5.37 change LDI to STI

5.39

6.24

7.32 7.34

optional: 5.58 6.26 (no extra point) （选做，不计分）

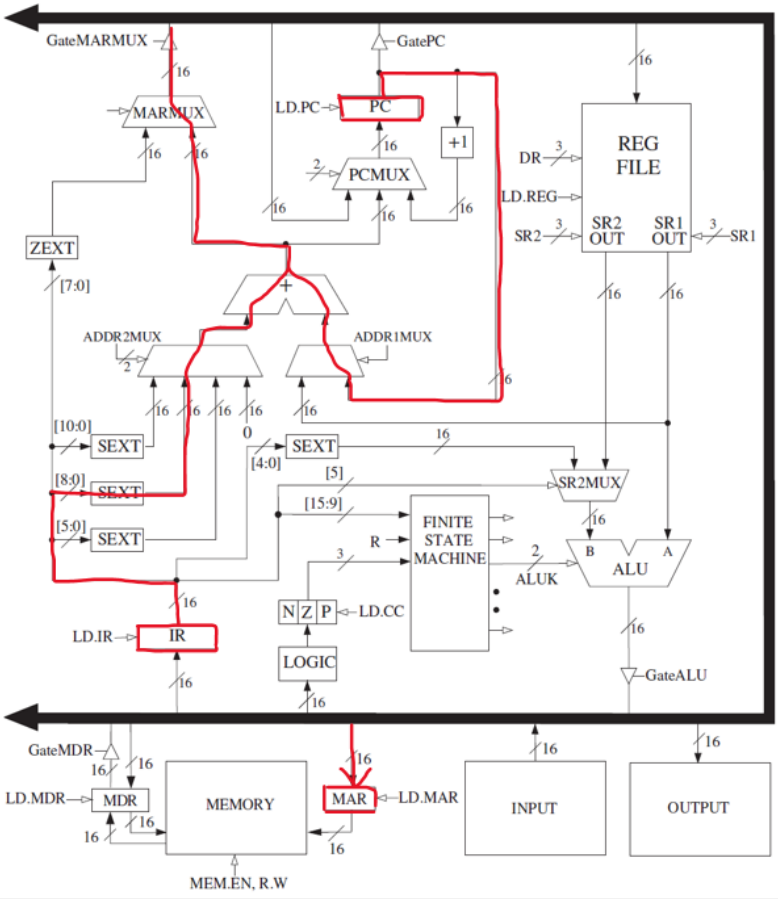
5.37

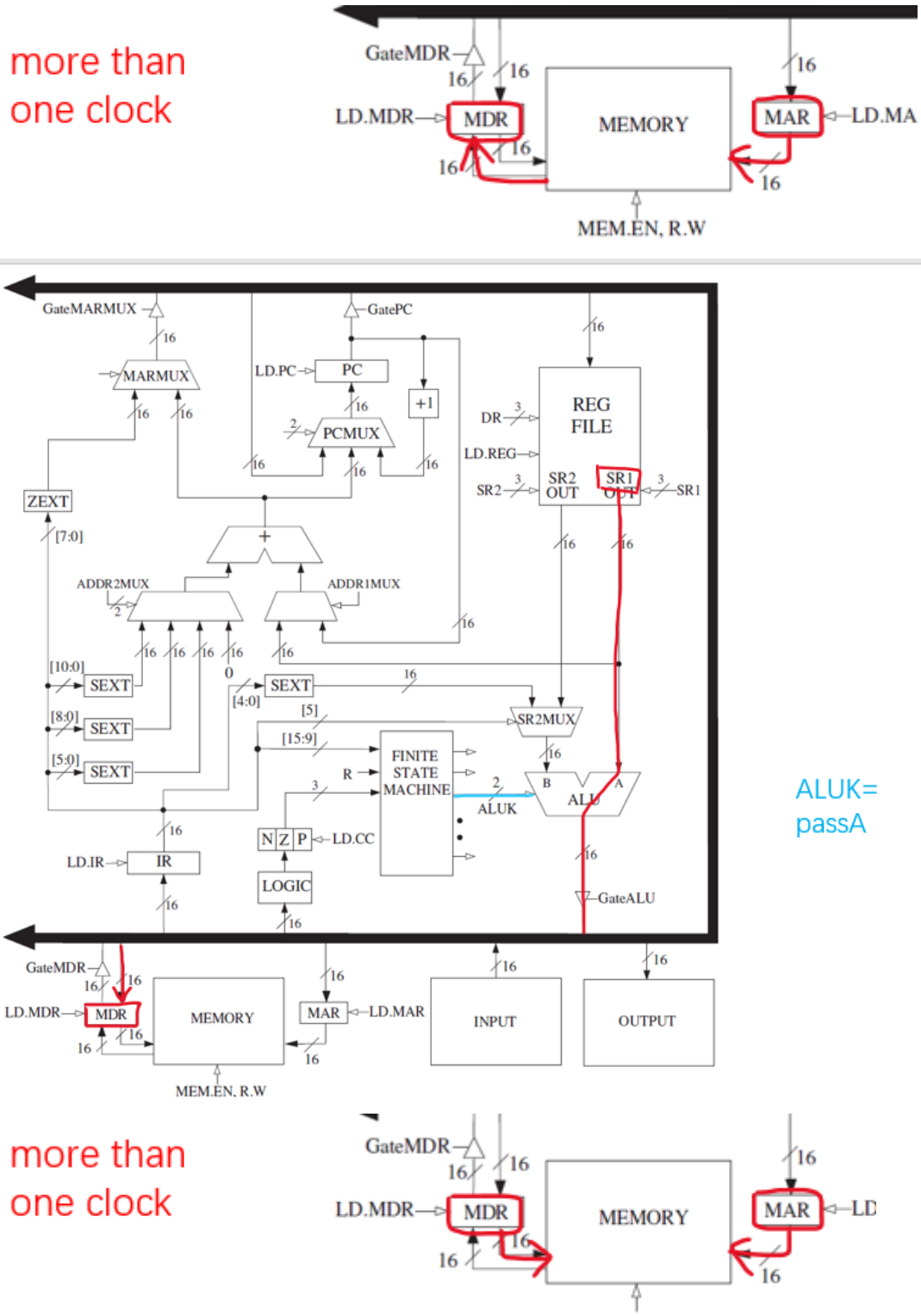
Using the overall data path in Figure 5.18, identify the elements that implement the STI instruction of Figure 5.8.

Instruction Fetch and Decode,then PC + SEXT imm[8:0] is computed then chosen by MARMUX ,next the Address will be sent to MAR , MEM fetch Data to MDR,result that the MEM fetch will be sent to MAR as the address to place DR's. So DR's data will pass SR1OUT ALUK=passA Gate ALu then to MDR.Finally , DR's data written in now MAR's address ,Condition Code will be set.STI finish

PC,MAR,MEM,MDR,ID,SEXT,ADDR1MUX ADDR2MUX,adder of them ,MARMUX,REGFILE ,ALU, LOFIC NZP ,FINITE STATE MACHINE ,Gate ALU Gate MARMUX GateMDR,Gate PC

STI
1 of 5



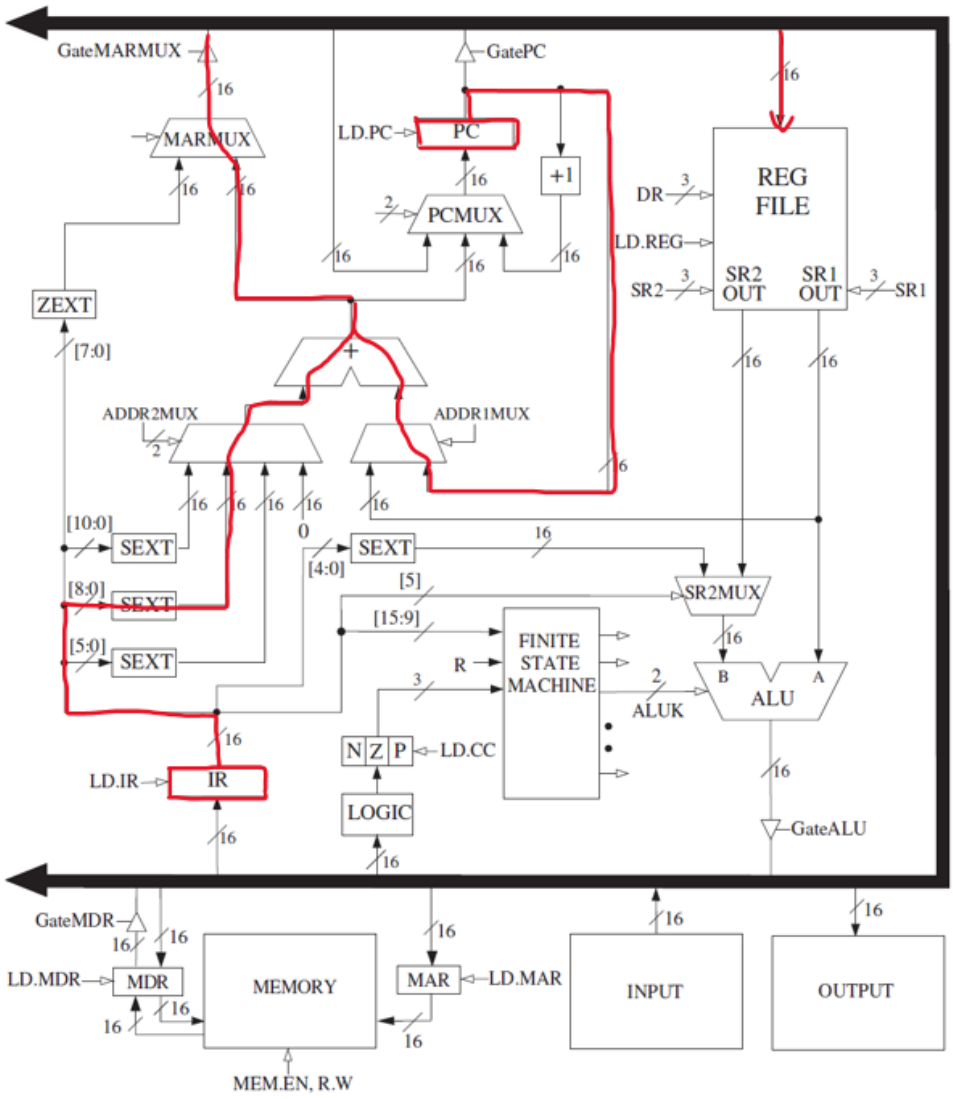


5.39

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

Inst Fetch and then Decode ,PC calculate ,write New PC in Reg,Finish

PC,IR,SEXT[8:0] ADRR1MUX ADDR2MUX the adder of them ,MARMUX,REG FILE,GateMARMUX,Gate PC



6.24

	Before	After
PC	x3050	x3051
R0	x2F5F	xFFFF
R1	x4200	x4200
R2	x0123	x0123
R3	x2323	x2323
R4	x0010	x0010
R5	x0000	x0000
R6	x1000	x1000
R7	x0522	x0522
M[x3050]	**x6???**	**x6???**
M[x4200]	x5555	x5555
M[x4201]	xFFFF	xFFFF

LDR DR,BaseR,offset

R0 was changed after that,and R0 = MEM[x4201]

x4201 = 4200 + 1

So it use R1 as the baseR with offset 1

LDR R0,R1,#1

0110 000 001 000001

7.32

Symbol table

symbol	Address
SKIP	x8009
A	x800A
B	x8011
BANNER	x801E
C	x801F

x8006 0010 0010 0000 0011

x8007 0000 0100 0000 0001

x8008 0011 0000 0000 1000

B .FILL #5 just store the value 5 in MEM before Program execute

and ST R0,B is a instruction , should after its execution then B is changed

7.34

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.

```
        .ORIG  x3000
        LD     R0,A
        LD     R1,B
X       ------------------ (a)
        ------------------ (b)
        ADD    R2,R2,R1
        ------------------ (c)
        ADD    R1,R1,#-1
        ------------------ (d)
        BRnzp  X
DONE    ST     R1,C
        TRAP   x25
A       .BLKW  1
B       .BLKW  1
C       .BLKW  1
        .END
```

while(A-B!=) A++ B--;

a.NOT R2,R0

b.ADD R2,R2,#1

c.BRz ,DONE

d.ADD R0,R0,#1

```
.orig X8003
LD R0,A
LD R1,B
X    NOT R2,R0
     ADD R2,R2,#1
     ADD R2,R2,R1
     BRz DONE
     ADD R1,R1,#-1
     ADD R0,R0,#1
     BR  X
DONE
     ST   R1,C
     TRAP x25

A    .FILL #2
B    .FILL #8
C    .BLKW 1
.end
```