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

EE 460N Fall 2014
Y. N. Patt, Instructor
Stephen Pruett, Emily Bragg, Siavash Zangeneh TAs
Final Exam
December 12, 2014

Name:_____

Problem 1 (20 points):_____

Problem 2 (10 points):_____

Problem 3 (15 points):_____

Problem 4 (15 points):_____

Problem 5 (20 points):_____

Problem 6 (20 points):_____

Problem 7 (25 points):_____

Total (125 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): We showed in class that multiplication of two 16-bit binary fixed point numbers consists of a sequence of 16 shifts and adds. Andrew Booth's algorithm reduced that number from 16 to what?

Part b (5 points): Speed-up with p processors is defined as T1/Tp, where T1 is the time to solve the problem with one processor and Tp is the time to solve the problem if you have p processors. What important requirement is there on T1?

Part c (5 points): A four processor system, each processor having its own cache, uses the Directory scheme to maintain cache coherence. The directory stores a bit vector for each "line" (or, "block") of memory, indicating its status relative to the caches.

Assume no cache has line A. Then in sequence: processor 1 wishes to read a value in line A, processor 2 wishes to write a value in line A, processor 3 wishes to read a value in line A. At the end of this sequence, what are the contents of the bit vector for line A.

Part d (5 points): One Pseudo-LRU replacement scheme for 4-way set-associative caches uses 3 additional bits for each set in the cache. What does each of the three bits specify?

Name:_____

Problem 2 (10 points)

Assume IEEE-like floating point, except a data element contains 7 bits, rather than 32 or 64. We can represent the values 14 and 1/32 as 7 bit numbers.

Furthermore, we can multiply 14 times 1/32 with a MUL instruction, which yields the result 7/16 (as we would expect), or we can execute the for loop:

```
result = 0
for (i = 0; i < 12; i = i+1)
    result = result + (1/32)
```

which yields the result 1/4. Why do we get 1/4 instead of the correct value 7/16?

Your job: Complete the specification of the IEEE-like 7-bit floating point number:

Exponent: ____

Fraction: ____

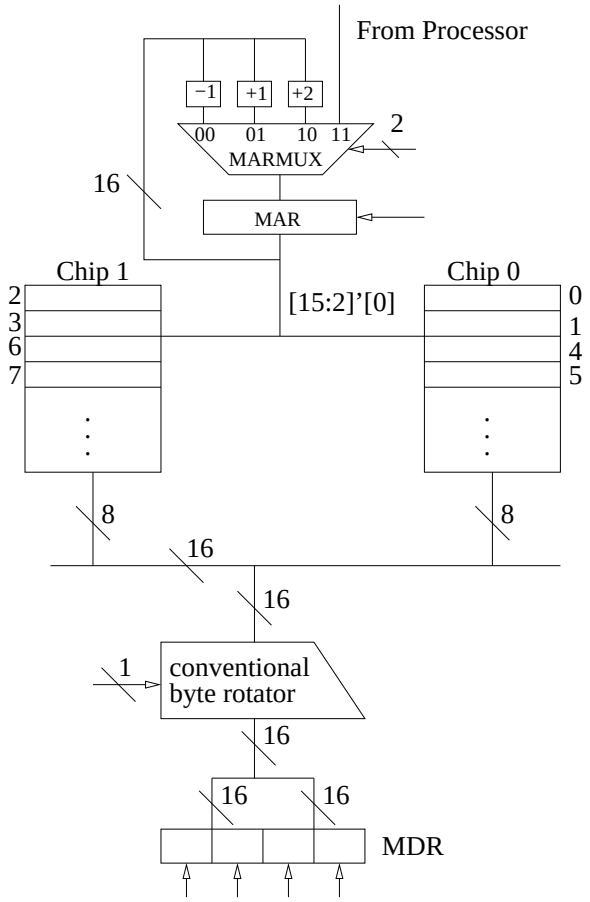
Excess (also called bias): ____

Name:_____

Problem 3 (15 points)

An Aggie messed up the wiring of an 8-bit addressable memory by leaving out bit 1 rather than bit 0, when he wired 15 bits of the MAR to the two memory chips. Instead of having chip 0 contain all even addresses and chip 1 contain all odd addresses, he ended up with what is shown below.

Your job is to implement the memory controller shown on the next page to support unaligned accesses of 32 bit loads.



Note: The conventional byte rotator simply takes a 16-bit vector and swaps the bytes.

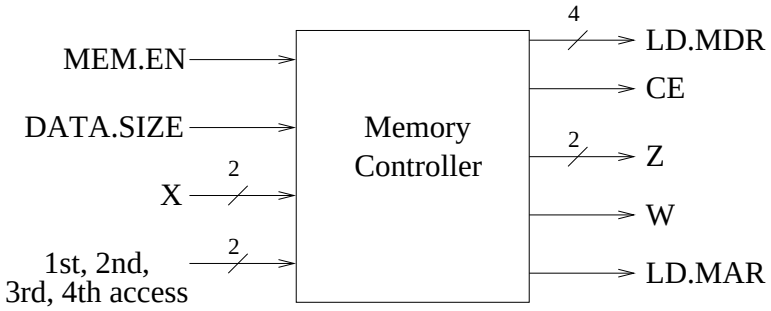
Note: The MDR is a 32 bit register.

Note: The output of the byte rotator is simultaneous applied to both MDR[15:0] and MDR[31:16].

Name:_____

Problem 3 continued

Part a: The memory controller shows a missing input (X) and two missing outputs (Z,W). Label them.



Part b: Given the byte rotator does a conventional byte rotate, what is the maximum number of bytes one can load into the MDR in each cycle.

Part c: For the case where data size is 32 bits, complete the truth table for the 4 input combinations labeled A, B, C, D:

	X	1st,2nd,3rd,4th access	LD.MDR3	LD.MDR2	LD.MDR1	LD.MDR0	Z1	Z0	W
	00	00	*****	*****	*****	*****	*****	*****	*****
	00	01	*****	*****	*****	*****	*****	*****	*****
	00	10	*****	*****	*****	*****	*****	*****	*****
	00	11	*****	*****	*****	*****	*****	*****	*****
A	01	00							
B	01	01							
C	01	10							
D	01	11							
	10	00	*****	*****	*****	*****	*****	*****	*****
	10	01	*****	*****	*****	*****	*****	*****	*****
	10	10	*****	*****	*****	*****	*****	*****	*****
	10	11	*****	*****	*****	*****	*****	*****	*****
	11	00	*****	*****	*****	*****	*****	*****	*****
	11	01	*****	*****	*****	*****	*****	*****	*****
	11	10	*****	*****	*****	*****	*****	*****	*****
	11	11	*****	*****	*****	*****	*****	*****	*****

Name:_____

Problem 4 (20 points)

Assume a Tomasulo-style, out-of-order execution machine that handles ADD and MUL instructions. Instructions are of the form ADD Rx,Ry,Rz and MUL Rx,Ry,Rz, as discussed in class. Each instruction requires a fetch cycle, a decode cycle, some number of execution cycles, and a final cycle to store the result into a register and/or a reservation station entry waiting for that result. A result is available to subsequent instructions after it is stored in a register or reservation station entry.

Assume a program consists of four instructions. The first instruction is fetched in cycle 1.

Shown below is a snapshot of the register file before cycle 1, the register file at the end of cycle x, and the reservation stations at the end of cycle x.

Reservation station entries are allocated in order, from top to bottom. That is, for example, the first MUL is allocated to the top reservation station entry associated with the mul functional unit, the second MUL with the second reservation station entry, etc. Each instruction remains in its reservation station until its result is stored in its destination register.

R0	1	–	1
R1	1	–	2
R2	1	–	3
R3	1	–	4

Figure 1: Registers Before Cycle 1

R0	1	–	5
R1	1	–	2
R2	1	–	3
R3	0	β	–

Figure 2: Registers After Cycle X

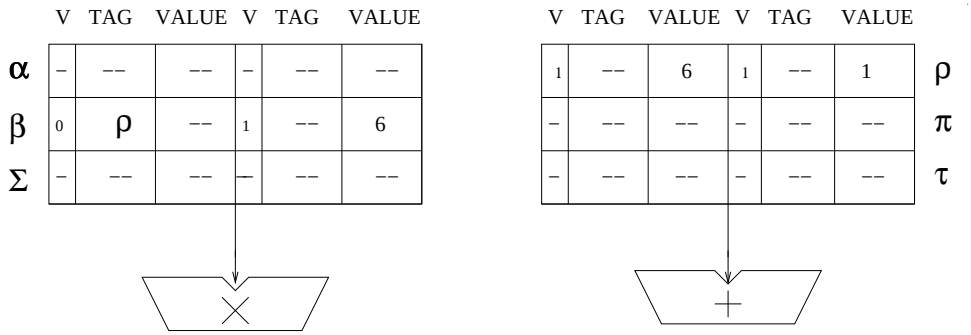


Figure 3: Reservation Stations after Cycle X

The machine has one add functional unit and one mul functional unit. Neither is pipelined. The timing diagram below indicates the cycles (with the letter E) that each functional unit is busy in the execution phase of an instruction.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Adder						E	E	E	E							
Multiplier			E	E	E	E					E	E	E	E		

Name:_____

Problem 4 continued

Part a (5 points): What is X (There are more than one correct answer. We are only asking for one of them)?

Part b (15 points): Complete the table by filling in the four instructions.

Instruction	Opcode	DR	SR1	SR2
I1				
I2				
I3				
I4				

Name:_____

Problem 5 (20 points)
Assume 16-bit physical address space, byte-addressible memory, and a 512 byte, 2-way set-associative, write-back cache with LRU replacement that allocates on a write-miss. Assume the cache is initially empty.

Part a (10 points):
Suppose we execute the following code segment which copies an array A[256][4], whose starting address is x1000 into an array B[256][4], whose starting address is x2000. Each element in both arrays is one byte.

```
for (i=0;i<255;i++)
    for (j=0;j<4;i++)
        A[i][j] = B[i][j];
```

We note that the cache hit ratio for executing this code segment is 7/8.

Your job: Determine the cache parameters:

Block Size: ☐

No. of Sets: ☐

Tag Size: ☐

Tag store size: ☐

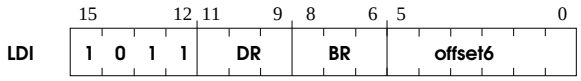
Part b (10 points)
Suppose instead, we execute the following code segment, starting with the same empty cache.

```
for (j=0;j<4;i++)
    for (i=0;i<255;i++)
        A[i][j] = B[i][j];
```

What is the cache hit ratio this time? ☐

Name:_____

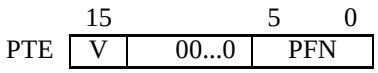
Problem 6 (20 points)
We wish to enhance the LC-3b by adding virtual memory and a new instruction, Load Indirect (LDI). Our virtual memory scheme will follow the VAX scheme which you have already studied. LDI will use the unused opcode 1011, and have the following format:



LDI operates as follows: A virtual address (call it A) is computed by adding the sign-extended offset to the contents of the register specified by BR. The contents of A is a virtual address B. The contents of B are loaded into the register specified by DR.

(If you took EE306, you studied the LDI instruction. Note, however, that the LDI instruction in EE306 computed the address A differently than is shown here.)

The virtual memory model is as follows: Virtual addresses are 16 bits. Physical addresses are 12 bits. Frame size is 64 bytes. Memory is partitioned into system space (x0000 to x7FFF) and user space (x8000 to xFFFF). Each PTE is 2 bytes and has the format



Name:_____

Problem 6 continued

Assume the state of the machine before this instruction is executed includes:

R0: x9500
R1: xA400
PC: x8200

A 4 entry TLB.

V	Page #	PTE	
		V	PFN
1	x090	1	x08
0	-	-	-
0	-	-	-
0	-	-	-

To process LDI R0,R1,#0, seven physcial memory accesses are needed. The table below shows the VA, PA, Data, and whether or not there was a TLB hit for each of these seven physical memory accesses in the order they occurred.

Your job: Complete the table and fill in the additional two boxes. You can assume no page faults occur in the processing of the LDI instruction.

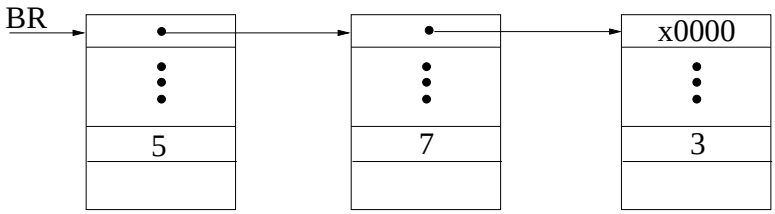
Virtual Address	Physical Address	Data	TLB Hit
	x978		
	x490		
	x600		
		x8006	
x3102		x801A	
xC07E		xC07E	

UBR:

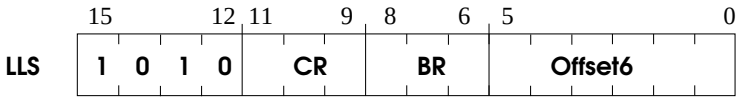
SBR:

Name:_____

Problem 7 (25 points):
The linked list is a useful data structure that connects nodes, each of which is a record consisting of values for a number of parameters. Usually, the first word in a node is a pointer to the next node. The first word of the last node is usually x0000 (the null pointer).



It is common to traverse a linked list searching for a particular value for a particular parameter.
A useful instruction to accomplish this could be added to the LC-3b ISA. Call it LinkedListSearch (LLS), which uses opcode 1010, and has the following format:



CR contains the value one is searching for. BR contains a pointer to the first node in the linked list. offset is the offset from the starting address of a node where the value of the parameter being queried is contained.
LLS operates as follows:

```
while (BR != 0) {
    if (CR == M[BR + Offset6])
        return
    else
        BR = M[BR]
}
```

After LLS executes, BR contains the address of the first node that contains the value we are looking for, or it will contain x0000, the null pointer signifying that no node contained that value.
Note that LLS modifies condition codes to an undefined state.

Name:_____

Problem 7 continued:
Part a : Complete the state machine for this instruction.

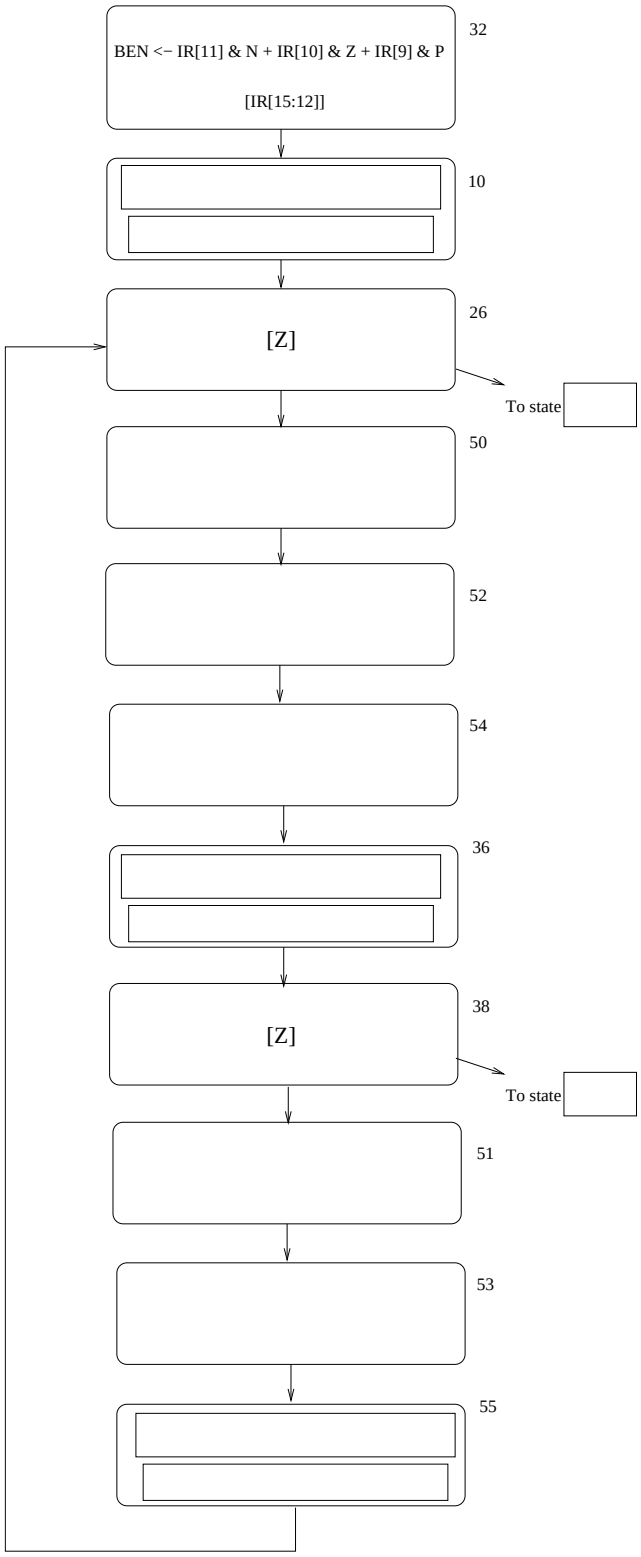


Figure 4: State diagram for LLS instruction

Name:_____

Problem 7 continued:
Part b : Add the additional data path required by filling in the dashed box.

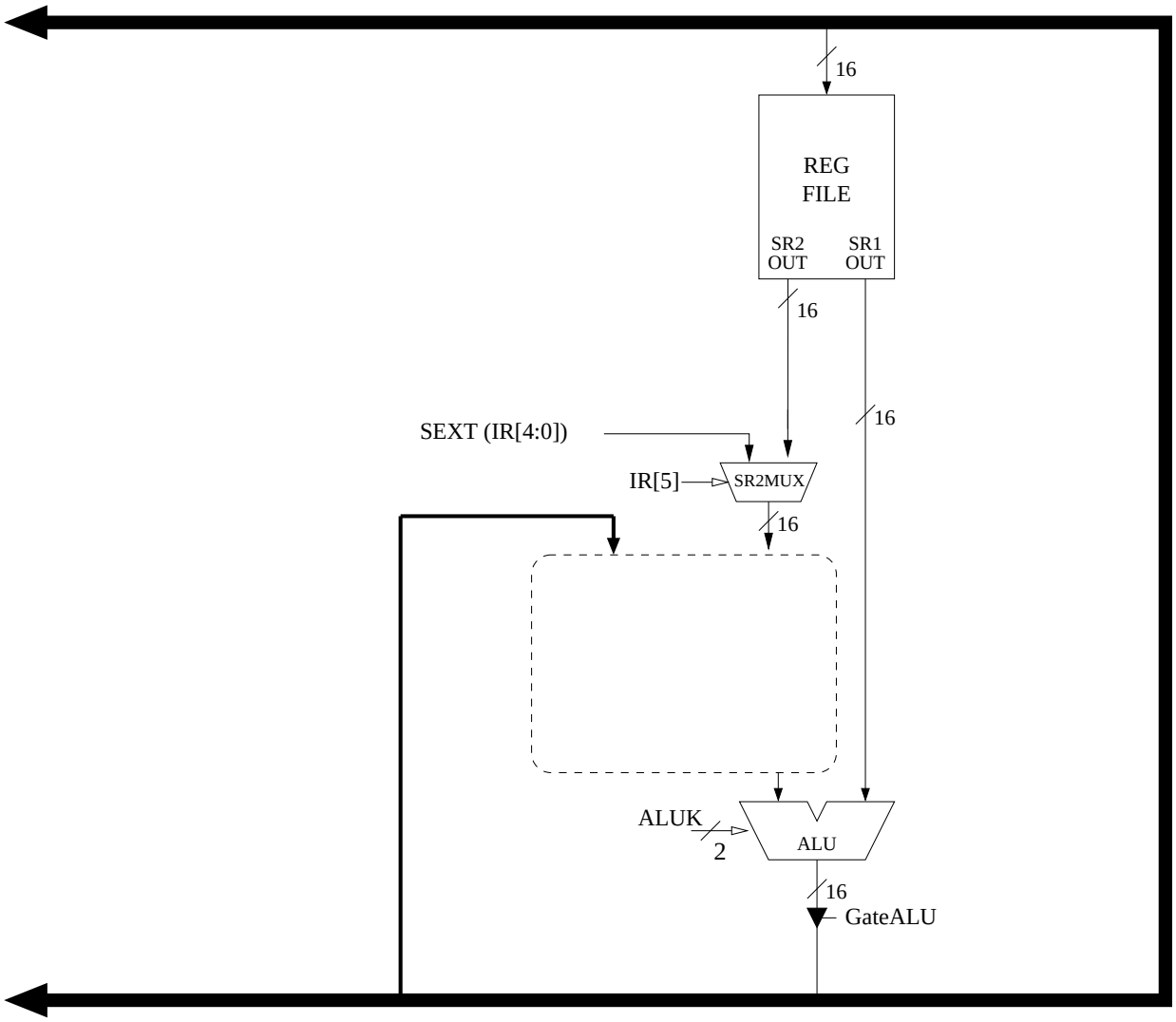
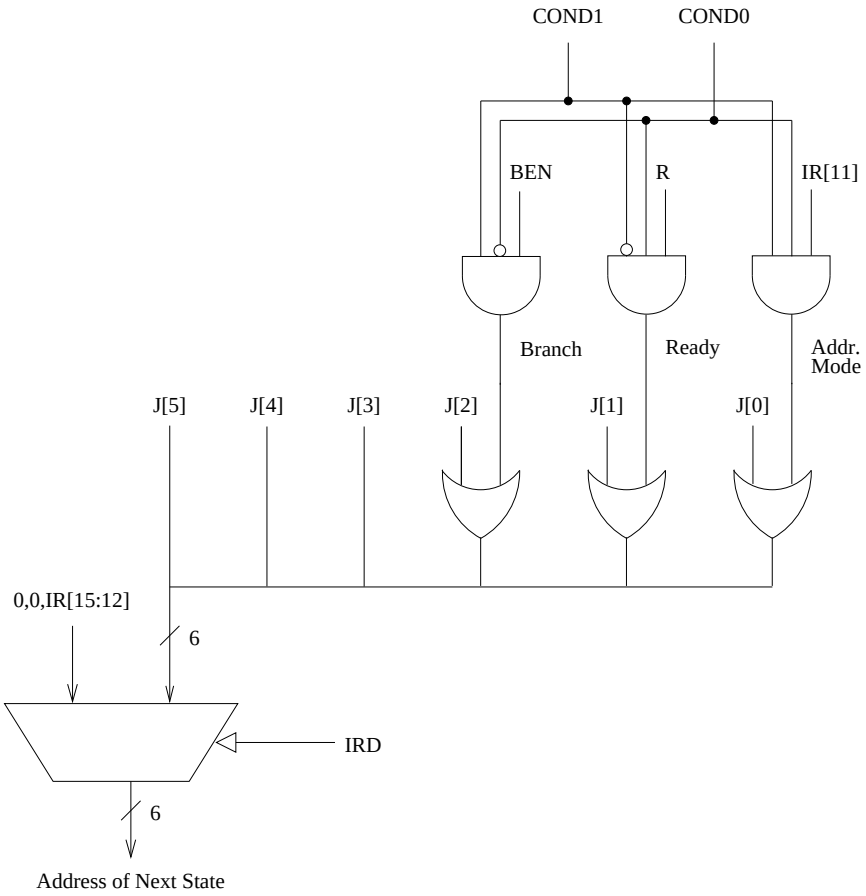


Figure 5: Modified Datapath for LLS instruction

Name:_____

Problem 7 continued:
Part c : Modify the microsequencer as necessary.



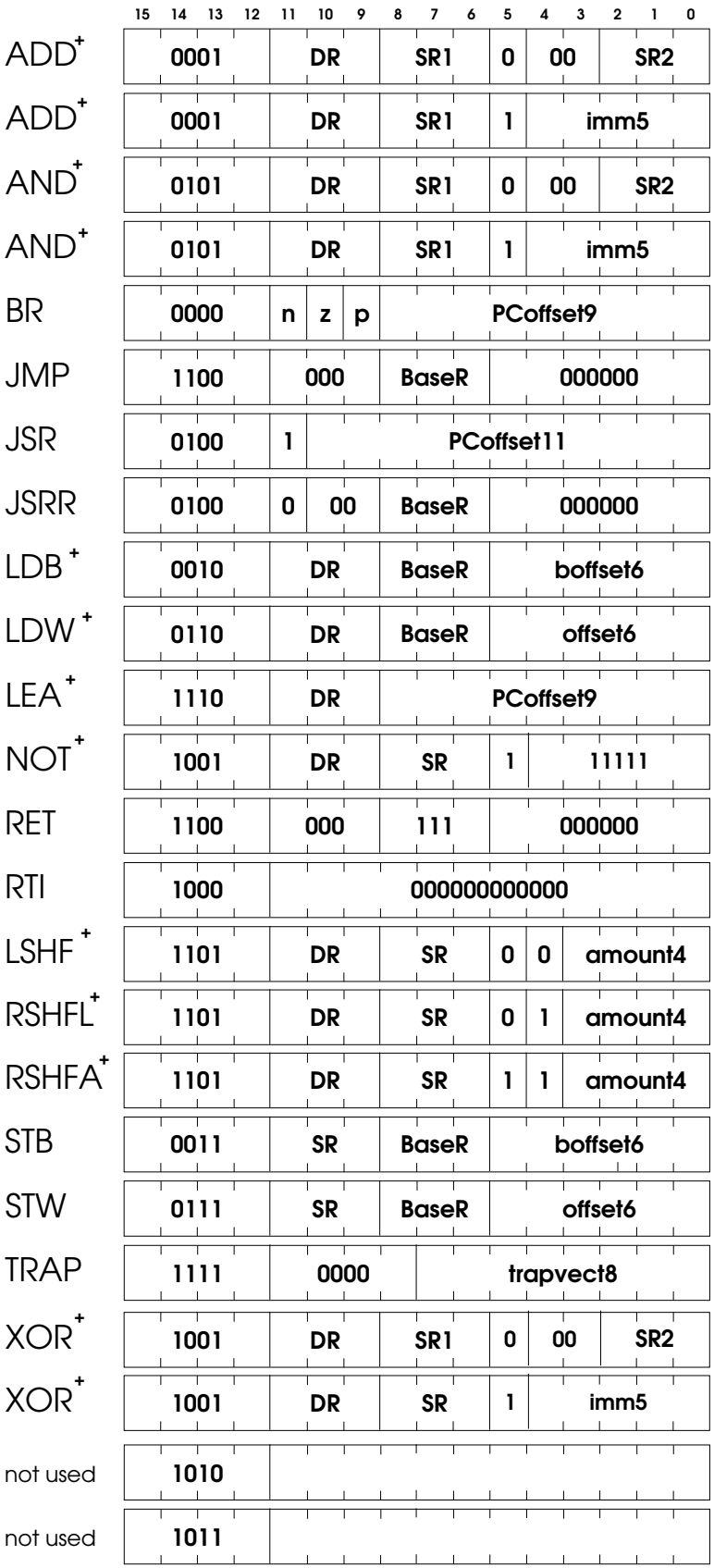


Figure 6: LC-3b Instruction Encodings

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	

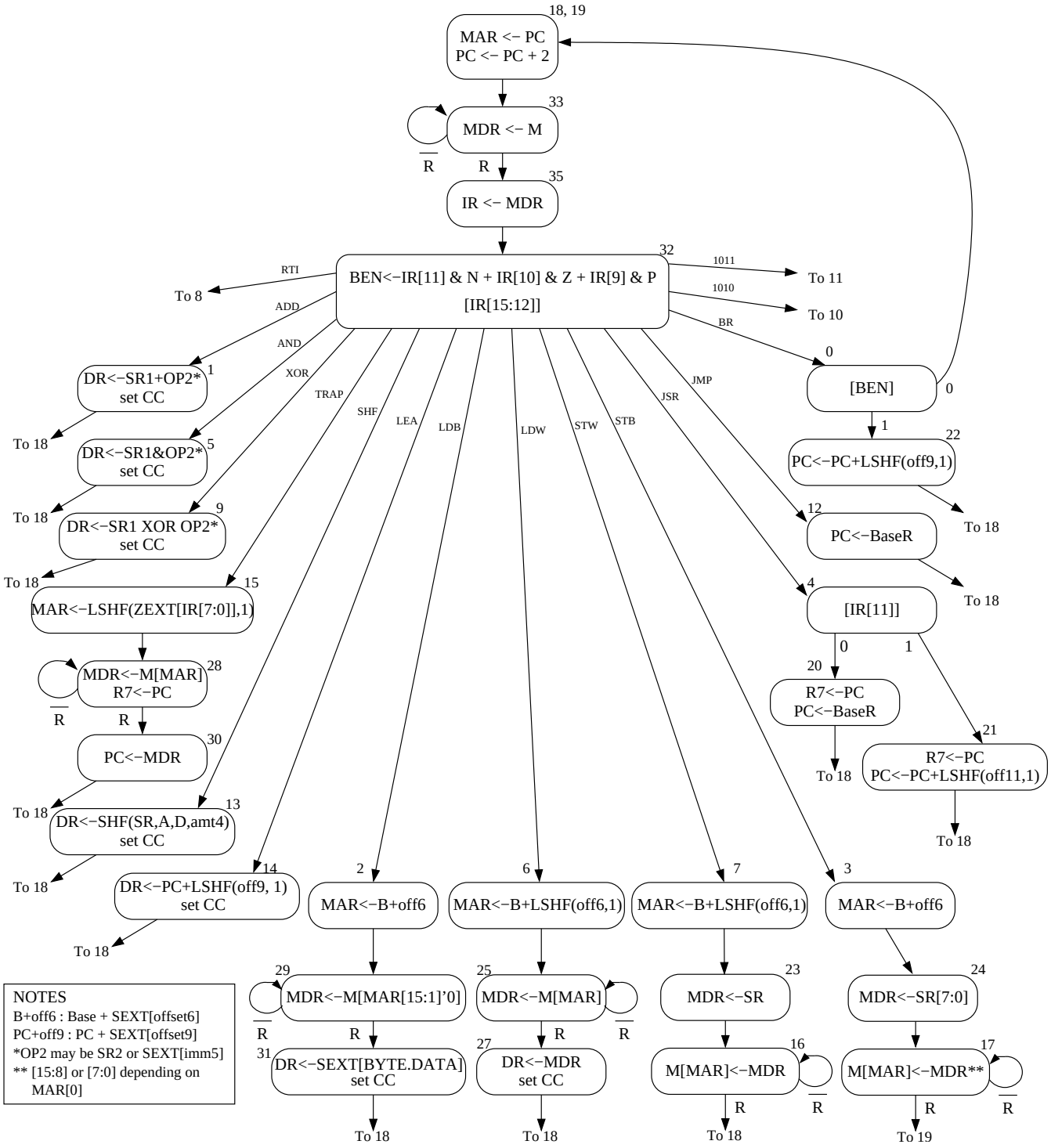


Figure 7: A state machine for the LC-3b

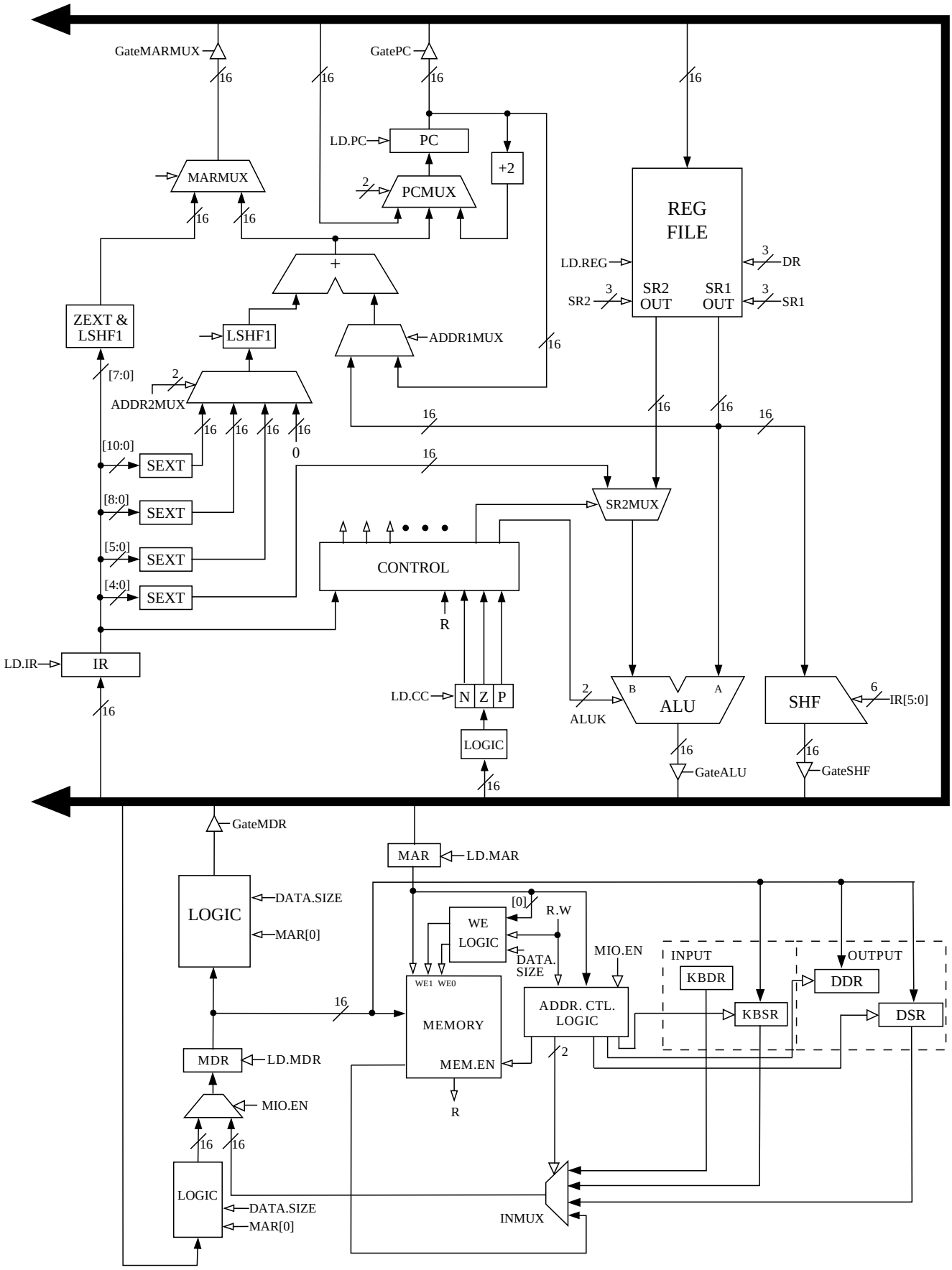


Figure 8: The LC-3b data path

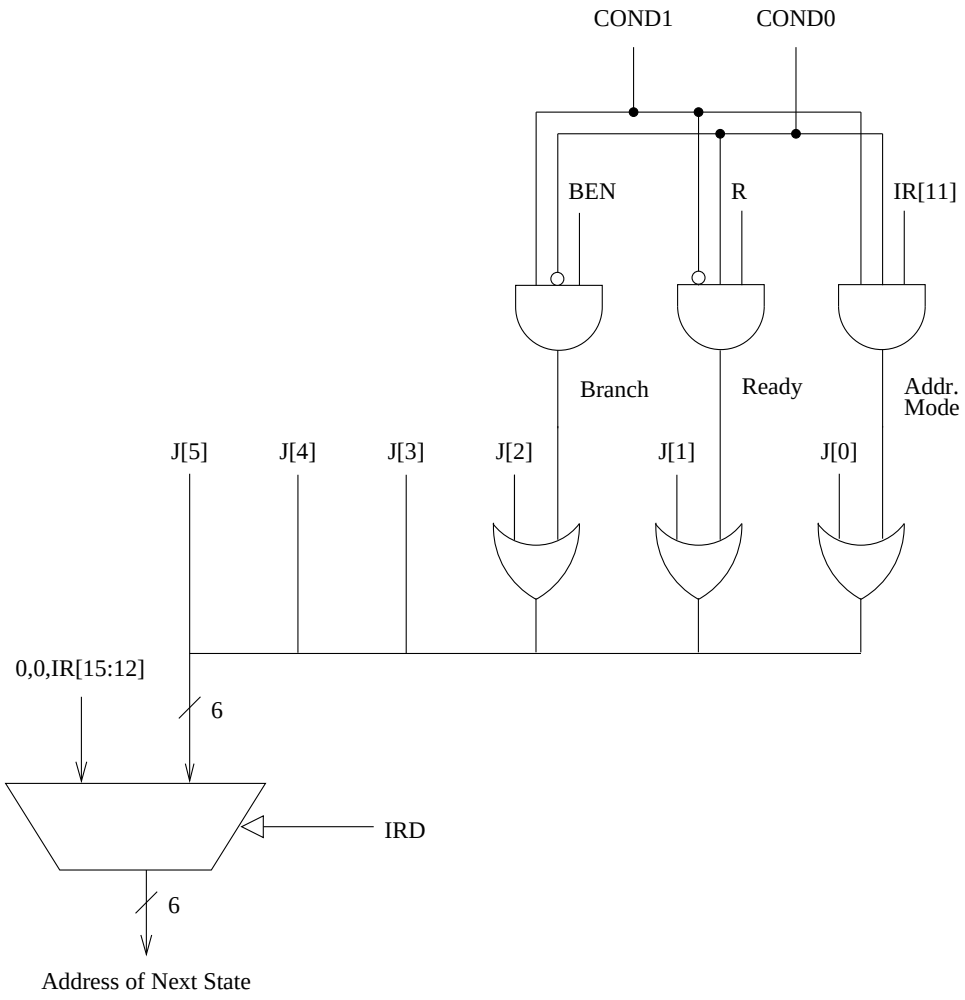


Figure 9: The microsequencer of the LC-3b base machine