Department of Electrical and Computer Engineering
The University of Texas at Austin

EE 460N Spring 2014
Aater Suleman, Instructor
Stephen Pruett, Jay Patel, Chirag Sakhuja, TAs
Exam 1
February 26, 2013

Name:____

Problem 1 (25 points):_______

Problem 2 (15 points):_______

Problem 3 (10 points):_______

Problem 4 (25 points):_______

Problem 5 (25 points):_______

Total (100 points):______________

You have 75 minutes to take this exam.
Note: Please be sure that your answers to all questions (and all supporting work that is required) are contained in the space provided.

Note: Please be sure your name is recorded on each sheet of the exam.

Please sign the following. I have not given nor received any unauthorized help on this exam.

Signature:____

GOOD LUCK!

Name:__________________________________

Problem 1 (25 points):

Part a (5 points): On a chip you designed, the two longest paths, P1 and P2, take 7 ns and 8 ns respectively. The reamining paths take only 2 ns each. For the same amount of engineering effort, you can *either* shrink P2 alone by 5 ns *or* shrink both P1 and P2 by 2 ns each. Which optimization will provide higher performance? Explain your answer.

Part b (5 points): An aggie hired by Little Computer Inc to design the memory circuit for LC-3b made an error: He connected MAR[3] to memory address pin 15 on the memory and connected MAR[15] to memory address pin 3 on the memory. Will this microarchitecture work?

Circle one: Yes / No

Specify the range of memory addresses, if any, for which this microarchitecture will fail.

Part c (5 points): Software can choose to ignore a particular interrupt by clearing the __________ bit corresponding to the interrupt.

Name:___________________________________

Problem 1 continued

Part d (5 points): When an exception is detected, the currently executing instruction is rolled back to its start before the exception service routine can execute. An aggie working at Little Computer Inc finds this rollback wasteful and wants to embark on a project to optimize exception handling. Will you allow her to continue this project or tell her to move on to the other pending tasks? Explain your answer.

Part e (5 points): Although a ________ cache is prone to cache consistency issues, such issues can be prevented if we can ensure that the sum of the number of ________ bits and ________ bits is less than or equal to the number of ________ bits.

Name:___________________________________

Problem 2 (15 points):

Answer parts (a-d) for a PIPT, write-back cache. Assume:

- Physcial address is 16-bits.
- The tag store entry is 1 byte.
- There are no unused bits in the tag store entry.
- Cache block size is 8 bytes.
- The cache uses a tree replacement policy that requires 7 bits.
  Hint: In the tree replacement policy, the replacement bits are not a part of the tag store entry.

Part a (2 points): What is the cache's associativity?

Part b (3 points): How many bits of the physical address are used as the index?

Part c (5 points): How big is the data store? Answer in **bytes**.

Part d (5 points): How big is the tag store? Answer in **bits**.

Name:____________________________________

Problem 3 (10 points):

We show two slightly different memory systems below. Please refer to them when answering the questions on the next page.
Note: The CE signal indicates whether a chip should be enabled for an access or not.

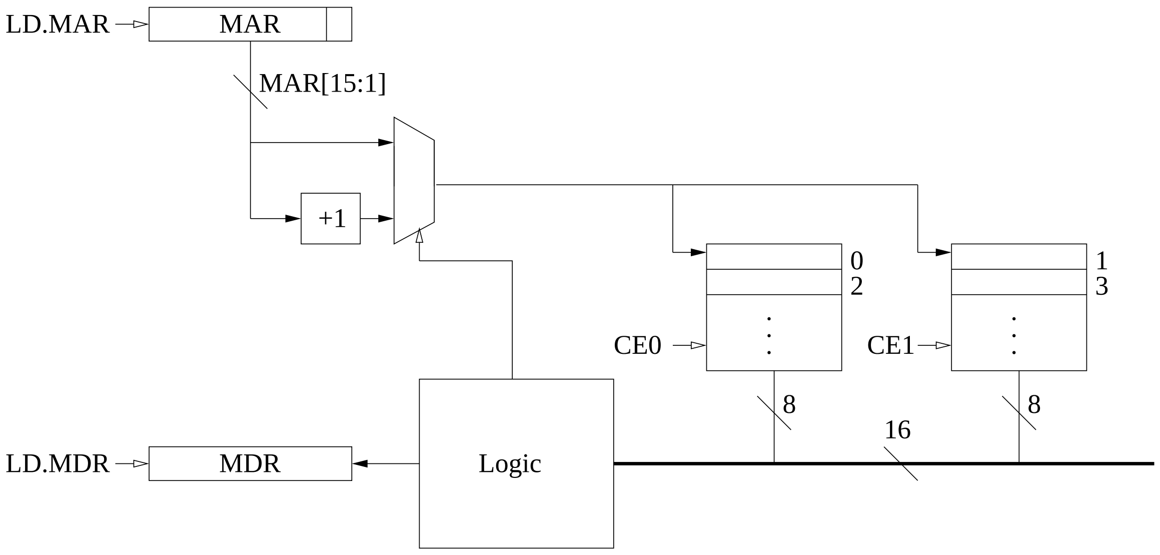


Figure 1: Memory system M1

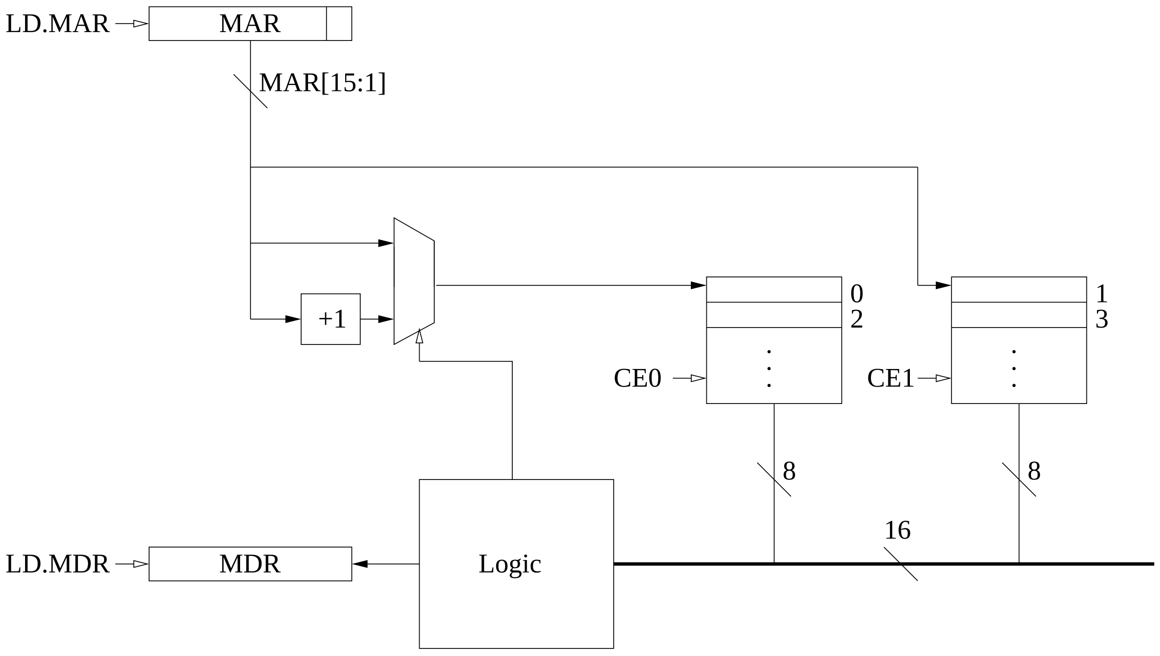


Figure 2: Memory system M2

Name:____________________________________

Problem 3 continued:

Assume:

- The logic blocks contain all the necessary gates and latches to complete the memory accesses.
- The logic blocks are implemented to maximize performance.
- Each memory chip takes 100 cycles to access.
- The system only performs reads (and no write operations).

Part a (5 points): How many cycles will it take to execute the data load (not instruction fetch) of LDW R0, x3000 on M1 and M2?

Cycles with M1:

Explain in less than 20 words.

Cycles with M2:

Explain in less than 20 words.

Part b (5 points): How many cycles will it take to execute the data load (not instruction fetch) of LDW R0, x3001 on M1 and M2?

Cycles with M1:

Explain in less than 20 words.

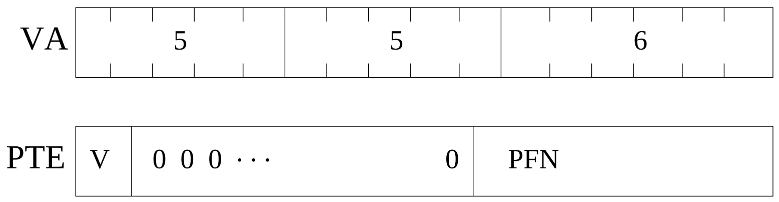
Cycles with M2:

Explain in less than 20 words.

Name:__________________________________

Problem 4 (25 points):

We have added an 8-entry, 2-way write-through PIPT cache and an x86 style 2-level virtual-to-physical translation scheme to the LC-3b. Each virtual address is 16 bits. The most significant 5-bits, VA[15:11], are used to access the system page table, the next 5 bits, VA[10:6], are used to access the process page table, and least significant 6-bits, VA[5:0], are the the byte in page bits. The PTE only contains a valid bit as its most significant bit and the PFN as its least significnat bits. All remaining bits are reserved and always set to zeros.



Notes:
1. Physical address is 10 bits.
2. The cache implements a perfect LRU replacement policy.
3. The cache block size is 4 bytes.
4. Instructions and data share the same cache.
5. The cache is initially empty.
6. The memory accesses to system page table, the user page table, and user pages are **all** cached.
7. No page faults happen during the execution of this instruction.
8. Only for the sake of this problem, assume that the LC-3b is **big endian** (most significant byte is in least significant address) and does not permit unaligned accesses.

The PC is loaded with address x3184 and one instruction is run to completion. The figure below shows the contents of the cache after the end of this instruction. **Your job:** Use this data to answer the questions on the next next page.

LRU	V0	Tag0	V1	Tag1	Data0				Data1			
0	0	101100	0	010010	x80	x70	xAE	xFD	x80	x06	x80	x77
1	1	011100	0	011100	x60	x43	x59	xAE	xFD	xE9	x46	x57
0	0	011000	0	110110	x53	x74	x65	x70	x68	x65	x6E	x21
0	1	111100	1	110000	x80	x0C	x00	x0F	x80	x07	x80	x06

*Cache after execution of first instruction
*The order of bytes in the data store is byte0 (00), byte1 (01), byte2 (10), byte3 (11).
*V0, Tag0, and Data0 correspond to Way 0 and V0, Tag1, and Data1 correspond to Way 1.
*If the LRU bit is 0, the data in Way 0 will be evicted next.

PROBLEM IS CONTINUED ON THE NEXT PAGE!!!

Name:____________________________________

Problem 4 continued

Part a (2 points): How big is each page (in bytes)?

Answer: $\boxed{}$

Part b (3 points): What is the size of a PTE (in bytes)?

Answer: $\boxed{}$

Part c (5 points): What is the value of SBR?

Answer: $\boxed{}$

Part d (10 points): Fill in the table in the order of physical memory accesses. Note that some rows in this table may be left blank.

Physical Address	Data	Explanation of data

Part e (5 points): What is the value of R0 and R1? (There are two possible answers to this question and either one will get full credit)

R0: $\boxed{}$

R1: $\boxed{}$

Name:____________________________________

Problem 5 (25 points):

We are adding a new instruction, ADD32, to the LC-3b ISA. ADD32 adds two 32-bit values. One of the operands is loaded from memory and the other operand is obtained by concatenating two registers. The final answer is stored to memory. ADD32 works as follows:

Assembler Format

ADD32 DR, SR1, SR2H, SR2L

Notation

SR1: Pointer to a source; one of R0..R7 which species the *address* of the memory location from which the first 32 bit operand is obtained.
SR2H: Source register; one of R0..R7 which specifies the register from which the high 16-bits of the second operand are obtained
SR2L: Source register; one of R0..R7 which specifies the register from which the low 16-bits of the second operand are obtained
DR: Pointer to destination; one of R0..R7 which species the *address* of the memory location where the result of the 32-bit add must be written to.

Encodings

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	DR			SR1			SR2H			SR2L		

Operation

MEM[DR+2]@MEM[DR] = MEM[SR1+2]@MEM[SR1] + SR2H@SR2L;
setCC();

*The @ symbolizes bit concatenation.

PROBLEM IS CONTINUED ON THE NEXT PAGE!!!

Name:____________________________________

Problem 5 continued:

Your job Implement ADD32 by modifying the datapath and the state diagram.

Part a (12 points): Complete the state diagram to implement ADD32 by filling in the shaded bubbles and state assignment for the last state.
HINT: You may find it helpful to look at part b while solving part a.

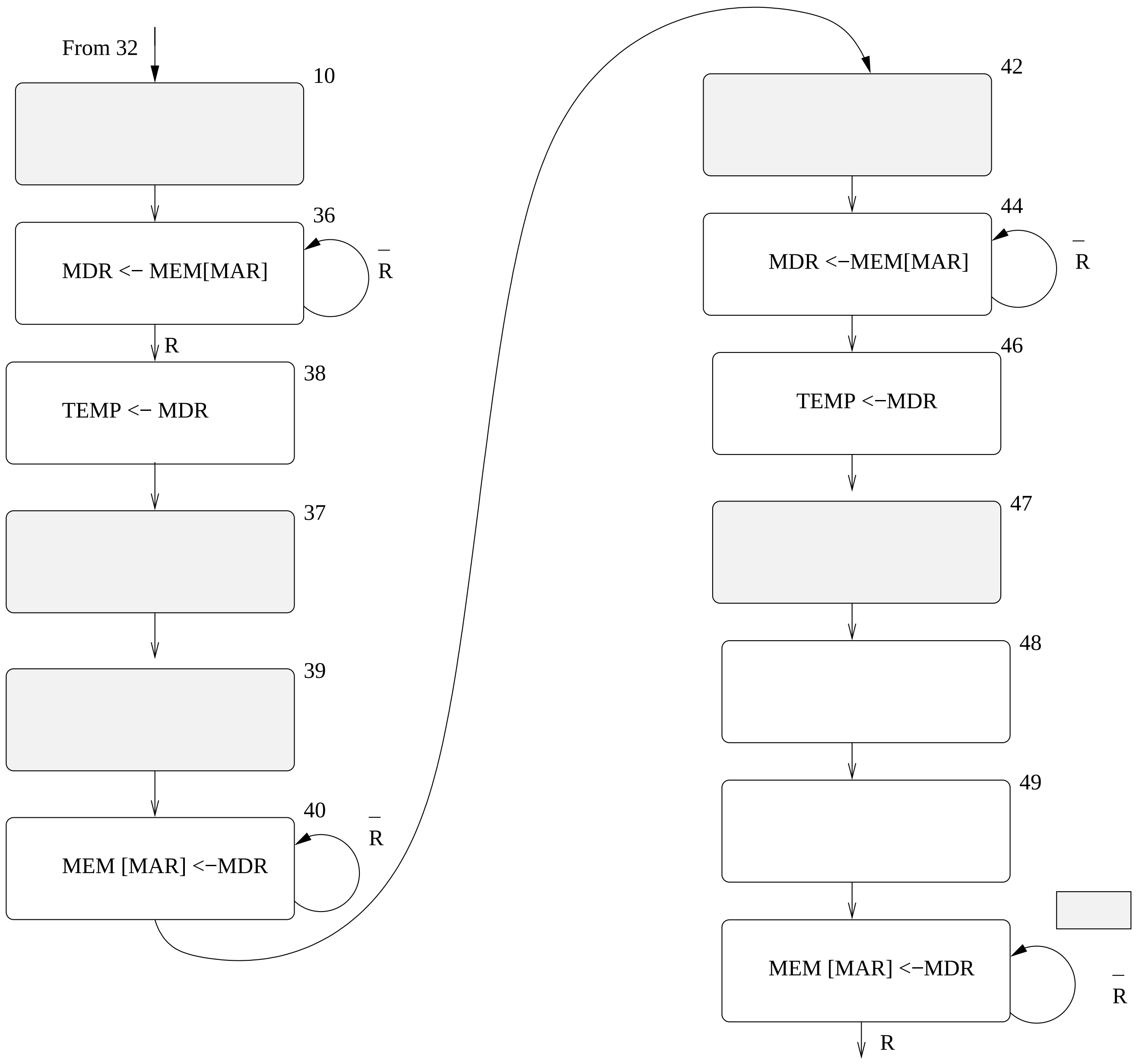
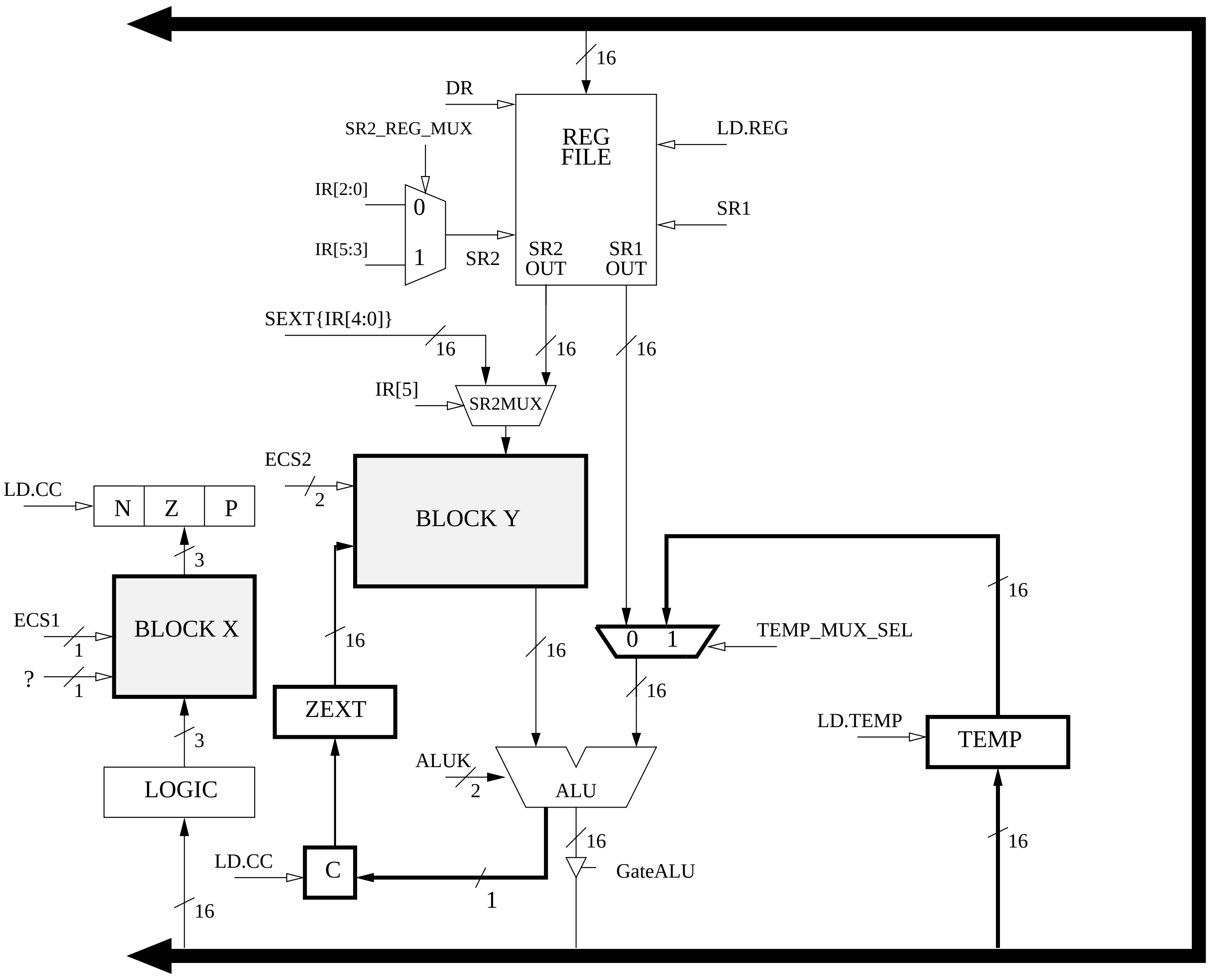


Figure 3: State diagram for the LC-3b

Name:__________________________________

Problem 5 continued:

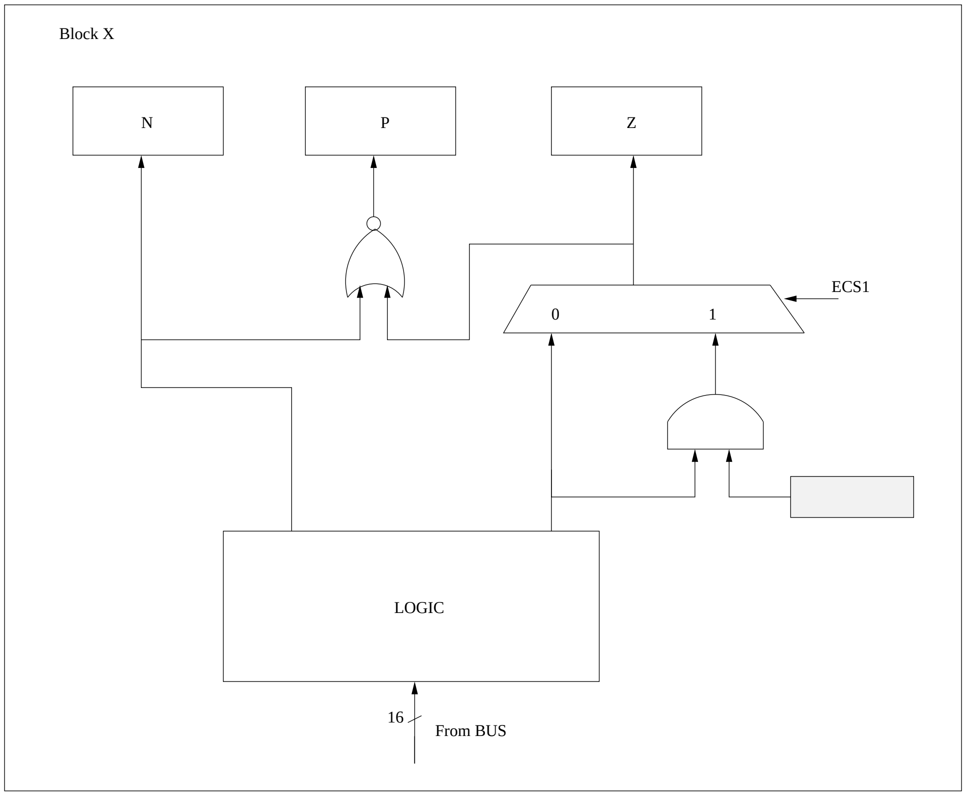
Part b (13 points): Below is the relevant portion of the LC-3b datapath. Two new blocks X and Y have been added. Addtionally, the ALU has been modified to output a carry bit. The carry bit is saved in the "C" condition code register. **Note:** the ALU is still only 16-bits wide.

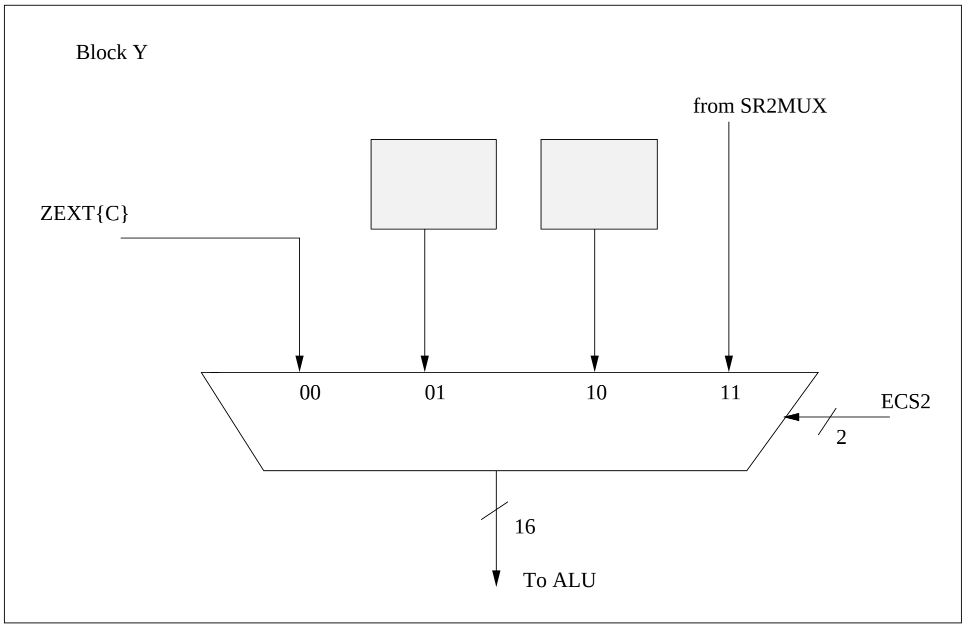


Name:__________________________________

Problem 5, Part b continued:

Complete the logic for blocks X and Y below by filling in *only* the three shaded boxes.





	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD$^+$	0001				DR			SR1			0	00		SR2		
ADD$^+$	0001				DR			SR1			1	imm5				
AND$^+$	0101				DR			SR1			0	00		SR2		
AND$^+$	0101				DR			SR1			1	imm5				
BR	0000				n	z	p	PCoffset9								
JMP	1100				000			BaseR			000000					
JSR	0100				1	PCoffset11										
JSRR	0100				0	00		BaseR			000000					
LDB$^+$	0010				DR			BaseR			boffset6					
LDW$^+$	0110				DR			BaseR			offset6					
LEA$^+$	1110				DR			PCoffset9								
NOT$^+$	1001				DR			SR			1	11111				
RET	1100				000			111			000000					
RTI	1000				000000000000											
LSHF$^+$	1101				DR			SR			0	0	amount4			
RSHFL$^+$	1101				DR			SR			0	1	amount4			
RSHFA$^+$	1101				DR			SR			1	1	amount4			
STB	0011				SR			BaseR			boffset6					
STW	0111				SR			BaseR			offset6					
TRAP	1111				0000				trapvect8							
XOR$^+$	1001				DR			SR1			0	00		SR2		
XOR$^+$	1001				DR			SR			1	imm5				
not used	1010															
not used	1011															

Figure 4: LC-3b Instruction Encodings

Table 1: Data path control signals

Signal Name	Signal Values	
LD.MAR/1:	NO(0), LOAD(1)	
LD.MDR/1:	NO(0), LOAD(1)	
LD.IR/1:	NO(0), LOAD(1)	
LD.BEN/1:	NO(0), LOAD(1)	
LD.REG/1:	NO(0), LOAD(1)	
LD.CC/1:	NO(0), LOAD(1)	
LD.PC/1:	NO(0), LOAD(1)	
GatePC/1:	NO(0), YES(1)	
GateMDR/1:	NO(0), YES(1)	
GateALU/1:	NO(0), YES(1)	
GateMARMUX/1:	NO(0), YES(1)	
GateSHF/1:	NO(0), YES(1)	
PCMUX/2:	PC+2(0)	;select pc+2
	BUS(1)	;select value from bus
	ADDER(2)	;select output of address adder
DRMUX/1:	11.9(0)	;destination IR[11:9]
	R7(1)	;destination R7
SR1MUX/1:	11.9(0)	;source IR[11:9]
	8.6(1)	;source IR[8:6]
ADDR1MUX/1:	PC(0), BaseR(1)	
ADDR2MUX/2:	ZERO(0)	;select the value zero
	offset6(1)	;select SEXT[IR[5:0]]
	PCoffset9(2)	;select SEXT[IR[8:0]]
	PCoffset11(3)	;select SEXT[IR[10:0]]
MARMUX/1:	7.0(0)	;select LSHF(ZEXT[IR[7:0]],1)
	ADDER(1)	;select output of address adder
ALUK/2:	ADD(0), AND(1), XOR(2), PASSA(3)	
MIO.EN/1:	NO(0), YES(1)	
R.W/1:	RD(0), WR(1)	
DATA.SIZE/1:	BYTE(0), WORD(1)	
LSHF1/1:	NO(0), YES(1)	

Table 2: Microsequencer control signals

Signal Name	Signal Values	
J/6:		
COND/2:	$COND_0$	;Unconditional
	$COND_1$	;Memory Ready
	$COND_2$	;Branch
	$COND_3$	;Addressing Mode
IRD/1:	NO, YES	

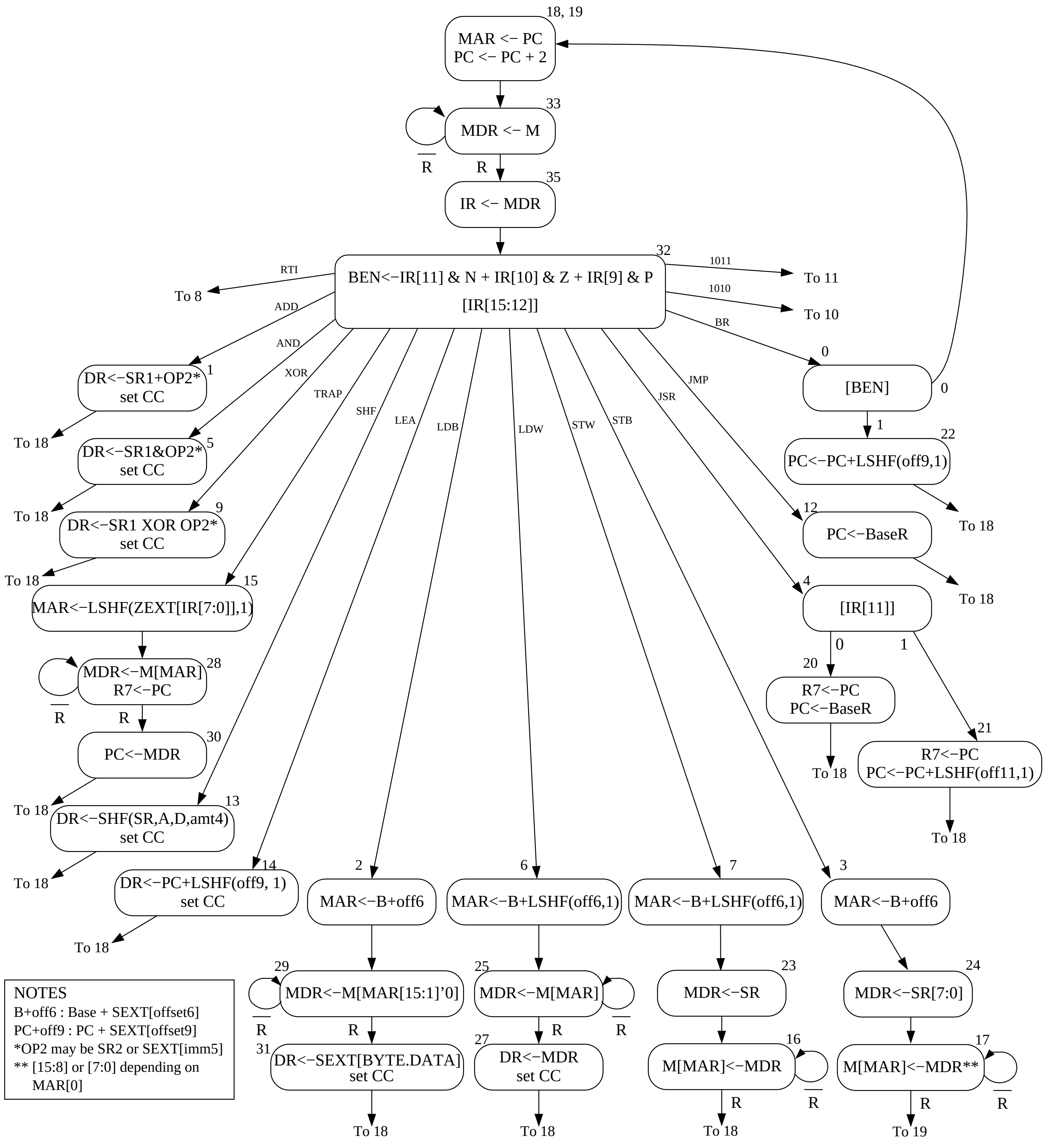


Figure 5: A state machine for the LC-3b

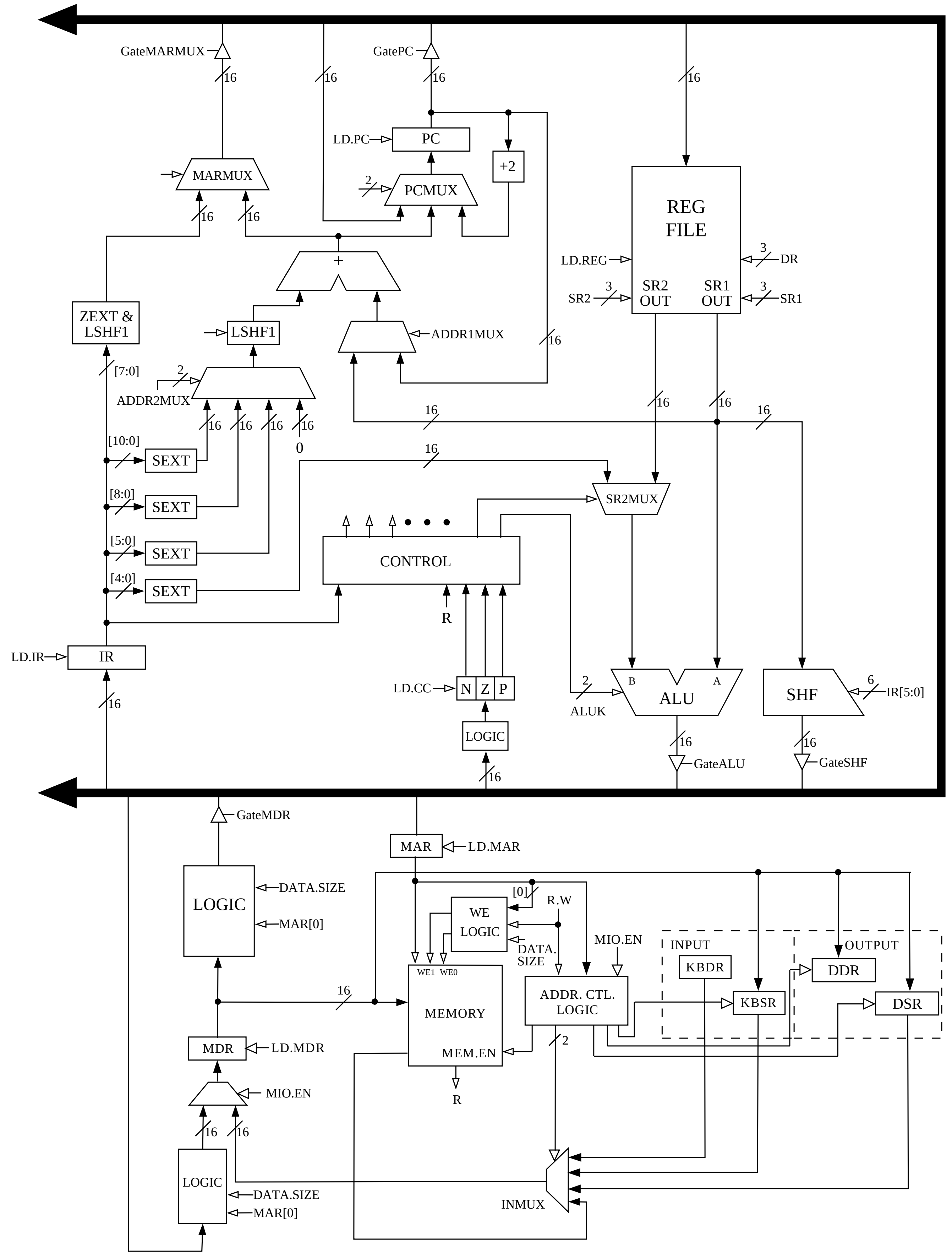


Figure 6: The LC-3b data path

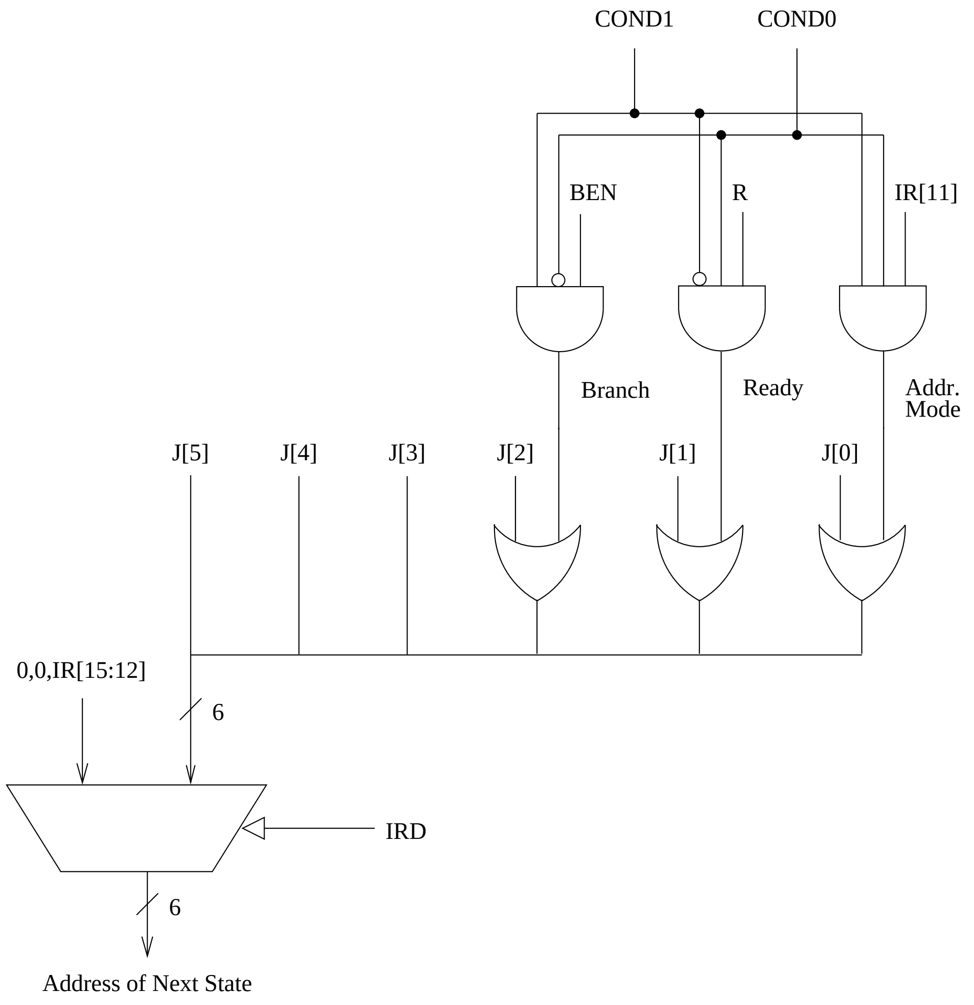


Figure 7: The microsequencer of the LC-3b base machine

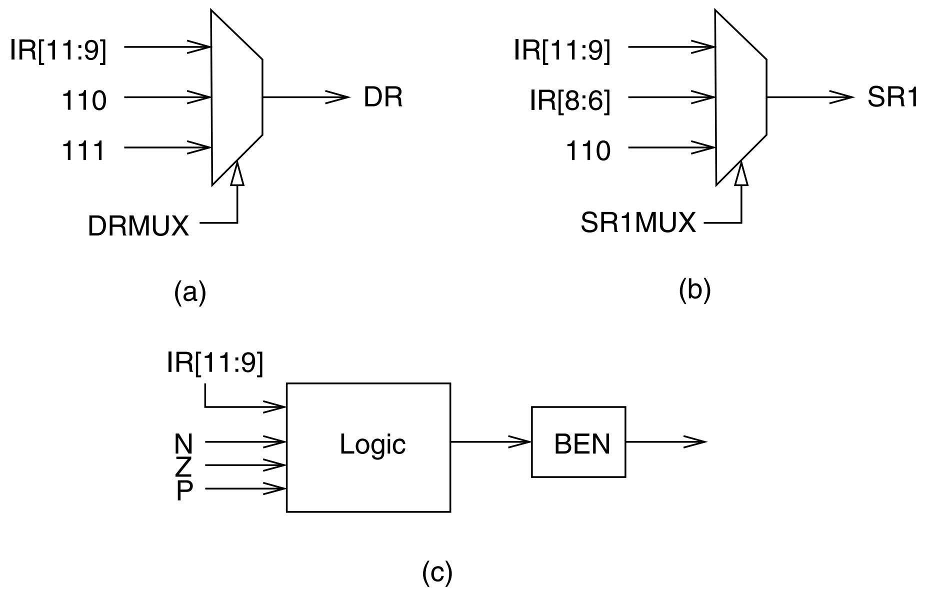


Figure 8: The microsequencer of the LC-3b base machine