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

EE 360N, Spring 2003
Yale Patt, Instructor
Hyesoon Kim, Onur Mutlu, Moinuddin Qureshi, Santhosh Srinath, TAs
Exam 1, March 5, 2003

Name:__________________________

Problem 1 (25 points):_______

Problem 2 (20 points):_______

Problem 3 (20 points):_______

Problem 4 (15 points):_______

Problem 5 (20 points):_______

Problem 6 (no points):_______

Total (100 points):______________

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.

GOOD LUCK!

Name:________________________

Problem 1 (25 points):

Part a (5 points): The main element of storage required to store a single bit of information depends on whether we are talking about DRAM cells or SRAM cells.

For DRAM cells it is:

For SRAM cells it is:

Part b (5 points):

The primary purpose of segmentation is:

The primary purpose of paging is:

Part c (5 points): The reference bit in a PTE is used for what purpose?

The similar function is performed by what bit or bits in a cache's tag store entry?

Part d (5 points): We note that condition codes get set by the three load instructions and the four operates in the last cycle of the instruction cycle when they load the destination register. So, someone suggested we get rid of the LD.CC control signal and use instead the LD.REG signal to load condition codes, If we did this, without changing anything else, would the LC-3b work correctly? Why/why not?

Part e (5 points): A cache has the block size equal to the word length. What property of program behavior, which usually contributes to higher performance if we use a cache, does not help the performance if we use THIS cache?

Name:__________________________

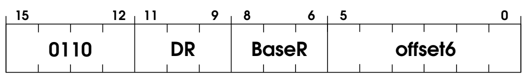
Problem 2 (20 points):

Little Computer Inc. has decided to support unaligned accesses in the LDW instruction. The specification of the LDW instruction is as follows:

Assembler Format

LDW DR, BaseR, offset6

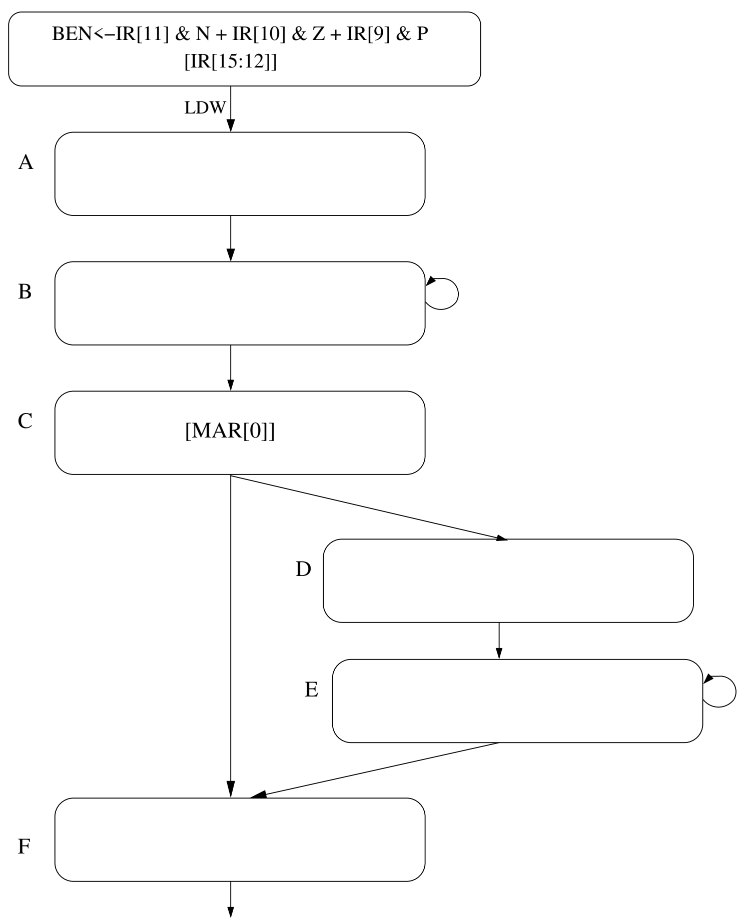
Encoding



Operation

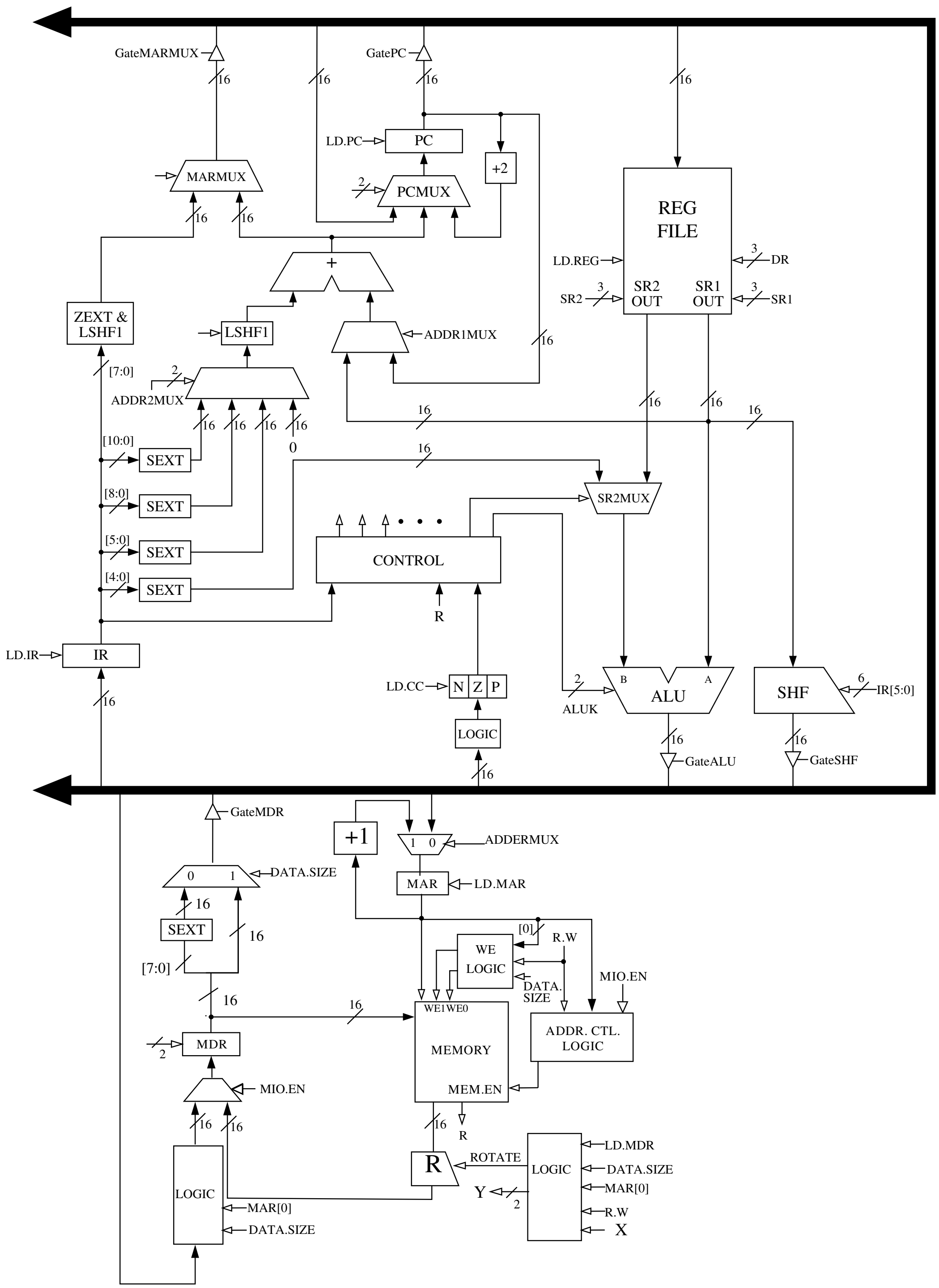
DR = MEM[BaseR+SEXT(offset6)];
setcc(DR);

Part a. We show below the states used to implement the LDW instruction. Using the notation of the LC-3b state diagram, describe inside each "bubble" what happens in each state. We have already given you what happens in state C. In this state, MAR[0] is tested and next state is determined based on the value of MAR[0]. **The modified datapath is shown on the next page.**



Name:________________________

Problem 2 continued:



Name:________________________

Problem 2 continued:

Part b. The modified datapath shown on the previous page contains a logic block whose inputs are LD.MDR, DATA.SIZE, R.W, MAR[0], and X. The outputs of this logic block are the two-bit signal Y and a 1-bit ROTATE signal. Identify precisely in the boxes below the signals X, Y[0], and Y[1]. Four or five words should be more than enough for each signal. Identify the specific value for X in each input combination of the truth table. Complete the output columns of the truth table.

Signal X:

Signal Y[0]:

Signal Y[1]:

R.W	DATA.SIZE	LD.MDR	MAR[0]	X	Y[1]	Y[0]	ROTATE
READ	BYTE	NO	0				
READ	BYTE	NO	1				
READ	BYTE	LOAD	0				
READ	BYTE	LOAD	1				
READ	WORD	NO	0				
READ	WORD	NO	1				
READ	WORD	LOAD	0				
READ	WORD	LOAD	1				

Name:________________________

Problem 2 continued:

Part c. The processing in each state (A, B, C, D, E, F) is controlled by asserting or negating each control signal. Enter a 1 or a 0 as appropriate for the microinstructions corresponding to states A, B, D, E, F. The control signals for state C are already filled in for you.

	LD.MAR	LD.MDR	LD.IR	LD.BEN	LD.REG	LD.CC	LD.PC	GatePC	GateMDR	GateALU	GateMARMUX	GateSHF	PCMUX		DRMUX	SR1MUX	ADDR1MUX	ADDR2MUX		MARMUX	ALUK		MIO.EN	R.W	DATA.SIZE	LSHF1	ADDERMUX	X
													PC+2, BUS, ADDR 00, 01, 10		IR[11:9](0), R7(1)	IR[11:9](0), IR[8:6](1)	PC(0), BaseR(1)	ZERO, offset6, PCoffset9, PCoffset11 00, 01, 10, 11		LSHF(ZEXT[IR[7:0],1)(0), adder(1)	ADD, AND, XOR, PASSA 00, 01, 10, 11			RD(0), WR(1)	BYTE(0), WORD(1)		BUS(0), MAR+1(1)	
state A																												
state B																												
state C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
state D																												
state E																												
state F																												

Name:________________________

Problem 3 (20 points):

We hired a new circuit designer from A&M to help us implement the LC-3b, and he loaded the microinstructions into the wrong control store locations, as noted on the state machine shown in Figure 1. No problem, we can fix it with some quick fixes to the microsequencer. Figure 2 identifies the "new" microsequencer.

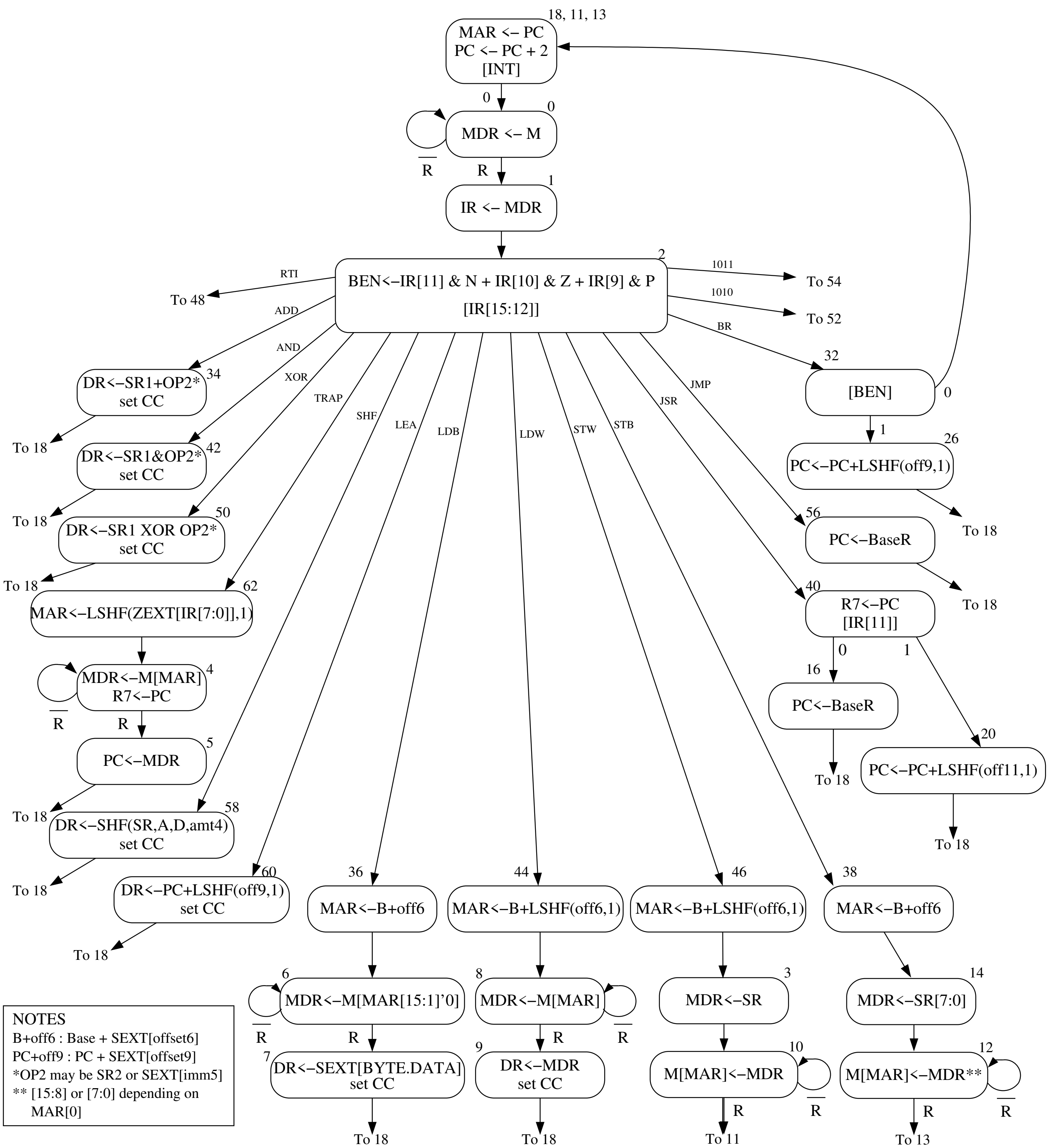


Figure 1

Name:__________________________

Problem 3 continued:

Cond[1:0]	Signal Values
00	Unconditional
01	Branch
10	Ready
11	Addressing Mode

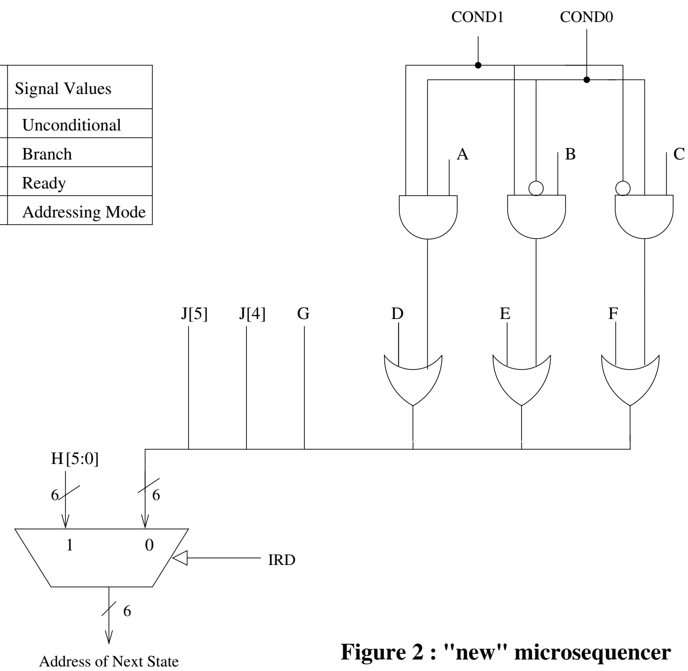


Figure 2 : "new" microsequencer

Part a. Identify the signals A through G in the boxes provided below. A few words at most should suffice for each box.

A	
B	
C	
D	
E	
F	
G	

Part b. Identify separately each bit of H[5:0].

H[5]	H[4]	H[3]	H[2]	H[1]	H[0]

Part c. In which state / states is IRD asserted?

Name:__________________________

Problem 4 (15 points):

An LC-3b system ships with a two-way set associative, write back cache with perfect LRU replacement. The tag store requires a total of 4352 bits of storage. What is the block size of the cache? This is one problem where you really do need to show all your work on the paper.

Hint: $4352 = 2^{12} + 2^{8}$.

Name:__________________________

Problem 5 (20 points):

A machine with 64KB, byte addressable virtual memory and 4KB physical memory has two-level virtual address translation similar to the VAX. The page size of this machine is 256 bytes. Virtual address space is partitioned into the P0 space, P1 space, system space and reserved space. The space a virtual address belongs to is specified by the most significant two bits of the virtual address, with 00 indicating P0 space, 01 indicating P1 space, and 10 indicating system space. Assume that the PTE is 32 bits and of the format 10000000..000PFN.

For a single load instruction the physical memory was accessed three times. The first access was at location x108 and the value read from that location (x108, x109, x10A, x10B) was x80000004. Hint: What does this value mean?

The second access was at location x45C and the third access was at location x942.

If SBR = x100, P0BR = x8250 and P1BR = x8350,

Part a. What is the virtual address corresponding to physical address x45C ?

VA = x [ ]

Part b. What is 32 bit value read from location x45C ?

Value = x [| | | | | | |]

Part c. What is the virtual address corresponding to physical address x942 ?

VA = x [ ]

Name:__________________________

Problem 6 (**optional - for those who finish early and wish a challenge**):

Many people have asked us to include an old popular addressing mode, available on Motorola's MC68000, Digital Equipment's PDP-11, and IBM's second generation RISC machine in the LC-3b. It is called pre-decrement addressing mode, whereby a source or destination operand address was obtained as follows: first decrement the register by the size of the operand in bytes. Then use the register as a pointer to the memory location to obtain the operand. The assembly notation is -(Rx). Evaluate the operand addresses sequentially, first source, then destination.

We will try this out by using our two unused opcodes 1010 and 1011 to do a copy instruction from source address to destination address using this new addressing mode for both. We will call 1010 MOVB for Move a byte from source to destination, and 1011 MOVW for the equivalent Move two bytes. For example, for MOVB, if R1 initially contained the value #4097, MOVB -(R1),-(R1) would copy the one byte in location #4096 into location #4095.

The encodings for MOVB and MOVW are as shown below.

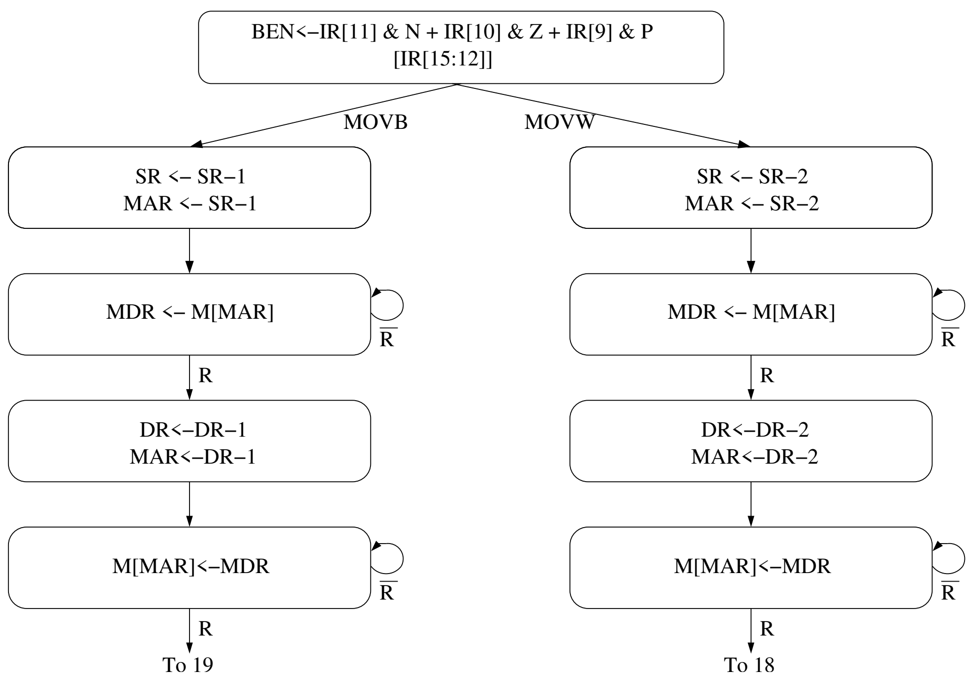
MOVB:

15–12	11–9	8–6	5	4–0
1010	DR	SR	1	11111

MOVW:

15–12	11–9	8–6	5	4–0
1011	DR	SR	1	11110

State machines for the two new opcodes are shown below:



Question: If we include MOVB and MOVW as described above in the LC-3b ISA, what additional storage structure would be needed in the data path specifically to allow the processor to handle page faults properly? We must not unnecessarily slow down the processor, so saving the register file before each MOVB or MOVW instruction is not an option. We would like to incur no extra cycles in processing MOVB or MOVW in the absence of a page fault.

Draw the storage structure that is needed to do this, with specific details as to the number of elements, size of each element, and size of each field.

Explain how the structure is used (in 25 words or less).

Explain why this structure is necessary (in less than 25 words, please).

Name:__________________________

Problem 6 continued:

Explain how the structure is used:

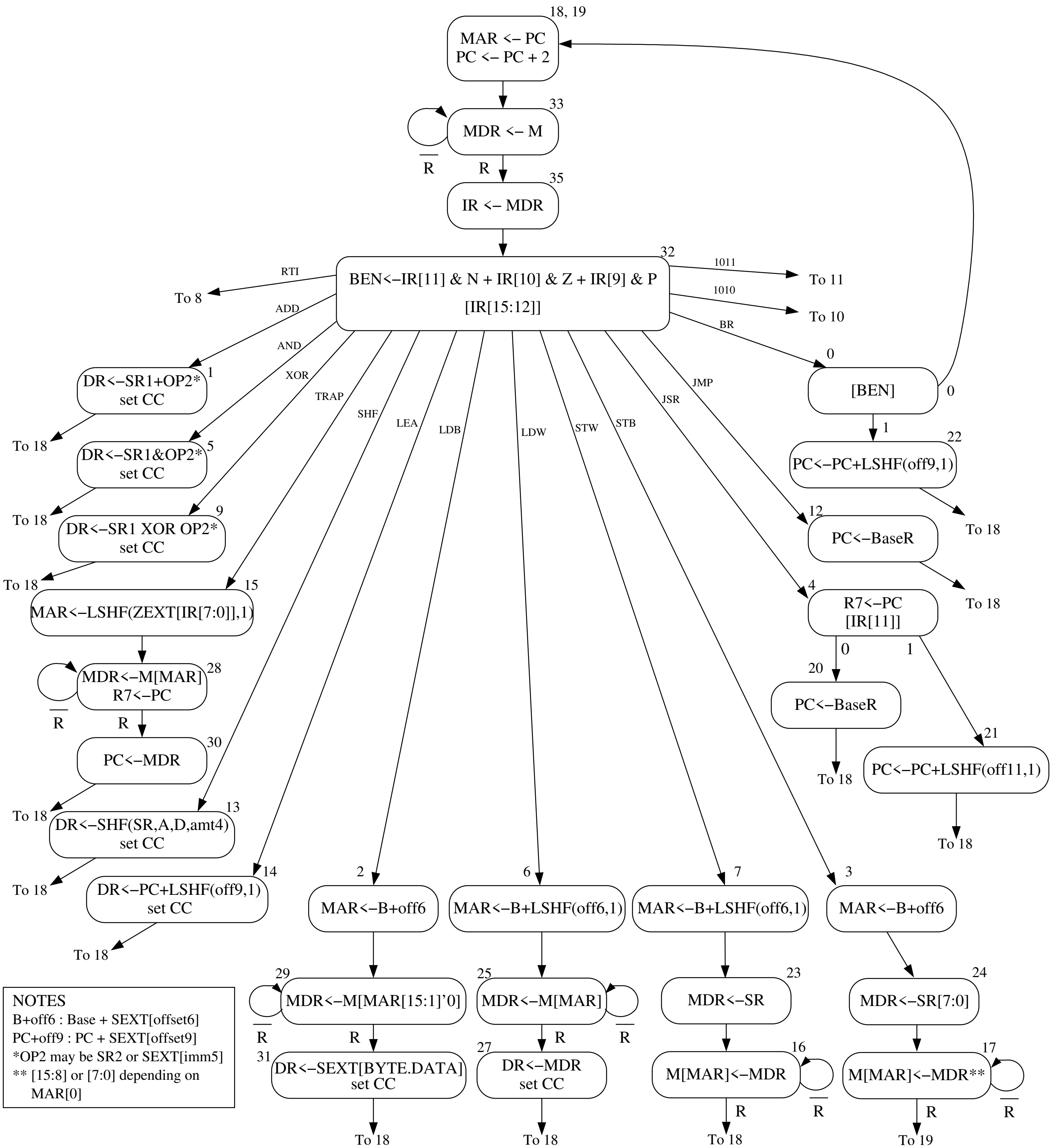
Explain why this structure is necessary:

LC-3b ISA

	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															

+ indicates instructions that modify condition codes.

A state machine for the LC-3b (from Appendix C)



NOTES
B+off6 : Base + SEXT[offset6]
PC+off9 : PC + SEXT[offset9]
*OP2 may be SR2 or SEXT[imm5]
** [15:8] or [7:0] depending on
MAR[0]