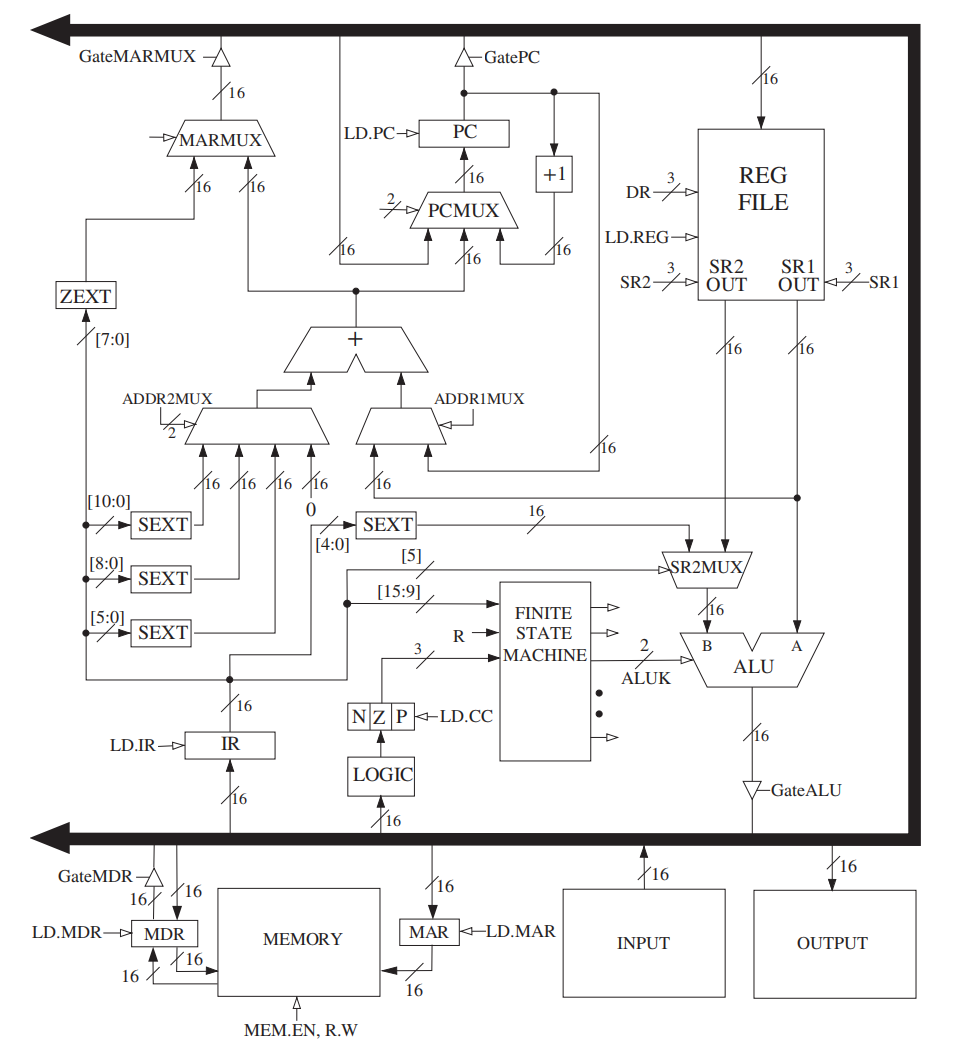
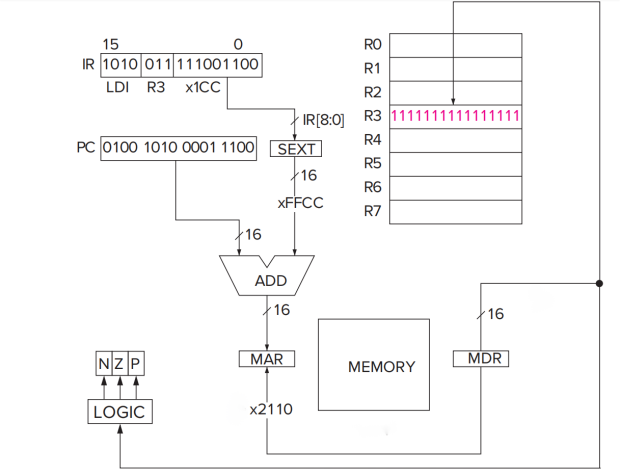
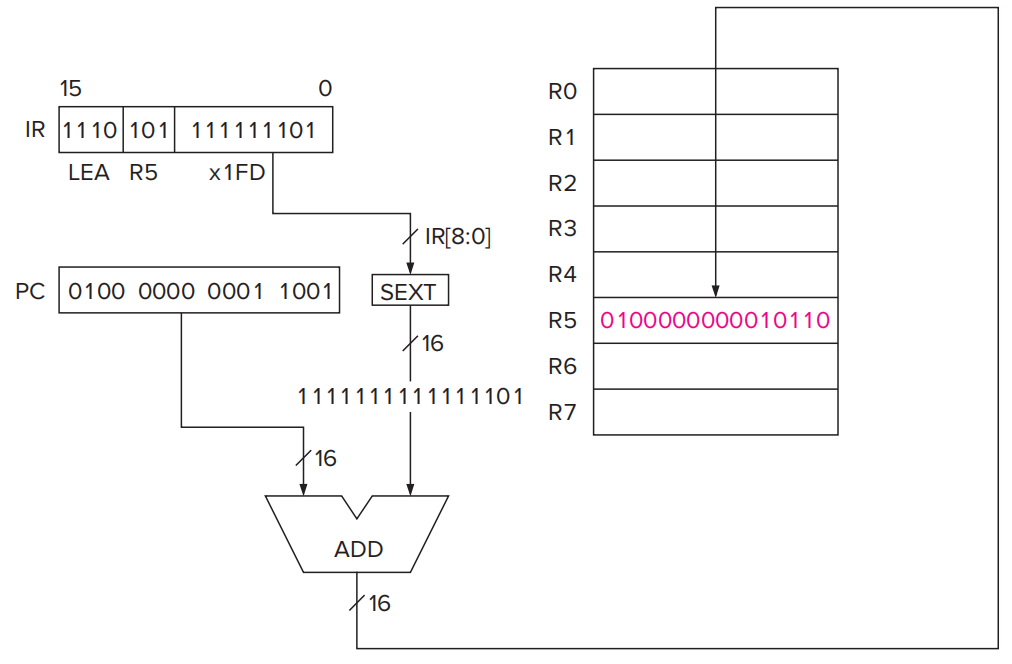
5.37 Using the overall data path in Figure 5.18, identify the elements that implement the STI instruction of Figure 5.8.

*Answer:*

Memory, MDR, MAR, IR, PC, Reg file, SEXT unit connected to IR [8:0], ADDR2MUX, ADDRIMUX set to PC, and their connected ADDER, MAXMUX and GateMARMUX implement STI instructions.

5.39 Using the overall data path in Figure 5.18, identify the elements that implement the LEA instruction of Figure 5.6.

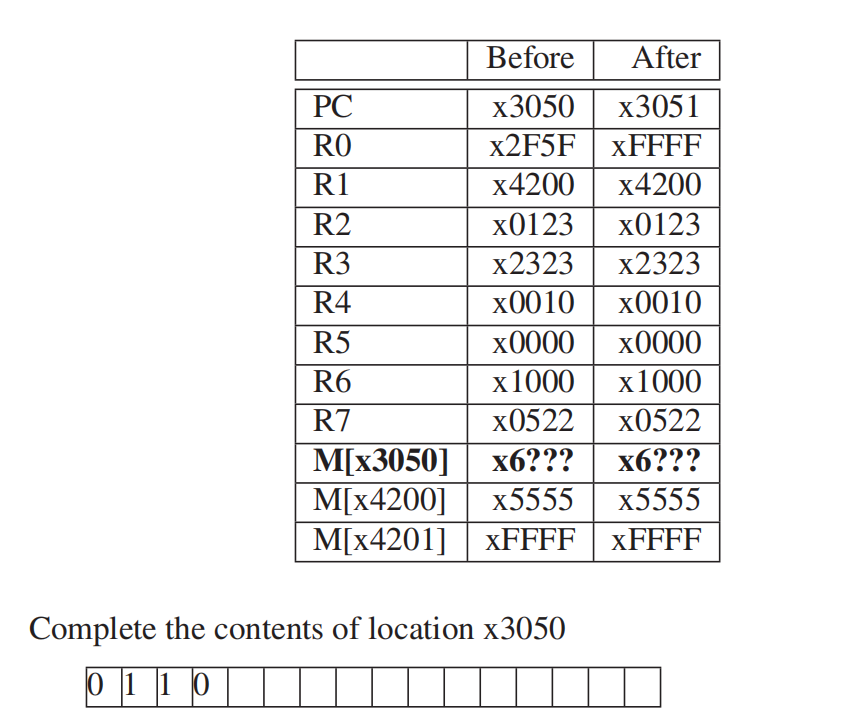


*Answer*:

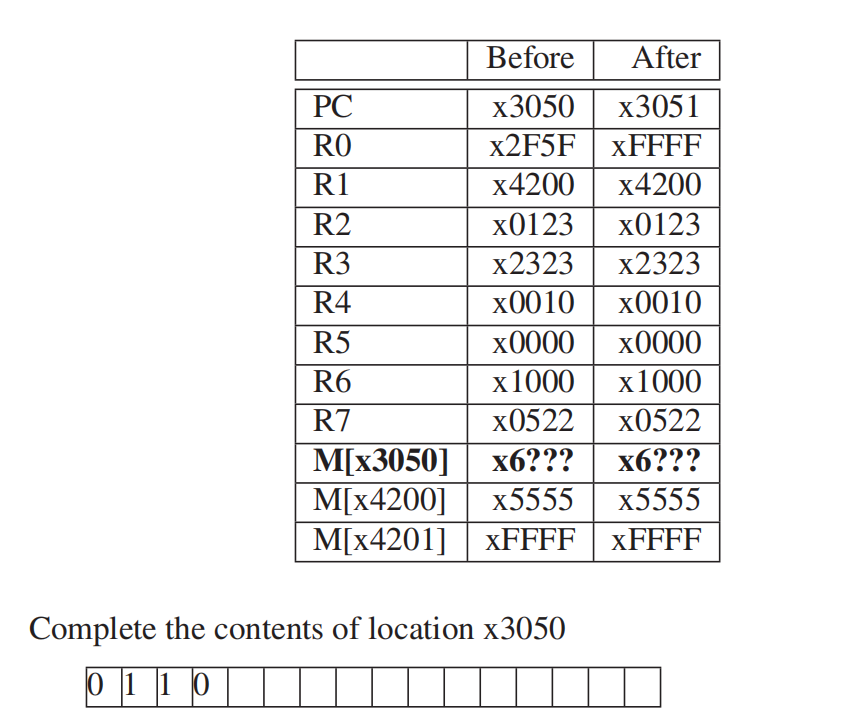
IR, PC, Reg file, SEXT unit connected to IR[8:0], ADDR2MUX, ADDR1MUX set to PC, and ADDER they connect to, MAXMUX and GateMARMUX implement LEA instruction.

6.24 A student is debugging his program. His program does not have access to memory locations x0000 to x2FFF. Why that is the case we will discuss before the end of the book. The term is “privileged memory” but not something for you to worry about today.He sets a breakpoint at x3050, and then starts executing the program. When the program stops, he examines the contents of several memory locations and registers, then hits single step. The simulator executes

one instruction and then stops. He again examines the contents of the memory locations and registers. They are as follows:

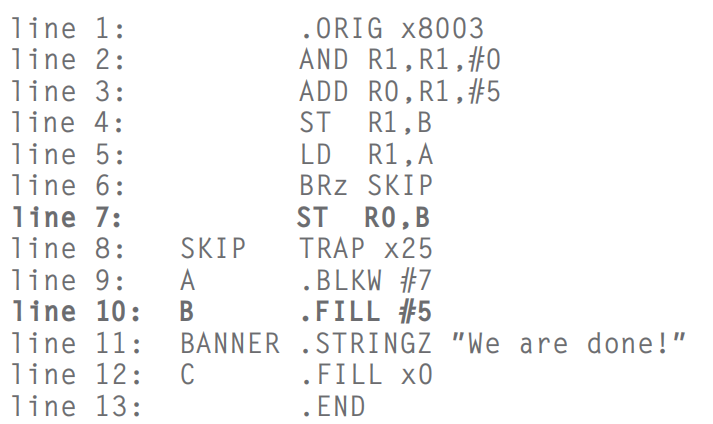


*Answer*:

LDR instructions adds the register and offset values to calculate the address and thus getting the value. The address is x4201, so the baseR is R1 and the offset is 1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |

7.32 Consider the following semi-nonsense assembly language program:



|  |  |
| --- | --- |
| SKIP | x8009 |
| A | x800A |
| B | x800B+6 |
| BANNED | x800C+6 |
| C | x800D+6 |

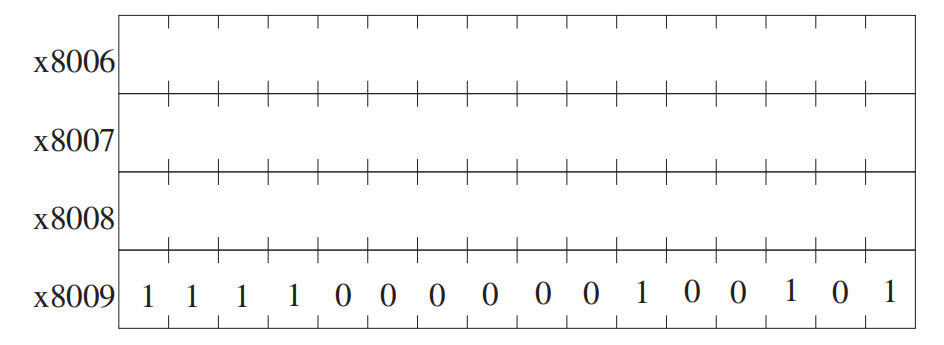
a.

A separate module will store a value in A before this program executes.

a. Construct the symbol table.

b. Show the result of assembly of lines 5 through 7 above. Note: the instruction at line 8 has already been assembled for you.

c. Note that two different things could cause location B to contain the value 5: the contents of line 7 or the contents of line 10.Explain the difference between line 7 causing the value 5 to be in location B and line 10 causing the value 5 to be in location B.



0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1

0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1

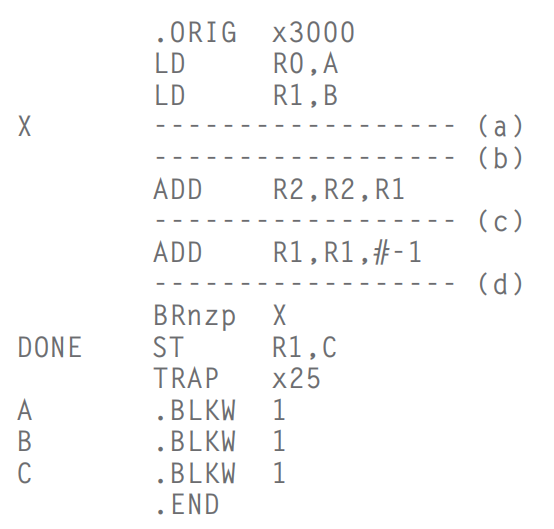
0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0

b.

c. Difference:

Line 7 store 5 in B at runtime, line 10 set 5 in B after loading

7.34 It is often useful to find the midpoint between two values. For this problem, assume A and B are both even numbers, and A is less than B. For example, if A = 2 and B = 8, the midpoint is 5. The following program finds the midpoint of two even numbers A and B by continually incrementing the smaller number and decrementing the larger number. You can assume that A and B have been loaded with values before this program starts execution. Your job: Insert the missing instructions.



*Answer:*

1. NOT R2,R0
2. ADD R2,R2,#1
3. BRz DONE
4. ADD R0,R0,#1