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

EE 460N Spring 2011
Y. N. Patt, Instructor
Faruk Guvenilir, Milad Hashemi, Yuhao Zhu, TAs
Exam 1
March 9, 2011

Name:_____

Problem 1 (25 points):_____

Problem 2 (15 points):_____

Problem 3 (20 points):_____

Problem 4 (20 points):_____

Problem 5 (20 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.

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): The basic performance equation is:

$$Performance = \frac{1}{x * y * z}$$

Decreasing x, y, or z improves performance. What are x, y, and z.

x: [ ] **y:** [ ] **z:** [ ]

y and z are always at odds with each other: decrease y at the expense of z, and vice-versa. Explain why that is in 20 words or fewer.

[ ]

Part b (5 points): The LC-3b data path shows a path that increments the PC by 2 and then loads the result into the PC. Some have been concerned that there is a problem since the incremented PC would then get incremented by 2 again, and again, continuously loading into the PC. Since this would happen many times during the clock cycle, by the end of the cycle the PC would contain PC + a big number. Why is this not a problem? Please answer in fewer than 20 words. An answer that simply names the structures that comprise the PC will receive zero points. A correct answer explains what happens and when.

[ ]

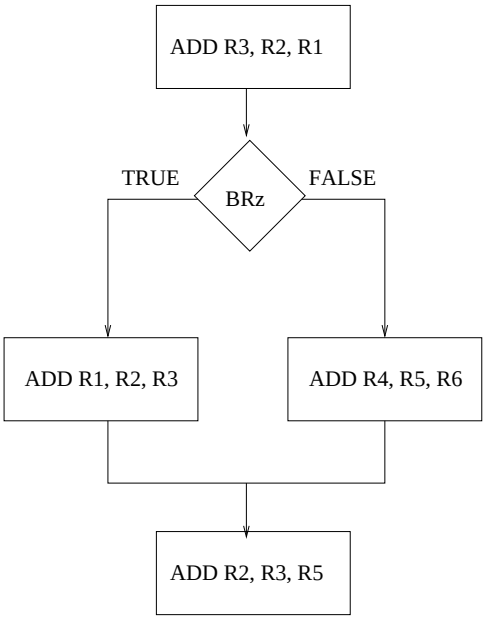
Part c (5 points): The XYZ company decided to redesign the LC-3b to be big-endian. Assume this redesign. Show the contents of locations x3000 and x3001 if the machine language instruction corresponding to the assembly language instruction RSHFA R3,R2,#6 is stored there.

7	6	5	4	3	2	1	0	
								x3000
								x3001

Name:_____

Problem 1 continued

Part d (5 points): In a pipelined microarchitecture, conditional branches are problematic because usually the condition has not been determined at the point where one needs to fetch the instruction after the branch. One solution to this problem is branch prediction. An alternative is predicated execution. A standard flow chart for a program fragment containing a conditional branch is shown below on the left. Construct on the right the flow chart corresponding to the code a compiler would construct if it predicated the branch.



Part e (5 points): Two successive accesses to a DRAM chip were to chip addresses xBC7842 and xBC7A12. The second access required asserting Row Address Strobe before asserting Column Address Strobe. What is the maximum number of bytes in each row buffer? Assume the row buffer is byte addressable. Assume the DRAM chip is part of only one bank. Show your work.

Name:_____

Problem 2 (15 points)

XYZ Computer Company wants to produce the LC-3b, but wants to make it a 32 bit machine (i.e., address space is 32 bits, registers are 32 bits, data path to memory is 32 bits, the LC-32b bus is 32 bits, etc.), and implement unaligned accesses. LC-32b, they wish to call it. In this problem we consider the implementation of the store halfword instruction in the LC-32b,

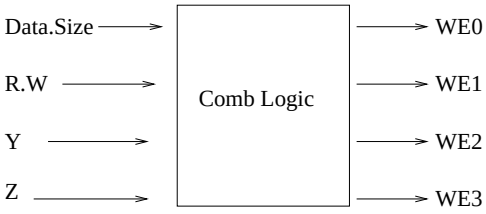
StH R1, A

which stores the low 16 bits of register 1 into memory locations A, A+1. All the generic features of the LC-32b are the same as they were in the LC-3b.

Part a (3 points): Assume R1 = #-10 (xFFFFFFF6) before this instruction executes. After decode, what gets loaded into the 32-bit MDR?

Part b (3 points): Assume A = xA805. After decode, what gets loaded into MAR?

Part c (4 points): For an unaligned write, one needs combinational logic to determine WE0, WE1, WE2, and WE3. The combinational logic block is shown below:



What are the two inputs Y and Z that are usually necessary to determine the write enable signals. Hint: Y is a two-bit value, Z is a one-bit value.

Y: **Z:**

Part d (5 points): For the values of Data.Size, R.W, Y and Z that are relevant for the above example, complete the one or two rows of the truth table that indicate the WE signals that are asserted to accomplish the StH instruction.

Data.Size	R.W	Y	Z	WE3	WE2	WE1	WE0
Halfword	Write						
Halfword	Write						

Name:_____

Problem 3 (20 points)

In this problem, we would like you to reconstruct the code that must have been executed by an out-of-order machine during cycles 0 through 23. The first instruction is fetched in cycle 0, the last instruction finishes in cycle 23.

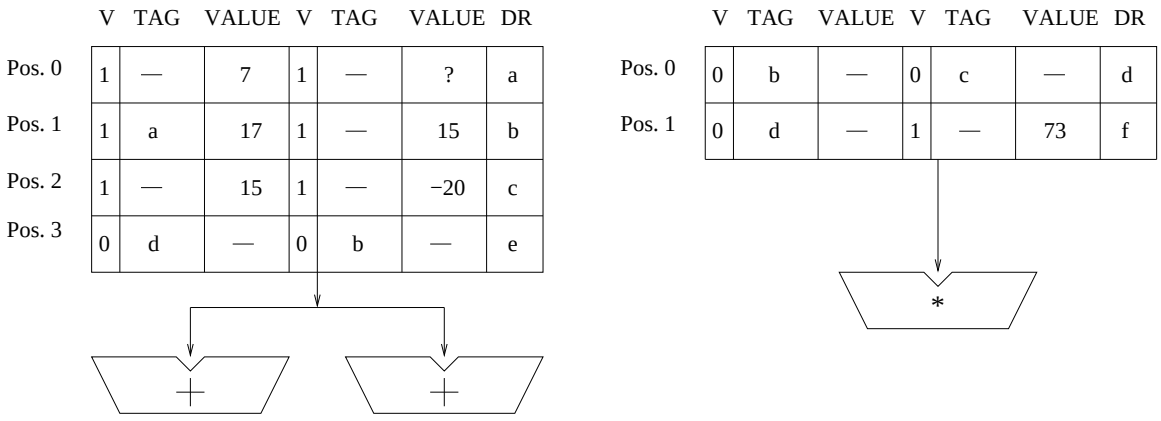
The diagram below shows what is going on in each cycle, from cycle 0 through cycle 23.

Cycles	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
ADD 0			x	x	x	x	x	x	x	x	x	x							x	x	x	x	x	
ADD 1					x	x	x	x	x															
MUL													x	x	x	x	x	x	x	x	x	x	x	x

Figure 1. Occupancy Chart.

The machine has a simple 4 stage pipeline. The stages are: fetch, decode, execute, and writeback. Fetch, decode and writeback take 1 cycle each. Execute takes a variable number of stages.

The execution unit has two adders and one multiplier. An 'x' in the chart above indicates that the functional unit is busy during that cycle. Figure 2 shows the state of the machine after cycle 7. A '?' indicates that you will need to figure out what that value is.



	V	TAG	VALUE
R0	1	—	5
R1	0	d	10
R2	0	f	73
R3	0	e	100
R4	1	—	15
R5	1	—	−20
R6	1	—	31
R7	1	a	17

DR = Destination Register Tag
V = Valid bit

Figure 2. The state of the machine after cycle 7.

Name:_____

Problem 3 continued

The functional units have reservation stations that are tied to either the multiplier or to the adders as shown.

Destination register rename tags are assigned alphabetically, starting with the letter 'a' in program order.

The machine utilizes reservation stations starting from position 0 and continuing down sequentially.

Assume data is forwarded to all waiting instructions the cycle after the computation has completed.

Part a: What is the latency of each functional unit?

Adder: | **Multiplier:**

Part b: From all the information shown above, construct the assembly language code that must have executed starting in cycle 0 and completing in cycle 23.

Please list the code in the following format.

Opcode DR, SR1, SR2
DR: Destination Register
SR1: The operand in the left reservation station entry.
SR2: The operand in the right reservation station entry.

Instruction 0	
Instruction 1	
Instruction 2	
Instruction 3	
Instruction 4	
Instruction 5	

Part c: What is the IPC of this code fragment?

IPC:

Name:_____

Problem 4 (20 points)

You have been given a black box containing a 1K by 16-bit-addressable memory system, and you are asked to figure out the organization of that memory – how many ranks, and the degree of interleaving. You are told that the memory is made up of 32 x 16 bit chips.

You are provided with a sequence of addresses to use as a memory access pattern, which is shown below in Table 1. Unaligned memory accesses are not allowed.

You can assume that the memory requests are generated successively, starting at cycle 0 and the memory controller sends each request to DRAM as quickly as it is able to. The results from this sequence of memory accesses is shown below in Table 2.

Table 1. Test addresses.

Access Number	Test Address
1	x284
2	x2A4
3	x2D1
4	x2FE
5	x2AB
6	x0C6
7	x0A0
8	x115

Table 2. Test results.

Cycle Number	Sent	Returned
0	x284	–
1	–	–
2	–	–
3	–	–
4	x2A4	x284
5	x2D1	–
6	x2FE	–
7	x2AB	–
8	x0C6	x2A4
9	–	x2D1
10	–	x2FE
11	–	x2AB
12	x0A0	x0C6
13	x115	–
14	–	–
15	–	–
16	–	x0A0
17	–	x115

The rank bits are guaranteed to start from the MSB and are contiguous. The interleaving bits are guaranteed to be contiguous as well.

What is the organization of the memory system? That is, please identify the rank bits and the interleave bits.

Rank bits: [ ] **Interleave bits:** [ ]

Explain your answer in the box below.

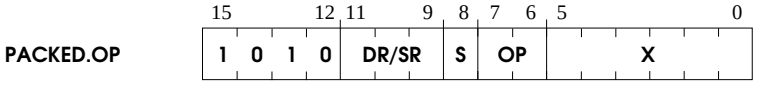
[ ]

Name:_____

Problem 5 (20 points):

Many ISAs use packed data, wherein one 16-bit word contains two 8-bit values. It saves space. Your task here is to implement PACKED.OP, an instruction that performs ADD, AND, or XOR on bytes. We will use the unused opcode 1010.

PACKED.OP has the following format:



Note that the two operands are the high and low byte of SR, and the sign-extended or zero-extended result is stored into DR. Note that SR and DR are the same register. Also, the S bit specifies whether to sign-extend (S=1) or zero-extend (S=0). OP specifies whether to ADD (00), AND (01), or XOR (10). Condition codes are set based on the result written to DR, as usual.

The function of X (bits[5:0]) will be left for you to figure out. ...later.

Example Assembler Formats

PACKED.ADDs SR1 ; sign extend
PACKED.ANDz SR1 ; zero extend
PACKED.XORz SR1 ; zero extend

Semantically, the instruction is processed as:

```
IF(S == 1)
      DR = SEXT(SR[15:8] OP SR[7:0])
ELSE
      DR = ZEXT(SR[15:8] OP SR[7:0])
setcc();
```

We can implement PACKED.OP with only two additional microinstructions and a few changes to the data path. We have made the changes to the data path. **We have left for you to identify certain information about the data path and also to generate the two new microinstructions.** Pages 9 and 10 deal with the data path, page 11 deals with the two new microinstructions. For this problem, we can ignore overflow conditions.

Name:_____

Problem 5 continued:

The data path you are familiar with has been modified (in bold face) as shown below.

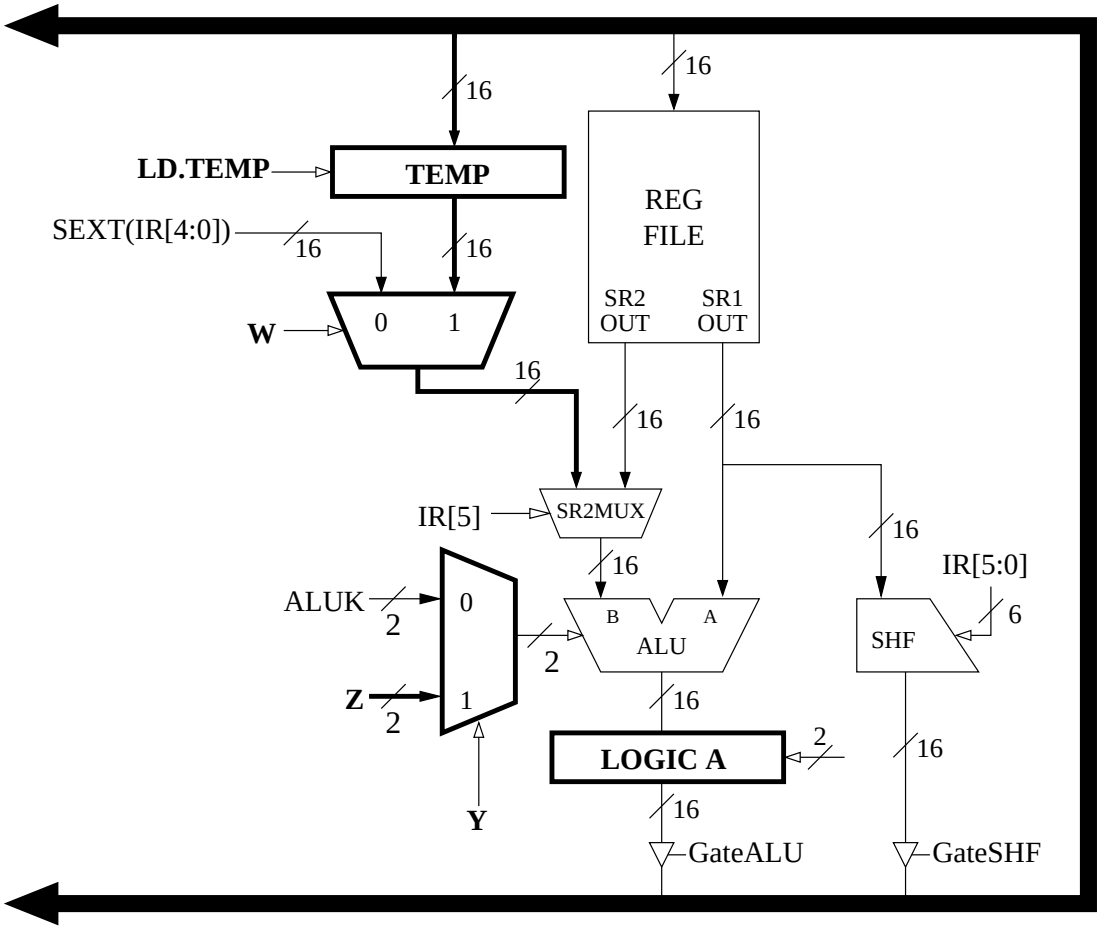


Figure 1: Modified datapath to support PACKED.OP instruction

Name:_____

Problem 5 continued:

Part a: Identify X, W, Y, Z. Note that your answers may be bits of the IR, existing control signals in the LC-3b, new control signals that may be added to the LC-3b, or arbitrary bit patterns. If you use any new control signals, call them ECS1, ECS2, ECS3, etc.

X: | **W:** | **Y:** | **Z:**

Part b: What does logic block A do? Note that logic block A doesn't include a shifter.

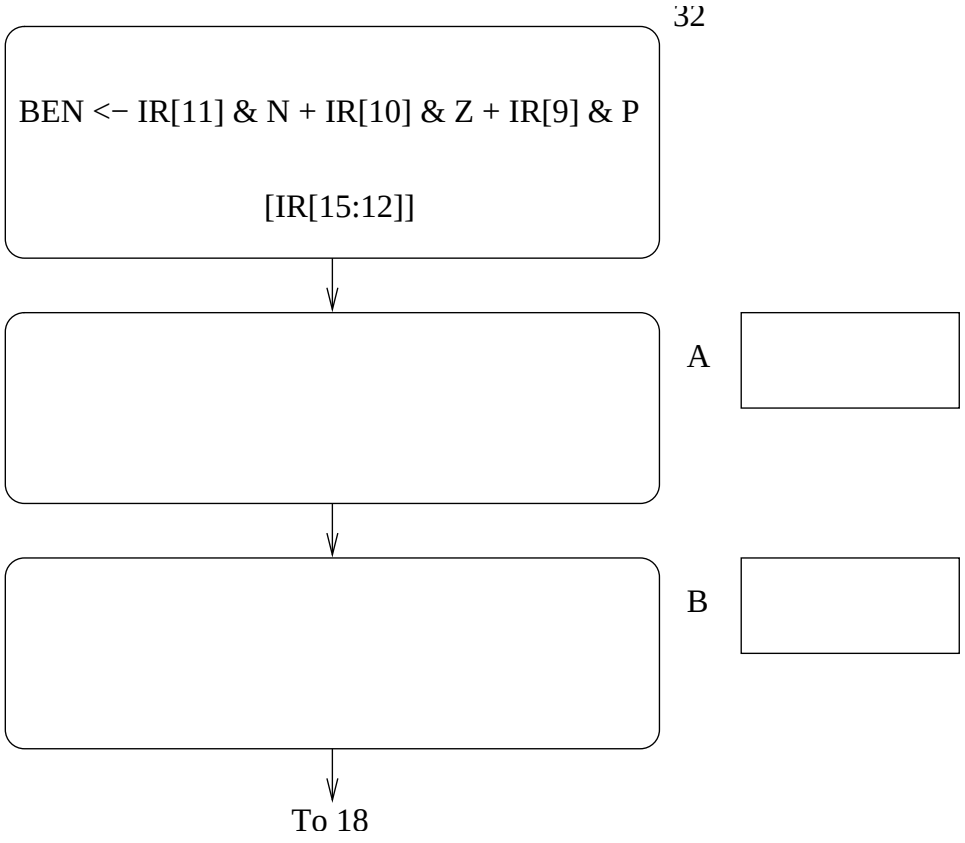
For each control signal to logic block A, describe what the values 0 and 1 mean.

Control Signal	**0**	**1**

Name:_____

Problem 5 continued:

Part c: Two states (A,B) are needed to implement PACKED.OP. Fill in the operation for states A and B, and pick valid state numbers.



Part d: A relevant portion of the microinstructions for states A and B are shown below. Fill in the table with the appropriate values, and cross out the columns for the extra control signals if you did not use them.

	LD.IR	LD.BEN	LD.REG	LD.CC	LD.PC	LD.TEMP	GatePC	GateALU	GateSHF	PCMUX[1:0]		DRMUX	SR1MUX	ALUK[1:0]		ECS1 (if needed)	ECS2 (if needed)	ECS3 (if needed)	ECS4 (if needed)
A																			
B																			

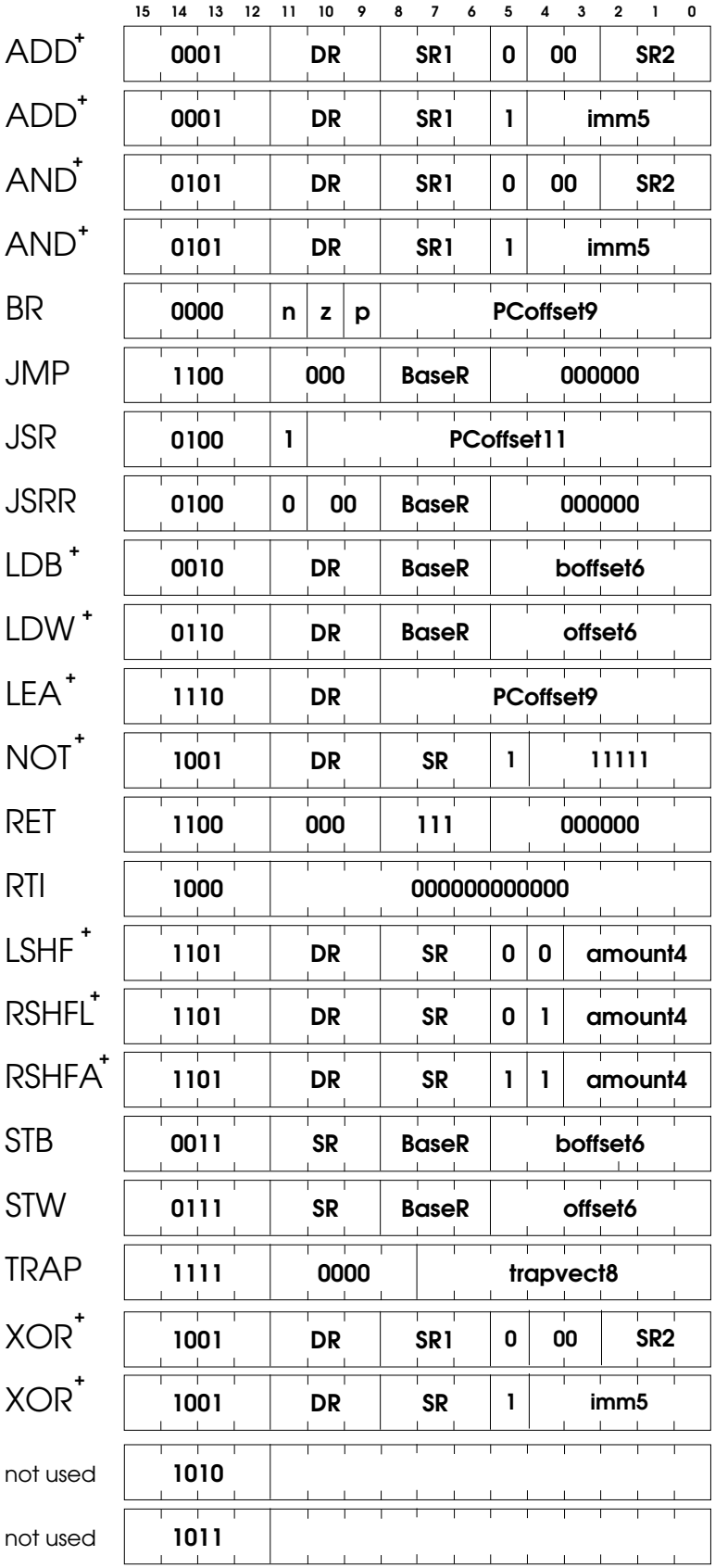


Figure 2: LC-3b Instruction Encodings

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	

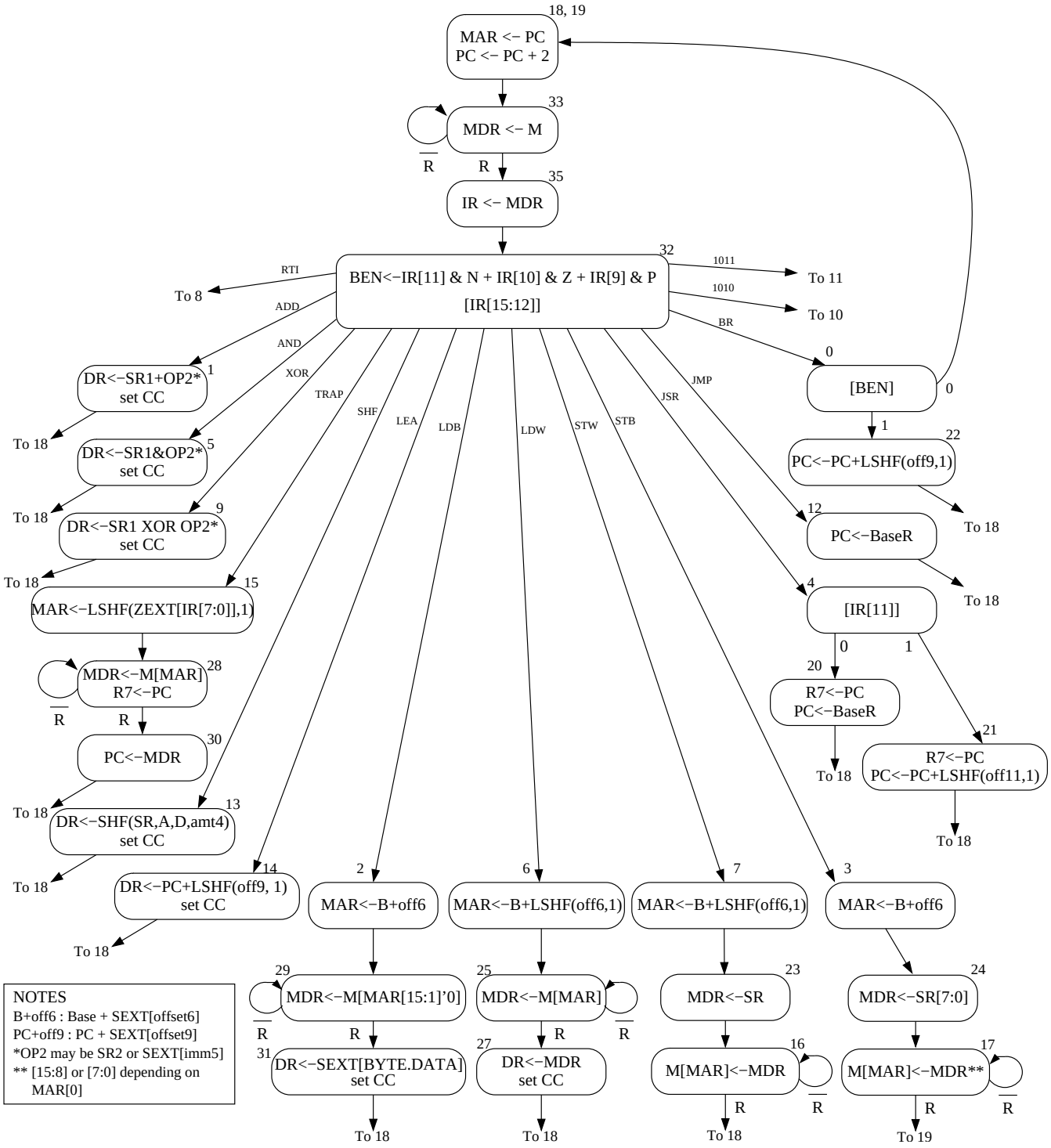


Figure 3: A state machine for the LC-3b

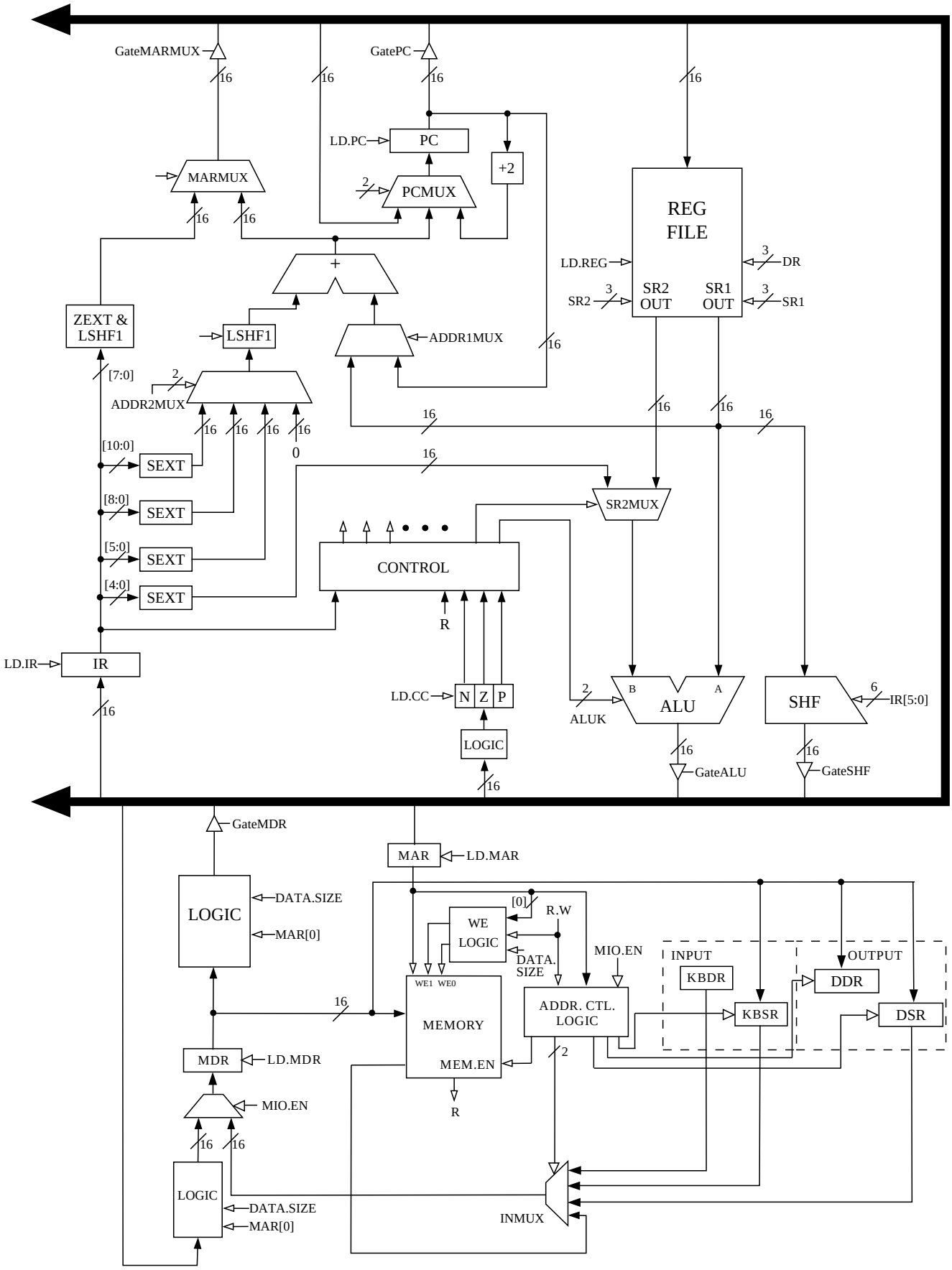


Figure 4: The LC-3b data path

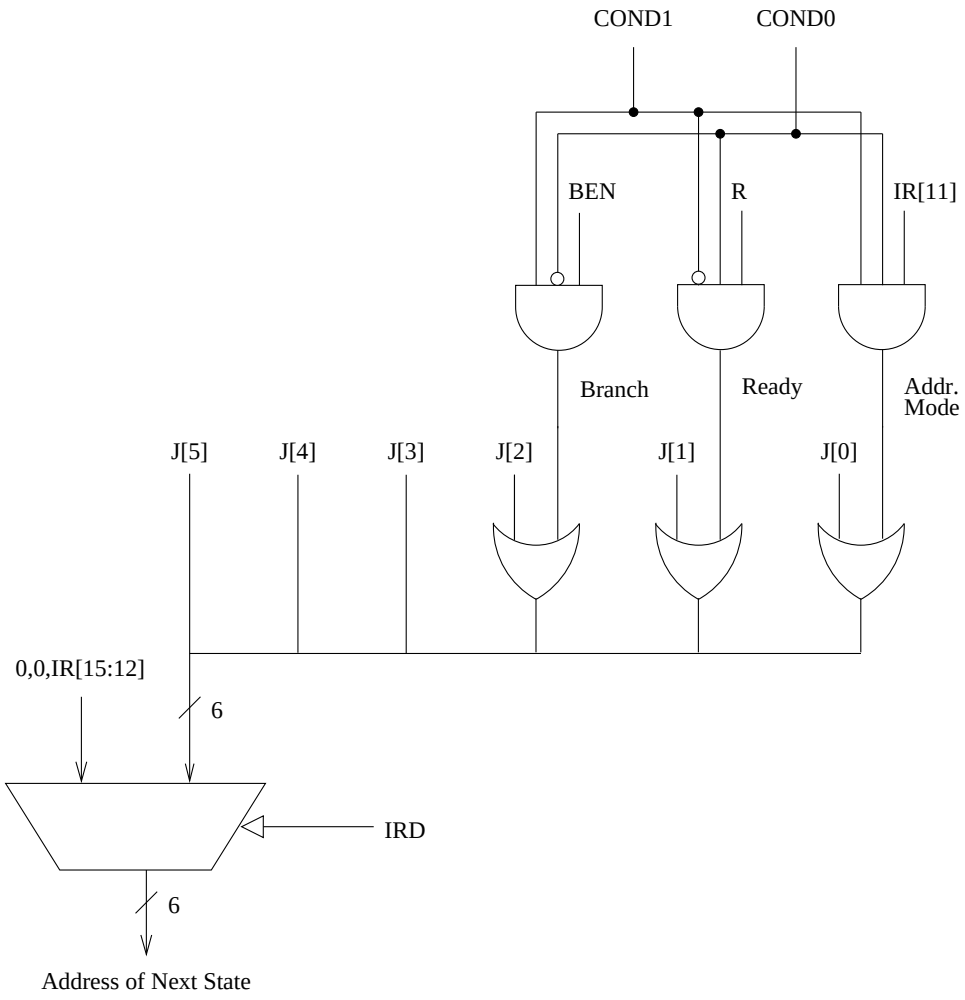


Figure 5: The microsequencer of the LC-3b base machine