

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

ECE 306 Fall 2023

Instructor: Yale N. Patt

TAs: Chester Cai, Sophia Jiang, Ali Mansoorshahi, Jaeyoung Park, Anna Guo, Asher Nederveld,  
Edgar Turcotte, Nadia Houston, Varun Arumugam,

Exam 2

Nov 13, 2023

Name and EID: \_\_\_\_\_

Problem 1 (15 points): \_\_\_\_\_

Problem 2 (20 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 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: \_\_\_\_\_

**Question 1 (15 points):** Answer the following questions.

**Note:** For each of the four answers below, if you leave the box empty, you will receive one point.

**Part a (5 points):** The LC-3 assembler sees the following line in an assembly language program:

```
A .FILL xF025
```

What does xF025 represent? Circle only ONE choice and explain.

- a. TRAP x25
- b. The 2's complement integer 1111 0000 0010 0101 (which is -4059 in decimal)
- c. Not a and not b
- d. We can not tell from the information provided

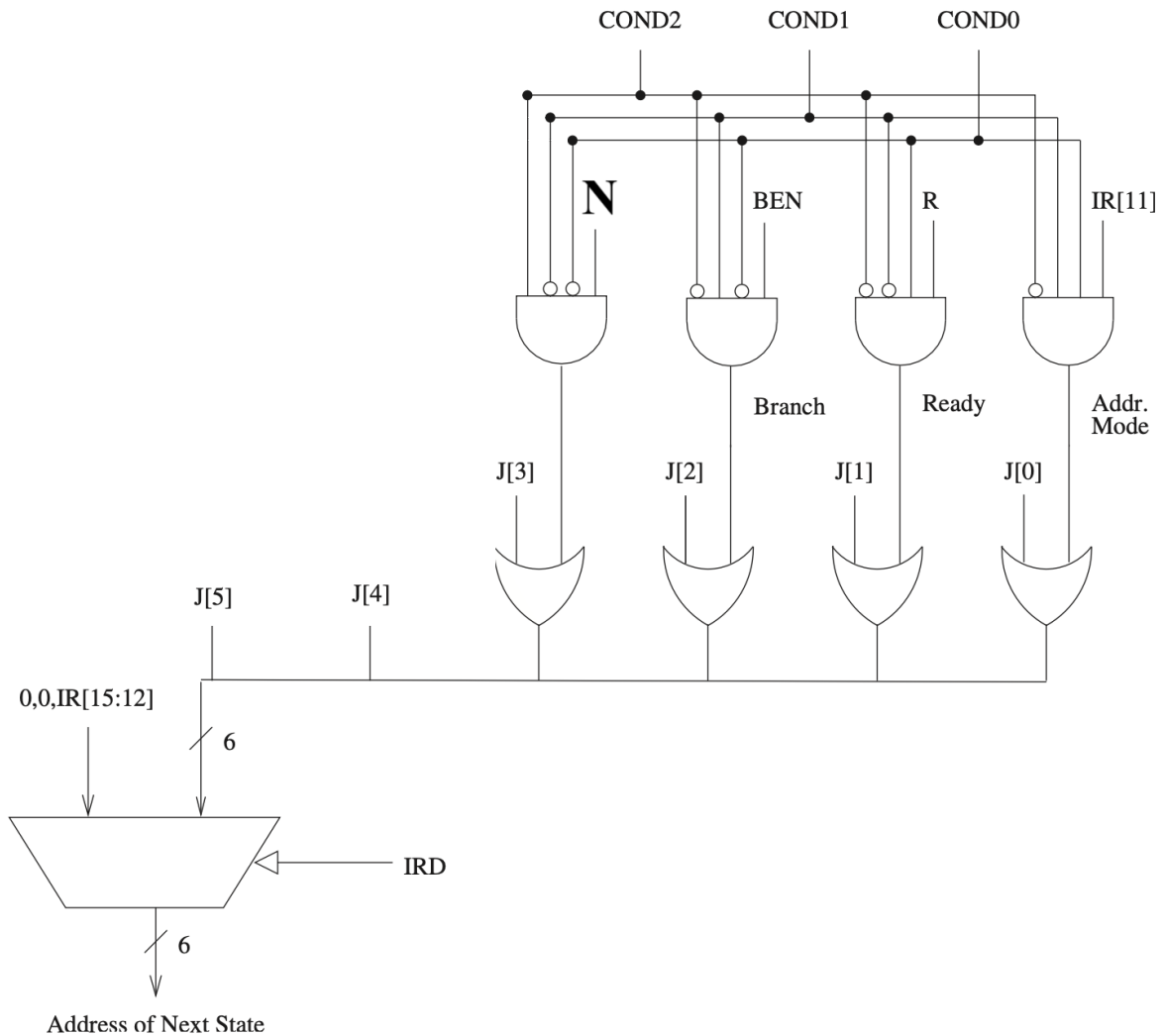
**Part b (5 points):** The following program fragment is in memory locations x4000, x4001, x4002, x4003. The PC contains x4000. The program fragment is allowed to execute. As a result of this being executed, what will be stored in locations x4010 to x4FFF?

```
0010 000 000000010
0011 000 000000000
1111 0000 00100101
0011 000 000000000
```

**Part c (5 points):** In the interest of saving clock cycles, a microarchitect decided to have a single state in the state machine do the following: **MAR←--PC+off9** and **MDR←--SR** without changing the datapath. Saving clock cycles is in general a good idea. Is this particular example a good idea or a bad idea? Explain.

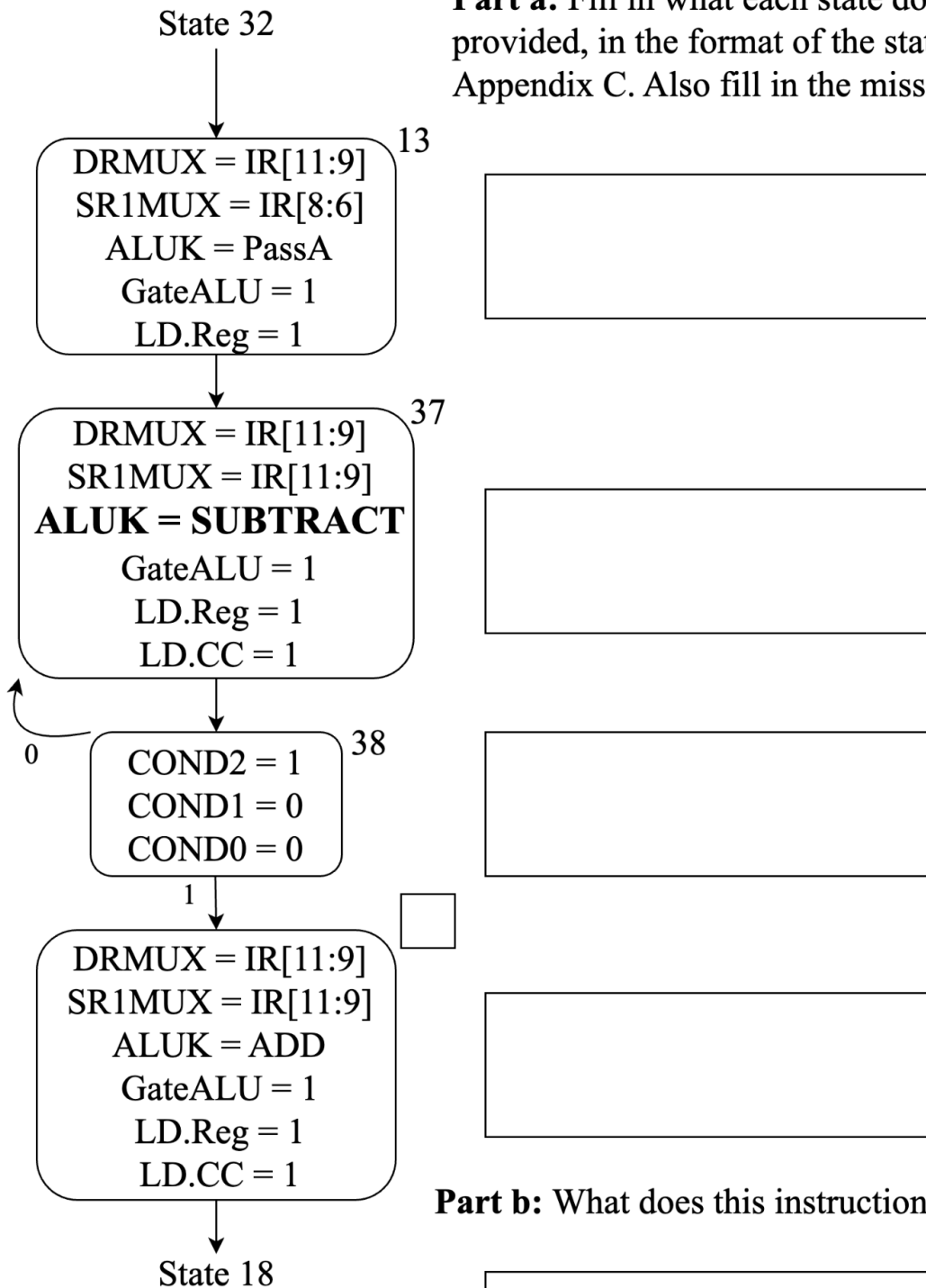
Name: \_\_\_\_\_

**Question 2 (20 points):** We wish to add a new instruction to the LC-3 using the unused opcode 1101. To implement this instruction, we made modifications to the state machine, microsequencer, and ALU. The changes to the microsequencer are shown below. The **N** signal refers to the N bit in the condition codes. The changes to the state machine are shown on the next page. A new SUBTRACT mode was added to the ALU, where the output of the ALU will be  $A - B$ .



Name: \_\_\_\_\_

**Part a:** Fill in what each state does in the box provided, in the format of the state diagram from Appendix C. Also fill in the missing state number

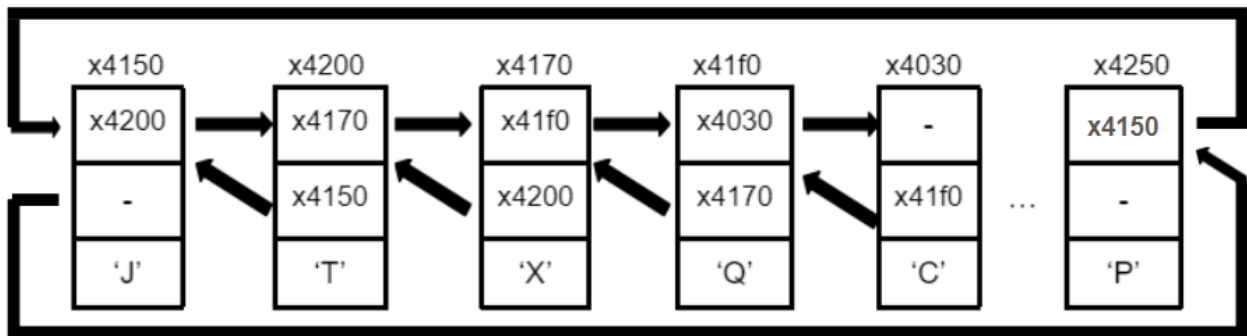


**Part b:** What does this instruction do?

Name: \_\_\_\_\_

**Question 3 (15 points):** Encryption is a mechanism that allows two people (John and Mary) to send messages to each other such that anyone else seeing the messages can not tell what the messages say. A very simple algorithm for doing this is the Caesar Cipher, which requires only that John and Mary both know the order of the letters in the English alphabet and a number  $N$ . It works as follows: John wants to send a message to Mary. He replaces each letter in his message ( $x$ ) with the letter ( $y$ ) which is  $N$  positions later in the alphabet, and sends the resulting message to Mary. For example, if John wanted to send the message BUY, and  $N=2$ , he would replace the letters of BUY with DWA and send that message to Mary. Note that Y is close to the end of the English alphabet, so we use A as the letter after Z. That is  $Y + 2 = A$ .

Since everyone knows the order of letters of the English alphabet, this encryption mechanism is easy to break by simple trial and error. Thus we decided to develop a better mechanism that is much harder to break. Instead of John and Mary having to know the order of the letters of the English alphabet, they now use a linked list to specify that order. That is, John and Mary both have copies of the following doubly linked list in addition to  $N$ .



Each node in the doubly linked list consists of three words, two pointers (the address of the next node and the address of the previous node) and a letter of the English language. As before,  $N$  specifies the number of nodes ahead to replace a given letter. That is, if  $N=2$ , and John wants to send the message TJX, he would replace those three letters with QXC, and send that to Mary.

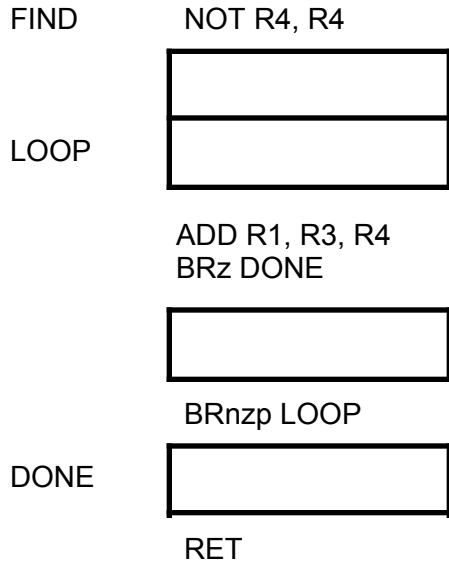
To account for letters close to the end of the linked list, we include a forward pointer from the last nodes to the first, and a backward pointer from the first node to the last.

**Your job:** Complete the two subroutines below by adding the missing instructions. John needs FIND to find the letter  $x$  in the linked list that is to be replaced, and REPLACE to find the letter to replace it with.

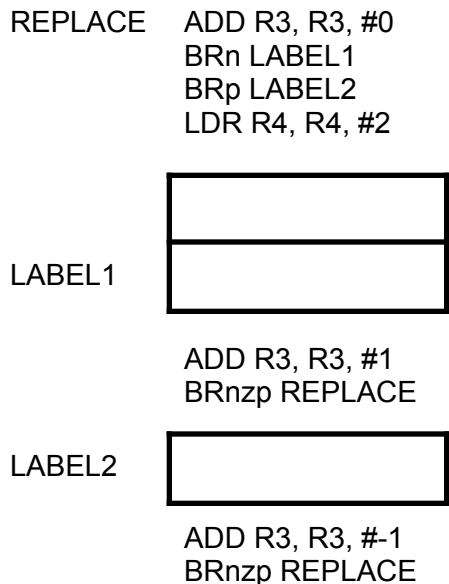
Name: \_\_\_\_\_

**Part A:** Before execution of FIND, R4 contains the ASCII code of the letter to be replaced (x). R2 contains the address of the node containing the letter "A." After execution of the subroutine, R4 contains the address of the node containing x.

**Note:** The address of a node is the address of the first word of that node. For example, the address of the node containing the letter T is x4200.



**Part B:** Before execution of REPLACE, R3 contains the shift amount N, and R4 contains the address of the node containing the letter x. After execution of the subroutine, R4 will contain the ASCII code of the letter y. Note that N can be either positive or negative.



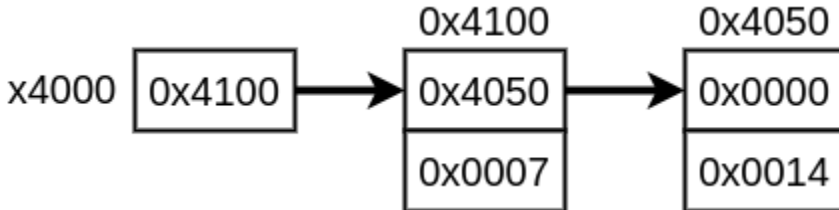
Name: \_\_\_\_\_

**Question 4. (25 points)** In class we implemented the stack data structure in sequential memory locations with the Stack Pointer pointing to the location that contains the top of the stack. In this problem, we will implement a stack using a linked list.

Each node in the linked list consists of two LC-3 words. The first word contains a pointer to the next node. The second word contains a 16-bit value.

The stack pointer is in memory location x4000. That is,  $M[x4000]$  contains the address of the top of stack. If the stack is empty,  $M[x4000]$  contains x0000.

**Part A** The stack below contains two 16-bit values, 7 (x0007) and 20 (x0014)..



The following subroutine pushes a value onto the stack. Before the PUSH routine is called, R0 points to the node to be pushed (i.e., R0 contains the address of the first word of the node to be pushed onto the stack).

```
PUSH  LDI  R1, HEAD
      STR  R1, R0, #0
      STI  R0, HEAD
      RET
HEAD  .FILL x4000
```

The value of the node to be pushed is x0003. The address of the node is x6000. After the PUSH completes successfully, the stack contains three nodes. Draw the linked list implementation of the stack, similar to the figure shown above.

Name: \_\_\_\_\_

**Part B** Your job here is to implement a subroutine that POPS a value from the stack, puts it in R0, and sets R5 to 0 if the POP routine executed successfully. If the POP was not successful, the POP routine puts the value 1 in R5. You should be able to do this in fewer than 15 instructions.

This one time only, you can assume the program calling POP will not need to use the values in any of the registers that the POP routine uses, after the POP routine executes.

<pre>POP  HEAD .FILL x4000</pre>
----------------------------------



Name: \_\_\_\_\_

**Question 5 (25 points):** The subroutine below (GETS) allows N characters typed on the keyboard to be stored in a one-dimensional memory array (i.e., successive memory locations), starting at the address specified in R0. N is specified in R1 before the subroutine is called. Assume the subroutine is allowed to access KBSR and KBDR.

The subroutine will also do the following:

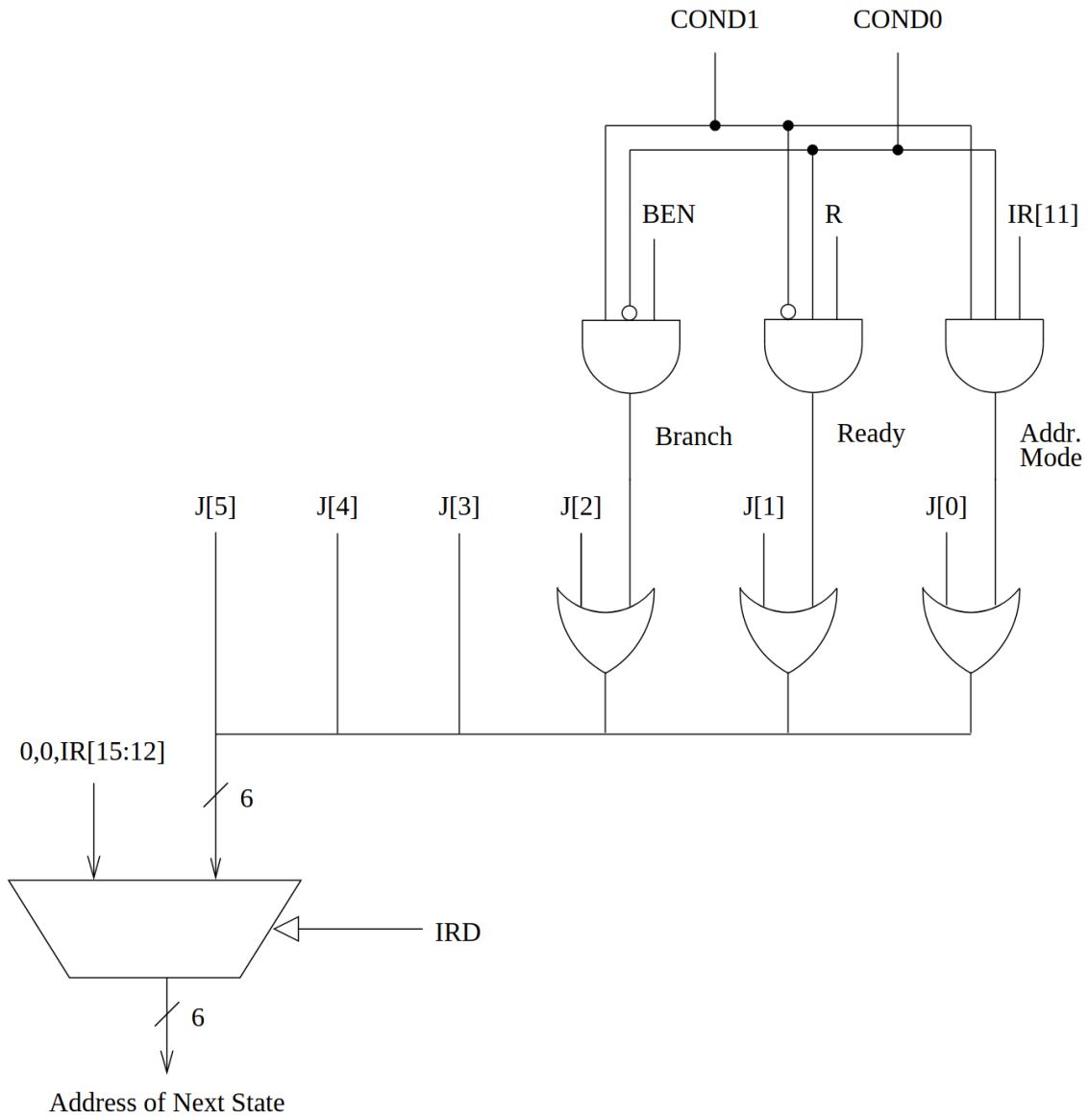
- It will stop early if the user types <ENTER> (ASCII x0A).
- <ENTER> will not be part of the array if it was typed by the user.
- It will store x0000 at the end of the array.
- It is responsible for saving and restoring registers as necessary.
- It can not call TRAP routines.

**Your Job: Fill in the missing instructions below.**

```
GETS      ST R0, SAVE_R0           ; R0 contains address of array
          ST R1, SAVE_R1           ; R1 contains number of inputs
          ST R2, SAVE_R2
          
          ADD R1, R1, #0
LOOP1     BRz DONE
LOOP2     
          BRzp LOOP2
          
          LD R3, NEG_ENTER
          
          
          STR R2, R0, #0
          
          ADD R1, R1, #-1
          BRnzp LOOP1
DONE      
          STR R2, R0, #0
          LD R0, SAVE_R0
          LD R1, SAVE_R1
          LD R2, SAVE_R2
          
          
SAVE_R0   .BLKW #1
SAVE_R1   .BLKW #1
SAVE_R2   .BLKW #1

KBSR     .FILL xFE00
KBDR     .FILL xFE02
NEG_ENTER .FILL xFFF6           ; negative of ASCII for <ENTER>
```

Name: \_\_\_\_\_



Name: \_\_\_\_\_

**This page is left blank intentionally. Feel free to use it for scratch work.  
You may tear the page off if you wish.  
Nothing on this page will be considered for grading.**