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

EE 360N, Fall 2004
Yale Patt, Instructor
Aater Suleman, Huzefa Sanjeliwala, Dam Sunwoo, TAs
Exam 1, October 6, 2004

Name:___________________________

Problem 1 (20 points):_______

Problem 2 (15 points):_______

Problem 3 (20 points):_______

Problem 4 (15 points):_______

Problem 5 (30 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.

GOOD LUCK!

Name:_________________________

Problem 1 (20 points):

Part a (4 points): The string move instruction copies a block of information from one region of memory to another. This instruction can usually be used to copy up to 64KB of data with a single instruction, although admittedly, it takes a long time to execute that instruction. This instruction is usually accomplished by using an inner loop that copies data, increments the two pointers, and decrements the counter indicating how much more is left to copy. This implementation detail (the inner loop) allows the string move instruction to be stopped after some but not all of the data has been copied, in order to

service and interrupt _________________________,

which improves interrupt latency _________________________.

Part b (4 points): Some ISAs have a fixed length, uniform decode instruction format. Other ISAs opt for a variable length format. The advantages of a fixed length, uniform decode format (in 20 words or fewer):

Decode logic is simpler

The disadvantages of a fixed length, uniform decode format (in 20 words or fewer):

Lower utilization of instruction bits. Limited number of opcodes, addressing modes, and operands.

Please put both answers in the above boxes.

Part c (4 points): The classical condition code mechanism gives you an extra piece of work tacked on to the instruction, thereby potentially decreasing the number of dynamic instructions needed to execute the program. A significant potential disadvantage to this mechanism is (in 20 words or fewer):

The branch instruction needs to be executed right after the instruction that generates the condition codes.

Part d (4 points): Critical path design requires you to look at the longest speed path and try to shorten it, thereby decreasing the cycle time. MIPS found on one of their early machines that the speed path associated with the cache access was twice as long as the ALU path and the control path. That is, the ALU was n nanoseconds, the control path 0.80n nanoseconds, and the cache access approximately 2n nanoseconds. The microarchitect set the cycle to what? How did he manage that? Answer in the box, 20 words or fewer:

n nanoseconds. Make the cache access a two cycle operation.

Part e (4 points): Mike Flynn observed that it did not matter how sophisticated we made our pipeline, or how many stages we had in it, we would never be able to complete on average more than one instruction each cycle as long as: (Finish your answer in the box in fewer than 20 words.)

we fetch only one instruction per cycle.

Name:__________________________

Problem 2 (15 Points):

Little Computer Inc. produces a machine in which all instructions execute in one cycle. However, the ISA does not include a multiply (MUL) instruction. To perform a multiplication, it takes 5 instructions from the existing ISA. Little Computer Inc. proposes a new ISA, which adds a MUL instruction. Everything else stays the same. In the new ISA, all instructions take one cycle, including the new MUL instruction. However, in order to accomplish this, the implementation of the proposed ISA has an increase in cycle time of 50%. Your job: to find out if the proposed ISA is a good idea.

Part a (8 points): Little Computer Inc simulates the proposed ISA and finds that on a set of representative benchmarks, 20% of the instructions executed on average are MUL instructions. Would it be a good idea to add the MUL to the ISA? Explain.

0.2 x 5 + 0.8 = 1.8

Since $1.8 > 1.5$, it is a good idea

Part b (7 points): What if, instead of 20%, only 10% of the instructions exectuted are MUL instructions? Good idea or bad idea? Explain.

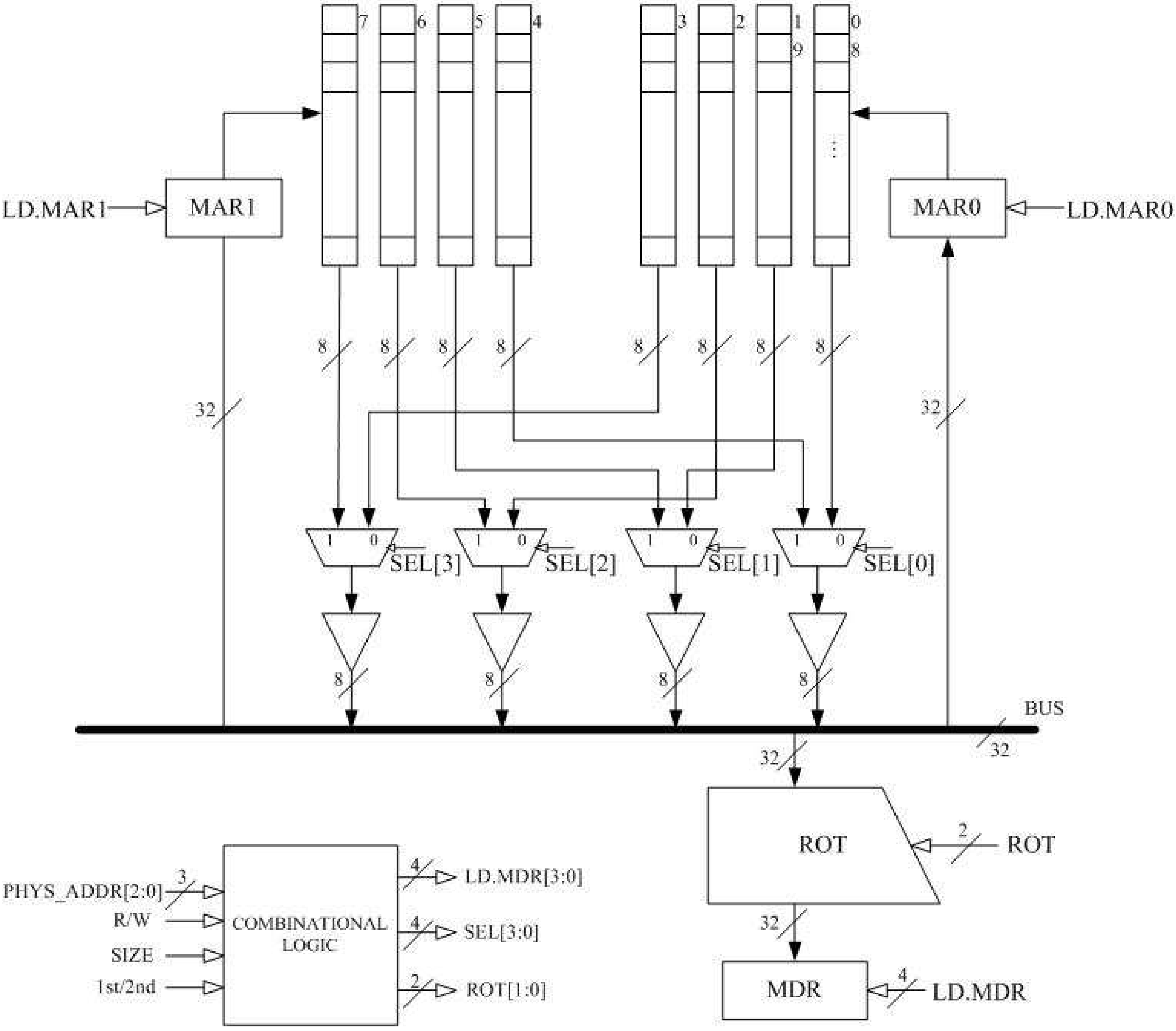
0.1 x 5 + 0.9 = 1.4

Since $1.4 < 1.5$, it is a bad idea

Name:__________________________

Problem 3 (20 points):

A two-way interleaved, byte addressable memory is shown in the figure. It takes **TEN** cycles after the MAR is loaded to load the MDR with the contents of the relevant memory locations. The processor supports unaligned memory accesses.



Name:

Problem 3 (Continued):

Part a (4 points) LDW R3, B loads a 32-bit word into R3. Consider the case where the address B is an integer of the form $(8 * k + 2)$, where k is an integer. What is the number of cycles required to get the contents of B into the MDR, after the cycle in which the processor puts the requested address on the bus.

10

Part b (4 points) Repeat part (a) where B is $(8 * k + 6)$, k is an integer.

11

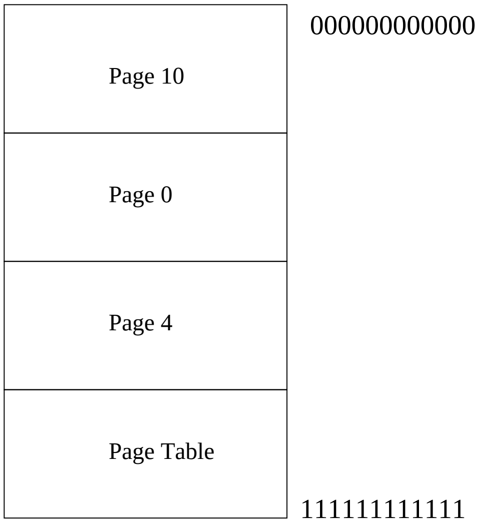
Part c (12 points) Construct the truth table to implement the combinational logic that controls the unaligned memory system shown in the figure for read accesses. You DO NOT need to worry about the datasize BYTE.

R/W	SIZE	1st/2nd	ADDR[2:0]	LD MAR[3:0]	ROT[1:0]	SEL[3:0]
R	H	1	000	0011	00	xx00
R	H	2	000	0000	xx	xxxx
R	H	1	001	0011	01	x00x
R	H	2	001	0000	xx	xxxx
R	H	1	010	0011	10	00xx
R	H	2	010	0000	xx	xxxx
R	H	1	011	0011	11	0xx1
R	H	2	011	0000	xx	xxxx
R	H	1	100	0011	00	xx11
R	H	2	100	0000	xx	xxxx
R	H	1	101	0011	01	x11x
R	H	2	101	0000	xx	xxxx
R	H	1	110	0011	10	11xx
R	H	2	110	0000	xx	xxxx
R	H	1	111	0001	11	1xxx
R	H	2	111	0010	11	xxx0
R	W	1	000	1111	00	0000
R	W	2	000	0000	xx	xxxx
R	W	1	001	1111	01	0001
R	W	2	001	0000	xx	xxxx
R	W	1	010	1111	10	0011
R	W	2	010	0000	xx	xxxx
R	W	1	011	1111	11	0111
R	W	2	011	0000	xx	xxxx
R	W	1	100	1111	00	1111
R	W	2	100	0000	xx	xxxx
R	W	1	101	0111	01	111x
R	W	2	101	1000	01	xxx0
R	W	1	110	0011	10	11xx
R	W	2	110	1100	10	xx00
R	W	1	111	0001	11	1xxx
R	W	2	111	1110	11	x000

Name:__________________________

Problem 4 (15 points):

A 16KB virtual address space is made up of 1KB pages. For this anemic system, operating system code and data structures are contained within the process' virtual address space. A snapshot of physical memory right now is shown below:



All pages that are resident belong to this process. The process consists of 12 pages of virtual address space. Since the last time the reference bits in all PTEs were cleared, the following memory accesses were made:

R-10, R-10, W-10, R-10, R-0, R-0, R-0, R-10, R-0,

where R-n means Read from page n, W-m means Write to page m.

Name:__________________________

Problem 4 (Continued):

Your job: Construct a snapshot of the COMPLETE page table for this process as it exists right now. Note, we have provided more than enough space to contain the COMPLETE page table. Use only what you need.

Each page table entry should consist of the minimum integer number of bytes needed to do the job. You may assign fields to the bits of the PTE anyway you wish.

Assume memory is protected on a page level of granularity. Assume two levels of privilege {Kernel, User}. Assume three kinds of access {None, Read, Write}. In the current process Kernel and User are allowed to read and write to all pages. Feel free to choose any code you feel is appropriate to represent the fact that Kernel and User can read and write to a page.

	V	M	R	Prot			PFN		
0	1	0	1	0	0	0	0	1	Page Table Base Register
1	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	
4	1	0	0	0	0	0	1	0	
5	0	0	0	0	0	0	0	0	
6	0	0	0	0	0	0	0	0	
7	0	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	
9	0	0	0	0	0	0	0	0	
10	1	1	1	0	0	0	0	0	
11	0	0	0	0	0	0	0	0	

Name:__________________________

Problem 5 (30 points):

We wish to add to the LC-3b the new instruction MWORDCPY, which copies the contents of k consecutive words to another set of k consecutive words. We will use the unused opcode 1010 for this purpose.

Encoding

	15 12	11 9	8 6	5	4		2 0
MWORDCPY DBaseR, SBaseR, CntR	**1010**	**DBaseR**	**SBaseR**	0	0	0	**CntR**

Operation

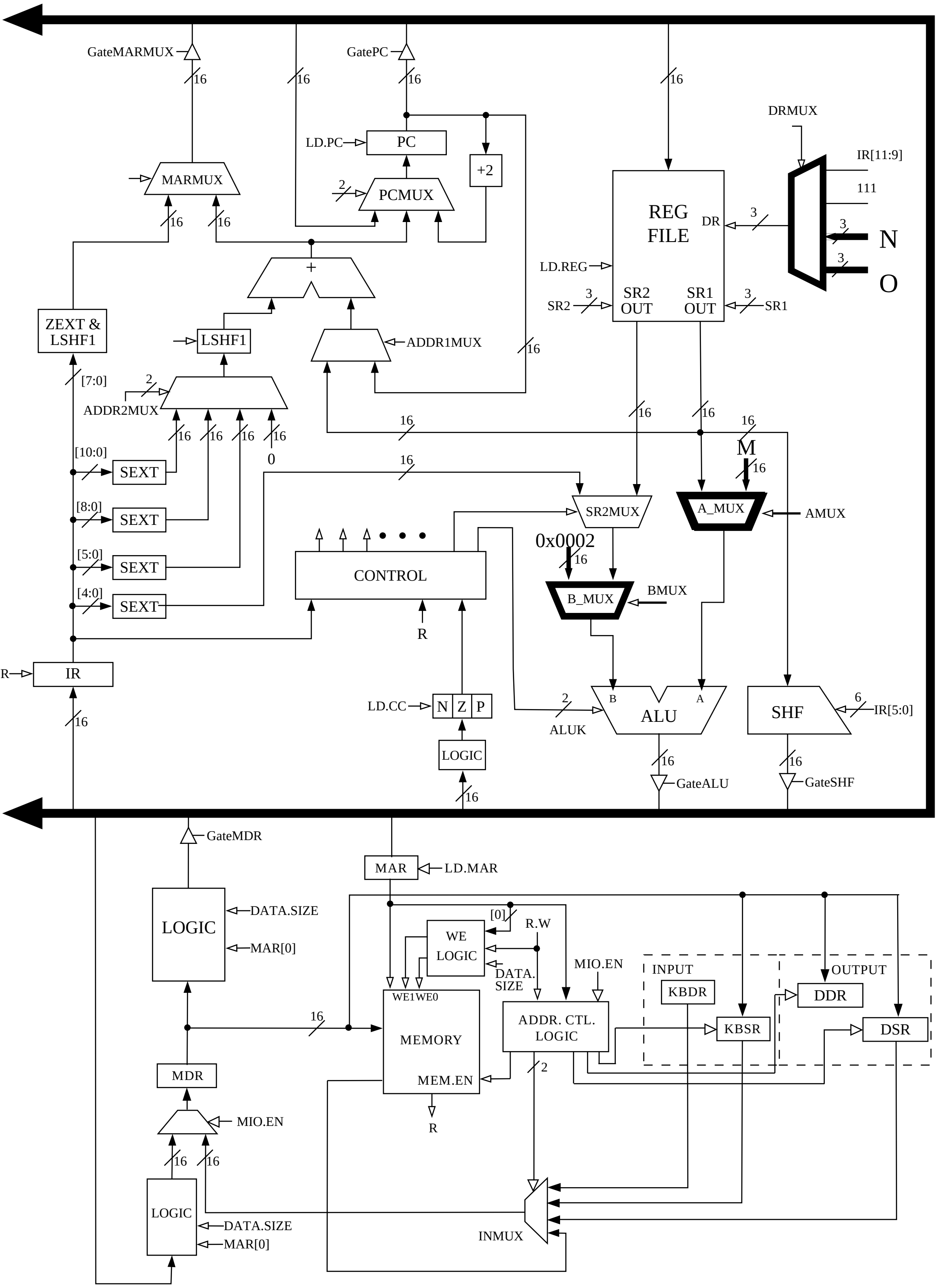
```
while(CntR > 0){
        MEM[DBaseR] = MEM[SBaseR];
        DBaseR = DBaseR + 2;
        SBaseR = SBaseR + 2;
        CntR = CntR - 1;
}
```

We will assume for simplicity here that DBaseR and SBaseR are different registers and that CntR contains a non-negative integer. You do not have to test for either of these.

In order to implement this instruction, we have added some hardware to the LC-3b datapath and some new logic in the microsequencer.

Name:_________________________

Problem 5 (Continued):



Name:__________________________

Problem 5 (Continued):

Part a (3 points) What are M, N, and O (see datapath on previous page). Be specific.

M -1 or 0xFFFF

N IR [8:6]

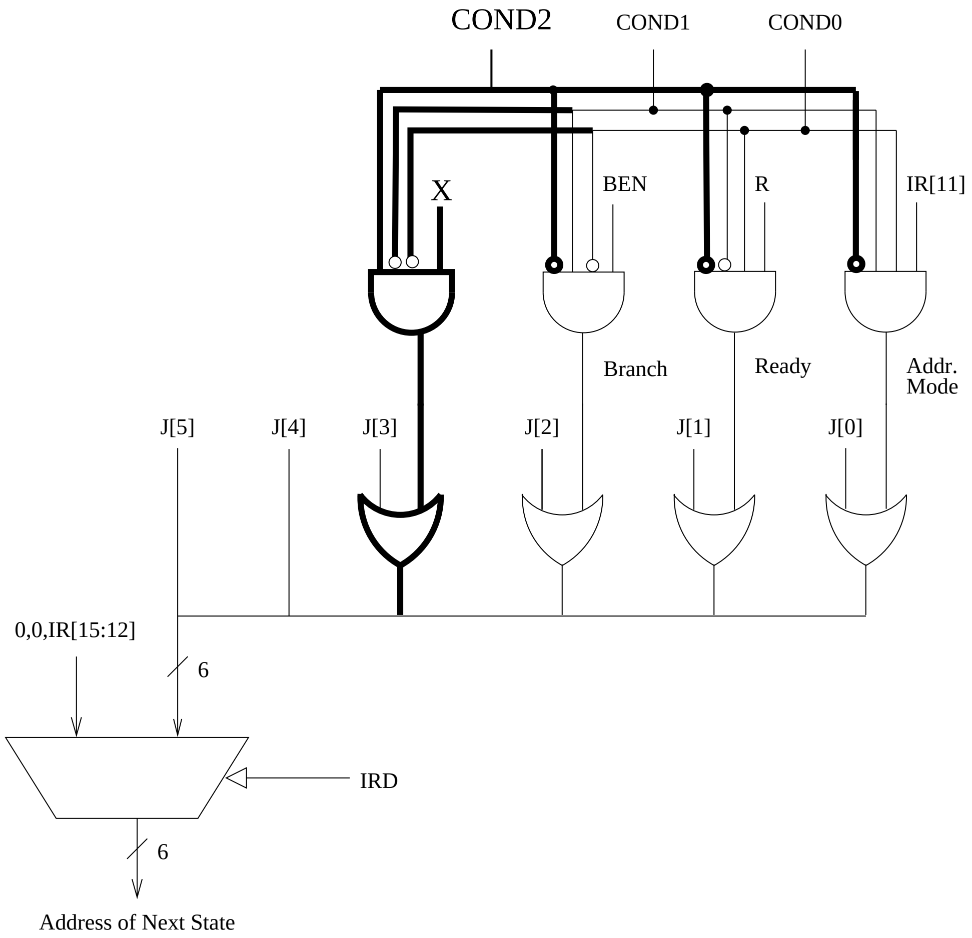
O IR [2:0]

Part b (2 points) What signal does X correspond to in the Microsequencer diagram shown below? (Hint: It is one of the condition code registers).

P

Part c (5 points) What is the Control Store address for C in the state machine on the next page.

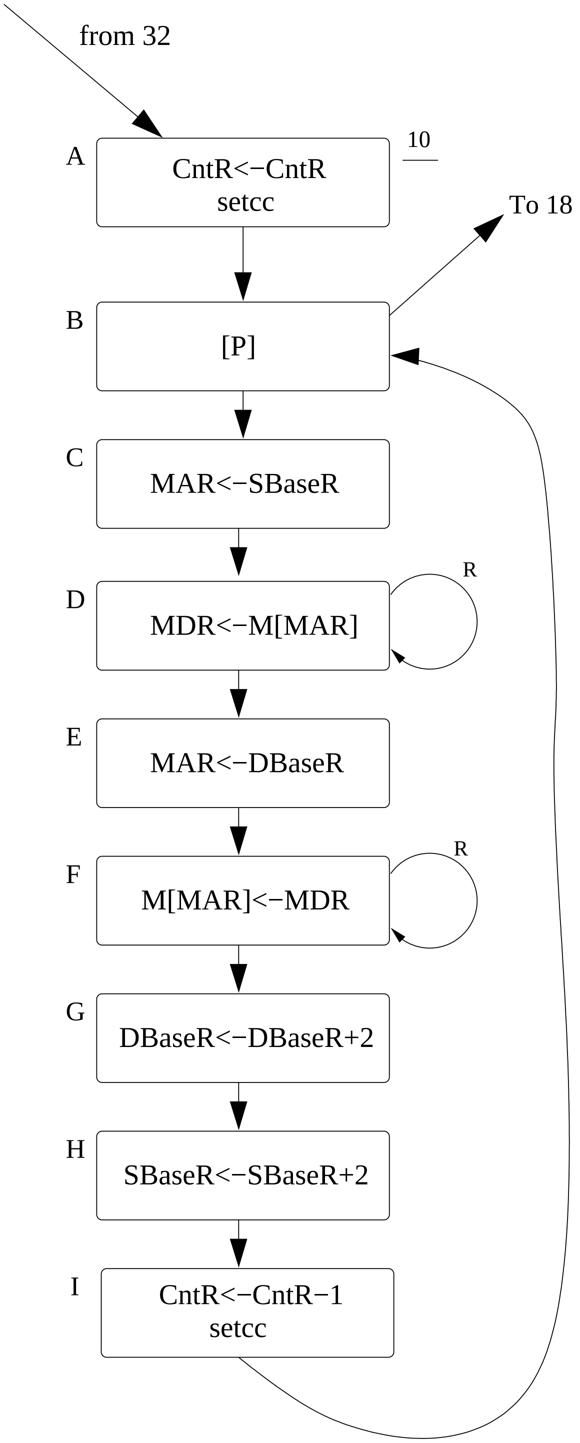
26



Name:__________________________

Problem 5 (Continued):

Part d (9 points) We show below the begining of the state diagram necessary to implement MWORDCPY. Using the notation of the LC-3b State Diagram, add the ”bubbles” you need to implement the MWORDCPY instruction. Describe inside each ”bubble” what happens in each state. You should be able to implement this in fewer than 15 states. (A TA found a solution that required only 8 states).



Name:__________________________

Problem 5 (Continued):

Part e (2 points) Give the values of the COND bits (COND0 , COND1, COND2) for the state labeled B.

COND2 __1___
COND1 __0___
COND0 __0___

Part f (9 points) The processing in each state you just added is controlled by asserting or negating each control signal. Enter a 1 or a 0 as appropriate for the microinstructions corresponding to the states you have added.

	LD.MAR	LD.MDR	LD.REG	LD.CC	GateMDR	GateALU	GateMARMUX	SR1MUX	AMUX	BMUX	DRMUX		ADDR2MUX		ADDR1MUX	MARMUX	ALUK		R.W	MIO.EN
A	0	0	1	1	0	1	0	0	1	0	1	1	0	0	0	0	0	1	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	1	0	0
D	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
E	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0
F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
G	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
H	0	0	1	0	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0
I	0	0	1	1	0	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0