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

EE 460N Spring 2017
Y. N. Patt, Instructor
Chirag Sakhuja, Sarbartha Banerjee, Jonathan Dahm, Arjun Teh, TAs
Final Exam
May 12, 2017

Name:_____

Problem 1 (20 points):_____

Problem 2 (10 points):_____

Problem 3 (10 points):_____

Problem 4 (20 points):_____

Problem 5 (20 points):_____

Problem 6 (25 points):_____

Problem 7 (25 points):_____

Total (130 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 read the following sentence, and if you agree, sign where requested: I have not given nor received any unauthorized help on this exam.

Signature:_____

GOOD LUCK!

Name:_____

Problem 1 (20 points): Answer the following questions.

Part a (5 points): Write a program fragment in LC-3b code that will take a value stored in R1, divide it by 4, and store the result in R2. Use as many instructions as you need.

Part b (5 points): A physical cache read access requires a TLB access, a Tag Store access, and a Data store access. In general we decrease latency substantially by doing the Tag Store access at the same time as we do the Data Store access. If we do that, a direct mapped cache has shorter latency than a 4-way set associative cache. What is the major reason for this?

Part c (5 points): Several device controllers are connected to the asynchronous bus discussed in class. What two things must be true for a device controller to assert SACK? What must all be true for a device controller to subsequently negate SACK?

To assert SACK

To negate SACK

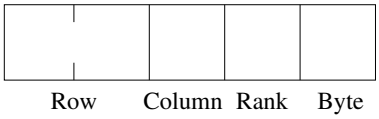
Part d (5 points): As you know, wobble is a problem in floating point arithmetic. Is it a problem in fixed point arithmetic? Why or why not? Explain in 15 words or fewer.

Name:_____

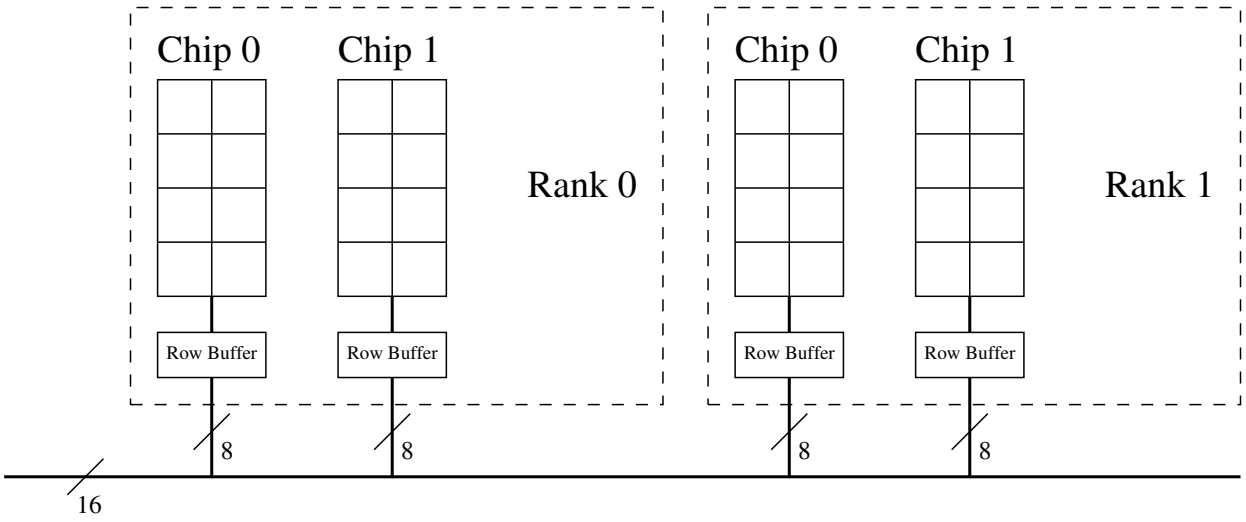
Problem 2 (10 points): Shown below are the addresses and contents of five memory locations.

Addr	Contents
x00	00000000
x01	00000001
x02	00000010
x04	00000011
x1E	00000100

Memory address bits are broken down as follows:



Part a (5 points): Your first job, identify the five locations specified above by putting their contents in the proper locations of the memory structure:



Part b (5 points): How many clock cycles would it take to read the contents of the five memory locations specified above in sequential order, if we are restricted to access the memory one byte at a time? Assume it takes 3 cycles to open a row, 1 cycle to access an open row, and 2 cycles to close a row.

Name:_____

Problem 3 (10 points): Three processors P1, P2, and P3; each has its own cache C1, C2, and C3. The caches are connected to the memory via a bus. Cache coherency is maintained by a Goodman Snoopy Cache protocol. Initially, cache lines A, B, and C are not contained in any of the three caches. P1, P2, and P3 access memory data from lines A, B, and C in the following order:

P1	read	A
P2	write	B
P1	write	A
P3	read	B
P2	write	B
P1	write	C
P3	read	C
P1	write	A

What is the state (Invalid, Valid, etc.) of each cache line in each cache after the 8 accesses have completed?

C1

A	
B	
C	

C2

A	
B	
C	

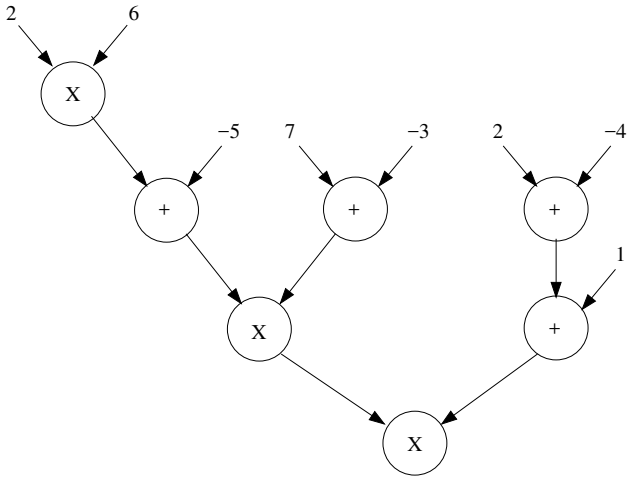
C3

A	
B	
C	

Name:_____

Problem 4 (20 points): Consider the LC-3b, augmented with a multiply instruction. In this problem, Both ADD and MUL are restricted to the single format OPCODE Ra,Rb,Rc. i.e., no immediates. ADD instructions take 1 cycle of execution, MUL instructions take 8 cycles of execution. The data path contains exactly one pipelined multiplier.

The dataflow graph of a seven-instruction fragment of a program that is in the process of execution is shown below:



Prior to fetching the first instruction, the Register file is as shown:

R0	7
R1	2
R2	6
R3	6
R4	-3
R5	-4
R6	-5
R7	1

At the time that the seventh of the seven-instruction fragment is stored in a reservation station, the Reorder Buffer is as follows:

V	R	Ret	DR	Value
1	1	1	R2	4
1	1	0	R2	6
1	0	0	R2	–
1	0	0	R2	–
1	1	0	R0	4
1	1	0	R1	-2
1	0	0	R0	–
1	1	0	R1	-1
1	0	0	R6	–

Each entry contains 3 bits to indicate its state. The Valid (V) bit indicates if the entry is in use, i.e. V=0 indicates an empty slot in the reorder buffer. The Ready (R) bit indicates that the entry corresponds to an instruction that has successfully completed execution. The Retirement bit (Ret) indicates that the executed instruction has retired.

PROBLEM CONTINUES ON NEXT PAGE

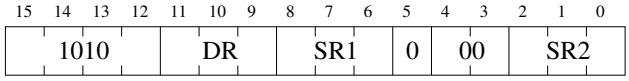
Name:_____

Part a (5 points): What is the value of R2 when the Reorder Buffer is as shown on the previous page?

Part b (15 points): Completely specify below the seven instructions in the seven-instruction fragment.

Name:_____

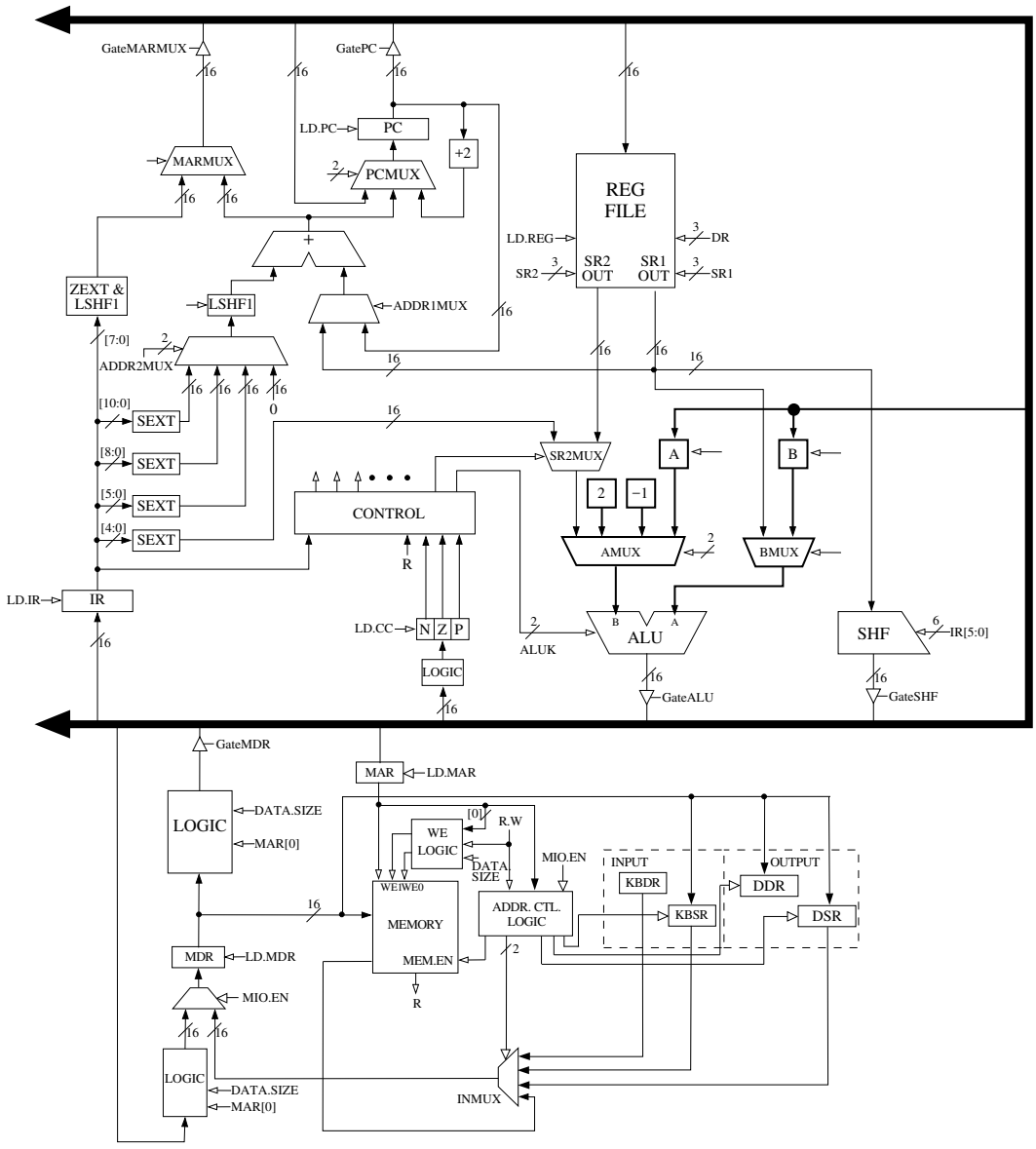
Problem 5 (20 points): We wish to add a new instruction to the LC-3b, called ARRAYCMP, which compares two **equal size, word arrays**. If they are identical, the Z bit will be set to 1. If not, the Z bit will be set to 0. We will use unused opcode 1010. The instruction format for ARRAYCMP is:



SR1 and SR2 contain the starting addresses of the two arrays. DR contains the size of the arrays in **words**. Note the side effects: SR1, SR2, and DR are clobbered by ARRAYCMP.

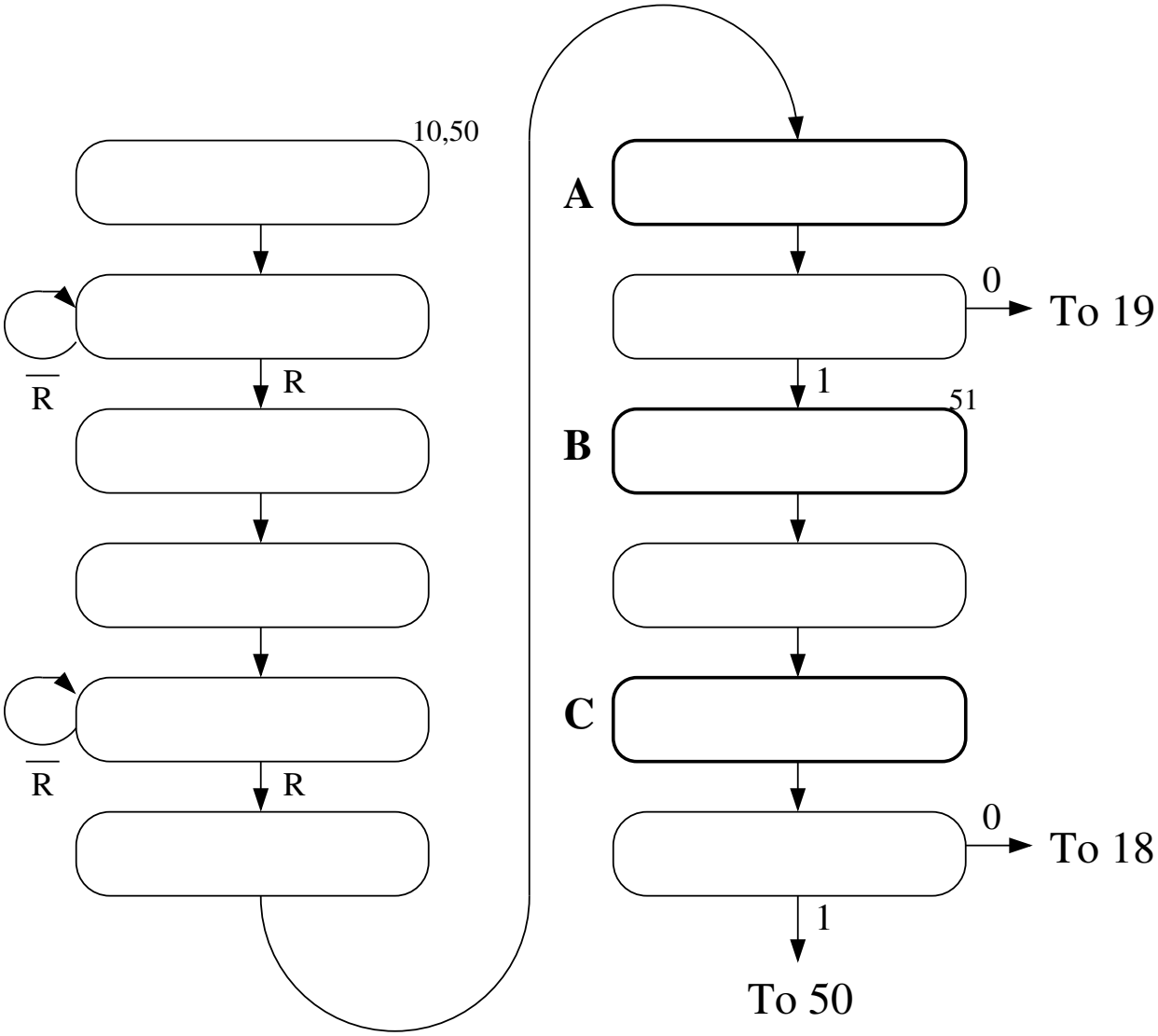
We implement ARRAYCMP by comparing the two arrays one word at a time. After each compare, we decrement DR. If the two arrays are identical, DR will equal 0, and the Z bit will be set. If a mismatch occurs, ARRAYCMP will stop execution with Z=0.

The datapath is modified to implement ARRAYCMP, shown below in bold. We have also added a third input to SR1MUX to select IR[2:0]. **Your job:** Complete the state machine and microsequencer and fill in the control store signals on the following two pages.



Name:_____

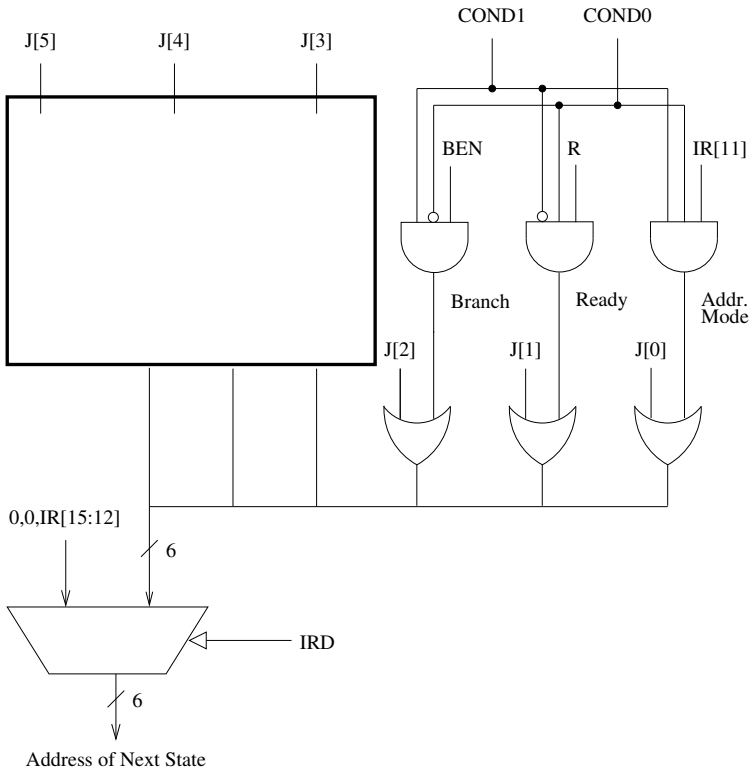
Part a (10 points): Complete the state machine to implement ARRAYCMP given the datapath modifications on the previous page.



PROBLEM CONTINUES ON NEXT PAGE

Name:_____

Part b (5 points): Add logic to the microsequencer to support the state machine from Part a. Put all your modifications in the bold box. You may use two additional control store signals called US1 and US2. You may also add additional inputs to the bold box.



Part c (5 points): We have added several control signals for controlling the data path, as shown below:

Signal name	**Signal values**	
LD.A/1:	NO, LOAD	
LD.B/1:	NO, LOAD	
AMUX/2:	SR2MUX	;select output of SR2MUX
	2	;select the value 2
	-1	;select the value -1
	A	;select the value in A
BMUX/1:	SR1OUT	;select the value in SR1OUT
	B	;select the value in B

Fill in the control signals needed to implement states A,B,C shown in bold in the state machine.

State	ALUK	LD.CC	LD.A	LD.B	AMUX	BMUX	GateALU
A							
B							
C							

Name:_____

Problem 6 (25 points): Consider the LC-3b augmented with a direct-mapped, write-back cache that contains instructions and data. Line size is 8 bytes. The cache is initially empty.

In the execution of a program, a number of accesses between memory and the cache occurred. The table below shows the first several cache line transfers between the cache and memory during execution of the program.

Hint: Recall, in a write back cache, transfers go from memory to cache and from cache to memory.
Hint: Recall, the LC-3b is little-endian.

Address	Cache Line
x3000	xE00030006000E001
x3008	xF0250BFDD0016200
x6010	x0000000000000000
xC020	x0000000000000000
x8040	x0000000000000000
x0080	x0000000000000000
x3000	xE00030006000E000
x0100	x0000000000000000
...	...

Part a (5 points): Shown below is part of the program that was executed. Your first job: Complete the table with the remaining instructions of the program. ...in assembly language, of course!

Label	Assembly
	.ORIG x3000
	LEA R0, A
	BRnp B
	HALT

Part b (5 points): Why is there an access to x3000 after the access to x0080? Please answer in 20 words or fewer.

PROBLEM CONTINUES ON NEXT PAGE

Name:_____

Part c (15 points): Finally, complete the specification of the cache, i.e., how many bits of index, how many bits of tag, how big is the data store? Please show your work.

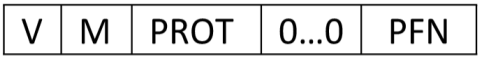
Number of index bits: [ ] Number of tag bits: [ ] Size of data store: [ ]

Name:_____

Problem 7 (25 points): Consider a byte-addressable memory system with two levels of virtual to physical translation (like the VAX).

Virtual Address Space:	64KB	Physical Memory Size:	4KB
User Space:	0x0000-0x7FFF	Page Size:	128B
System Space:	0x8000-0xFFFF	PTE Size:	2 bytes

A PTE is shown below:



A PTE includes a Valid bit, a Modify bit, and a 4 bit PROT field. The low bits (the exact number is for you to determine) are used for the PFN. In user mode, the processor can read/write to all user space pages and does not have any access to system pages. The system pages can only be read/written in supervisor mode. To implement this protection, the encoding of the PROT bits is 1111 for user page PTEs and 1100 for system page PTEs.

Each process, as you know, has its own user space page table. Process page tables are stored in contiguous system virtual memory as follows: Process1 user space page table is stored immediately after Process0 user space page table, and Process2 user space page table is stored immediately after Process1 user space page table.

All user space page tables start at the beginning of a page. The system space page table starts at the beginning of a frame.

Process0 user space Length Register (LR): 128 Process1 user space LR: 64 Process 2 user space LR: 64

The processor executes the following code:

```
Process0: ADD R0, R1, R2
Process0: RSHFL R3, R1, #1
<Context switch>
Process1: ADD R1, R2, R3
<Context switch>
Process2: ADD R1, R2, #0
Process2: AND R2, R2, R3
```

Note: context switches are required if the processor moves from executing code from one process to executing code from another process.

PROBLEM CONTINUES ON NEXT PAGE

Name:_____

The microarchitecture includes an 8-entry TLB. Its state, before the user code starts execution, is shown below. Note that the Process ID is included so the TLB is not flushed on a context switch. The TLB only contains PTEs for user space.

Process	Page Number	Frame Number
-		
1	x000	x04
-		
2	x003	x05
1	x013	x1B
2	x042	x08
-		
-		

Part a (12 points): Fill in the following values:

SBR [ ]

Process0 user space BR [ ]

Process1 user space BR [ ]

Process2 user space BR [ ]

Part b (13 points): The following table shows successive entries for successive memory accesses due to execution of the user code shown above. Memory operations due to the OS during context switches are not included.
Your job: Complete the table. Some entries may remain blank.

Process ID	VA	PA	Data	TLB Hit?
			xB01B	
	xA004		xBC02	
	x0104			
		x212		
	x01FE			
		xF86		
		xE88		
		x080		

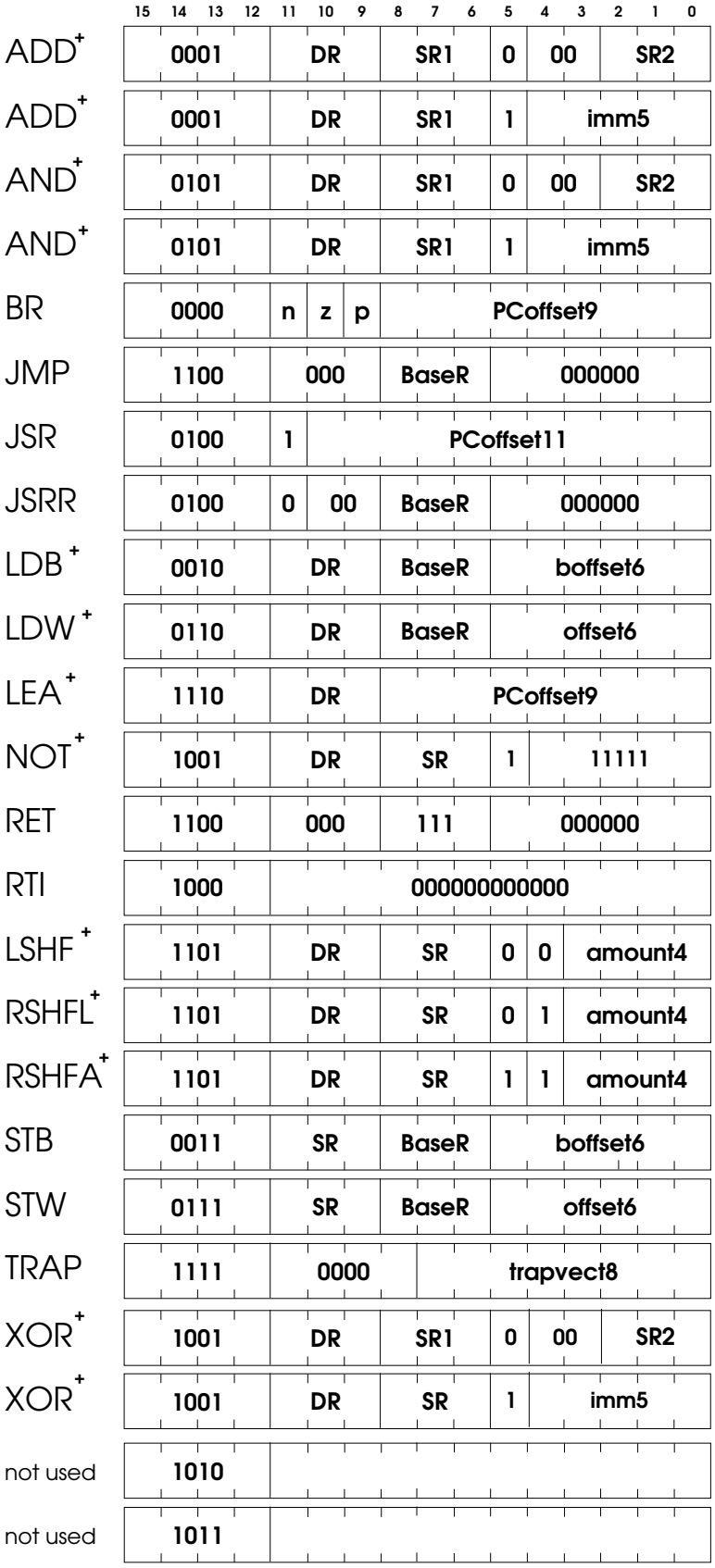


Figure 1: LC-3b Instruction Encodings

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	

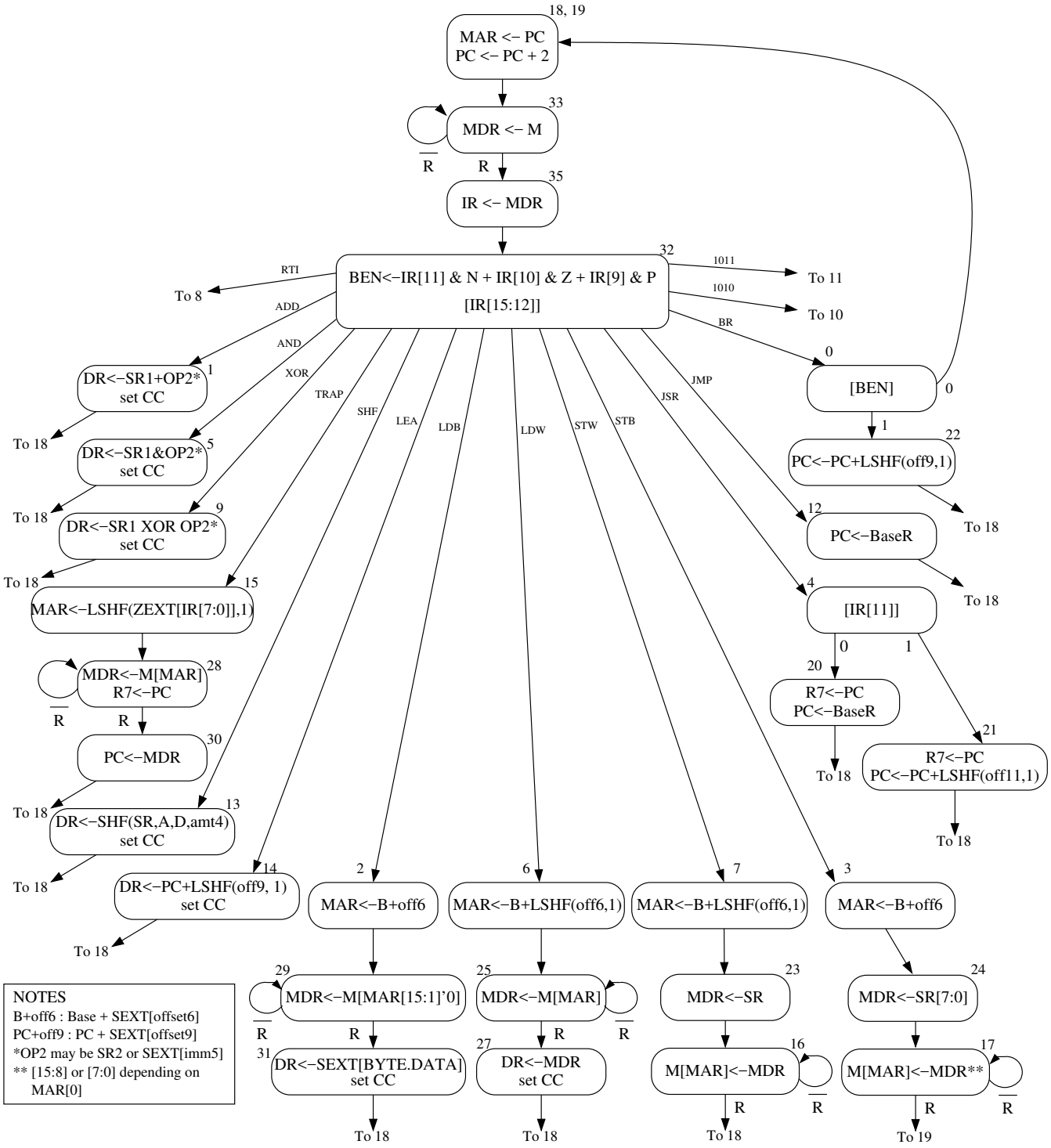


Figure 2: A state machine for the LC-3b

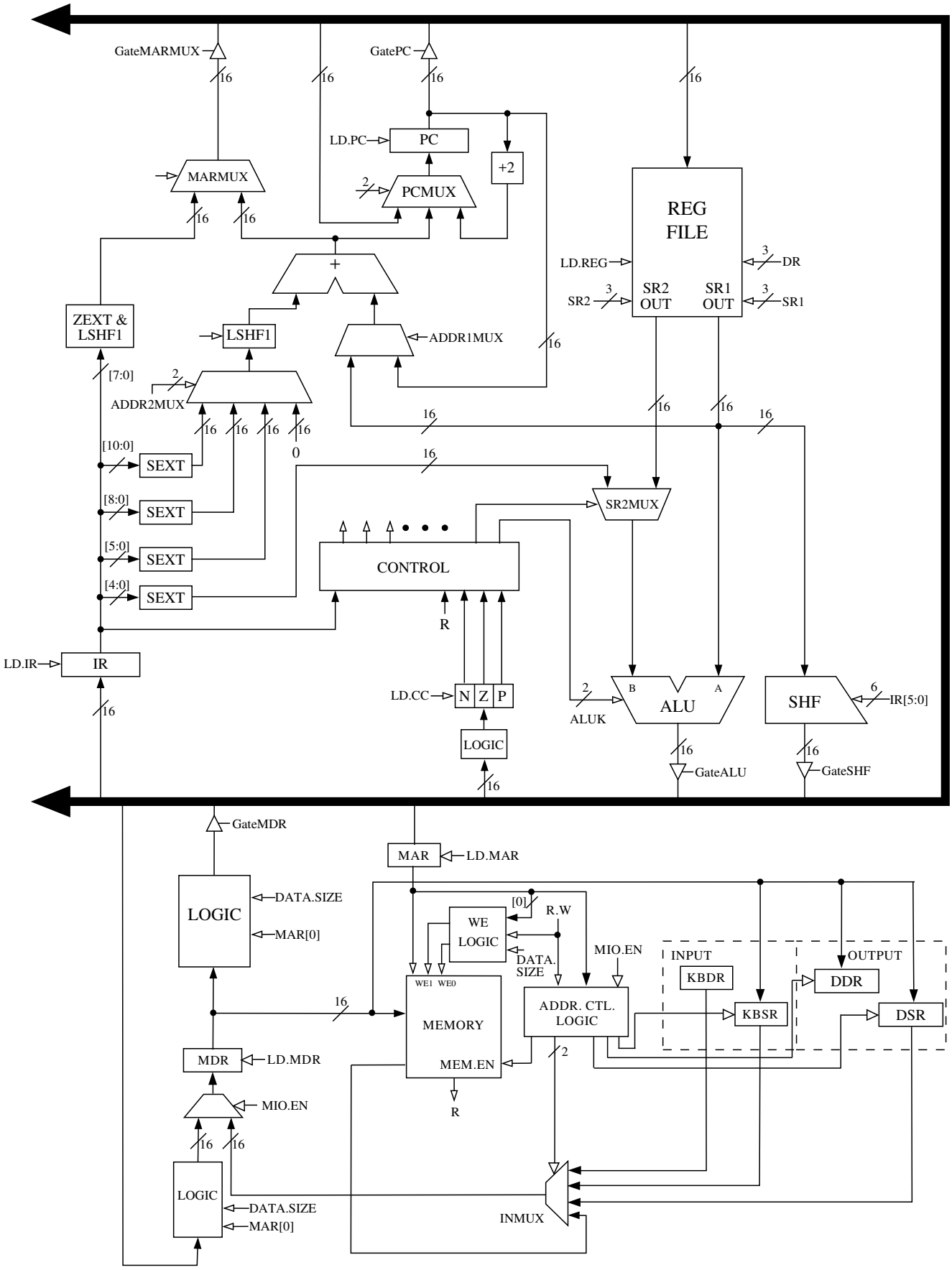


Figure 3: The LC-3b data path

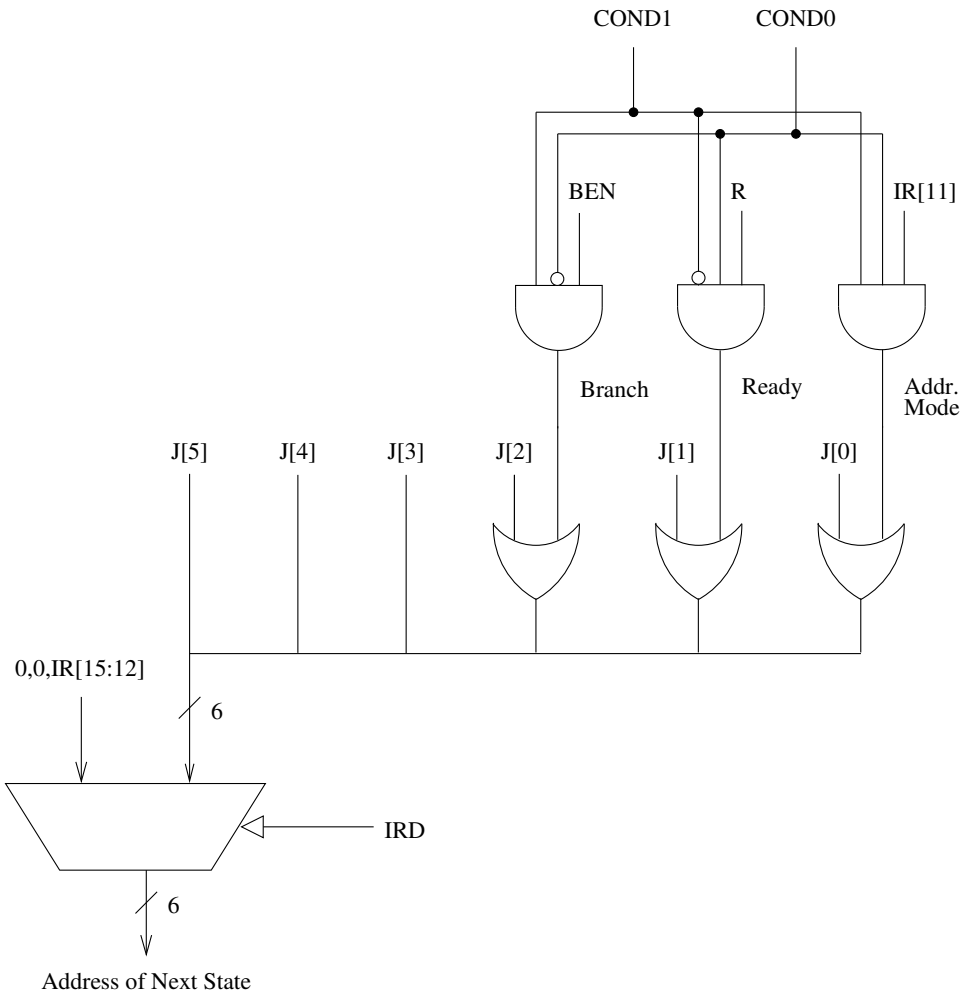


Figure 4: The microsequencer of the LC-3b base machine