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
Exam 2
April 19, 2017

Name: Solution

Problem 1 (20 points):_____

Problem 2 (15 points):_____

Problem 3 (20 points):_____

Problem 4 (20 points):_____

Problem 5 (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 read the following sentence, and if you agree, sign where requested: I have not given nor received any unauthorized help on this exam.

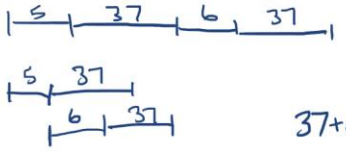
Signature:_____

GOOD LUCK!

Name: Solution

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

Part a (5 points): Vector chaining speeds up execution of vector instructions. Suppose I have back to back (the result of the first is a source of the second) vector instructions that are executed in pipelined functional units. The first functional unit has 5 pipeline stages, the second functional unit has 6 pipeline stages. Assume the vector length register contains the value 38. How many cycles are **saved** by vector chaining?



: 37+6+37+5 : 85

85 - 48

37+6+5 = 48

37 cycles

Part b (5 points): IEEE floating point has four rounding modes. The default is **unbiased round to nearest**. What does the word unbiased mean in this context? Why is that word a reasonable description of how rounding is achieved? Please be specific.

It rounds nearest unless the number is exactly in the middle.
In which case the number is rounded up or down based on LSB.

Part c (5 points): The x86 architecture calls it a Task State Segment. The VAX architecture called it a hardware process control block. What is it used for? Identify three items contained in it.

What it is used for:

A data structure that contains all the state information about a single process.

Item 1: PC Item 2: Registers Item 3: CR3

(there are other items as well)

Part d (5 points): What needs to be added to each tag store entry of a cache to make it a "sector cache"? Why is this useful for a policy where space in the cache is allocated on a write miss but the line is not loaded from memory.

Dirty bits for each of the sectors in a cache block.
This is so that we know which bytes to write back when the block is evicted.

Name: Solution

Problem 2 (15 points): Consider the following 8-bit floating point numbers.

011 → 2^1

3 → 1

BIAS = 3 - 1

21	0	1	1	0	0	1	0	1	10101.00
$3\frac{1}{8}$	0	0	1	1	1	0	0	1	11.001
$\frac{3}{8}$	0	0	0	0	1	1	0	0	0.0110 ← subnormal
$\frac{29}{32}$	0	0	0	1	1	1	0	1	0.11101

Part a (6 points): The decimal values represented by the above numbers are as follows, in no particular order:

$$3\tfrac{1}{8},\quad 21,\quad \tfrac{29}{32},\quad \tfrac{3}{8}$$

Given this information, how many bits specify the exponent and fraction? What is the bias?

BIAS: 2 EXPONENT: 3 FRACTION: 4

Part b (9 points): Using this scheme, is it possible to represent the value $6\frac{3}{8}$ perfectly? Why or why not? Explain in fewer than 20 words.

No, we need 5 bits of fraction to represent $6\frac{3}{8}$ but we only have 4.

Write down its representation, regardless of whether or not it can be represented perfectly. Use unbiased rounding if necessary.

0	1	0	0	1	0	1	0

110.01 ($6\frac{1}{4}$) 110.011 ($6\frac{3}{8}$) 110.10 ($6\frac{1}{2}$) ← ends with 0

Name: Solution

Problem 3 (20 points): We have an asynchronous bus as discussed in class. In addition to the processor, a DMA controller and a byte-addressable memory system are attached to the bus. There is only one bus request level. The BG line is daisy-chained between devices. There are separate address and data lines. The address and data buses are both 32 bits wide.

The DMA controller can copy an arbitrary amount of contiguous data from one part of memory to another part of memory without requiring a sequence of instructions that LD from one part and ST to the other part.

For purposes of this problem only, we will assume that the two parts of memory do not overlap. (After the exam, we can talk about what we would have to do to make this work if they did overlap.)

In order for the DMA controller to do this, it first needs to be told by the processor where the contents of memory to be copied starts, where it is to be copied to, and how much is to be copied.

The processor stores the starting address of the data to be copied into the DMA Controller's R_ADDR register, the starting address of the destination into the DMA Controller's W_ADDR register, and the amount to be copied into the DMA Controller's COUNT register.

The DMA Controller has a 4th register, the 32-bit DATA register. The transfer is implemented by the DMA controller sending R_ADDR, loading DATA with data it receives from memory, and then outputing W_ADDR and DATA. Hint: We can assume that two controllers do not create a problem if they both gate the same value onto the same bus at the same time. Each time a transfer occurs, internal logic in the DMA Controller increments R_ADDR and W_ADDR by four, and decrements COUNT by 4. The DMA Controller also has a one-bit signal COUNT=0.

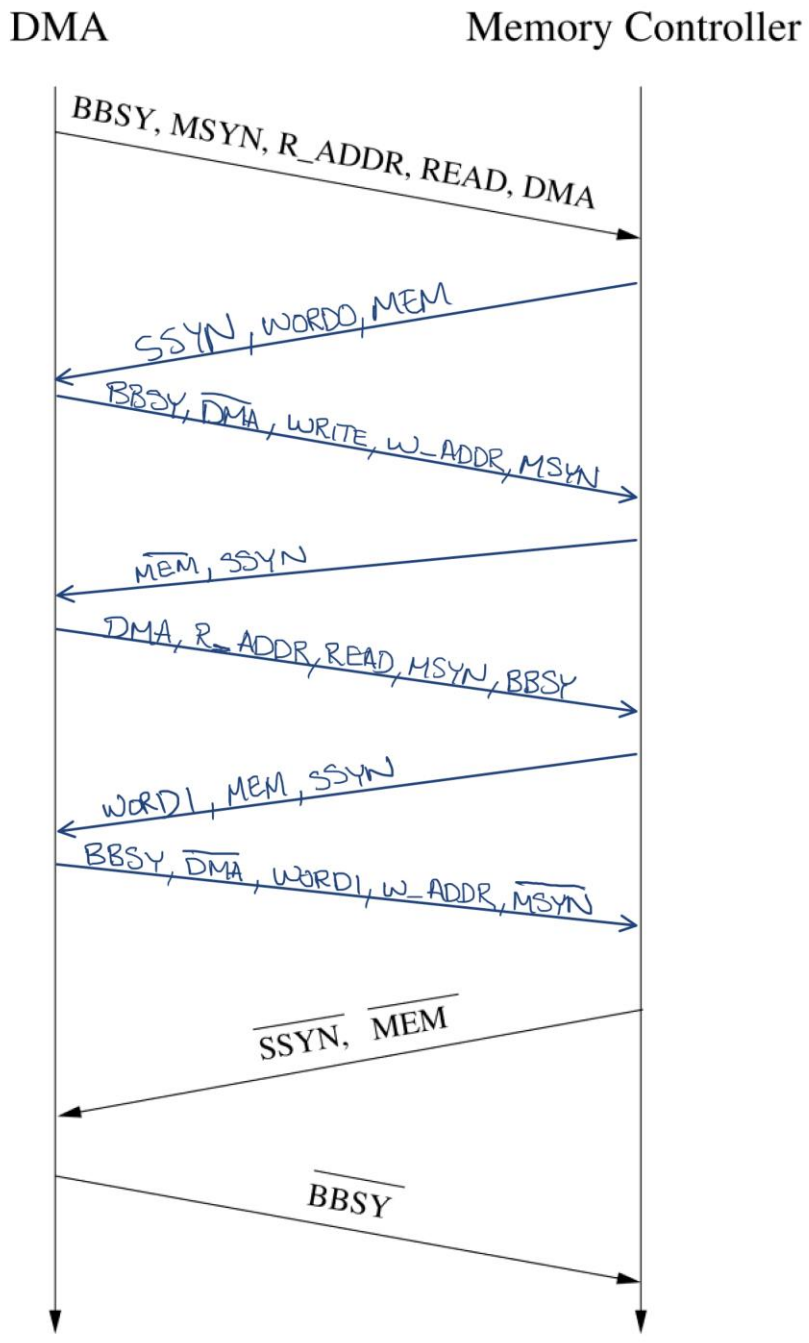
There are six relevant bus signals between the DMA controller and memory that must be controlled:

- **BBSY**: 1 if the bus is busy; 0 if it is free.
- **MSYN**: 1 if the master wants to continue the transaction; 0 if it is finishing.
- **SSYN**: 1 if the slave acknowledges MSYN=1; 0 if the slave acknowledges MSYN=0.
- **TYPE**: READ if the DMA controller wishes to read from the memory system; WRITE if the DMA controller wishes to write to the memory system.
- **DMA**: An extra signal you may find useful in your implementation.
- **MEM**: An extra signal you may find useful in your implementation.
- **COUNT=0**: A signal indicating that the DMA transfer is complete.

PROBLEM CONTINUES ON NEXT PAGE

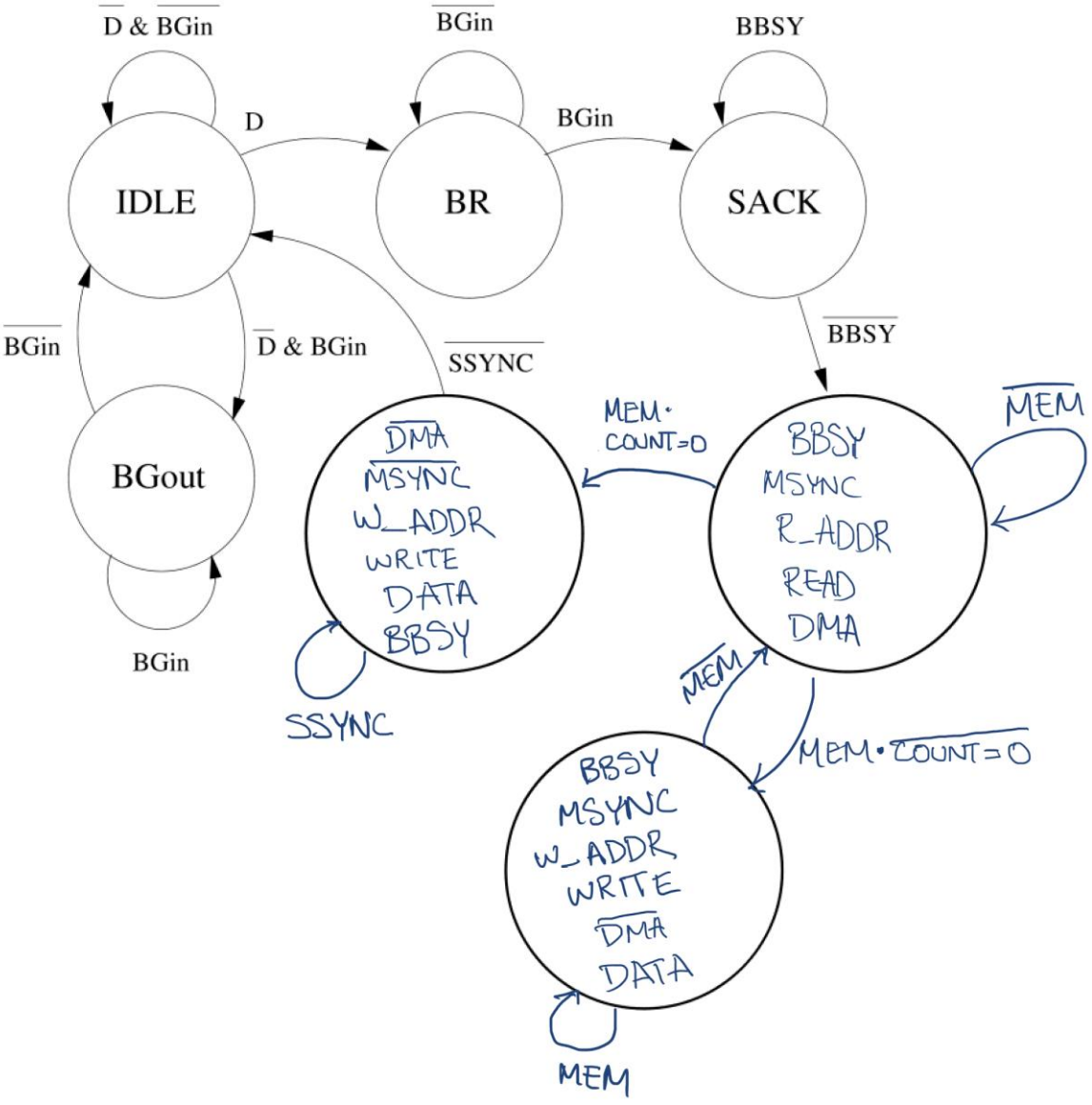
Name: Solution

Part a (10 points): Complete the transaction diagram below between the DMA Controller and the memory system when copying 2 32-bit words. When showing the words sent to the DMA Controller from the memory system, use WORD0 and WORD1 for clarity.



Name: Solution

Part b (10 points): Complete the DMA controller's state diagram below. We have given you the states used to secure the bus; your job is to complete the states and transitions used to accomplish the transaction. You should be able to do this in no more than 3 states.



Name: Solution

Problem 4 (20 points): The LC-3b has been augmented with VAX-style virtual memory. Virtual memory is split into two regions: system space, which consists of addresses x0000 to x2FFF, and user space, which consists of addresses x3000 to xFFFF. (We will assume, for purposes of this question only, that the memory-mapped addresses of I/O devices are part of user space, even though we all know better.) PTEs are 2 bytes each.

Part a (2 points): Write an expression to compute the number of pages in user space given a page size N.

$\frac{x D000}{N}$

Part b (3 points): Write an expression to compute the number of pages the user space page table occupies.

$\frac{2(xD000)}{N^2}$

Consider the following program: A breakpoint is set at the address of the HALT instruction. Then the program is run on the LC-3b (augmented with virtual memory). Before execution starts, the only data resident in physical memory is the system page table. The total number of page faults each instruction generates is listed next to the instruction. You may assume the first entry of the user space page table is at the beginning of a page.

```
    .ORIG x3000
    LEA R0, ADR      ; 2 page faults
    LDW R0, R0, #0   ; 0 page faults
    LDW R1, R0, #0   ; 1 page fault
    LDW R2, R0, #2   ; 2 page faults
    HALT
ADR .FILL xAFFE
    .END
```

Part c (5 points): Why does the first instruction generate 2 page faults? Please answer in fewer than 20 words.

1 time for reading the location x3000 and 1 time for reading the PTE containing location x3000

Part d (10 points): What is the page size?

2^8

[x3000, x8000) on one page of PTE

size of page table: $\frac{(x8000 - x3000) \cdot 2}{N}$

$\frac{\text{size of page table}}{N} = 1 \text{ page}$

$\frac{(x8000 - x3000) \cdot 2}{N^2} = 1$

$(x8000) \cdot 2 = N^2$

$N = 2^8$

Name: Solution

Problem 5 (25 points): We've implemented a 2-way set associative, 512 byte, write back physical cache for the LC-3b. The line size is 8 bytes, the cache uses perfect LRU replacement, the policy is allocate on write miss, that is, on a write miss, the cache line is loaded before the write is performed. The machine has 16KB of physical memory.

Consider the following piece of code that carries out component-wise addition of two integer arrays B and C and stores the result in A.

```
for (i = 0; i < 128; i++) {
   int temp_B = B[i];
   int temp_C = C[i];
   A[i] = temp_B + temp_C;
}
```

$\frac{512}{8}$ = # of lines in cache = 2^6

$\frac{2^6}{2}$ = # of sets = 2^5

All arrays consist of 128 16-bit integers, and are resident in physical memory. The arrays are in contiguous memory; that is, B begins immediately after A, and C begins immediately after B. The first element in each array is aligned with the beginning of a cache block. Note: temporary variables in the program are stored in processor registers.

Part a (13 points): After the program executes, how many cache misses will have occurred? Assume the cache is empty when the program begins.

Each array has 256 bytes => 32 lines

Cache Misses: 3(128) = 386

A[0], B[0] and C[0] collide. Since the three alternate in accesses, Each access will access the line that was just evicted. Every access will be a miss.

A UT student who got A's in both 460N and 360C decides to speed up the performance of the program by reconstructing it as follows:

```
for (i = 0; i < 128; i++) {
   A[i] = B[i];
}
for (i = 0; i < 128; i++) {
   int temp = A[i] + C[i];
   A[i] = temp;
}
```

Part b (12 points): How many cache misses will occur during execution of the reconstructed program?

Cache Misses: 96

The first loop brings in A and B, the second loop evicts B and brings in C.

Each line in the arrays miss once, 32+32+32

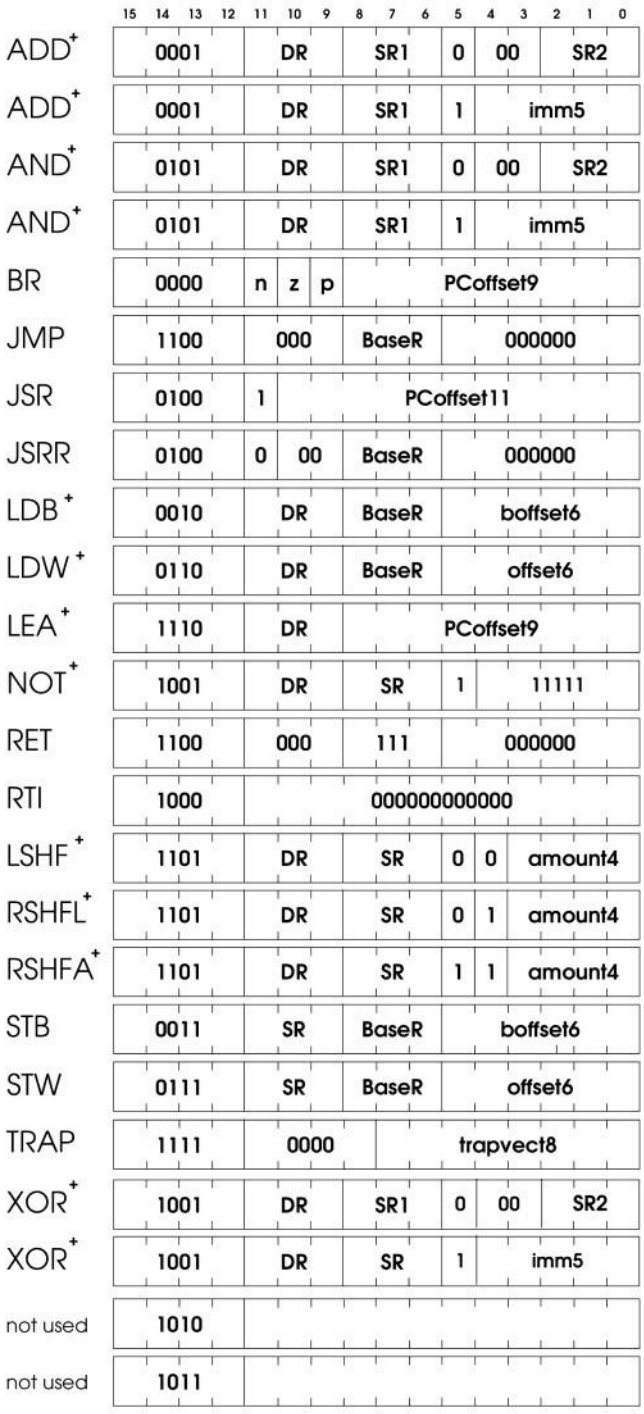


Figure 1: LC-3b Instruction Encodings

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	

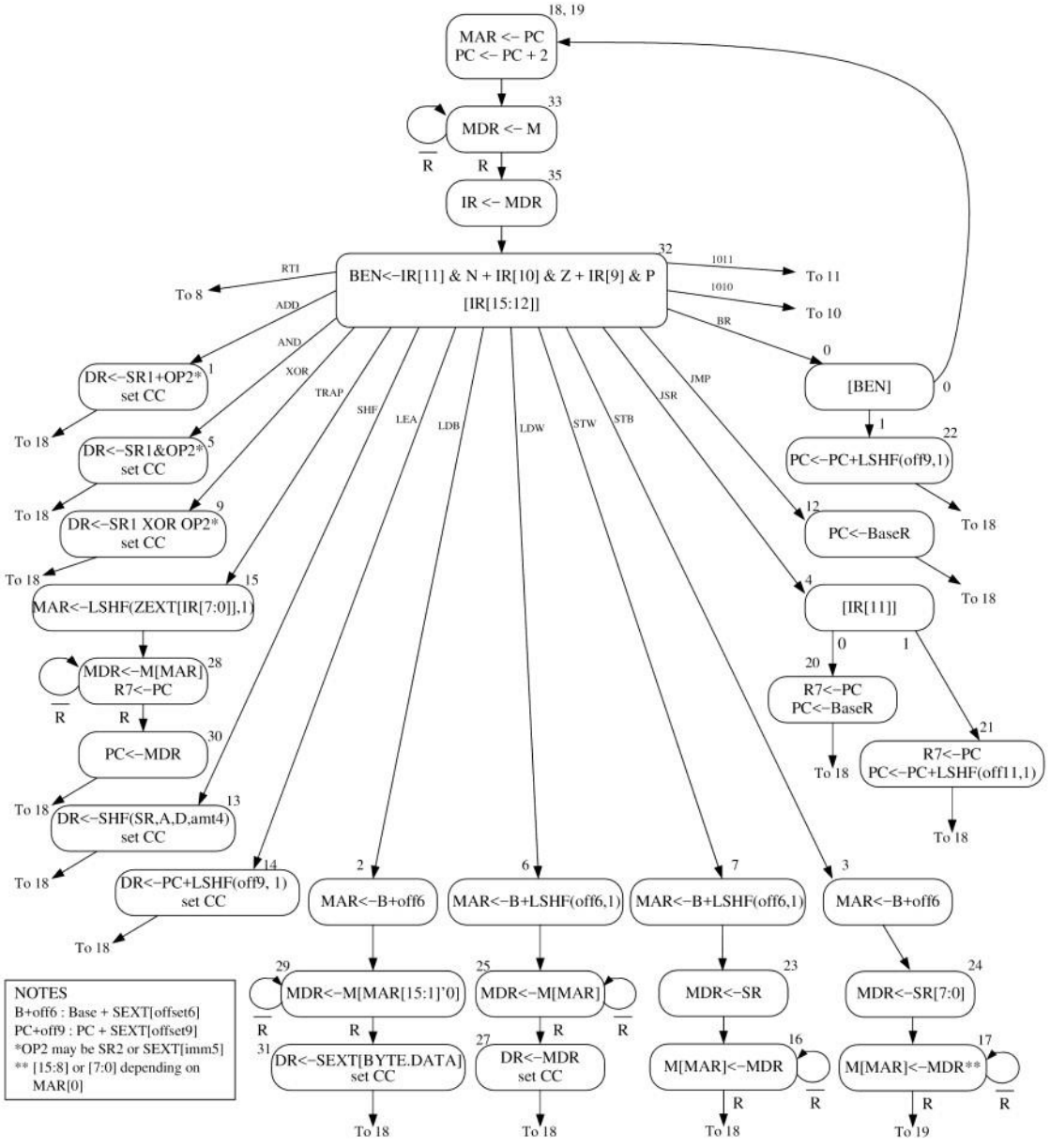


Figure 2: A state machine for the LC-3b

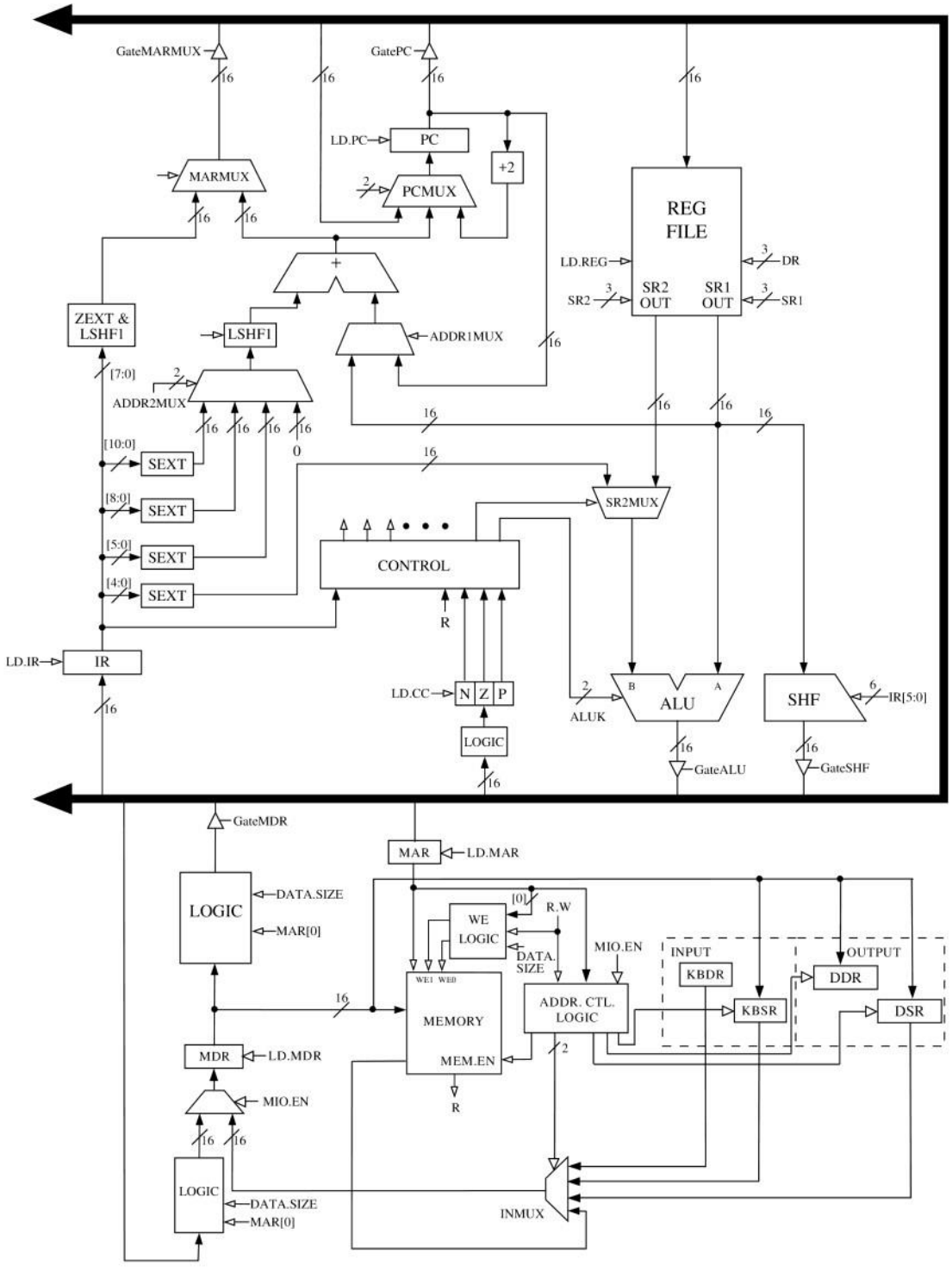


Figure 3: The LC-3b data path

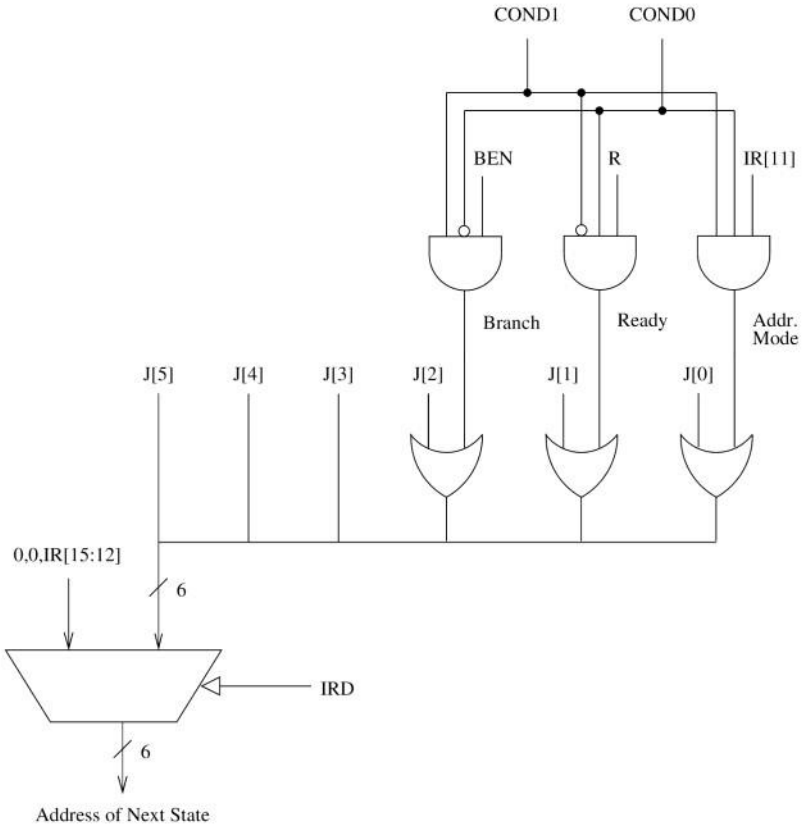


Figure 4: The microsequencer of the LC-3b base machine