

Initials:

8. [Bonus] Mystery Instruction [60 points]

A pesky engineer implemented a mystery instruction on the LC-3b. It is your job to determine what the instruction does. The mystery instruction is encoded as:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1010				DestR			SR1			0	0	0	SR2		

The instruction is only defined if the value of SR2 is greater than the value of SR1.

The modifications we make to the LC-3b datapath and the microsequencer are highlighted in the attached figures (see the four pages at the end of this question). We also provide the original LC-3b state diagram, in case you need it. (As a reminder, the selection logic for SR2MUX is determined internally based on the instruction.)

The additional control signals are

LD_TEMP1/1: NO, YES

LD_TEMP2/1: NO, YES

GateTEMP3/1: NO, LOAD

Reg_IN_MUX/1: BUS, Mystery2 – (Assume BUS is asserted if this signal is not specified)

Mystery_MUX/2: SR2MUX, PASS_1 (outputs value 1), PASS_0 (outputs value 0)

Additional Signals for ALUK: PASS_B (outputs the value from input B), SUB (A-B)

Also note that both of DRMUX and SR1MUX can now choose DR, SR1, and SR2

COND/4:

COND₀₀₀₀ ;Unconditional

COND₀₀₀₁ ;Memory Ready

COND₀₀₁₀ ;Branch

COND₀₀₁₁ ;Addressing mode

COND₀₁₀₀ ;Mystery 1

COND₁₀₀₀ ;Mystery 2

Initials:

The microcode for the instruction is given in the table below.

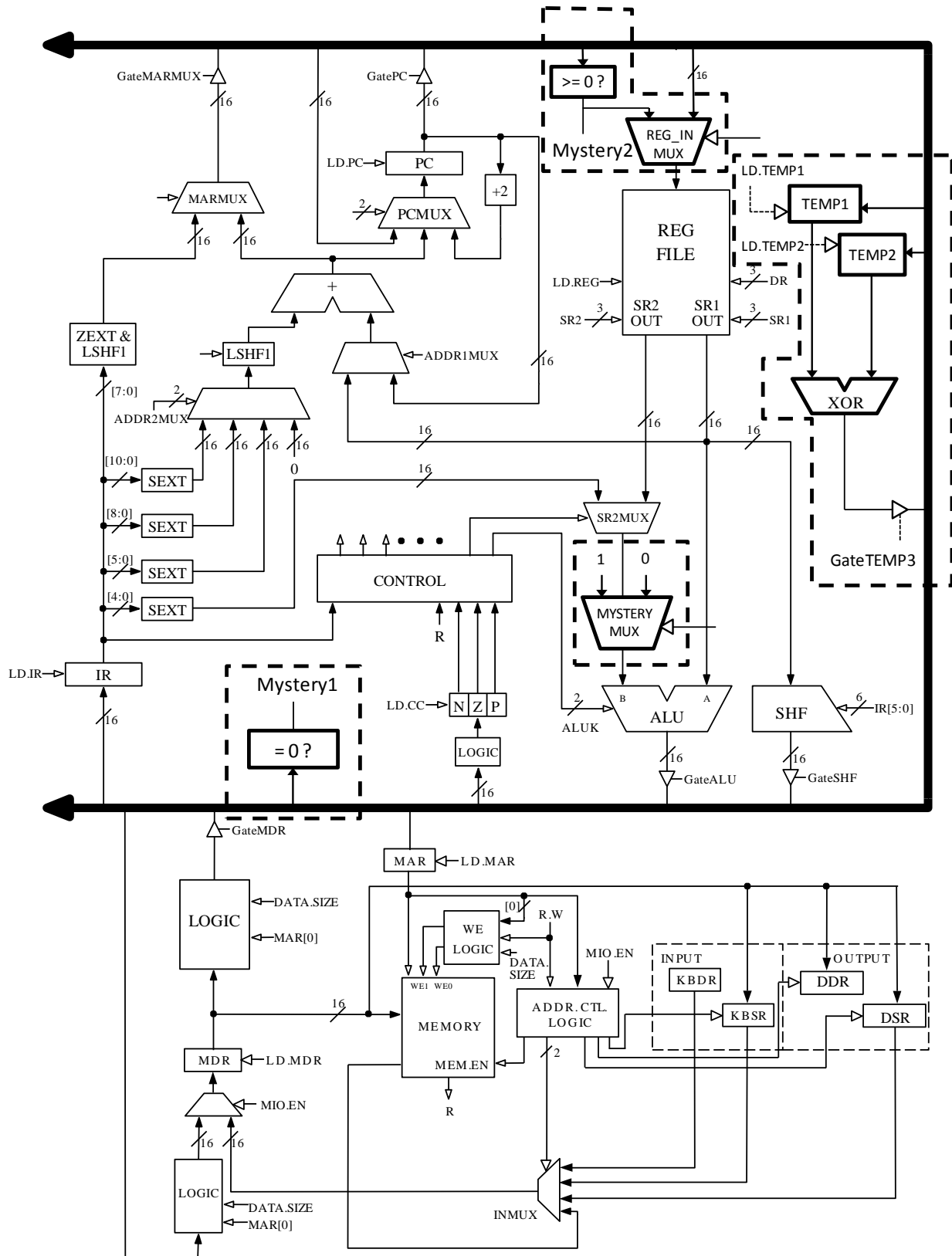
State	Cond	J	Asserted Signals
001010 (10)	COND ₀₀₀₀	001011	LD.REG, DRMUX = DR(IR [11:9]), GateALU, ALUK = PASS_B, MYSTERY_MUX = PASS_0
001011 (11)	COND ₀₀₀₀	110001	LD.MAR, SR1MUX = SR1(IR[8:6]), ADDR1MUX = SR1OUT, ADDR2MUX = 0, MARMUX = ADDER, GateMARMUX
110001 (49)	COND ₀₀₀₁	110001	LD.MDR, MIO.EN, DATA.SIZE=BYTE, R.W = R
110011 (51)	COND ₀₀₀₀	100100	GateMDR, LD.TEMP1, DATA.SIZE=BYTE
100100 (36)	COND ₀₀₀₀	100101	LD.MAR, SR1MUX = SR2(IR[2:0]), ADDR1MUX = SR1OUT, ADDR2MUX = 0, MARMUX = ADDER, GateMARMUX
100101 (37)	COND ₀₀₀₁	100101	LD.MDR, MIO.EN, DATA.SIZE=BYTE, R.W = R
100111 (39)	COND ₀₀₀₀	101000	GateMDR, LD.TEMP2, DATA.SIZE=BYTE
101000 (40)	COND ₀₁₀₀	010010	GateTEMP3
110010 (50)	COND ₀₀₀₀	101001	LD.REG, DRMUX = SR1(IR[8:6]), GateALU, ALUK = ADD, SR1MUX = SR1(IR[8:6]), MYSTERY_MUX = PASS_1
101001 (41)	COND ₀₀₀₀	101010	LD.REG, DRMUX = SR2(IR[2:0]), GateALU, ALUK = SUB, SR1MUX = SR2 (IR[2:0]), MYSTERY_MUX = PASS_1
101010 (42)	COND ₁₀₀₀	001011	LD.REG, DRMUX = DR (IR[11:9]), Reg_IN_MUX = MYSTERY2, GateALU, ALUK = SUB, SR1MUX = SR1(IR[8:6]), MYSTERY_MUX = SR2MUX

Describe what this instruction does. Show your work for partial credit.

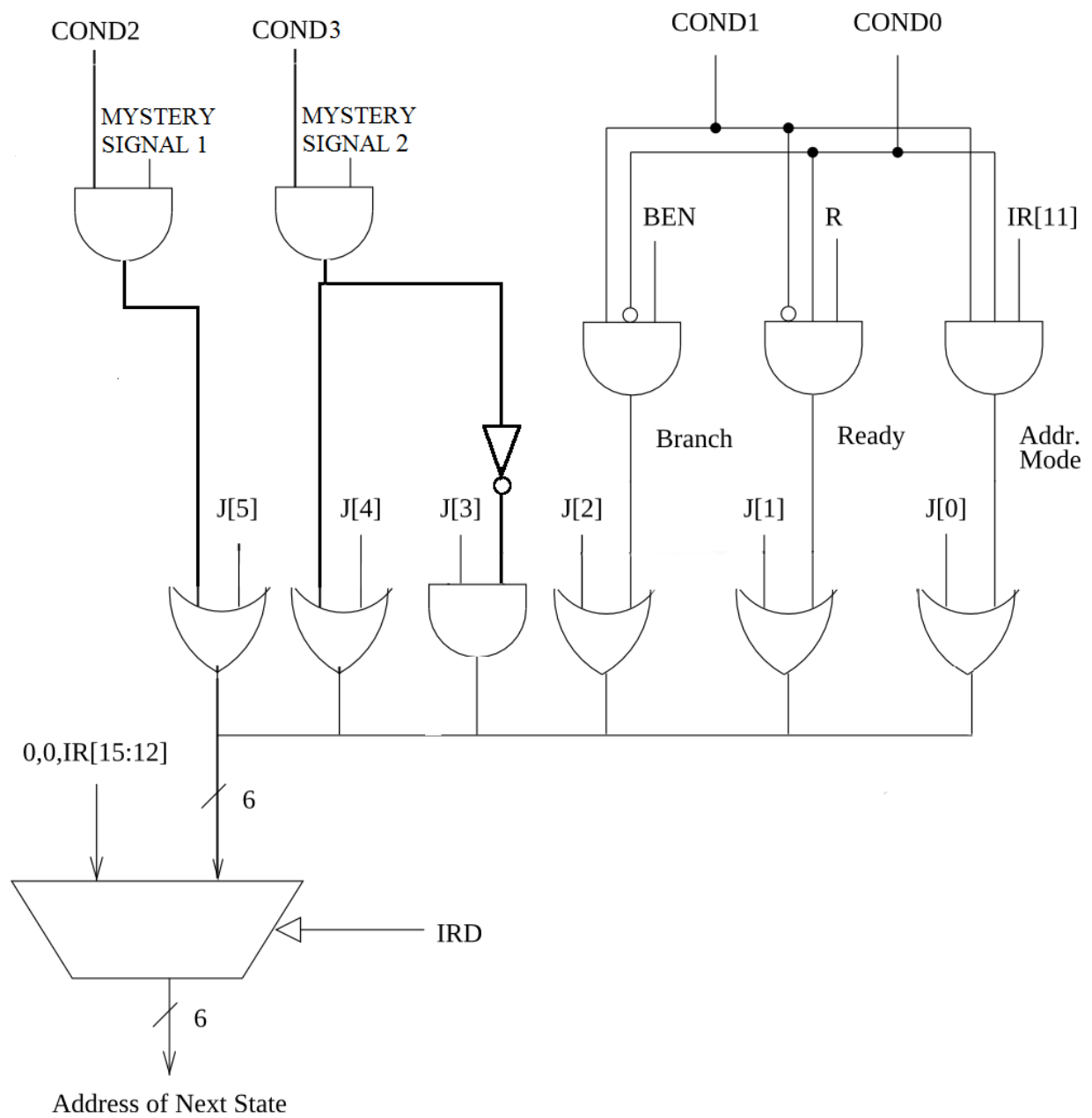
This instruction checks if the given string is a palindrome.

```
Code:
(char * sr1, *sr2;)
destR = 0;
while(sr1 < sr2){
    if (mem[sr1] != mem[sr2])
        return(fetch next instruction)
    sr1++;
    sr2--;
}
destR = 1;
return(fetch next instruction)
```

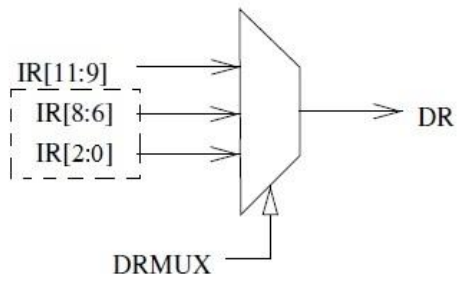
Initials: _____



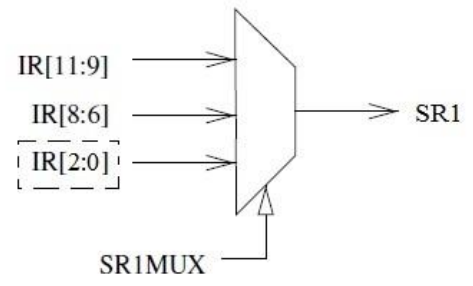
Initials: _____



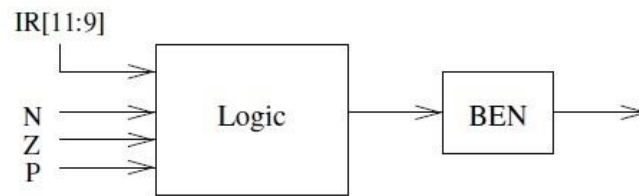
Initials:



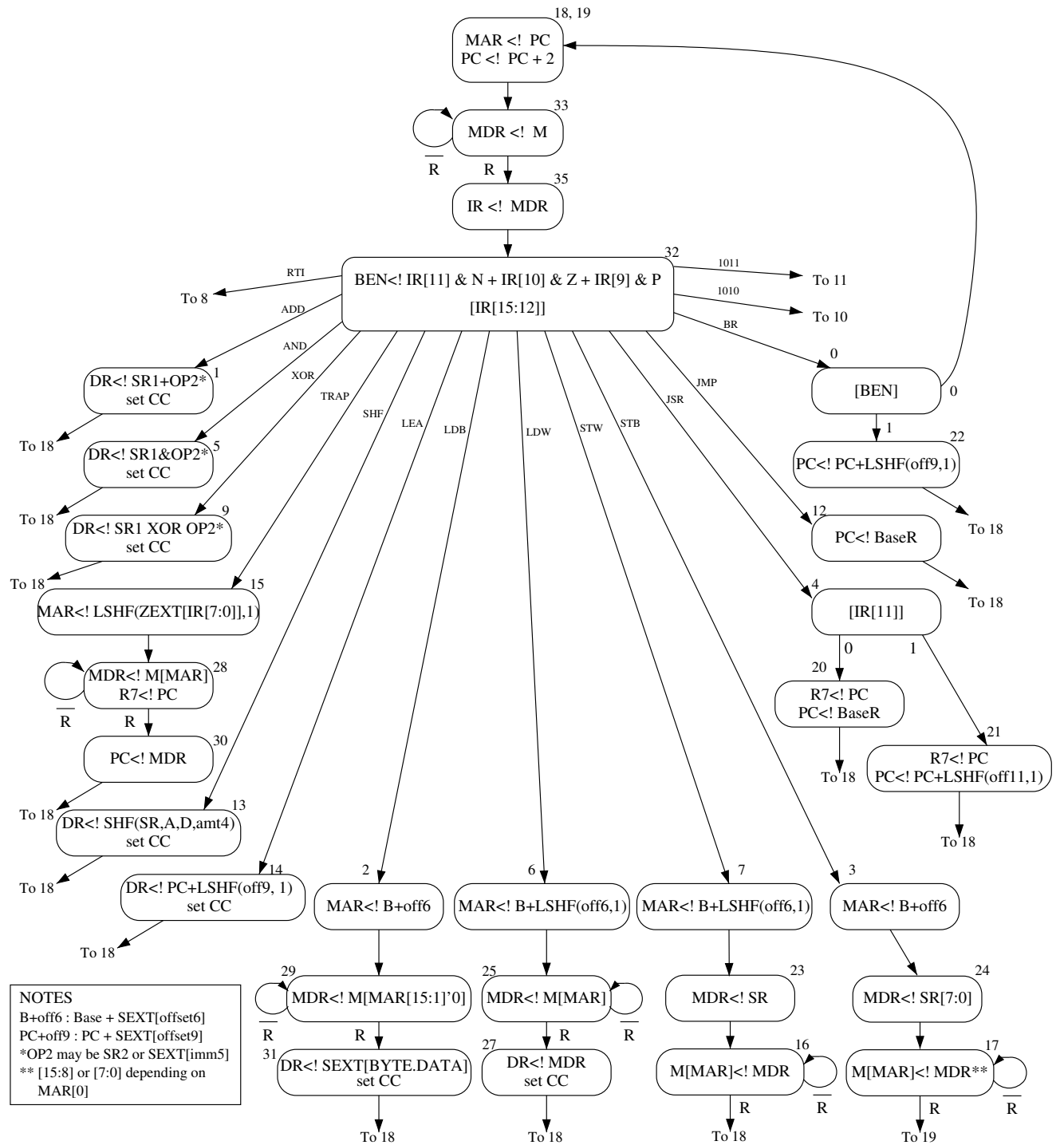
(a)



(b)



(c)



Initials: _____

Stratchpad

VLIW Instruction	ALU	MU	FPU
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			

Initials:

VLIW Instruction	ALU	MU	FPU
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			

Initials:

VLIW Instruction	ALU	MU	FPU
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			