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

EE 306, Fall 2021
Yale Patt, Instructor
TAs: Sabee Grewal, Ali Fakhrzadehgan, Ying-Wei Wu, Michael Chen, Jason Math, Adeel Rehman
Exam 2, November 17, 2021

Name:_____

Problem 1 (25 points):_____

Problem 2 (10 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.

I will not cheat on this exam.

_____
Signature

GOOD LUCK!

Name:_____

Problem 1. (25 points):

Part a. (5 points): If a memory location labeled A contains the contents B, then you know that `LDI R0, A` will load the contents of memory location B into R0. We can get rid of the LDI instruction by replacing each occurrence of it with two instructions that are currently in the LC-3 ISA. Specify completely (in LC-3 assembly language) the two instructions from the LC-3 ISA that accomplishes exactly the same thing as `LDI R0, A`. Note that `LDI R0, A` only uses one register (R0 in this case). Therefore, your solution should only use one register (R0 in this case).

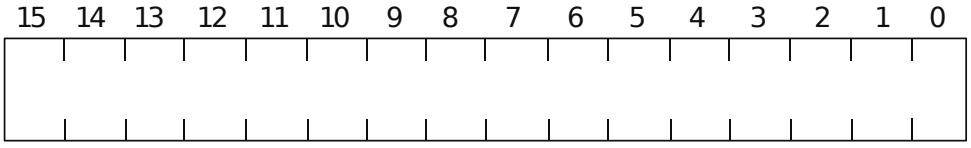
Part b. (5 points): A student wrote the following:

```
AND R0, R0, #0
LD R0, LABEL
```

What useful purpose, if any, does the AND instruction provide? Answer in 10 words or fewer.

Part c. (5 points): Many ISAs have an explicit MOV (or COPY) instruction that copies the value in one register into another register. For the LC-3, the COPY instruction is a special case of more than one existing instruction. That is, there are already LC-3 instructions which copy the value that is in SR into DR.

Choose an LC-3 instruction that copies the value that is in R4 into R5 and fill in the bits of the resulting instruction below.



Part d. (5 points): An aggie is trying to assemble a program but it won't assemble. The LC-3 assembler outputs:

"LD R0, A. error: cannot encode as 9-bit 2's complement integer."

What is the issue? Explain in 15 words or fewer

Part e. (5 points): The following subroutine does in software what a particular structure does in hardware.

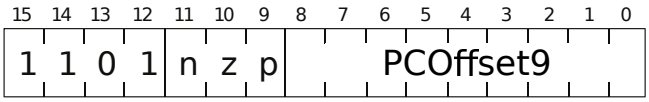
```
SUBROUTINE  ADD R0, R0, #0
            BRz ZERO
            ADD R3, R1,#0
            RET
ZERO        ADD R3, R2, #0
            RET
```

What is that structure? Answer in 10 words or fewer.

Name:_____

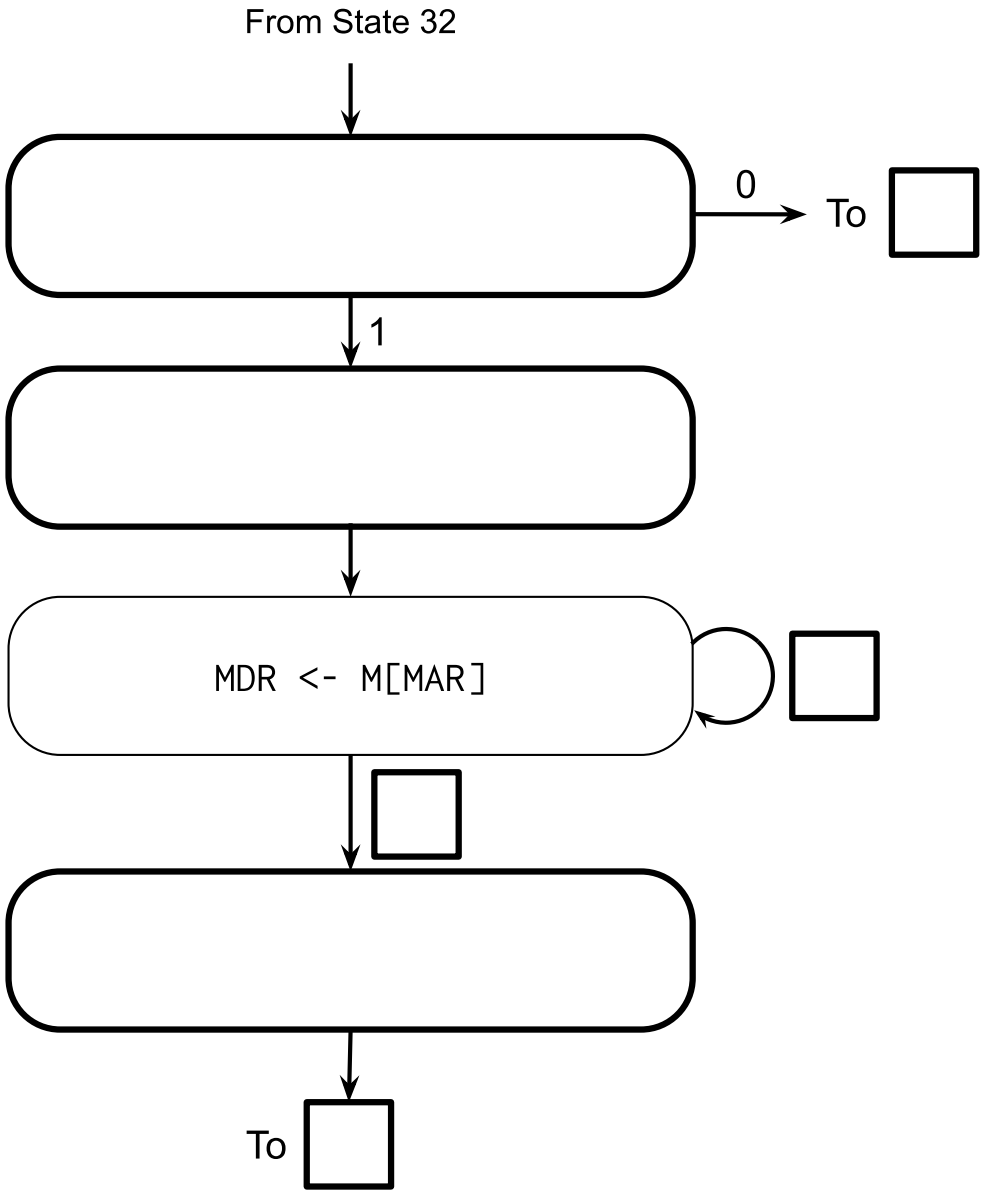
Problem 2. (10 points):

The BR instruction is limited by the 9-bit offset (similar to the LD and ST instructions). We fix this by using the unused opcode 1101 to add a new instruction, BRI (Branch Indirect). BRI works just like BR, except BRI branches to the address contained in the memory location formed by PC + PCoffset9. The format of the BRI instruction is shown below.



To implement BRI, we need to add four states to the LC-3 state machine, as shown below.

Your job: Fill in the missing information, showing clearly what each state does.



Name:_____

Problem 3. (20 points):

A mathematical expression is often written with parentheses — "(" is called the open parenthesis and ")" the closed. For an expression to be valid the number of open parentheses must equal the number of closed parentheses — and they must be in the right order. For example, here is a valid expression:

$$(x + 2)(x + 3)$$

Note that $(x+2\ (x+3)$ is not valid because it is missing a ")" close parentheses. And, $)x+2((x+3)$ is not valid because we have a close parenthesis before an open parenthesis.

We use computers to evaluate mathematical expressions all the time. One thing we have to do is check to make sure that the mathematical expression is valid. As a first step, we want to write a program in LC-3 assembly language that makes sure a mathematical expression has the correct parentheses.

Part a. (15 points): The program on the next page checks a mathematical expression for valid use of parentheses. The mathematical expression is input as a character string starting at location x3200. The program writes a 0 in RESULT if the sequence is valid, and a 1 in RESULT if the sequence is not valid.

Your job: Fill in the missing instructions.

Note: The PUSH and POP subroutines are not shown; they are the ones discussed in class. Both subroutines use R0 for the value to be pushed or popped. R5 is used to indicate success (R5=0) or failure (R5=1) of the push or pop. Assume that the stack is initially empty.

Part b. (5 points): Fill in the symbol table for this program. Use as many entries as you need.

Symbol	Address

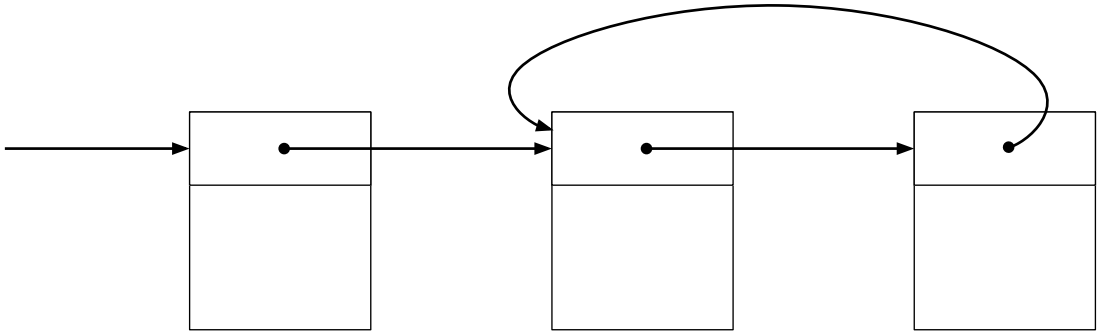
PROBLEM CONTINUES ON NEXT PAGE

```
            .ORIG x3000
            AND R4, R4, #0
            LD R1, MATH
            LD R2, OPENPAREN
            LD R3, CLOSEDPAREN
            LD R6, SP

LOOP        LDR R0, R1, #0
            BRz DONE
            ADD R1, R1 #1
            [                    ]
            BRz OPEN
            [                    ]
            BRz CLOSE
            BRnzp LOOP

OPEN        JSR PUSH
            BRnzp LOOP

CLOSE       JSR POP
            [                    ]
            BRp ERROR
            BRnzp LOOP

DONE        JSR POP
            ADD R5, R5, #0
            BRp STORE

ERROR       ADD R4, R4, #1
STORE       ST R4, RESULT
            HALT

MATH        .FILL x3200
SP          .FILL x4000 ; "SP" = "Stack Pointer"
OPENPAREN   .FILL #-40  ; Negative of ascii code for "("
CLOSEDPAREN .FILL #-41  ; Negative of ascii code for ")"
RESULT      .BLKW #1
            .END
```

Name:_____

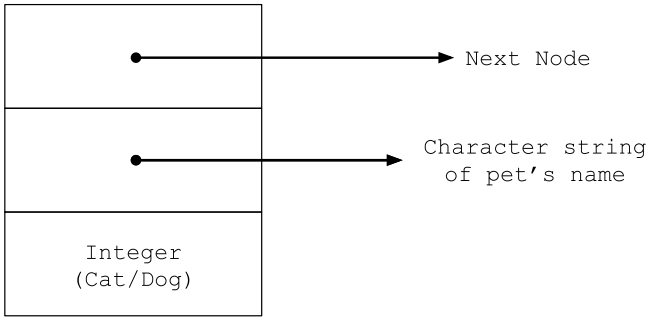
Problem 4. (20 points):

The veterinary clinic you helped in Programming Lab 3 has another problem. They noticed that some of their linked lists have *loops*. A linked list has a loop when a node points to an earlier node of the list, as shown below.



Part a. (4 points): What is the problem with a linked list having a loop? Explain in 20 words of fewer.

Part b. (16 points): The program on the next page determines whether or not a given linked list contains a loop. Assume each node has the same structure as the nodes in Programming Lab 3, as shown below.



Recall, the third word of the node uses bit 0 to indicate a cat or a dog. The program below may use bits 15 through 1 as necessary.

The address of the first node is contained in memory location x4000. The program writes 1 in memory location x4001 if the list has a loop and a 0 otherwise.

Your Job: Fill in the missing instructions.

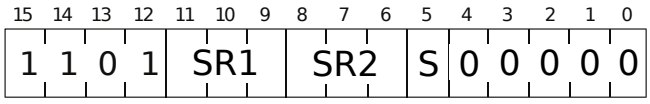
PROBLEM CONTINUES ON NEXT PAGE

```
                .ORIG   x3000
                LDI R0, LIST
                BRz DONE

REPEAT          [                    ]
                ADD R1, R1, #0
                BRnp DONE

                JSR MARK_VISITED
                [                    ]
                BRnp REPEAT
                AND R1, R1, #0

DONE            [                    ]
                HALT

CHECK_VISITED   LDR R1, R0, #2
                [                    ]
                BRz NOT_VISITED
                AND R1, R1, #0
                ADD R1, R1, #1
                RET
NOT_VISITED     AND R1, R1, #0
                RET

MARK_VISITED    ST R1, SAVE_R1
                LDR R1, R0, #2
                AND R1, R1, x-3
                ADD R1, R1, x2
                STR R1, R0, #2
                [                    ]
                RET
SAVE_R1         .BLKW #1

LIST            .FILL   x4000
OUTPUT          .FILL   x4001
                .END
```

Name:_____

Problem 5. (25 points):

Using the unused opcode 1101, we implemented a new instruction with the following format.



The instruction has two addressing modes. Bit 5 is the steering bit that determines which addressing mode is used. Note also that to support this new instruction we have added a special purpose register to the datapath called TempR.

The LC-3 executes the instruction 1101 001 010 0 00000. Shown below are snapshots of the relevant register values taken before and after executing the instruction.

Before	
R1	x1111
R2	x2222
TempR	x0000
M[x1111]	xAAAA
M[x2222]	xFFFF

After	
R1	x2222
R2	x1111
TempR	x1111
M[x1111]	xAAAA
M[x2222]	xFFFF

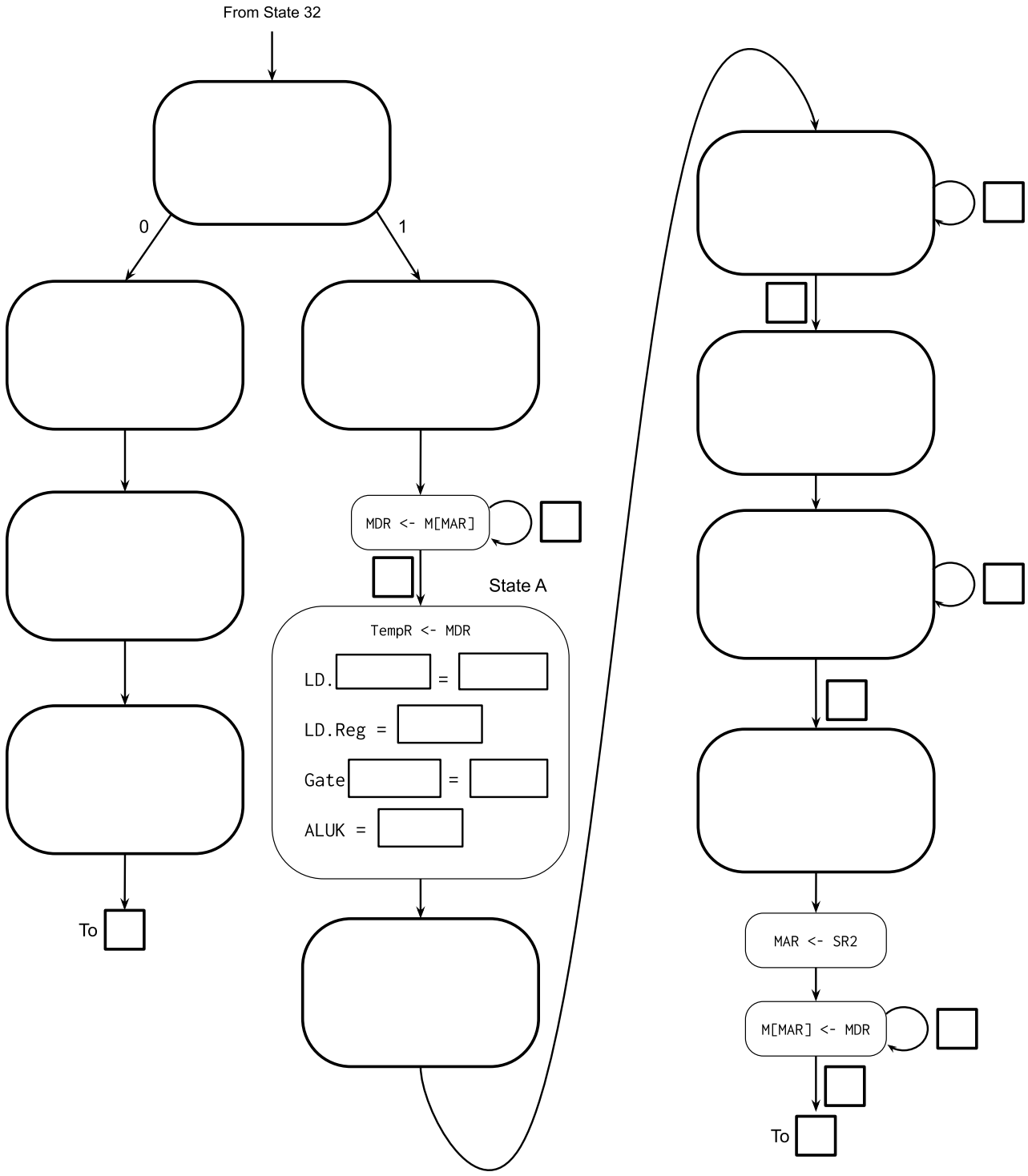
The LC-3 executes the instruction 1101 001 010 1 00000. Shown below are snapshots of the relevant register values and memory contents taken before and after executing the instruction.

Before	
R1	x1111
R2	x2222
TempR	x0000
M[x1111]	xAAAA
M[x2222]	xFFFF

After	
R1	x1111
R2	x2222
TempR	xAAAA
M[x1111]	xFFFF
M[x2222]	xAAAA

Part a. (5 points): What does the new instruction do? Explain the difference between the two addressing modes; i.e., what happens when bit 5 of the instruction is 0 and what happens when bit 5 of the instruction is 1. Explain in 25 words or fewer.

Part b. (12 points): Fill in the missing information of the state machine to support the new instruction. In addition, for the state labelled "State A", we are also asking you to specify the individual control signals. Recall that the ALUK control signal can be AND, ADD, NOT, or PASSA.



PROBLEM CONTINUES ON NEXT PAGE

Part c. (8 points): Draw the necessary additions to the datapath for the new instruction to work correctly. Be sure to show the necessary logic and control signals to implement this new instruction.