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

EE 360N, Spring 2009
Yale Patt, Instructor
Ramapriyan Chakravarthy, Khubaib, Vivekanand Venugopal, TAs
Exam 1, March 11, 2009

Name:_____

Problem 1 (20 points):_____

Problem 2 (10 points):_____

Problem 3 (10 points):_____

Problem 4 (25 points):_____

Problem 5 (15 points):_____

Problem 6 (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.

GOOD LUCK!

Name:_____

Problem 1 (20 points)

Part a (4 points): Consider memory that is protected by an even parity bit for every 8 bits of data. If we wish to transmit 10001001, we would instead transmit 100010011 where the last bit is the parity bit. Suppose the first and second bits were transmitted incorrectly, i.e., 010010011. Would the error be detected? Yes/No (circle one). If yes, explain how. If no, is this in general of serious concern? Use no more than 20 words.

Part b (4 points): In a 32 location content addressable memory, how many address bits are used to identify the location that one wishes to access? Explain in no more than 20 words.

Part c (4 points): "Critical Path design focuses on reducing the longest logic path among the various parts of the processor. Generally, the access time to on-chip memory (i.e., cache), the ALU path, and the microsequencer are the obvious culprits that one focuses on. In designing a version of the LC-3b with a single cycle cache, I first computed the delay for each of these three paths by measuring (a) the time it takes to load MDR, once the MAR is loaded, (b) the time it takes the ALU to read SR1 and SR2, perform an XOR, and load DR with the result, and (c) the time it takes to compute the address of the Control Store ROM, read its contents and latch the result into the microinstruction register. Since (a) was the longest, I focused on shortening that path."
What is wrong with the approach described above? Explain in no more than 20 words.

Name:_____

Problem 1 continued

Part d (4 points): Is the size of the TLB part of the ISA or part of the microarchitecture? Explain in no more than 20 words.

Part e (4 points): The MIPS ISA originally allowed three-cycle MULs to be followed by single cycle ADDs (as back to backs) in a pipelined implementation without a hardware interlock. Explain how they did it.

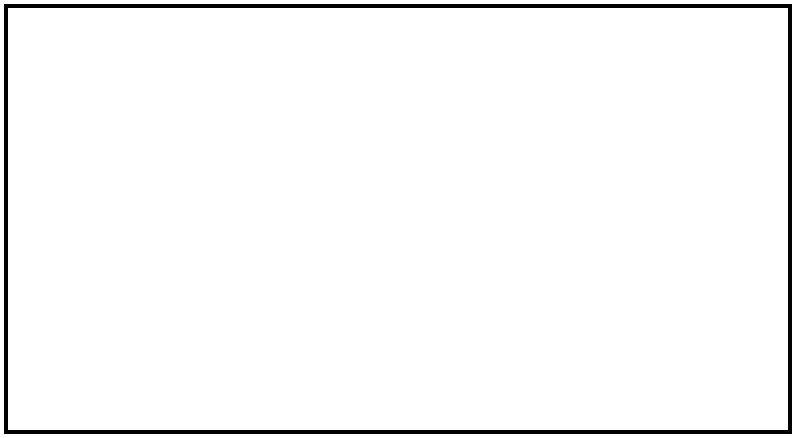
Name:_____

Problem 2 (10 points)

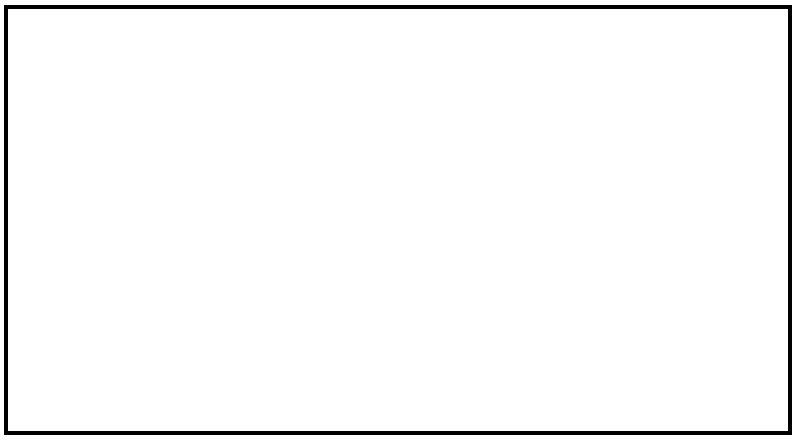
Part a (2 points): The LC-3b is little endian. That means `STW R0,A` means R0[15:8] gets stored into location [ ] and R0[7:0] gets stored in location [ ]. Assume A is an even address.

Part b (8 points): An enterprising engineer suggests that it would be worthwhile to produce a big endian LC-3b that could be used with data sets consisting of big endian values. Assume a compiler is available to produce code in big endian form.

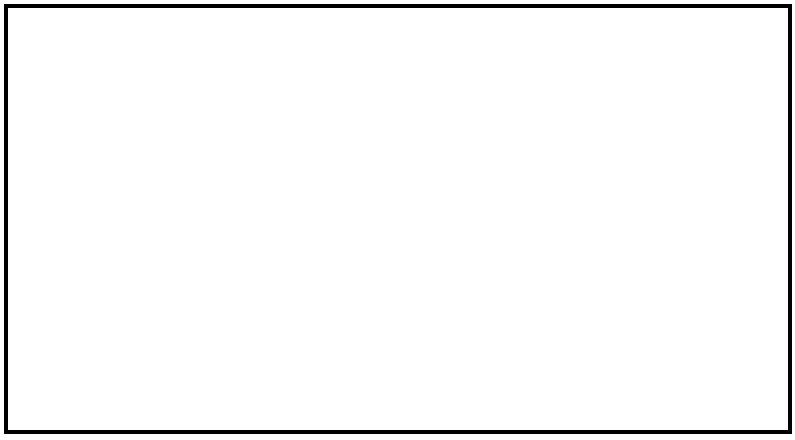
What very small changes to the data path and/or microsequencer and/or state machine are necessary to change LC-3b to a big endian LC-3b? Show by means of diagrams all changes that are necessary:



Datapath



Microsequencer



State Machine

Name:_____

Problem 3 (10 points)

Suppose our ISA had two different add instructions, the classical load-store `ADD Ri,Rj,Rk` that is present in the LC-3b, and an `ADDM Ri,Rj,Rk` instruction where all operands are in memory. That is, the semantics of `ADDM` is:

M[Ri] ← M[Rj] + M[Rk].

Part a (5 points): A programmer wrote the high level language statement A[i] = B[i] + C[i] that is part of the loop body of a for loop. Should the compiler generate the ADD instruction or the ADDM instruction? Explain why.

Part b (5 points): A programmer wrote the high level language statement SUM = SUM + A[i] that is part of the loop body of a for loop. Should the compiler generate the ADD instruction or the ADDM instruction? Explain why.

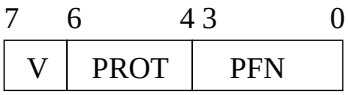
Name:_____

Problem 4 (25 points)

Consider a byte-addressable memory system with two levels of virtual to physical translation (like the VAX).

Virtual Address Space: 256B, partitioned into equal regions of process space and system space. Bit[7] of the VA identifies the region, bit[7]=0 identifies process space. Page size: 4 bytes.

The system **does not** include a Translation Lookaside Buffer. The Page Table Entries are one byte each and the format is as follows:



For the Process P1:

PBR (process base register) is x90.
PLR (process length register) is 9.

SBR (system base register) is x18
The SLR (system length register) is 8.

Shown below are the contents of some physical memory locations:

x0	x94	x8	x9B	x10	x82	x18	x8A
x1	x0C	x9	xB8	x11	x62	x19	x85
x2	x01	xA	x04	x12	xFF	x1A	x8C
x3	x02	xB	x91	x13	xF4	x1B	x99
x4	x34	xC	x1F	x14	xF3	x1C	x82
x5	x86	xD	x82	x15	x94	x1D	x05
x6	xAF	xE	x9A	x16	x00	x1E	x80
x7	x80	xF	x12	x17	x10	x1F	x09

Part a (8 points): Given Virtual Address 00001101 (x0D), what is the corresponding physical address?

Answer:

Name:_____

Problem 4 continued

Part b (3 points): How many pages of process page table and system page table are resident in physical memory?

process page table: [ ] **system page table:** [ ]

Part c (3 points): Which frame(s) contain the process page table and which frame(s) contain the system page table?

process page table: [ ] **system page table:** [ ]

Part d (5 points): Which address ranges in process space will generate a page fault when accessed?

[ ]

Part e (6 points):
If page X of Process P1 and page Y of Process P2 are mapped to the same frame, page X of P1 is said to be shared with page Y of P2.

Is the virtual address 00000101(x05) of Process P2 in a page of P2 which is shared with a page of Process P1? If yes, what is the corresponding virtual address in Process P1? The PBR of Process P2 is x84 and the PLR of Process P2 is 4.

Answer: [ ]

Name:_____

Problem 5 (15 points)

We wish to use memory chips having 30 nsec access time with a 100 MHz processor. Our objective is to design a 32KB byte-addressable physical memory. The system design restricts us to using exactly eight memory chips. The processor/memory data bus is 16 bits.

Part a (1 point): How many processor cycles does a single memory access take?

Part b (2 points): Can interleaving improve the performance of this memory for loads from sequential memory locations? If yes, what is the minimum number of banks required for optimal performance? If no, why not?

Part c (6 points): Three types of memory chips are available for our design:

A. 4K by 4 bits

B. 8K by 4 bits

C. 4K by 8 bits

Will 8 memory chips of type **A** satisfy the above performance and capacity design constraints? Explain.

Name:_____

Problem 5 continued

Will 8 memory chips of type **B** satisfy the above performance and capacity design constraints? Explain.

Will 8 memory chips of type **C** satisfy the above performance and capacity design constraints? Explain.

Part d (3 points):
Next step is for you to design a full 32KB, optimal performance memory system to go with our 100 MHz processor, using whichever of the three types of memory chips you wish. Which type of memory chip have you selected?

Identify on the detailed address register shown below what each of the 15 address bits are used for (rank, chip address, interleave, byte on bus):

Part e (3 points):
The next step is to wire it up. Unfortunately, an Aggie was hired to do the job. After he finished, we noticed that instead of memory locations 2,3 being on bank 1, they were on bank 0, along with locations 4,5,6, and 7. Memory location 8 was on bank 1, memory location 16 on bank 2, etc.

On the address register shown below, identify what each of the 15 address bits were actually used for after the Aggie wired it up.

Name:_____

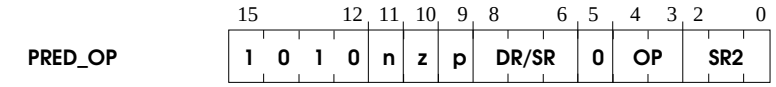
Problem 6 (20 points)

We wish to use the unused opcode 1010 to implement a new instruction PRED_OP, which conditionally executes the operation OP, depending on the state of the condition codes the same way that BR conditionally branches. OP can be ADD, AND, or XOR.

Semantically, the instruction is processed as:

```
IF([N AND n] OR [Z AND z] OR [P AND p])
      DR/SR = DR/SR OP SR2
      setcc();
```

PRED_OP has the following format:



Note that bits[8:6] specifies a register that is both source and destination, like the two-address machine x86.

Bits[4:3] are encoded as follows:

ADD/00
AND/01
XOR/10
11 will never be used

Name:_____

Problem 6 continued

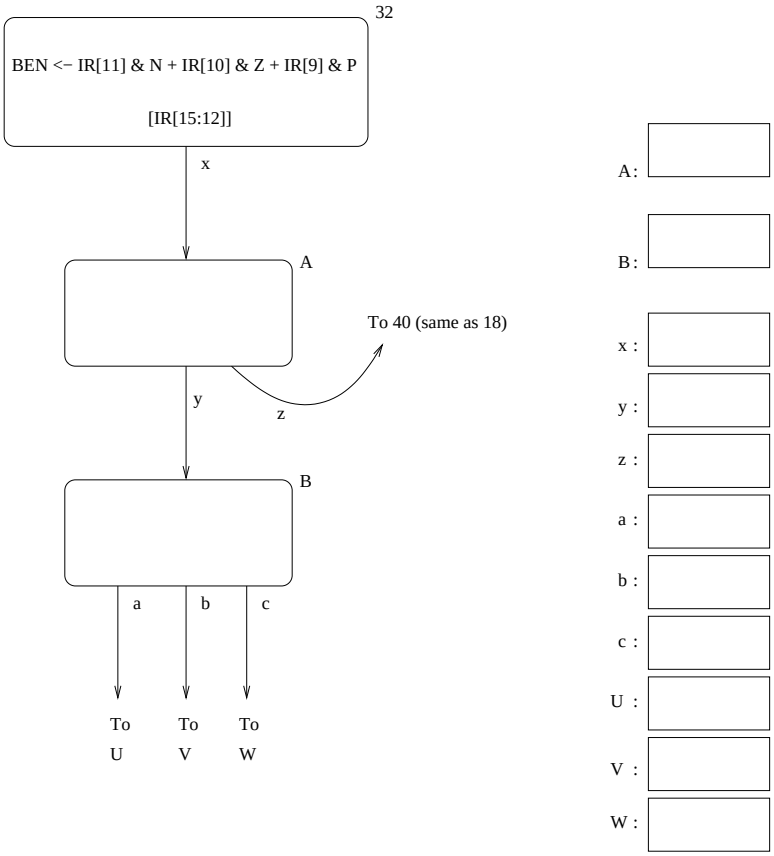
Changes to the state machine (10 points):

To implement PRED_OP, minor changes to the microsequencer and data path of the LC-3b are required. The result is the updated state machine, as shown below.

Your job: Make the changes to the state machine, the microsequencer and the data path required to add PREP_OP to the instruction set.

First describe what goes on in states A and B, and identify the binary representations for A,B,x,y,a,b,c,d,U,V,W in the state machine.

Note that the microinstruction at state 40 is the same as the microinstruction at states 18 and 19.



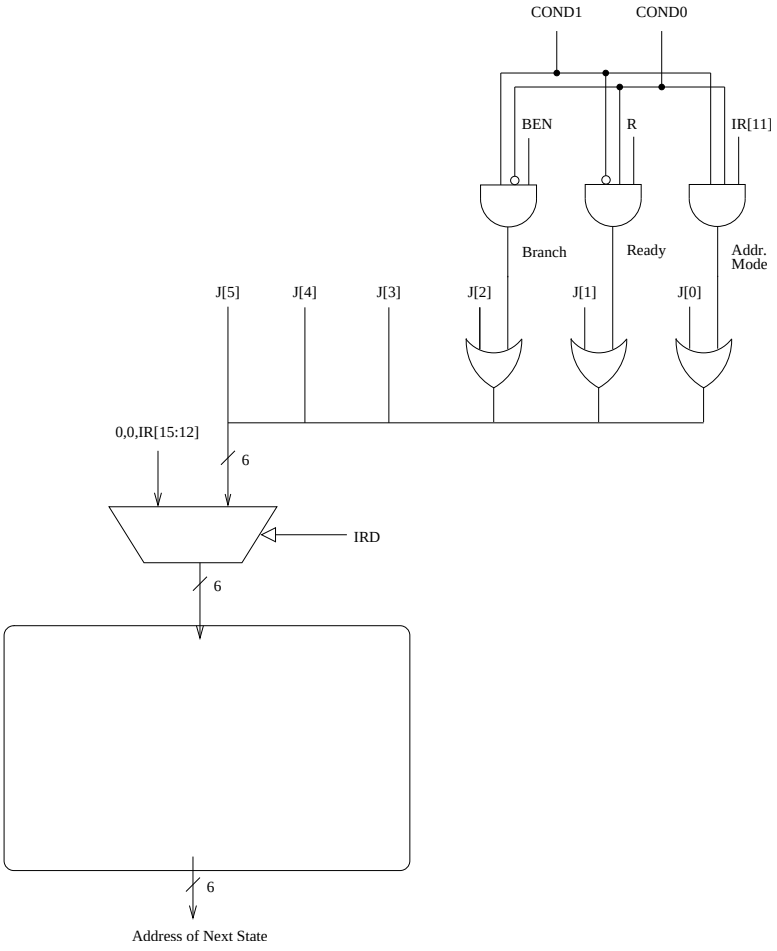
Name:_____

Problem 6 continued

Changes to the Microsequencer (6 points):

To have the state machine behave as described, you must do a 4-way branch from state B. One control signal plus a very simple logic block is all that is needed to make that happen. Add the logic block and control signal in the box provided to make that happen.

In what states is the new signal asserted? [ ]

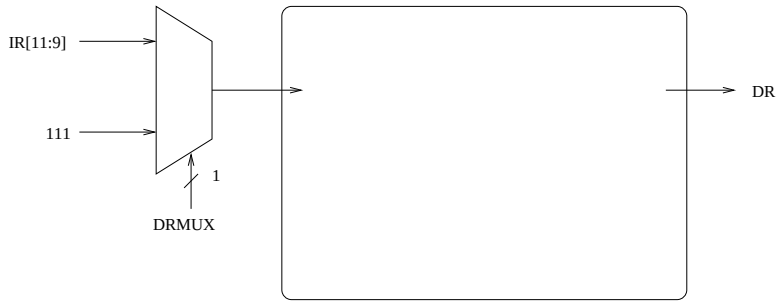


Name:_____

Problem 6 continued

Changes to the Data Path (4 points):

Only one small change is needed to the data path. Add the logic required to make that happen in the box provided.



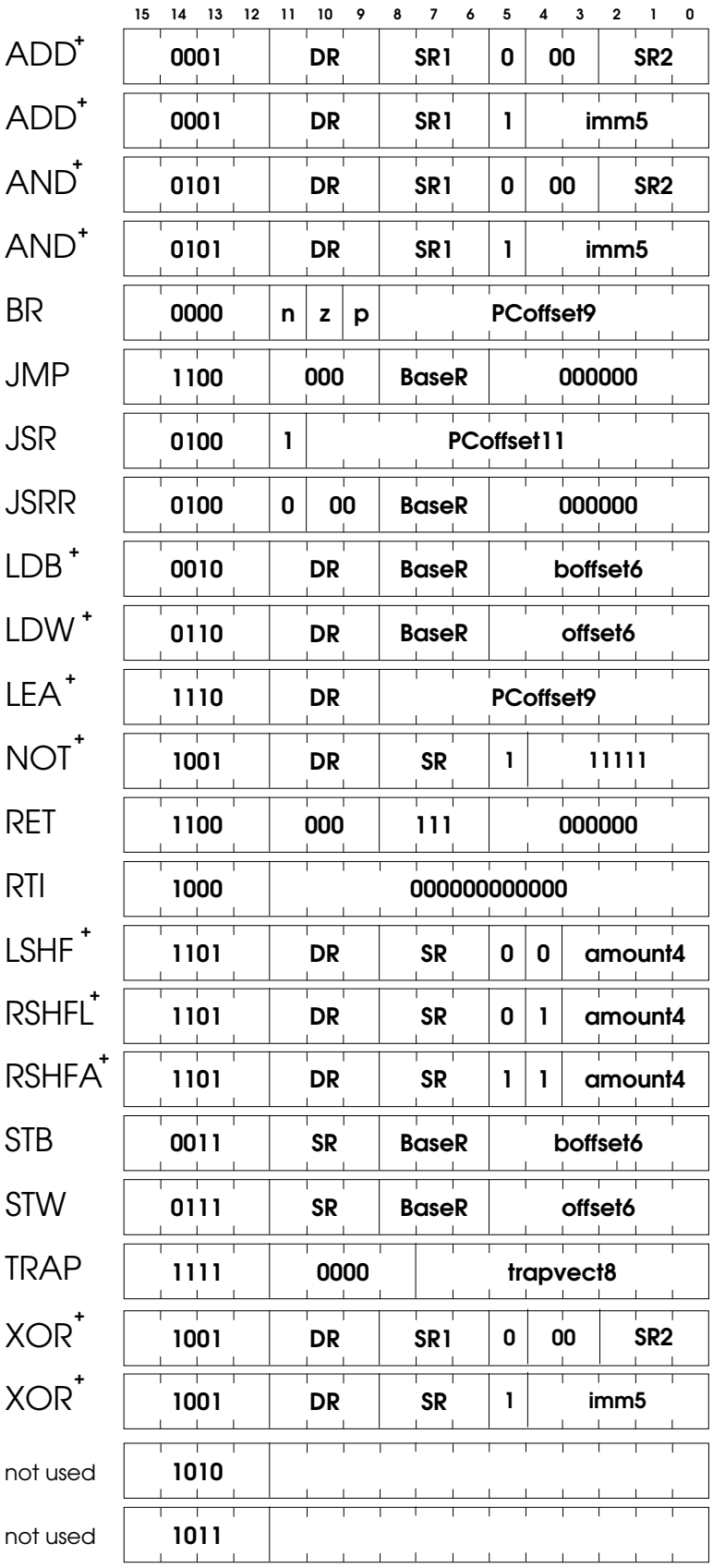


Figure 1: LC-3b Instruction Encodings

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	

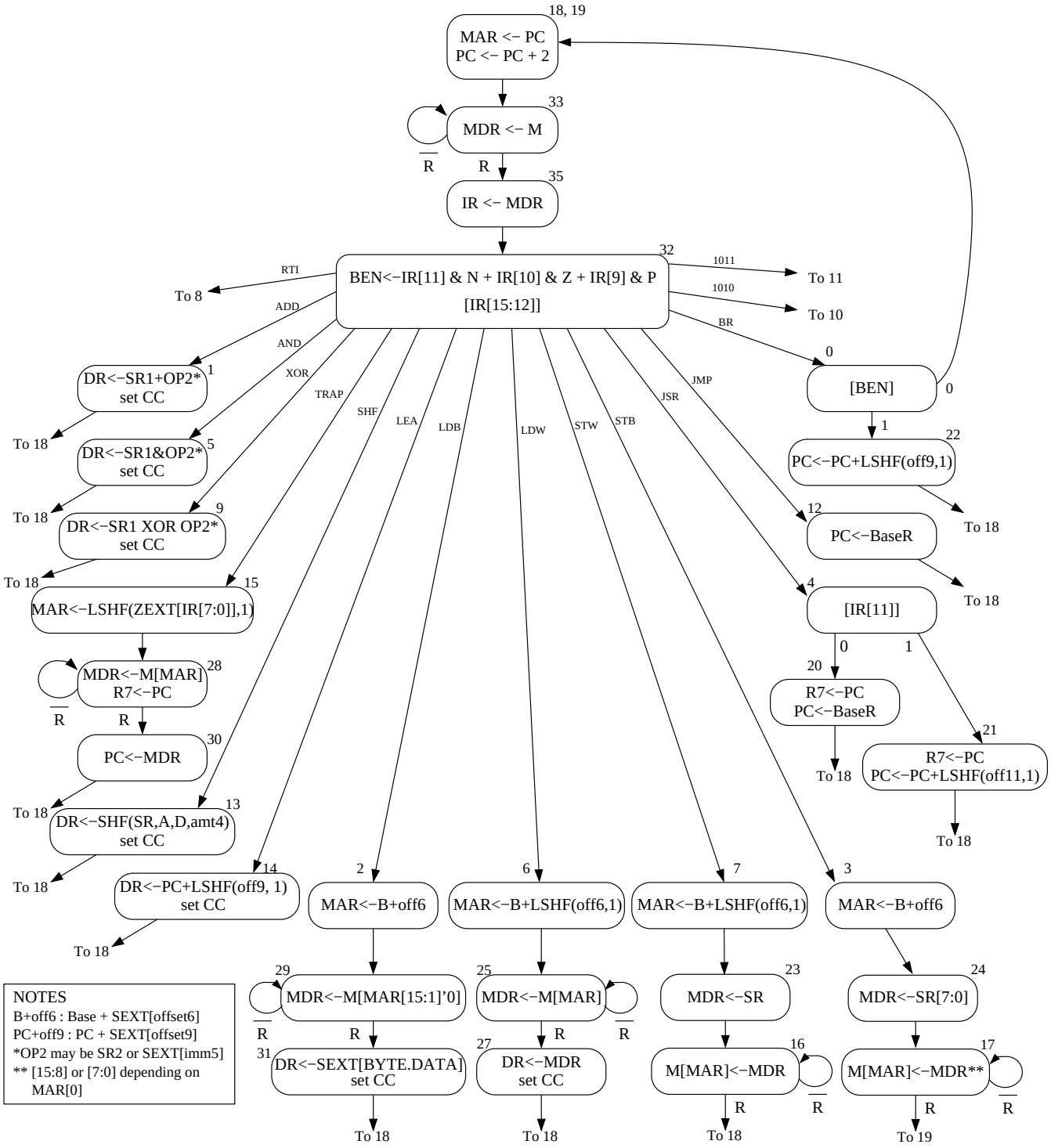


Figure 2: A state machine for the LC-3b

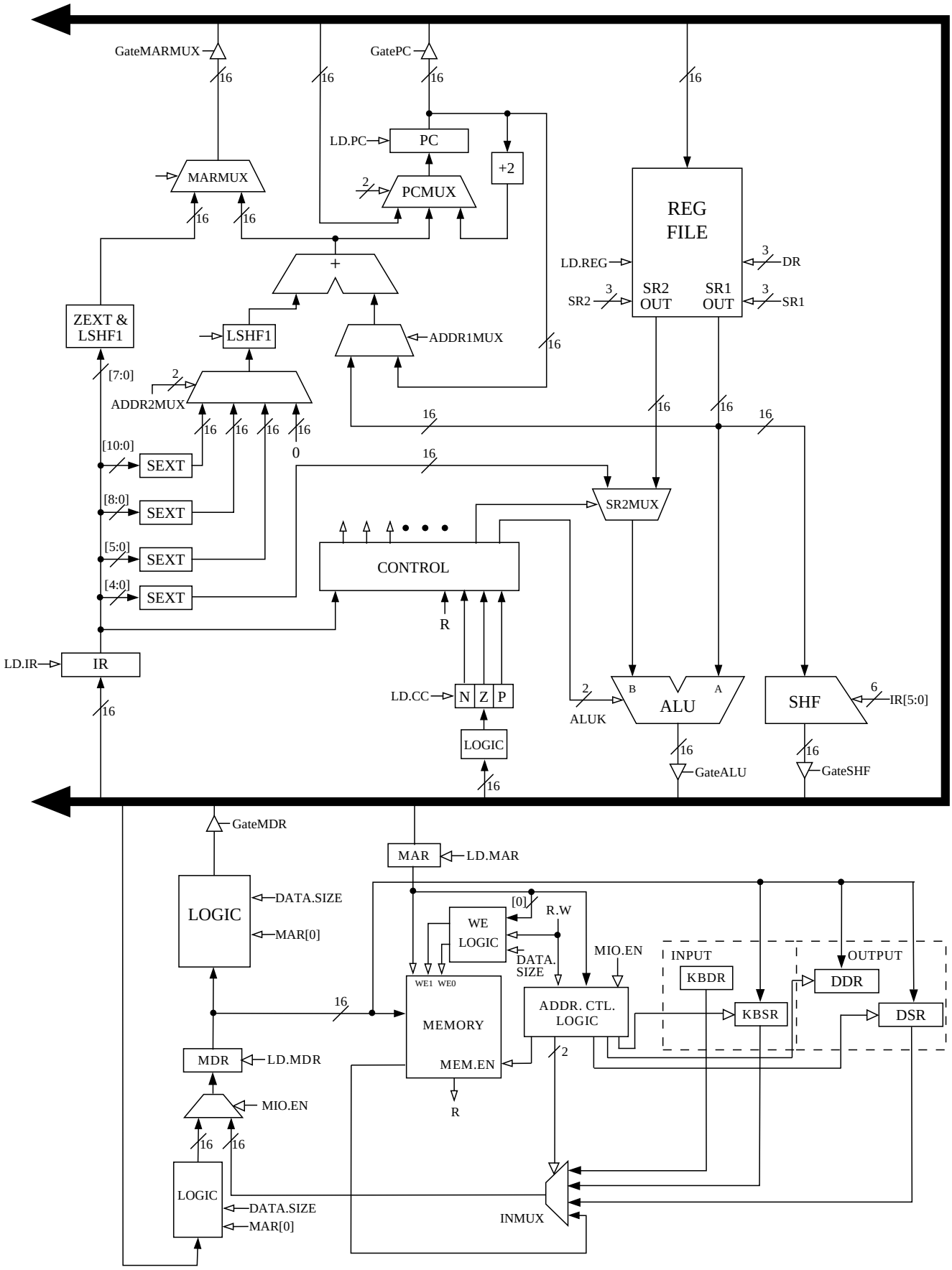


Figure 3: The LC-3b data path

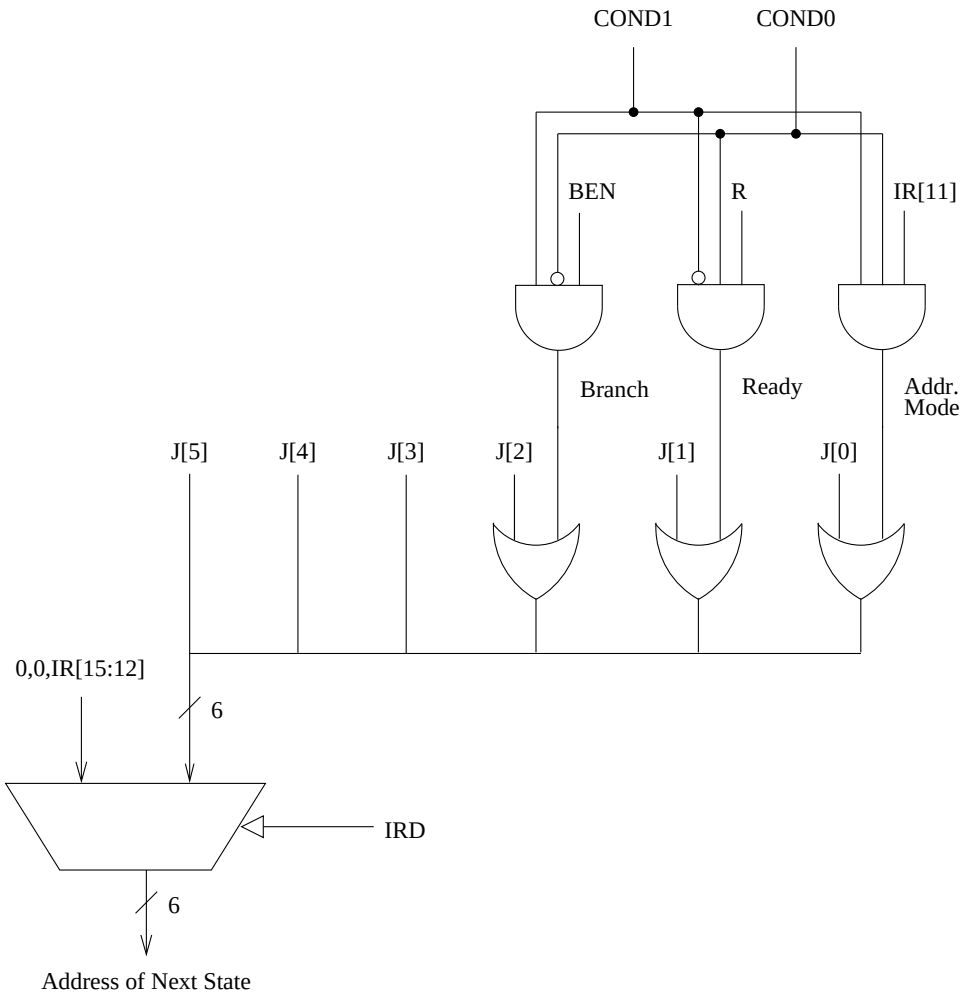


Figure 4: The microsequencer of the LC-3b base machine