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

EE 460N Fall 2016
Y. N. Patt, Instructor
Siavash Zangeneh, Ali Fakhrzadehgan, Steven Flolid, Matthew Normyle TAs
Exam 1
October 5, 2016

Name: Solution

Problem 1 (20 points):_____

Problem 2 (15 points):_____

Problem 3 (15 points):_____

Problem 4 (25 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 sign the following. I have not given nor received any unauthorized help on this exam.

Signature:_____

GOOD LUCK!

Name:_____

Problem 1 (20 points): Please answer any four of the following five parts. Please draw a line through the box of the part you choose not to answer.

Part a (5 points): A zero-address machine explicitly specifies NONE of the three relevant addresses (two source operands, one destination operand) of an operate instruction. How does the microarchitecture know where to get the sources and where to store the result?

The sources are popped from the stack
The result is pushed onto the stack

Part b (5 points): The Alpha 21164 chip had a 96KB L2 cache. 96 is not a power of 2. Why did the designers implement such an unusual size cache?

The designers were too aggressive in their expectation of the number of transistors on the chip.
They designed 128 KB, settled for 96 KB.

Part c (5 points): An important tradeoff exists in the decision as to whether or not to use condition codes. The positive of using condition codes is that it gives you an extra piece of information without requiring an extra instruction to get that piece of information. The negative is:

Since CC gets changed by almost every instruction, the branch is serialized to come right after the instruction that sets CC.

Part d (5 points): We would like to fetch a full packet of useful instructions from the on-chip instruction storage each cycle. Three things can prevent that from happening. They are:

Branch Misprediction

Instruction Cache Misses

Branch taken in middle of a packet

Part e (5 points): A recent term in the vocabulary of microarchitects is Dark Silicon. What does it refer to, and how can it be a feature, rather than a bug?

Dark silicon refers to the condition, where all transistors can not be turned on all the time.
The feature: Turn off when not in use.

Name:_____

Problem 2 (15 points)
An array of x1000 16-bit 2's-complement integers are stored in contiguous memory locations, starting at address x5000. The following program sums the positive integers contained in the array (ignores the negative integers) and stores the sum in R3.

```
          .ORIG x4000

          LEA   R5, DATA_LOCATION
          LDW   R0, R5, #0        ; r0 <- M[DATA_LOCATION]
          LEA   R5, LENGTH
          LDW   R1, R5, #0        ; r1 <- M[LENGTH]
          AND   R2, R2, #0

LOOP      LDW   R3, R0, #0        ; get the next integer
          BRnz  SKIP
          ADD   R2, R2, R3
SKIP      ADD   R0, R0, #2        ; increment the pointer
          ADD   R1, R1, #-1       ; decrement the iteration count
          BRp   LOOP              ; go to next iteration

          HALT

DATA_LOCATION  .FILL x5000
LENGTH         .FILL x1000

          .END
```

The program is executed on a computer whose microarchitecture supports:
(a) virtual memory,
(b) a 12 stage pipeline,
(c) a per-branch last time taken branch predictor, and
(d) 8-way interleaved physical memory.

Memory accesses take 5 cycles.

If the 2's-complement integers in the array are sorted, the program takes approximately 100 nanoseconds to execute. If the integers are unsorted (i.e., stored in random locations within the array), the program takes approximately 200 nanoseconds to execute.

Part a (3 points): Is this performance difference explained by the ISA or the microarchitecture?

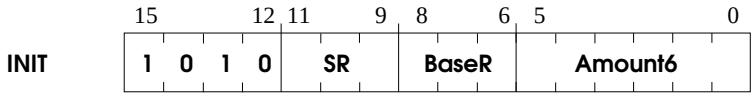
Microarchitecture

Part b (12 points): Considering the 5 possibilities (i.e., the ISA and the four microarchitecture structures **a** to **d** above), explain your best guess as to what is causing the enormous difference (100 nanoseconds vs 200 nanoseconds) in execution time.

Last time taken predictor works great if data are sorted. BRnz SKIP is not taken on positive numbers, and is taken for negative numbers. If data is not sorted, branch direction keeps changing, making it hard to predict.

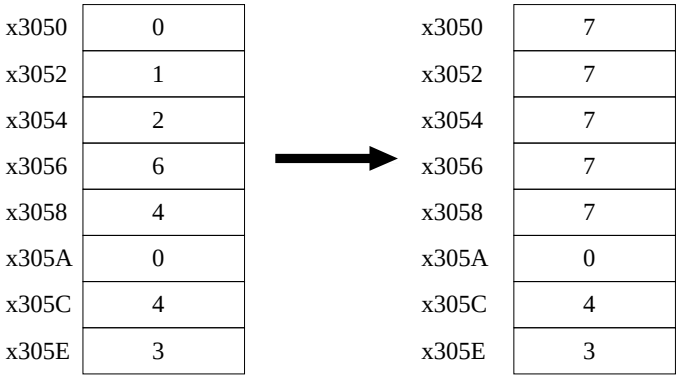
Name:_____

Problem 3 (15 points):
We wish to use one of the unused opcodes to define a new instruction, which we will call INIT. INIT initializes a region of up to 63 memory words with a specific value in each location. The instruction format for INIT is:



where, the starting address of the region is specified in BaseR, the number of words is specified in Amount6, and the value to be written to each location is specified in SR.

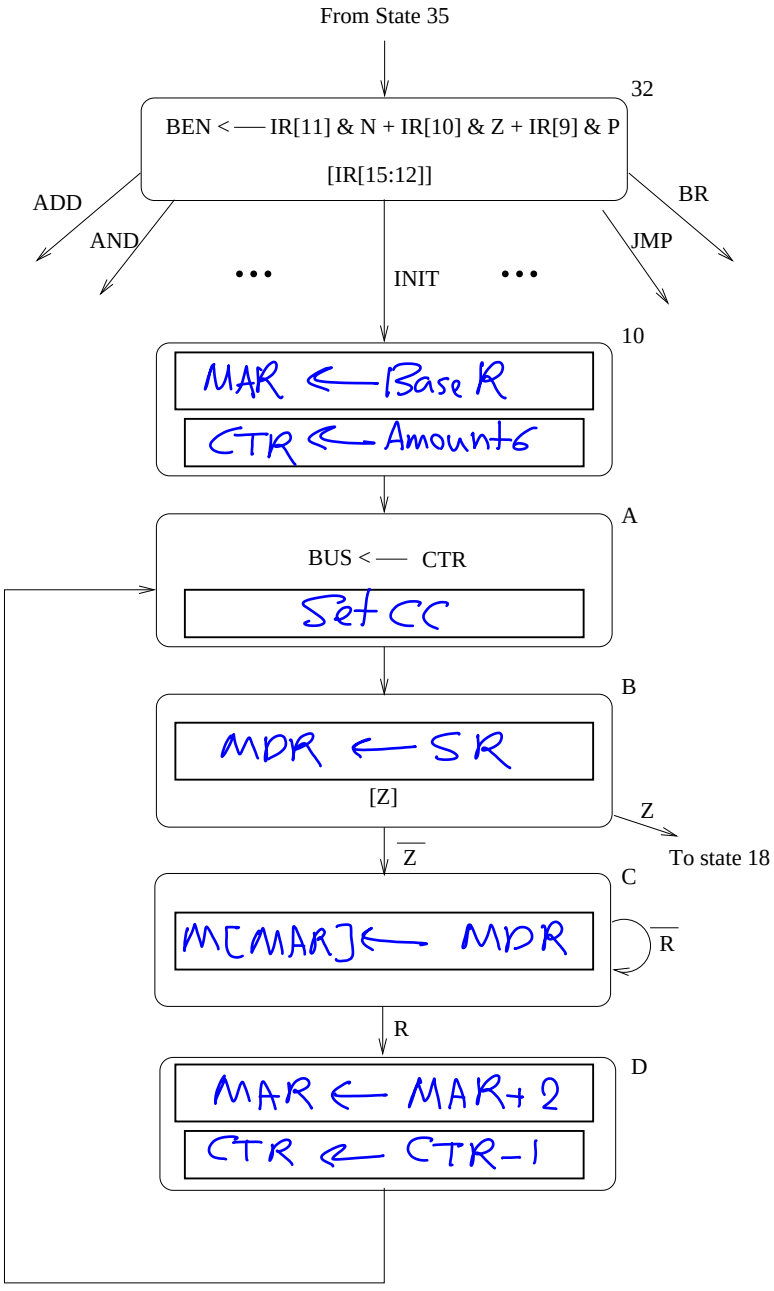
For example, if R0=x3050, R1=x0007, execution of INIT R1, R0, #5 would produce the result shown below.



Your job: Implement INIT on the LC-3b by making the required changes to both the state machine and data path, shown on the next two pages. (We will save the microsequencer changes for another day).

Name:_____

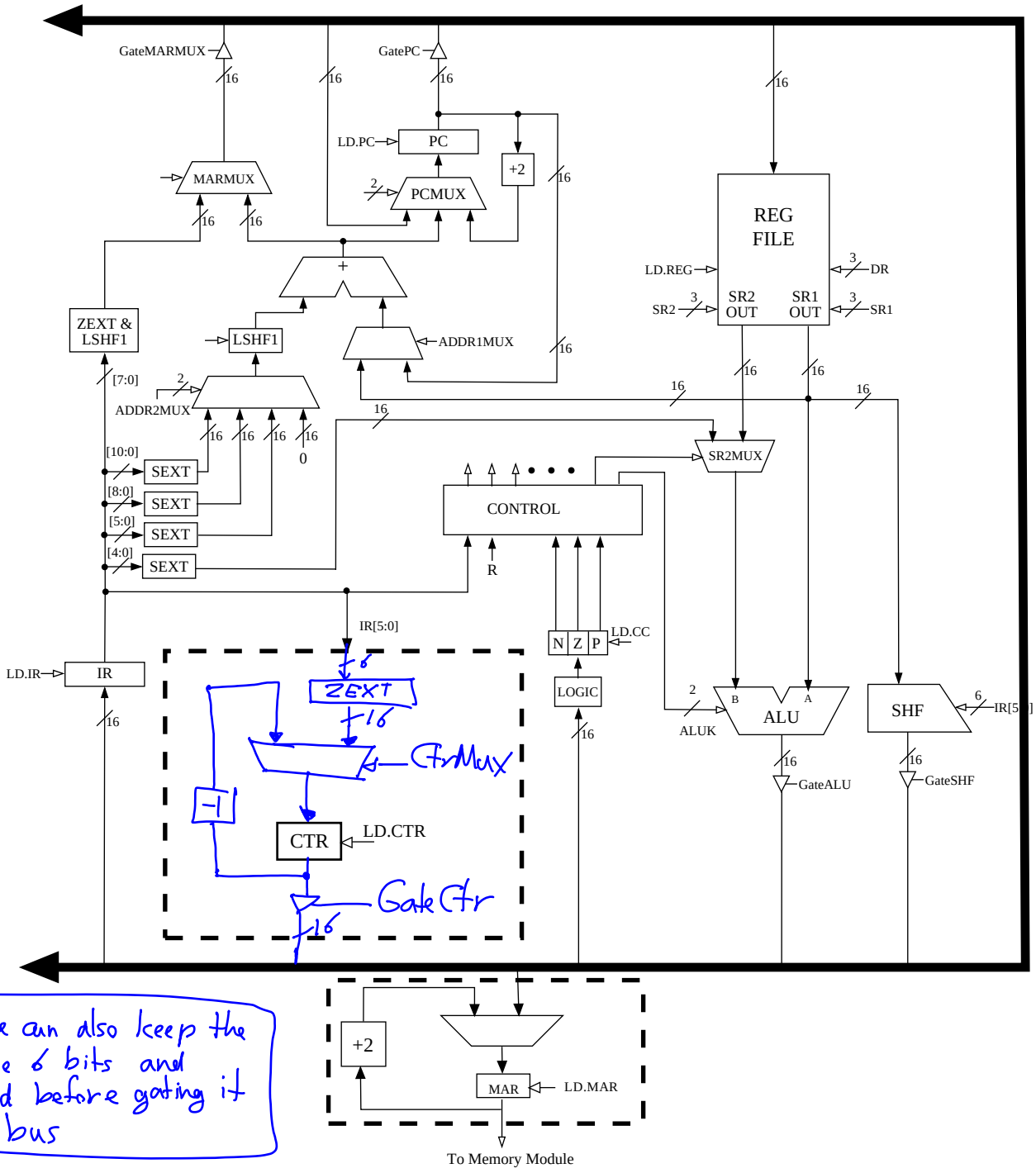
Problem 3 continued:
Part a (9 points): Fill in all boxes to complete the state machine for the path 1010.
Note: we have not asked you to specify which states (A, B, C, D) will be used. That is a microsequencer problem which we will save for another day.



Name:_____

Problem 3 continued:

Part b (4 points): The data path changes can be accomplished within the two dashed boxes shown. We have made the changes for one of them. Your job: the other.



Note: One can also keep the CTR to be 6 bits and zero-extend before gating it onto the bus

Part c (2 points): The instruction just before INIT sets the condition codes. Can the instruction after INIT use those condition codes? Why or why not? Explain;

No, because INIT updates condition codes (sets Z to 1)

Name:_____

Problem 4 (25 points)

Shown on the next page is a non-interleaved memory module containing a single byte addressable memory chip, and the logic to control the memory. Address space is 16 bits. As Faruk described in class the address is broken into row bits (bits[15:8]) and column bits (bits[7:0]). A memory location is accessed in two stages. The first stage takes 8 cycles. In the first cycle, the row bits are loaded into the row address register, accompanied by the load control signal RAS (row address strobe). The following seven cycles are needed to load the row buffer with the contents of all locations in the row. Then, the column bits are used to extract the desired byte from the row buffer. This takes one cycle. That is, a memory access takes 9 cycles total in general.

However, if the next memory access is to the same row, we do not need to load the row buffer (since it is already loaded). We can immediately extract the byte from the row buffer in one cycle. In that case, a memory access takes only one cycle.

Note the 3-bit counter (CTR) which is useful in controlling the memory. It is initially set to zero. When COUNTUP is asserted, CTR is incremented.

Part a (3 points): What is the purpose of the register *TMP*? Holds the address of row in the row buffer

Part b (3 points): What does *X*=1 indicate? Row buffer Hit

Part c (3 points): What does *Y*=1 indicate? Last cycle of row access

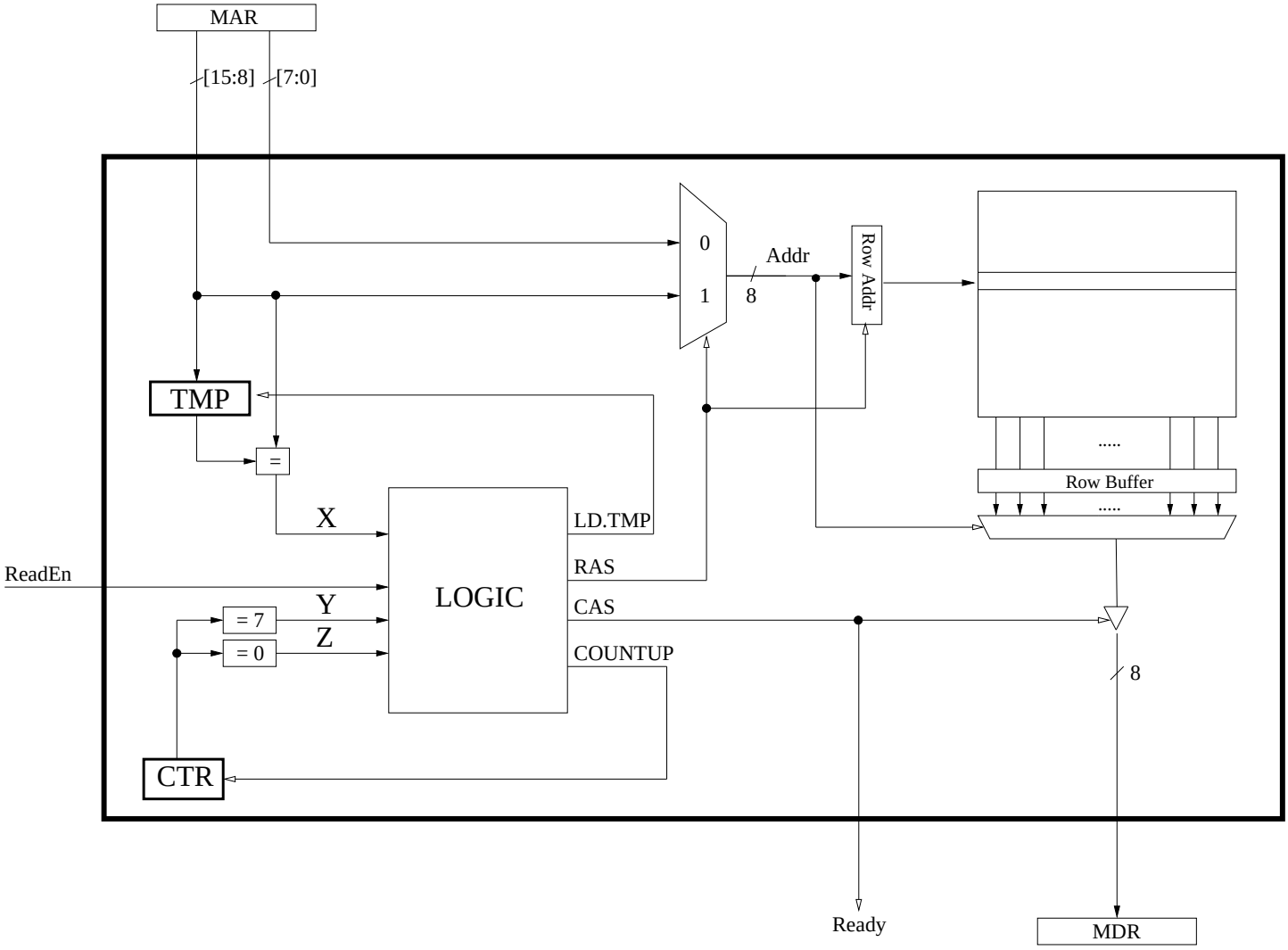
Part d (16 points): Complete the output functions of the truth table. Note that some entries (labeled x) are don't cares.

	ReadEn	X	Y	Z	RAS	CAS	LD.TMP	COUNTUP
(1)	0	x	x	x	0	0	0	0
(2)	1	0	0	0	0	0	0	1
(3)	1	0	0	1	1	0	0	1
(4)	1	0	1	x	0	0	1	1
(5)	1	1	x	x	0	1	x	0

(1) No access
(2) Middle cycle of row access, just count up
(3) First cycle of row access, RAS = 1 to update row address register and count up
(4) Last cycle of row access, LD.TMP = 1 to update current row number and count up to reset the counter to zero.
(5) Row buffer, simply access the column

Name:_____

Problem 4 continued:



Name:_____

Problem 5 (25 points)

The following program sums the contents of all memory locations in an array.

```
        .ORIG x3000

           LEA   R5, DATA_LOCATION
           LDW   R0, R5, #0          ; R0 <- M[DATA_LOCATION]:x6FFE
           LEA   R5, LENGTH
           LDW   R1, R5, #0          ; R1 <- M[LENGTH]:x400
           AND   R2, R2, #0

LOOP       LDB   R3, R0, #0          ; get the next integer
           ADD   R2, R2, R3
           ADD   R0, R0, #1          ; increment the pointer
           ADD   R1, R1, #-1         ; decrement the iteration count
           BRp   LOOP                ; go to next iteration

           HALT

DATA_LOCATION   .FILL x6FFE
LENGTH          .FILL x400

        .END
```

Assume the program executes on an implementation of the LC-3b that supports virtual memory. The 16-bit addresses you are familiar with are virtual addresses. Physical memory is 8KB. Page size is 512 bytes.

Part a (1 point): How many frames of physical memory are there? 16

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. The high bit specifies whether you are in user space or system space. A PTE is 16 bits. For purposes of this question only, we will assume the PTE has the following form:

V	00..0	PFN

Part b (1 point): How many bits in PFN? 4

Also assume for this problem that the microarchitecture has an 8-entry TLB which contains PTEs for user space only. Assume the TLB is empty before the above program executes.

The table on the next page lists in sequence the first nine physical memory accesses required by the LDB instruction to fetch data from the memory array. The table ignores all physical memory accesses due to fetching instructions.

When the program starts executing, memory locations x6FFE, x6FFF, ..., x73FD all contain the value #5.

Name:_____

Problem 5 continued:

Part c (16 points): Complete the table.

Virtual Address	Physical Address	Data	TLB Hit
—	x1202	x800F	X
x821E	x1E1E	x800D	X
x6FFE	x73FE	x5	X
x6FFF	x1BFF	x5	✓
—	x1202	x800F	X
x8220	x1E20	x800A	X
x7000	x1400	x5	X
x7001	x1401	x5	✓
x7002	x1402	x5	✓

Part d (4 point):

System Base Register: x1200

User Base Register: x81B0

Part e (3 points): How many physical memory accesses are required to satisfy all data accesses of the LDB instruction in the execution of the entire program? x406

(3) is a TLB hit since x6FFF is on the same page as x6FFE
Thus, PFN of (3) & (4) are the same

(5) is a TLB miss again as it crosses page boundaries

(6) Data in x1202 is given in first row of the table

(7) from x6FFE to 73FD, we hit pages x37, x38, x39
So total access are (x400 data accesses + 3x2 PTE accesses)

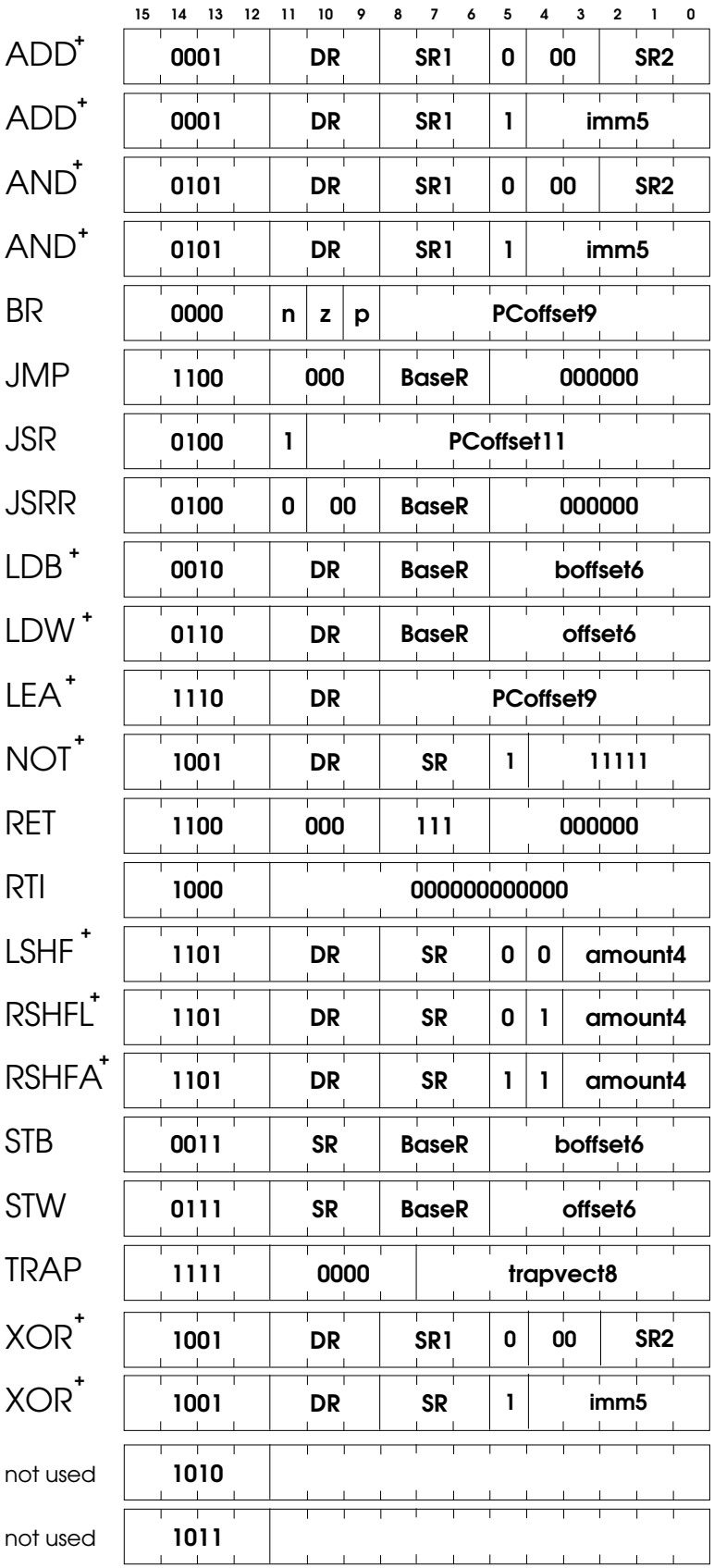


Figure 1: LC-3b Instruction Encodings

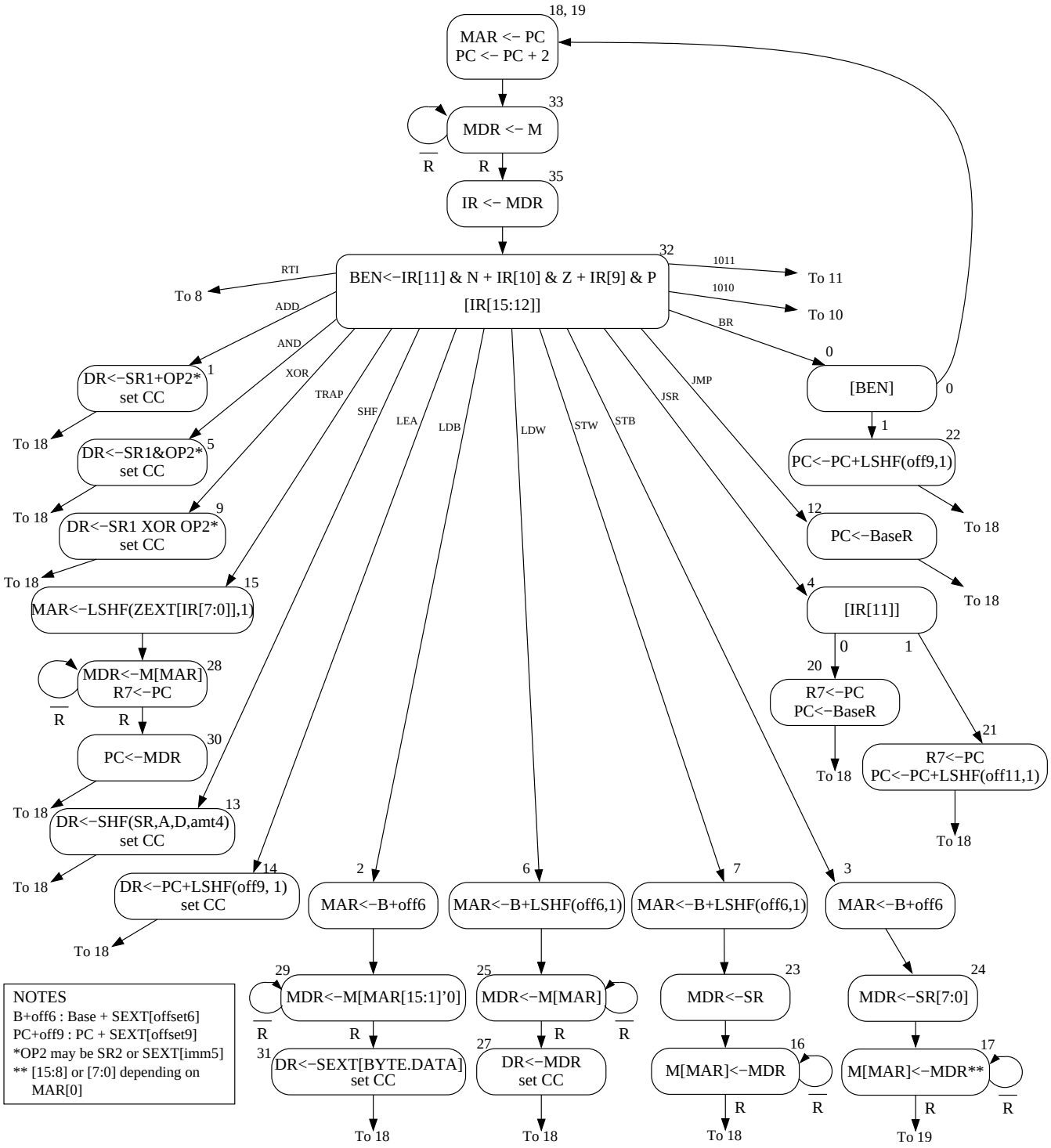


Figure 2: A state machine for the LC-3b

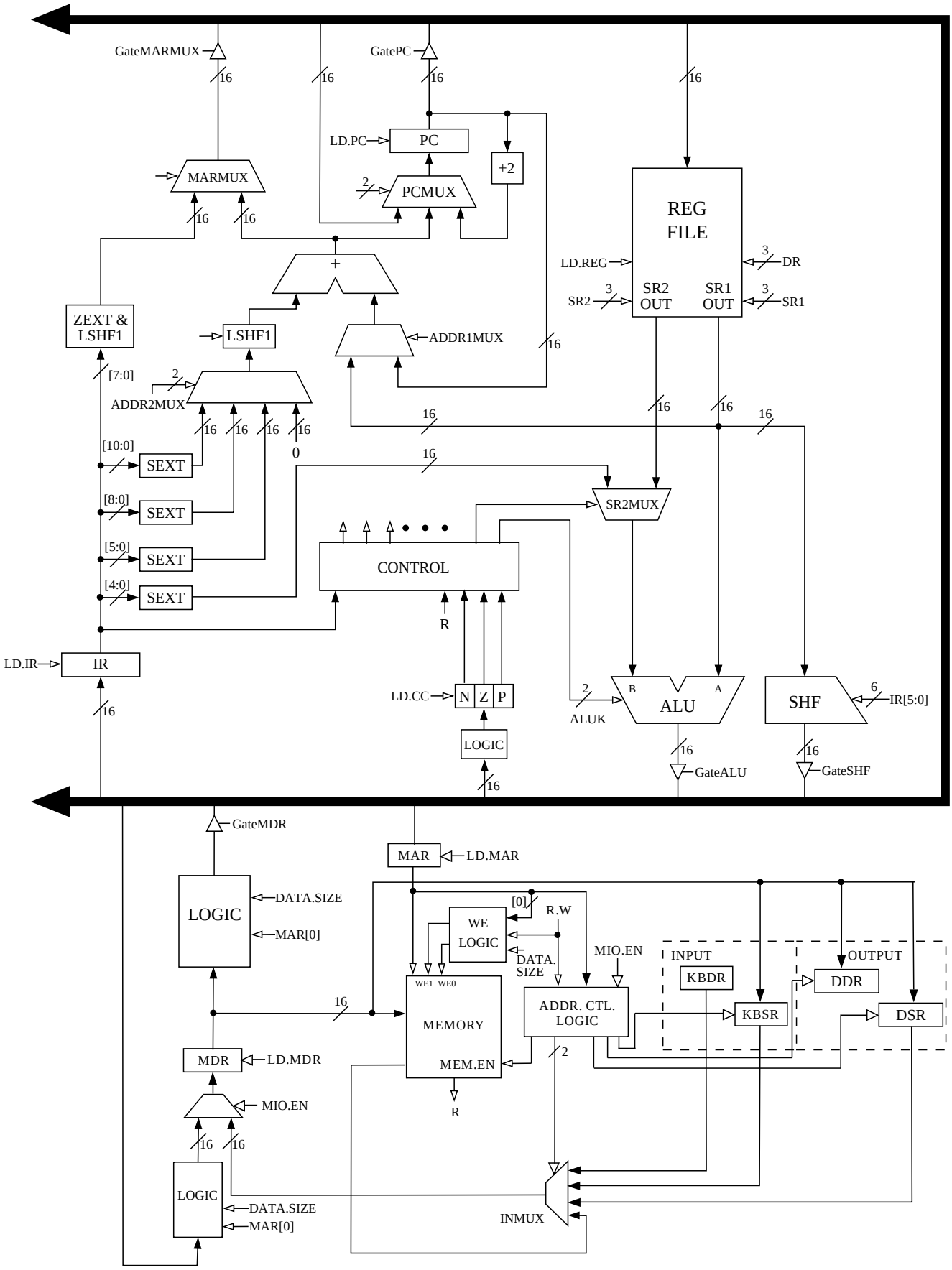


Figure 3: The LC-3b data path