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

EE 460N Spring 2015
Y. N. Patt, Instructor
Ben Lin, Kishore Punniyamurthy, Will Hoenig TAs
Exam 1
March 11, 2015

Name:____

Problem 1 (20 points):_______

Problem 2 (30 points):_______

Problem 3 (25 points):_______

Problem 4 (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.

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

Signature:____

GOOD LUCK!

Name:___________________________________

Problem 1 (20 points)

Part a (5 points): The x86 ISA has variable length instructions. This characteristic of the ISA has pluses and minuses.

The major plus is

The major minus is

Part b (5 points): If two locations are in the same row buffer on a DRAM, the only bits of their corresponding addresses that are different are the

Part c (5 points): If the contents of a memory location is protected with a parity bit, and two bits are inverted during transmission, what happens? Is this a problem? Why or why not?

Part d (5 points): If two processes translate different virtual addresses in their own virtual address space to the same physical address, and the cache is virtually indexed, physically tagged, the location corresponding to that one physical address can be present in two different locations in the cache. We call the two instances of the same physical cache line $\boxed{}$ and the problem is referred to as the $\boxed{}$.

Name:____________________________________

Problem 2 (30 points)
A computer system contains a physically-indexed, physically-tagged, 2-way set associative, write-back cache. The ISA specifies an n-bit physical address space, and an 128 byte page size.

With the cache initially empty, the processor makes ten consecutive memory reads, as shown in the table below. Note that some result in cache hits, others result in cache misses. Note that the actual number of bits of physical address is not shown.

	Physical Address	Hit/Miss
1	00...0000010000	Miss
2	00...0100000101	Miss
3	00...0000011000	Hit
4	00...0110000111	Miss
5	00...0111100001	Miss
6	00...0100000010	Hit
7	00...0110100111	Miss
8	00...0100100000	Miss
9	00...0111100010	Miss
10	00...0100001111	Hit

In order to concurrently access the TLB, the Tag Store, and the Data Store, the index bits are selected from the page offset, i.e., they are not affected by the virtual to physical address translation.

Part a (10 points): What is the cache line size? Why?

Part b (10 points): What are the index bits? Why?

Part c (5 points): What is the size of the cache (i.e., how many bytes of data can be stored in the cache)?

Part d (5 points): Given that the tag store requires 68 bits, and that the cache uses perfect LRU replacement, what is the physical address space of memory for this computer system.

Name:____________________________________

Problem 3 (25 points)

The following program fragment executes on a machine that supports virtual memory.

```
LD R0, A        ;R0 <-- M[A]
LD R1, B        ;R1 <-- M[B]
ADD R1, R1, R0 ;R1 <-- R1 + R0
ST R1, C        ;M[C] <-- R1
```

A, B, and C are virtual addresses.

The ISA specifies:
Virtual address space: 64KB
Physical address space: 8KB
A page is 256 bytes

The memory management system uses the two-level page table scheme similar to the VAX. Virtual memory is partitioned into two halves. User space starts at x0000, System space starts at x8000. A PTE is 16 bits. For purposes of this exam only, we will assume that a PTE has the following form:

V	00..0	PFN

Assume no cache and no shared memory(that is, no two pages map to the same frame). Assume the TLB only contains PTEs for pages in user space.

For the snippet of code above, if one gets no TLB hits, the processor makes nine accesses to physical memory (we are ignoring the fetching of instructions). The table below shows the sequence of nine memory accesses needed to do the job.

Virtual Address	Physical Address	Data
	x1AA8	
x5400	x0700	x5410
		x8008
x80C2		
	x0646	
		x8016
		x8005
x7618		x5824

On the other hand, IF the TLB initially contained the entries shown below,

V	Page No	PTE
1	x14	x8001
1	x23	x8006
1	x28	x8004
0	-	-

the processor would only need to make seven accesses to physical memory.

Your job: Complete the table.

Name:__________________________

Problem 4 (25 points)

Let's use one of the unused opcodes to add an instruction AVG (i.e., average) to the LC-3b ISA. Its format will be

	15			12	11 9	8 6	5	4 0
AVG	1	0	1	0	DR	SR1	1	imm5

	15			12	11 9	8 6	5 3	2 0
AVG	1	0	1	0	DR	SR1	000	SR2

depending on whether the second operand is an immediate or the contents of a register. AVG will sum the n ($n > 0$) 16-bit integers in consecutive memory locations starting at the location specified by SR1, divide that sum by n, and load the result into DR, setting the condition codes. The value n is either an immediate value or the contents of SR2. Assume that the n integers are aligned in memory (i.e. SR1 holds an even number), and assume that DR, SR1, and SR2 (if SR2 is being used) all refer to different registers.

For this problem, you can assume no overflows will occur. Note: execution of this instruction will destroy the initial contents of SR1.

Your job: augment the LC-3b state machine, the data path and the microsequencer as necessary to add AVG to the LC-3b ISA.

Part a (8 points): The state machine (see page 6). From state 32 (the decoder) we go immediately to the eight states needed to carry out the work of the AVG instruction. One of the states has been specified for you, and another (state 39) has been partially specified. **Your job is to complete the specifications of all the states and add the missing state numbers**

Part b (8 points):
The data path (see page 7). To implement AVG you will need additional structures. Four are shown in boldface on the data path diagram.

The DIVIDE UNIT takes two inputs X,Y and produces a result X/Y. The divide is a multi-cycle operation that latches X, Y internally in the first cycle, and signals completion with a DIV_R (for Divide Ready) signal when done. The DIVIDE UNIT starts processing when the control signal DIV is asserted. **Your job is to connect the DIVIDE UNIT to other elements of the data path as needed.**

The CTR is a step-down counter. It can be loaded when a LD.CTR control signal is asserted, and it can be decremented when a DEC.CTR control signal is asserted. **Your job is to connect it to other elements of the data path as needed.**

Registers containing x0000 and x0002 have also been added to the data path. **Your job is to connect it to other elements of the data path as needed.**

Feel free to add tri-state devices for any signals you wish to put on the bus.

There is also a dashed box in the data path. **Your job is to put in that box any other necessary structures(register or combinational logic) to complete the data path for implementing AVG.**

Part c (9 points): The microsequencer (see page 8). The augmented state machine requires additional control provided by the microsequencer. **Your job is to augment the microsequencer.** You will need an additional COND bit, call it COND2, which will be used to modify J[5] and J[4] when necessary. The necessary OR gates have already been put in place. Add whatever additional logic structures you need.

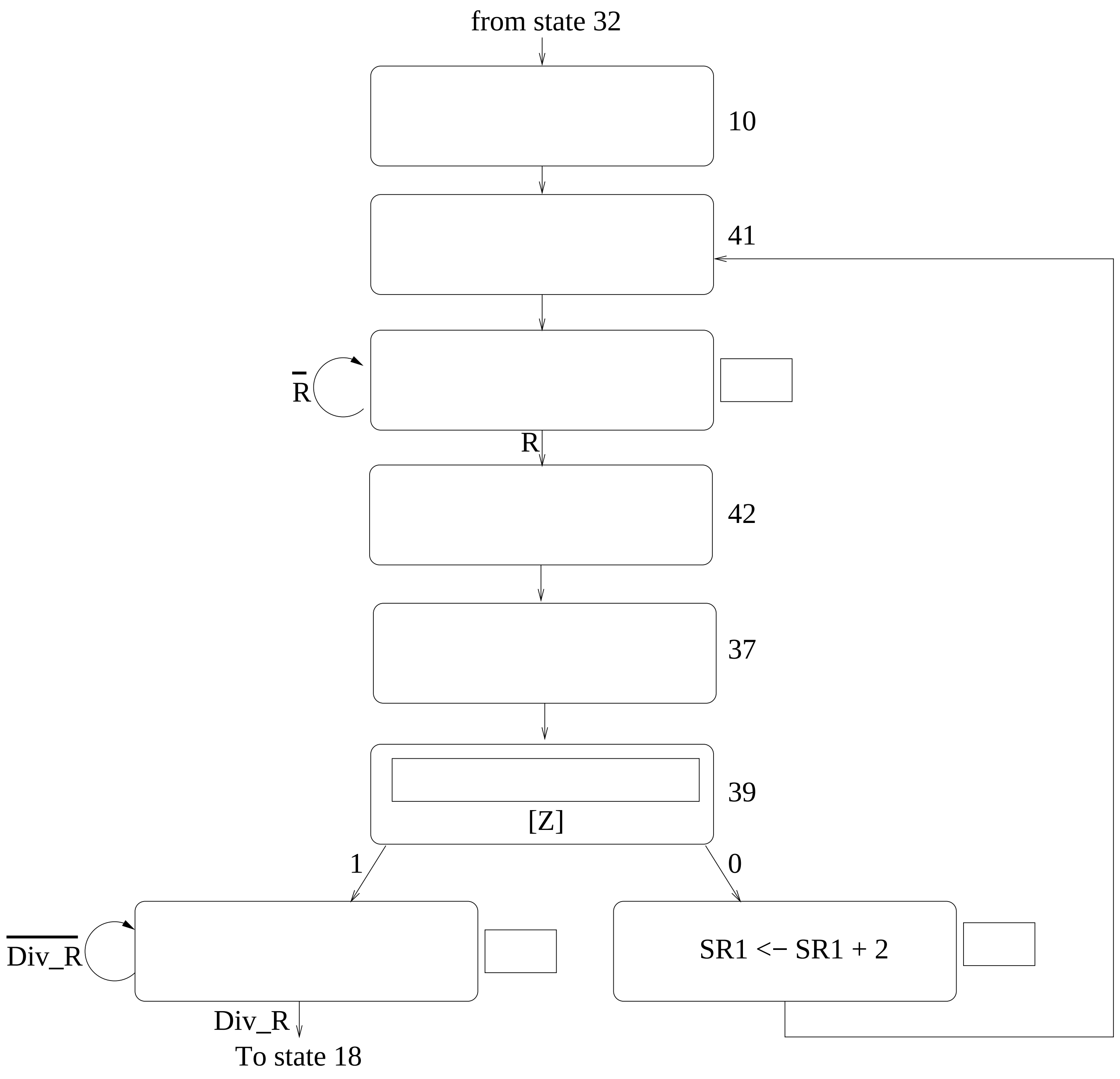

Figure 1: State diagram for AVG instruction

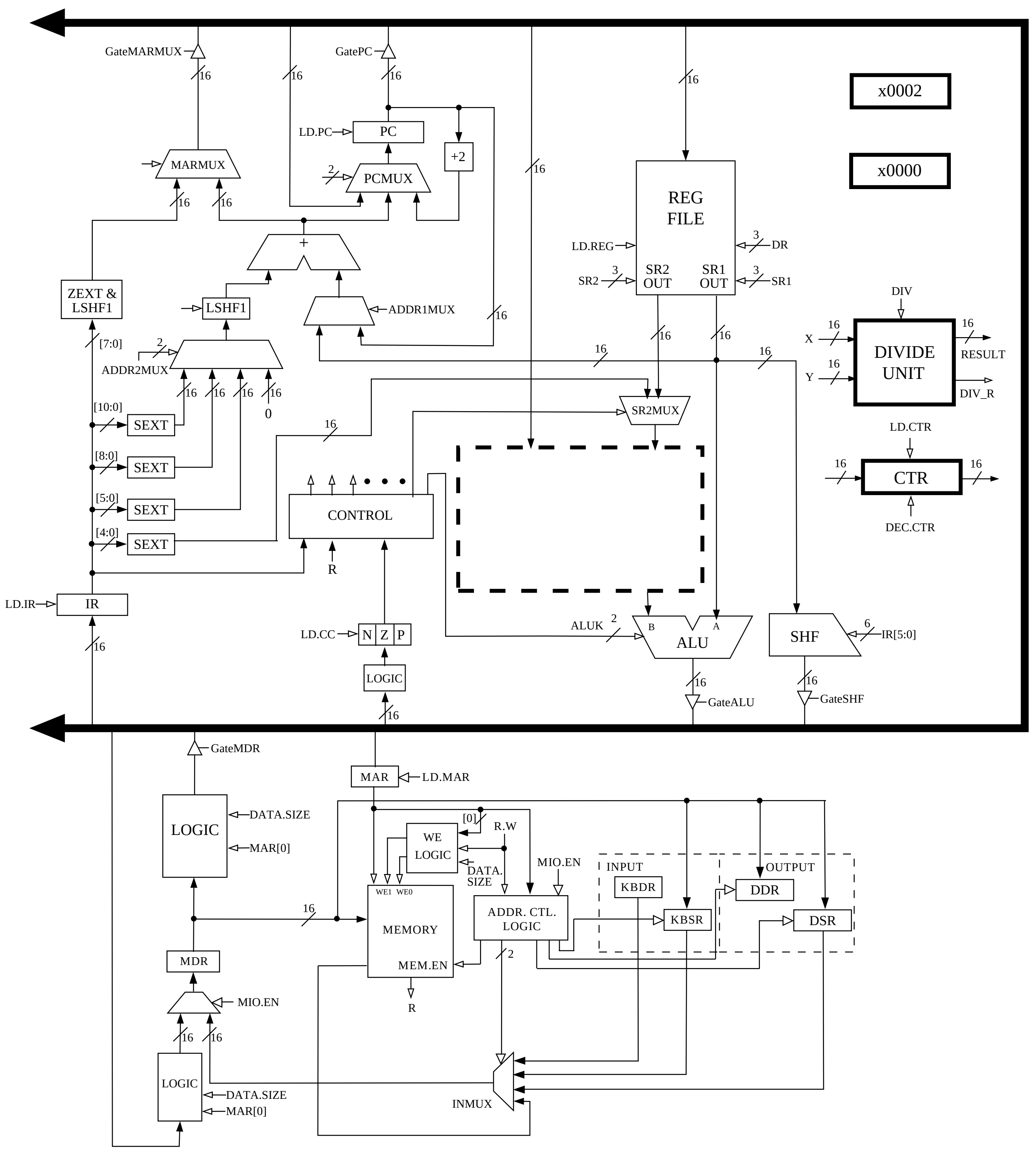

Figure 2: Data path for AVG instruction

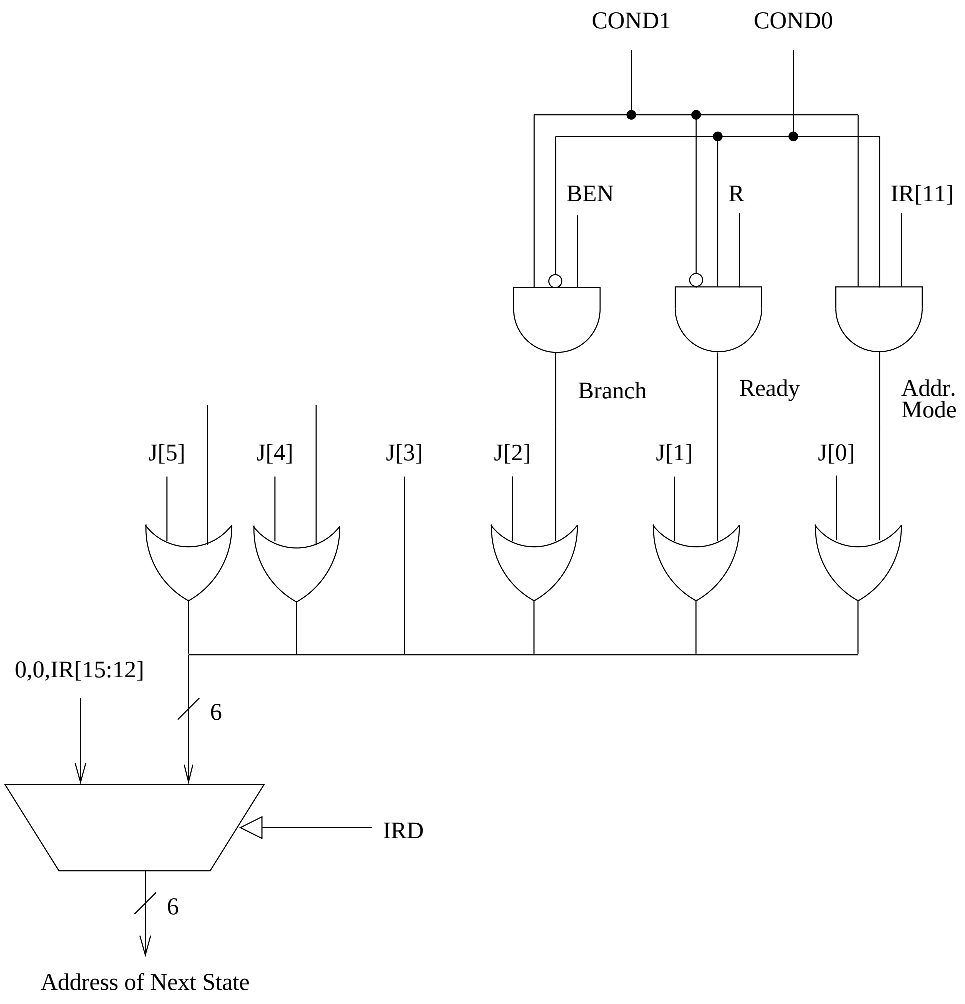

Figure 3: Microsequencer for AVG instruction

	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 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	

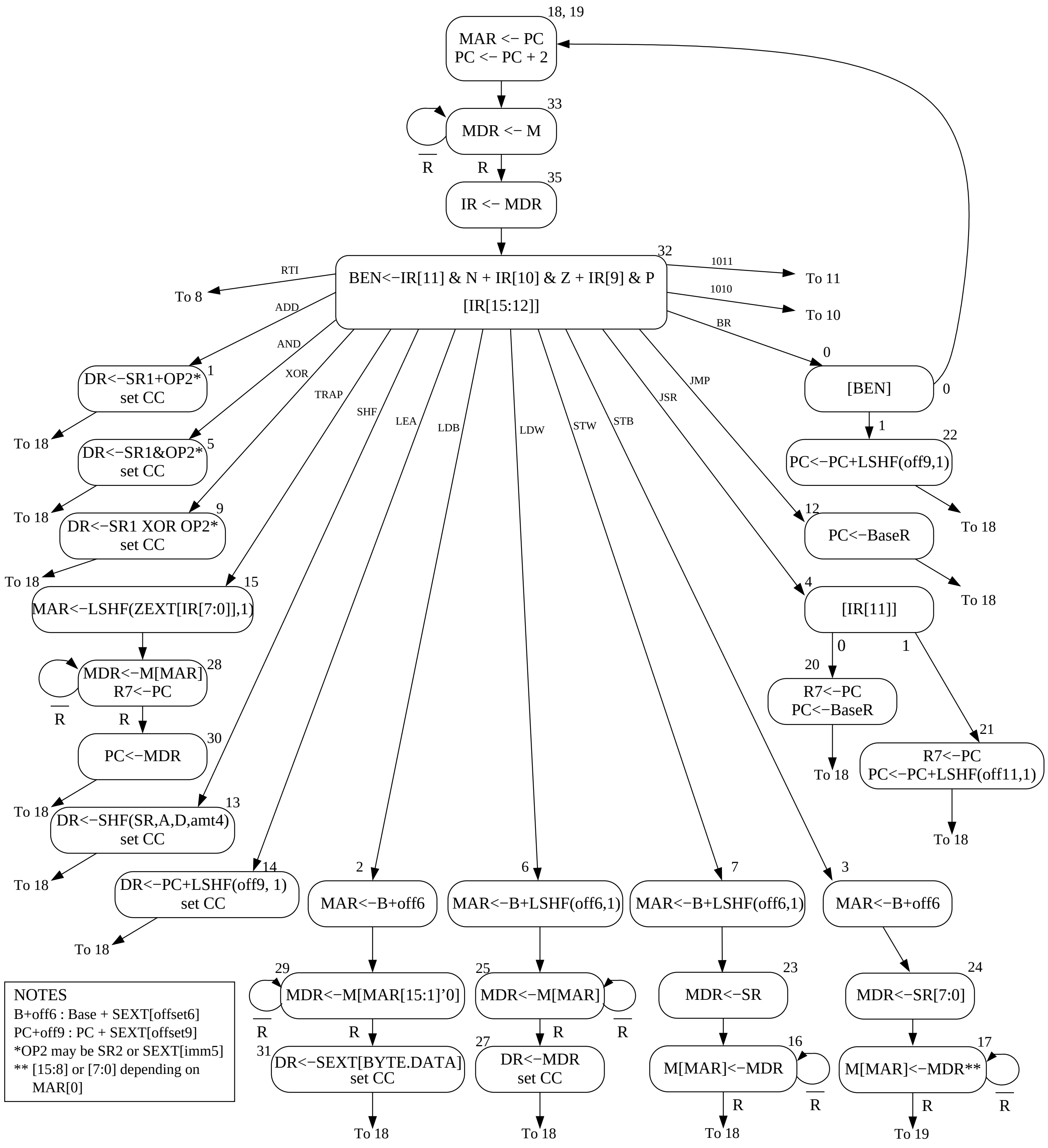

Figure 2: A state machine for the LC-3b

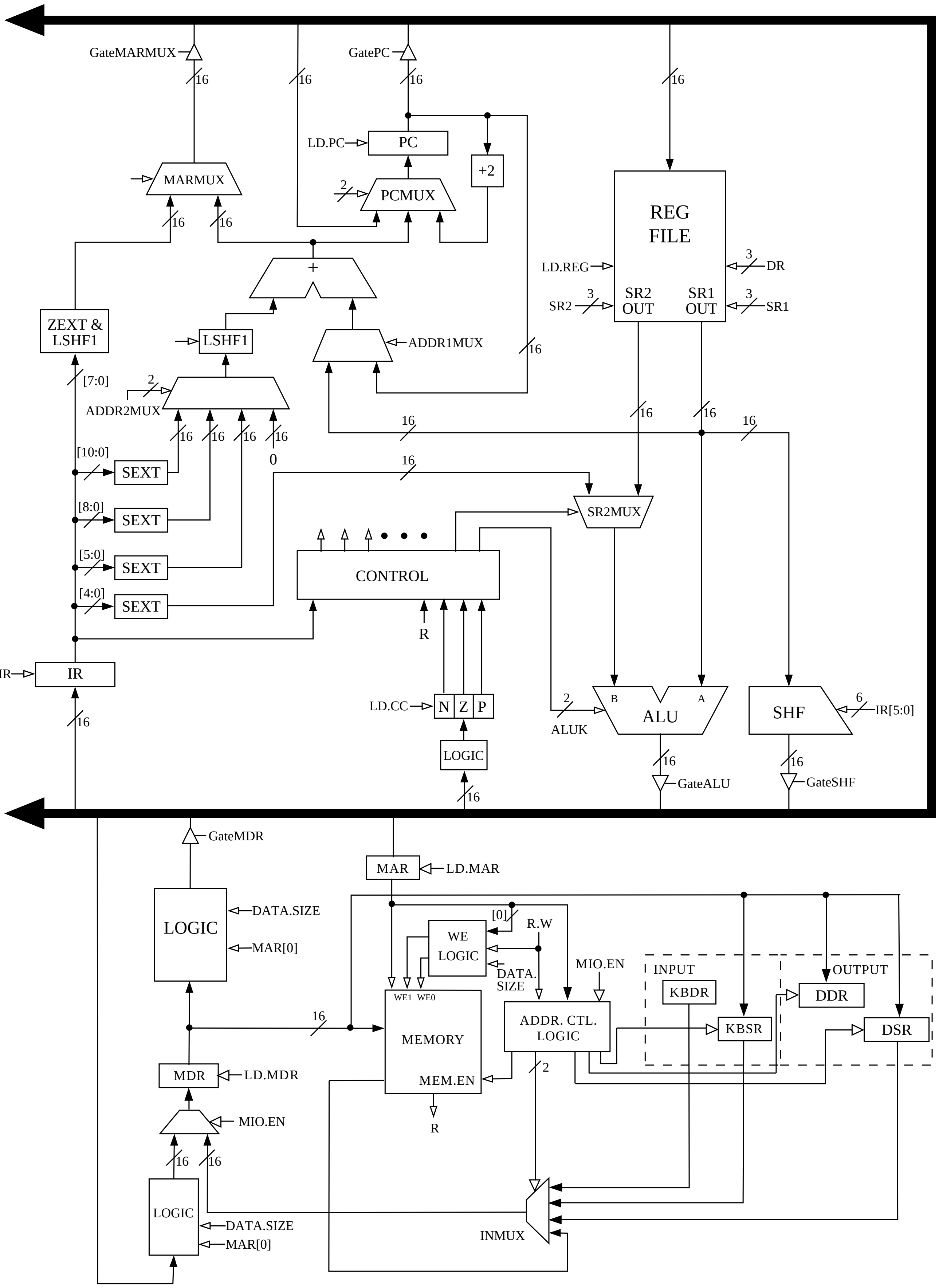

Figure 3: The LC-3b data path

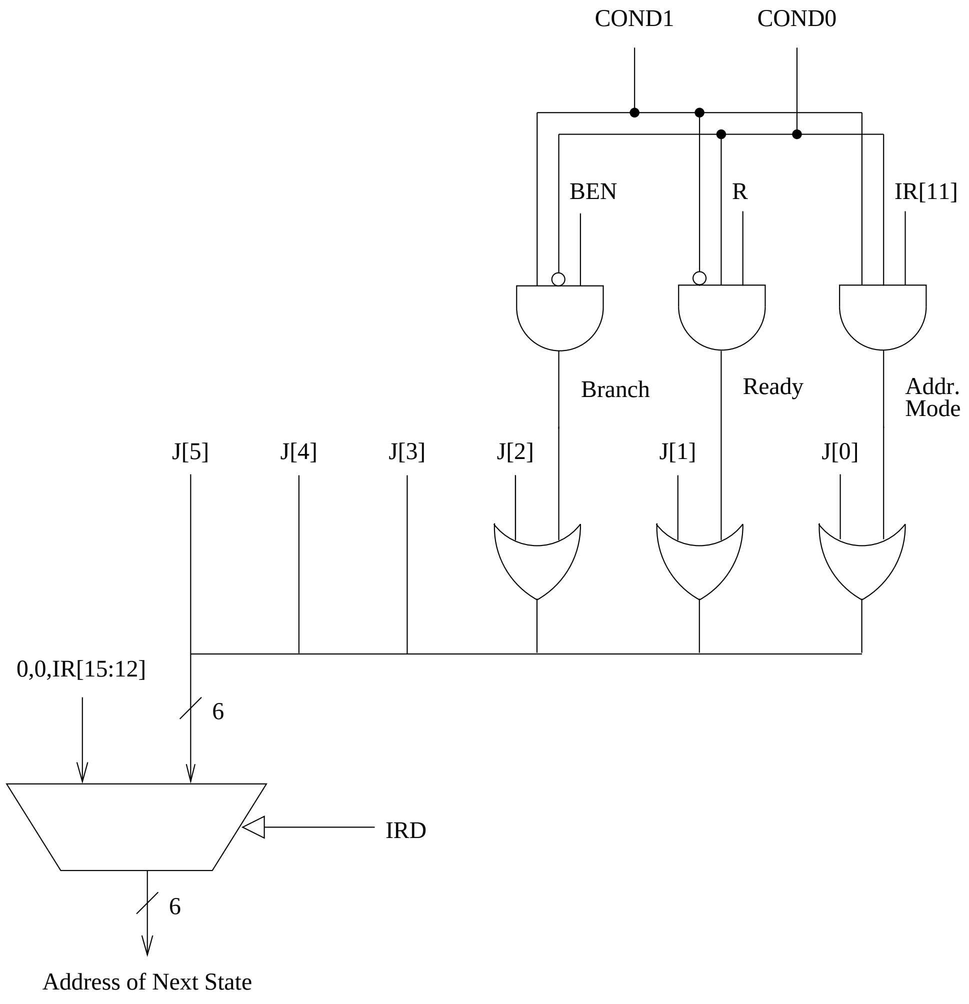

Figure 4: The microsequencer of the LC-3b base machine