the op code part of the instruction uses IR [high] and that the address is found in IR [low]. Write the fetch–execute cycle for an ADD instruction on this machine.

PC → MAR

MDR → IR [high]

PC+1 → PC

PC → MAR

MDR → IR [low]

IR [low] → MAR

MDR → A

PC+1 → PC

7.12 The Little Prince Computer (LPC) is a mutant variation on the LMC. (The LP is so named because the differences are a royal pain.) The LPC has one additional instruction. The extra instruction requires two consecutive words:

0XX

0YY

This instruction, known as move, moves data directly from location XX to location YY without affecting the value in the accumulator. To execute this instruction, the Little Prince would need to store the XX data temporarily. He can do this by writing the value on a piece of paper and holding it until he retrieves the second address. The equivalent in a real CPU might be called the intermediate address register, or IAR. Write the fetch–execute cycle for the LPC MOVE instruction.

PC → MAR

MDR → IAR

IAR [address] → MAR

XX data → MDR

PC+1 → PC

PC → MAR

MDR → IAR

IAR [address] → MAR

MDR → YY data

PC+1 → PC

7.13 Generally, the distance that a programmer wants to move from the current instruction location on a BRANCH ON CONDITION is fairly small. This suggests that it might be appropriate to design the BRANCH instruction in such a way that the new location is calculated relative to the current instruction location. For example, we could design a different LMC instruction 8CX. The C digit would specify the condition on which to branch, and X would be a single-digit relative address. Using 10’s complement, this would allow a branch of −5 to +4 locations from the current address. If we were currently executing this instruction at location 24, 803 would cause a branch on negative to location 27. Write a fetch–execute cycle for this BRANCH ON NEGATIVE RELATIVE instruction. You may ignore the condition code for this exercise, and you may also assume that the complementary addition is handled correctly. The single-digit address, X, is still found in IR [address].

I have no idea with this question.