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

EE 360N, Spring 2007
Yale Patt, Instructor
Chang Joo Lee, Rustam Miftakhutdinov, Poorna Samanta, TAs
Exam 1, March 7, 2007

Name:_____

Problem 1 (25 points):_____

Problem 2 (10 points):_____

Problem 3 (20 points):_____

Problem 4 (20 points):_____

Problem 5 (25 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.

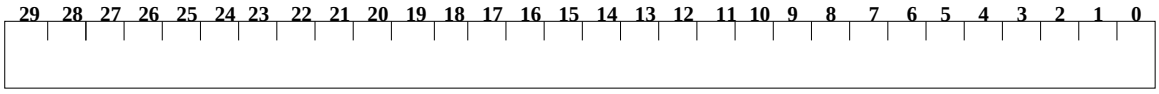
Note: Please be sure your name is recorded on each sheet of the exam.

GOOD LUCK!

Name:_____

Problem 1 (20 points)

Part a (5 points): A 1GB physical memory system is byte addressable. It has a 64-bit bus to the processor. It is 16-way interleaved. The memory is made out of 8MB chips, each with 8 data pins. Identify which bits in the address are row address bits, chip address bits, byte of bus bits and interleave bits.



Part b (5 points): If we add ECC protection to the LC-3b, then each 16 bit word would require another 5 bits to be able to correct a one bit error. Suppose we did, and we got back the following 21 bit pattern.

20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	1	1	0	1	1	1	0	0	0	1	0	0	0	1	1	1

Which bit was in error?

Part c (5 points): The atomic unit of processing is:

Name:_____

Problem 1 continued

Part d (5 points): Some of the following are part of the ISA, the rest are part of the microarchitecture. Put a check mark next to each that is part of the ISA.

page size: _____

MDR: _____

condition codes: _____

memory ready bit (R): _____

trap vector: _____

Part e (5 points): The xyz machine, which is bigendian, executes LD32 R1,A. Relevant memory locations before the instruction executes is as shown below:

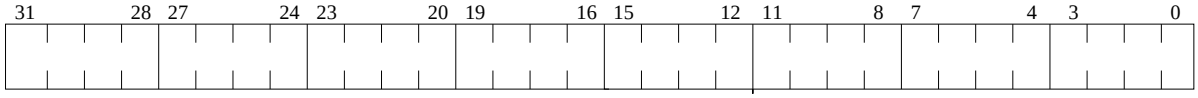
A: 11110000

A+1: 11111111

A+2: 10101010

A+3: 00000000

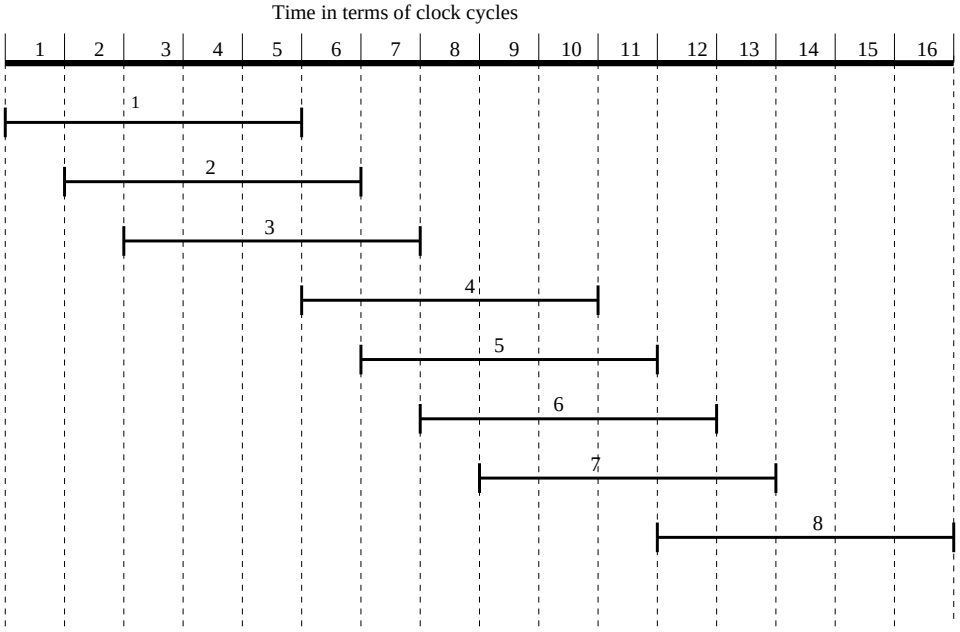
After execution, R1 contains:



Name:_____

Part c (10 points)

We have not talked about pipelining yet in class. When we do, you will see a pipelined machine can easily issue a memory request every cycle. At the memory controller side, however, we may need a queue to buffer the memory addresses if there are memory bank conflicts. In this example, eight successive memory accesses have arrived at the memory controller, and are buffered until their banks are free. Accesses must be done in the order in which they arrived. The figure below shows which memory accesses are active during each cycle.



Note that memory access 1 is initiated in cycle 1 and returns data at the end of cycle 5. Memory has an access time of five cycles, and is four way interleaved.

Your job: Identify which bank each memory access goes to and fill in the table below accordingly. Memory access 1 has already been entered. (Note: there are several correct solutions: any one of them will receive full credit.)

BANK 3	BANK 2	BANK 1	BANK 0
			1

Name:_____

Problem 3 (20 points)

An x86 assembly language programmer complained that the LC-3b did not have what to her was the most valuable addressing mode which is available in the x86 ISA. Recall that the x86 instruction is variable length. One of the optional bytes in that instruction is called SIB (for Scale/Index/Base). It allows one to construct an address by scaling (multiplying) the contents of one register (the Index) and adding the result to the contents of another register (the Base). That is, Address = Base + Scale*Index.

NO PROBLEM, we say. We will use an unused opcode to provide the same capability with the LC-3b ISA. We will call the new opcode SIB:

```
SIB  DR, BaseR, Scale, IndexR
```

which will load DR with the address computed by multiplying the IndexR register by 2^{Scale} and adding the result to the contents of the BaseR register.

We thus get the same effect as the x86 SIB byte, only it takes two LC-3b instructions. That is,

```
SIB  R5,R3,#3,R2
LDW  R1,R5,#0
```

will load R1 with the contents of memory whose address is obtained by adding R3 to the product of R2 and 2^3.

The next page shows the data sheet for the SIB instruction in the style of Appendix A.

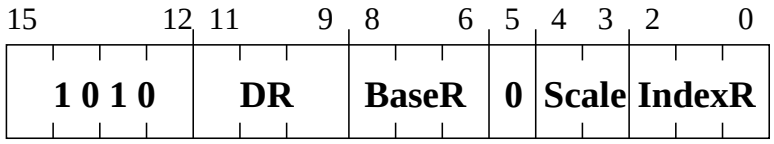
Part a (5 points): We can implement the SIB instruction with either one extra state or two extra states in the state diagram of the LC-3b. Which is better? Why?

SIB — Scale, Index, Base

Assembler Format

SIB DR, BaseR, Scale, IndexR

Encoding



Operation

DR = BaseR + (IndexR * 2^{Scale});
setcc(DR);

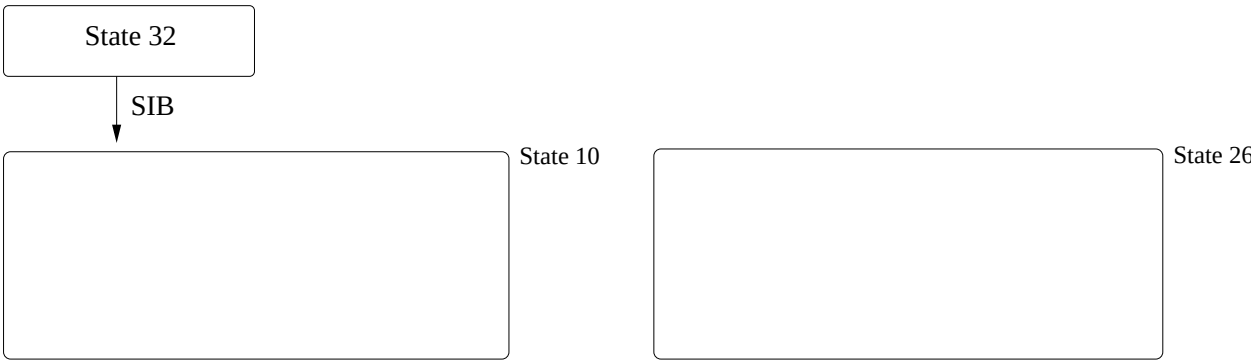
Description

Load the register specified by DR with the address formed by multiplying the index register IndexR by 2^{Scale} and adding the result to the base register BaseR. The condition codes are set, based on whether the value loaded is negative, zero, or positive.

Example

```
SIB     R5, R3, #3, R2      ;    R5 is loaded with 8*R2 + R3
```

Part b (15 points): Your job here is to implement the SIB instruction with two extra states (state 10, and state 26). Using the notation of the LC-3b State Diagram, describe what happens in these states inside their corresponding bubbles. Show all output arc(s) to indicate the next state after state 10 and state 26.



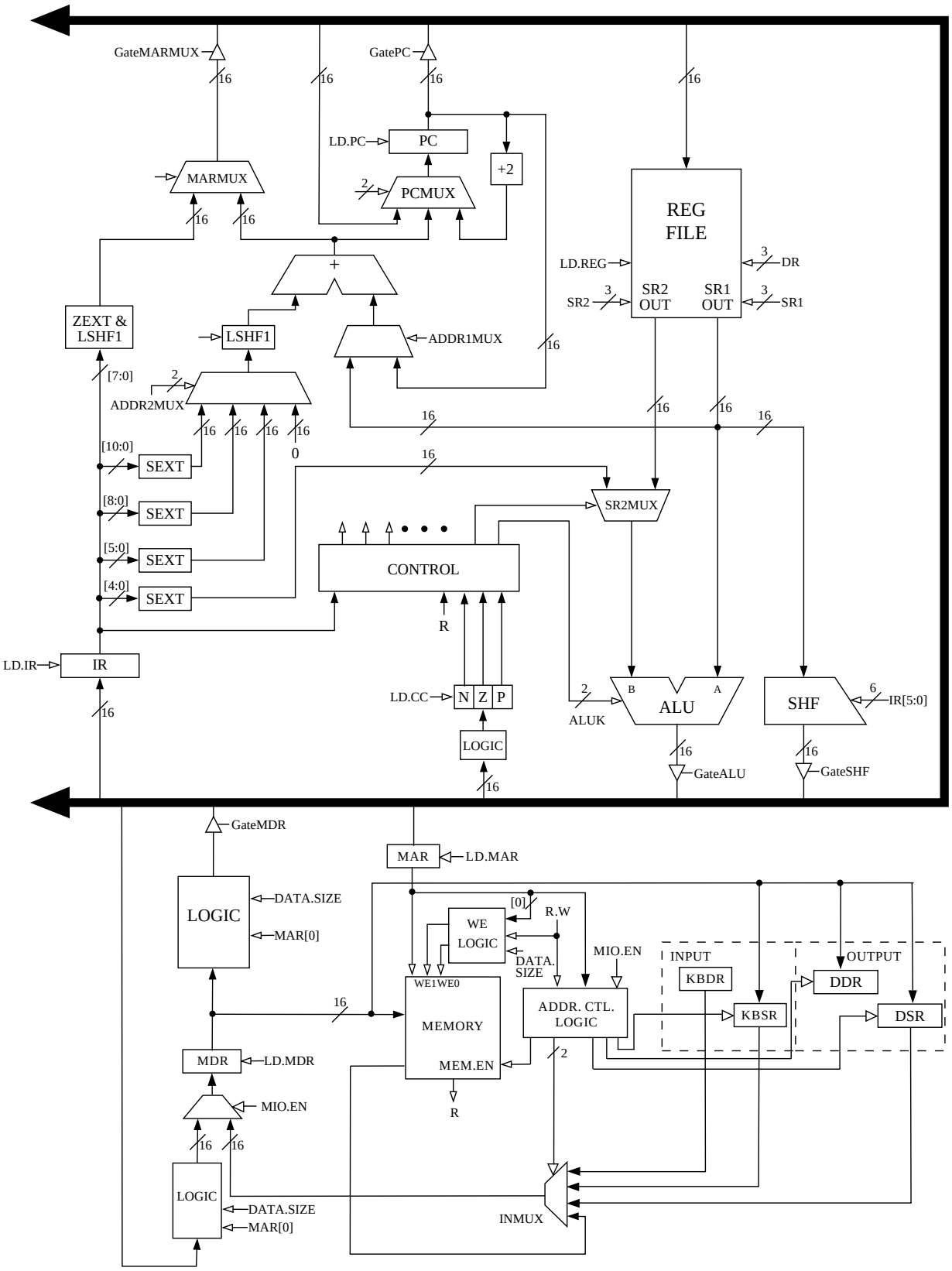
Using the data path diagram labeled "SIB with two extra states" on the next page, add any additional structures and any control signals needed to implement SIB as specified by the two states shown in the bubbles. Label any additional control signals "ECS 1" (for "extra control signal 1"), "ECS 2" etc.

Show the values in the figure below for each control signal corresponding to states 10 and 26.

	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	ECS1	ECS2	ECS3	ECS4	ECS5	ECS6
													(PC+2, BUS, ADDR 00, 01, 10				(ZERO, offset6, PCoffset9, PCoffset11 00, 01, 10, 11		(ADD, AND, XOR, PASSA 00, 01, 10, 11					as needed					
State 10													—				—		—										
State 26													—				—		—										

	IRD	COND	J					
State 10		—						
State 26		—						

SIB with two extra states



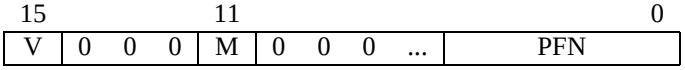
Name:_____

Problem 4 (20 points)

Consider the following **two level** virtual memory system for the LC-3b:

Virtual Address Space:	64KB	Physical Memory Size:	4KB
User Space Range:	x0000 to x7FFF	Page Size:	256 bytes
System Space Range:	x8000 to xFFFF	Page Table Entry Size:	2 bytes

The system **does not** include a Translation Lookaside Buffer. The Page Table Entry format is as follows:



Part a (2 points): How many bits are allocated for the Page Frame Number (PFN) in the PTE? Show the computation.

Answer:

Part b (18 points): The machine stopped at a breakpoint and the following state information was observed:

PC	SBR	UBR	R0	R1	R2	R3	R4	R5	R6	R7
x3100	x000	x9000	x30F0	xF000	x4050	xF000	x0000	x1299	x8000	x1000

Note: SBR is the System Page Table Base Register and UBR is the User Page Table Base Register. Each points to the first entry of the corresponding page table.

After execution resumed, the machine issued the following successive six physical memory requests, uninterrupted by any page faults, access control violations, or anything else. Note that each entry is incomplete. Your job: complete the six entries.

Access #	PA	Data	Identity of Item Being Read
1		x8004	
2		x8001	
3		x6200	
4			
5		x8007	
6		x8004	

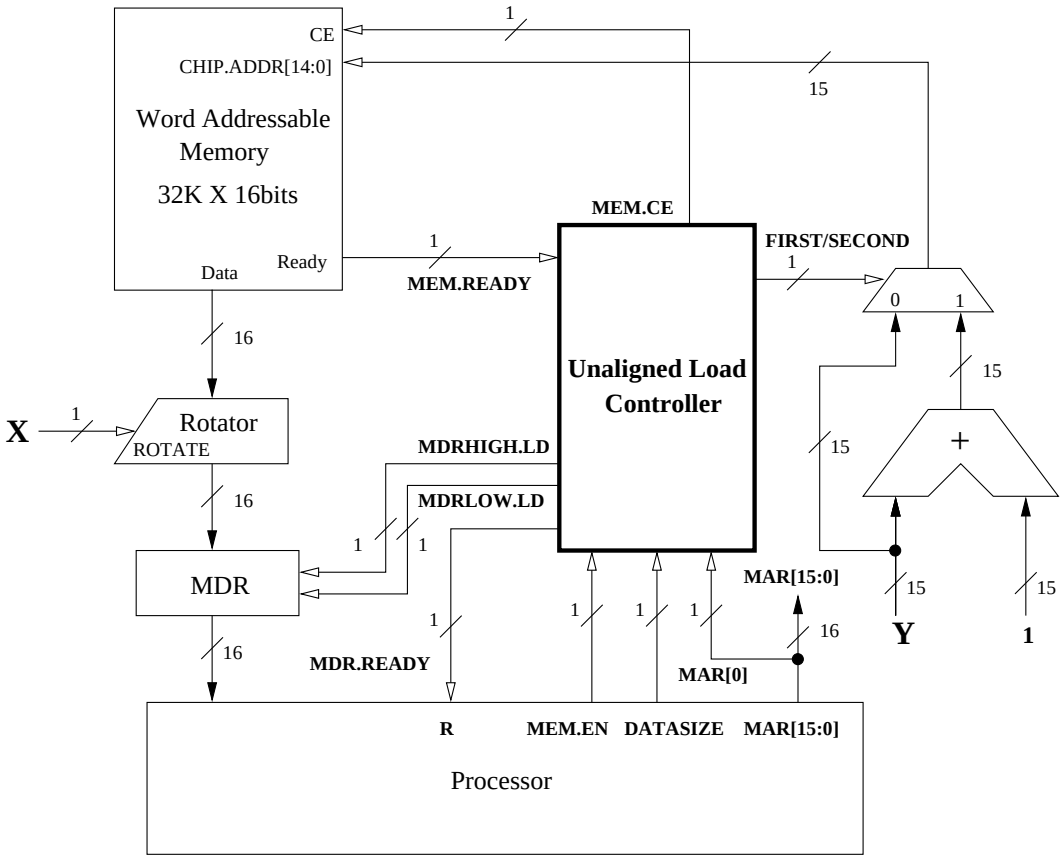
Note: The last column should identify what is being read specifically. For example: “The instruction JSR HELP,” “The PTE for User Virtual Page 0,” “PTE for System Virtual Page 0,” “Data due to load instruction,” etc.

Name:_____

Problem 5 (25 points):

The LC-3b assumes that all accesses are aligned. Once we have loaded the MAR, we enter a state (for example, states 25, 29, etc), where the MDR is loaded each cycle until the ready signal from memory tells the state machine that the MDR contains good data, and processing can continue.

We wish to augment the LC-3b to support unaligned loads **without** changing the control store of the LC-3b. (We will not deal with stores in the interest of time on this exam.) The diagram below shows how we can do it.



Note that we have a new piece of logic, an **Unaligned Load Controller**.

Like in our original machine (your 3rd programming lab), the processor provides the MAR, MEM.EN, DATASIZE, in state 25, 29, etc.) and waits for the memory system to return the ready bit R. The Unaligned Access Controller has as input: MEM.EN, DATASIZE, MAR[0] and the MEM.READY signal from Memory. The Unaligned Load Controller controls the unaligned load with MEM.CE, MDRHIGH.LD, MDRLOW.LD, FIRST/SECOND and a MDR.READY signal to let the processor know that MDR contains good data. The processor uses MDR.READY as it used the R signal from memory in the aligned case.

Name:_____

Problem 5 continued:

The control bits are encoded as follows:

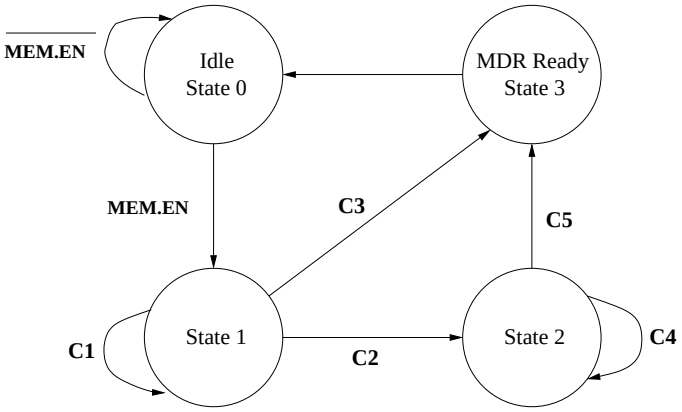
DATASIZE:	0 = Byte
	1 = Word
MDRHIGH.LD:	0 = No Load
	1 = Load High Byte
MDRLOW.LD:	0 = No Load
	1 = Load Low Byte
ROTATE:	0 = No Rotation
	1 = 8-bit Rotation
FIRST/SECOND:	0 = First Access
	1 = Second Access
MDR.READY:	0 = MDR Not Ready
	1 = MDR Ready

Part a (5 points): Identify the 1-bit signal X and the 15-bit signal Y shown on the diagram. (Note: they are generated by the processor.)

X: [ ]

Y: [ ]

Part b (5 points): The state machine for the Unaligned Load Controller is shown below. State 0 is an idle state, where no memory access is occurring. In state 3, the state machine sets MDR.READY which the processor interprets as the old R bit from memory, so it can move on and read the value in the MDR.



Briefly explain what is accomplished in States 1 and 2.

State 1: [ ].

State 2: [ ].

Name:_____

Problem 5 continued:

Part c (10 points): The Unaligned Load Controller has signals C1, C2, C3, C4, and C5, which are used to transition the controller through its states. Complete the logic equations for those control signals. (Note: No control signals are required to transition form state 3 to state 0.)

C1 = _____

C2 = _____

C3 = _____

C4 = _____

C5 = _____

Part d (5 points): The Unaligned Load Controller is best thought of as a Moore Machine. That is, its outputs are associated with the state. Complete the table below, identifying the value of each output signal for each state.

	MEM.CE	FIRST/SECOND	MDRHIGH.LD	MDRLOW.LD	MDR.READY
State 0					
State 1					
State 2					
State 3					

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	

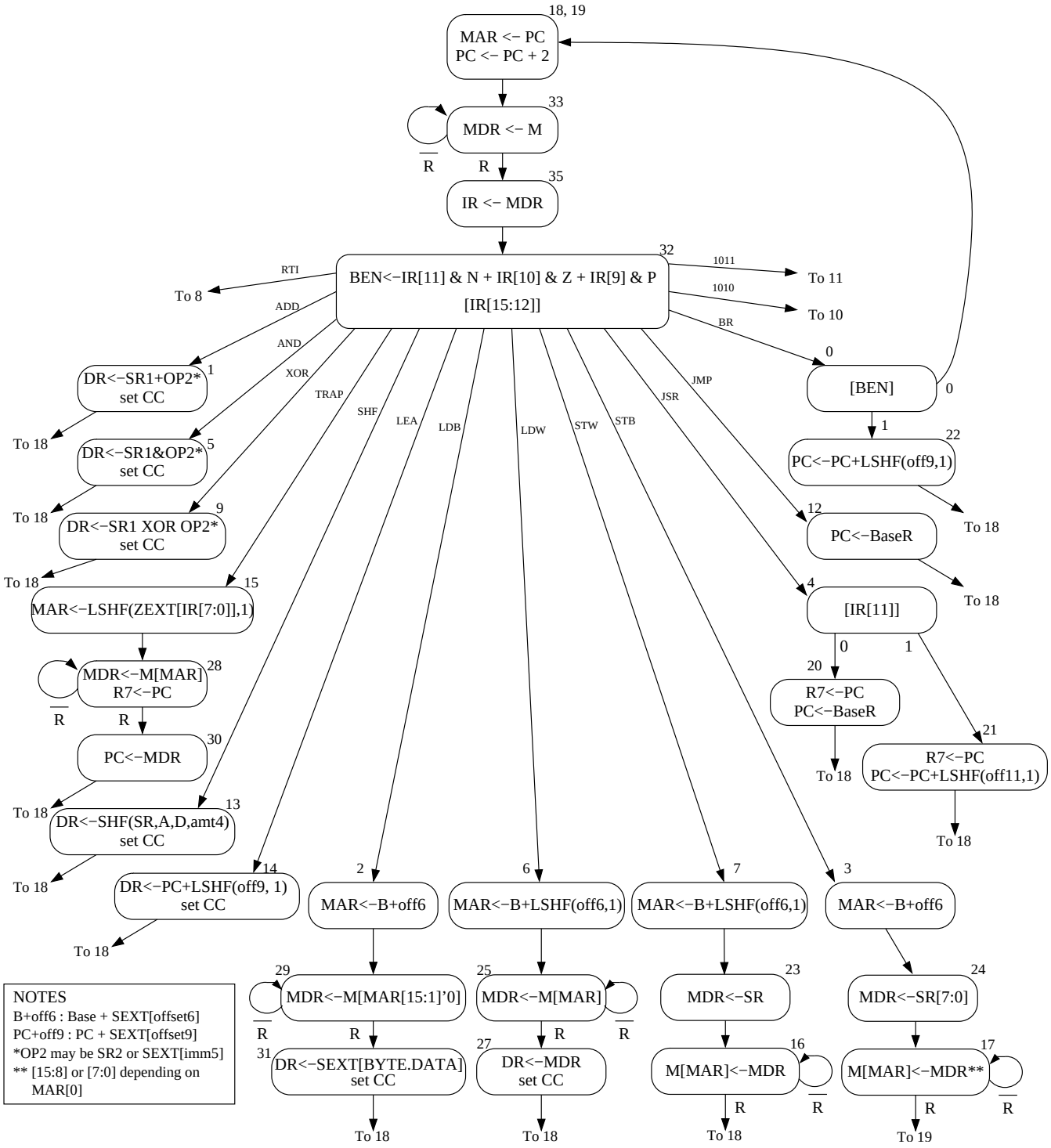


Figure 1: A state machine for the LC-3b

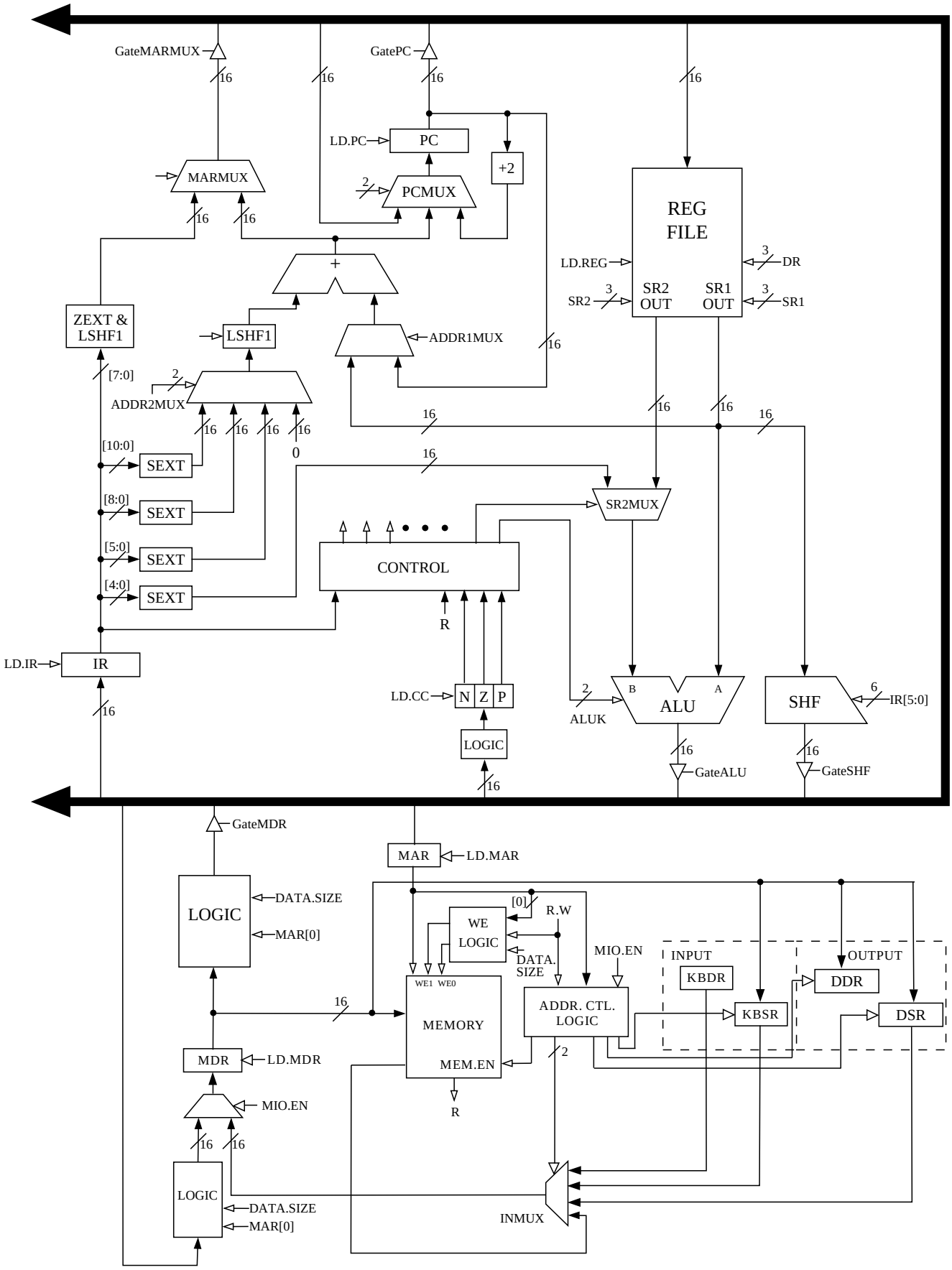


Figure 2: The LC-3b data path

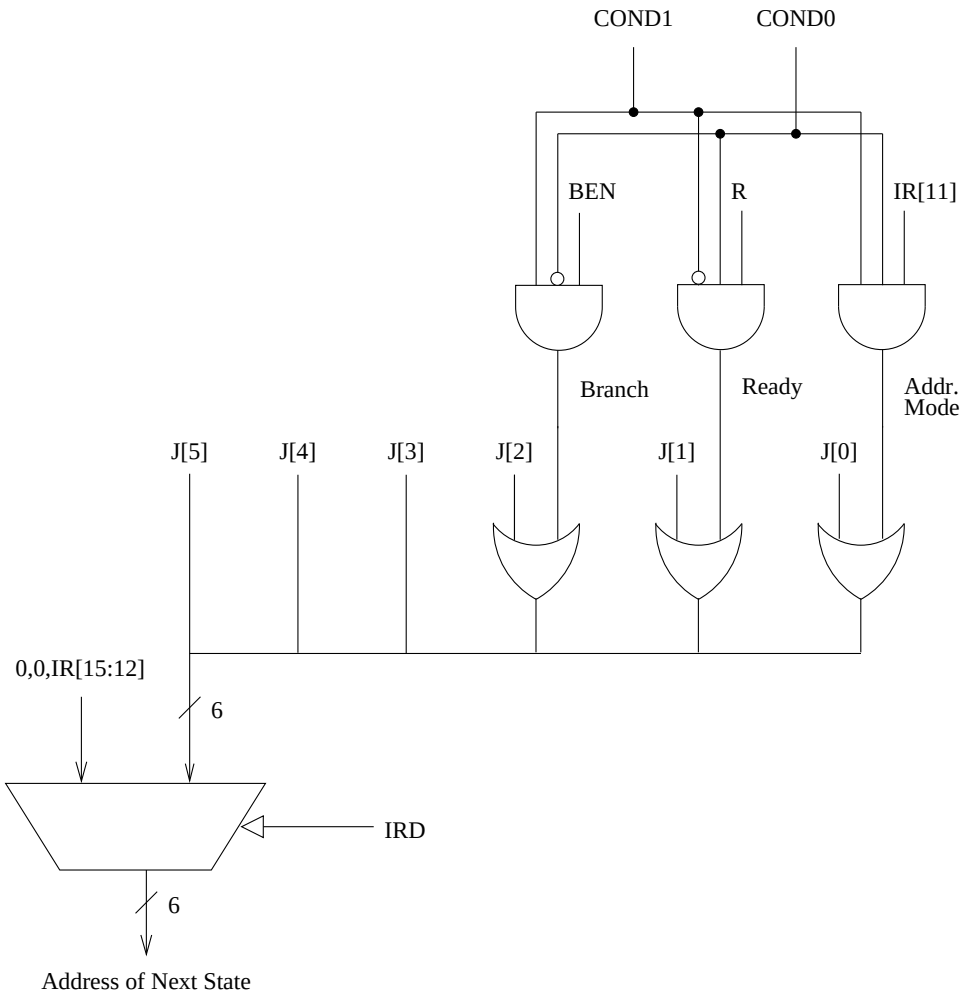


Figure 3: The microsequencer of the LC-3b base machine