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

EE 306, Fall 2019
Yale Patt, Instructor
TAs: Sabee Grewal, Arjun Ramesh, Joseph Ryan, Chirag Sakhuja, Meiling Tang, Grace Zhuang
Exam 2, November 20, 2019

Name:_____

Problem 1 (15 points):_____

Problem 2 (15 points):_____

Problem 3 (10 points):_____

Problem 4 (15 points):_____

Problem 5 (20 points):_____

Problem 6 (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. (15 points):

Part a. (5 points): How many of the 15 LC-3 instructions assert the LD.PC control signal during its instruction cycle? Explain in 10 words or fewer.

Part b. (5 points): Consider the following program written in LC-3 assembly language:

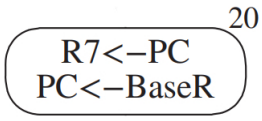
```
            .ORIG x3000
            LEA R0, LABEL
            LD  R1, LABEL
            STI R0, LABEL
            LDR R2, R1, #0
            HALT
LABEL       .FILL x4000
            .END
```

The program is loaded into the LC-3 simulator, a breakpoint is set at x3004, and the program is run. What are the resulting values in R0, R1, and R2?

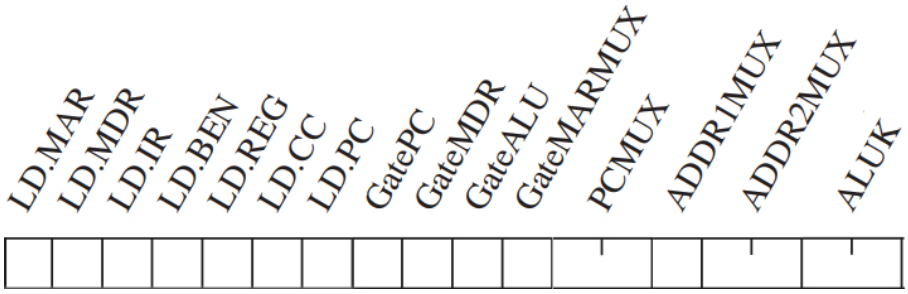
R0: R1: R2:

Name:_____

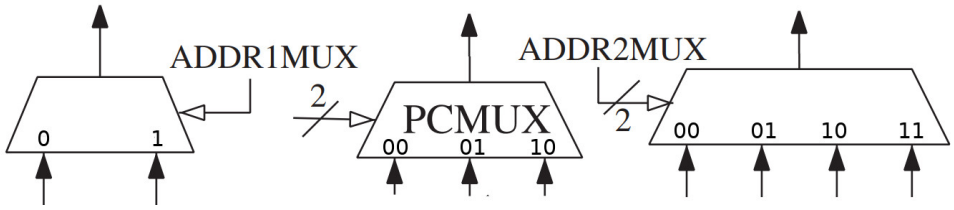
Part c. (5 points): State 20 of the LC-3 state machine is shown below.



Your Job: Fill in the output control signals for state 20. That is, fill in each control signal with 1, 0, or X depending on whether the corresponding control signal must be 1, 0, or you don't care.



We have provided information below about some of the muxes that you might find helpful. ALUK works as follows: 00 is ADD, 01 is AND, 10 is NOT, and 11 is PASSA.



Name:_____

Problem 2. (15 points):

Part a. (7 points): Consider the following program written in LC-3 assembly language:

```
        .ORIG x3000
        LDI R0, INPUT
        BRzp SKIP
        NOT R0, R0
        ADD R0, R0, #1

SKIP    ADD R1, R0, #0
        AND R2, R2, #0
LOOP    ADD R2, R2, R0
        ADD R1, R1, #-1
        BRp LOOP

        STI R2, OUTPUT
        HALT
INPUT   .FILL x4000
OUTPUT  .FILL x4001
        .END
```

What does the program do? Answer in 10 words or fewer.

Part b. (8 points): The program below operates on two linked lists. The address of the first node of the first list is stored at location x4000. The address of the first node of the second list is stored at location x4001. The first word of each node contains a pointer to (i.e., the address of) the next node.

```
        .ORIG x3000
        LD R0, LIST1
LOOP    LDR R1, R0, #0
        BRz NEXT
        LDR R0, R0, #0
        BRnzp LOOP
NEXT    LDI R1, LIST2
        STR R1, R0, #0
        HALT
LIST1   .FILL x4000
LIST2   .FILL x4001
        .END
```

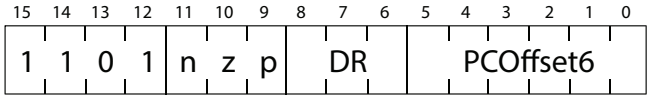
What does the program do? Answer in 10 words or fewer.

Name:_____

Problem 3. (10 points):

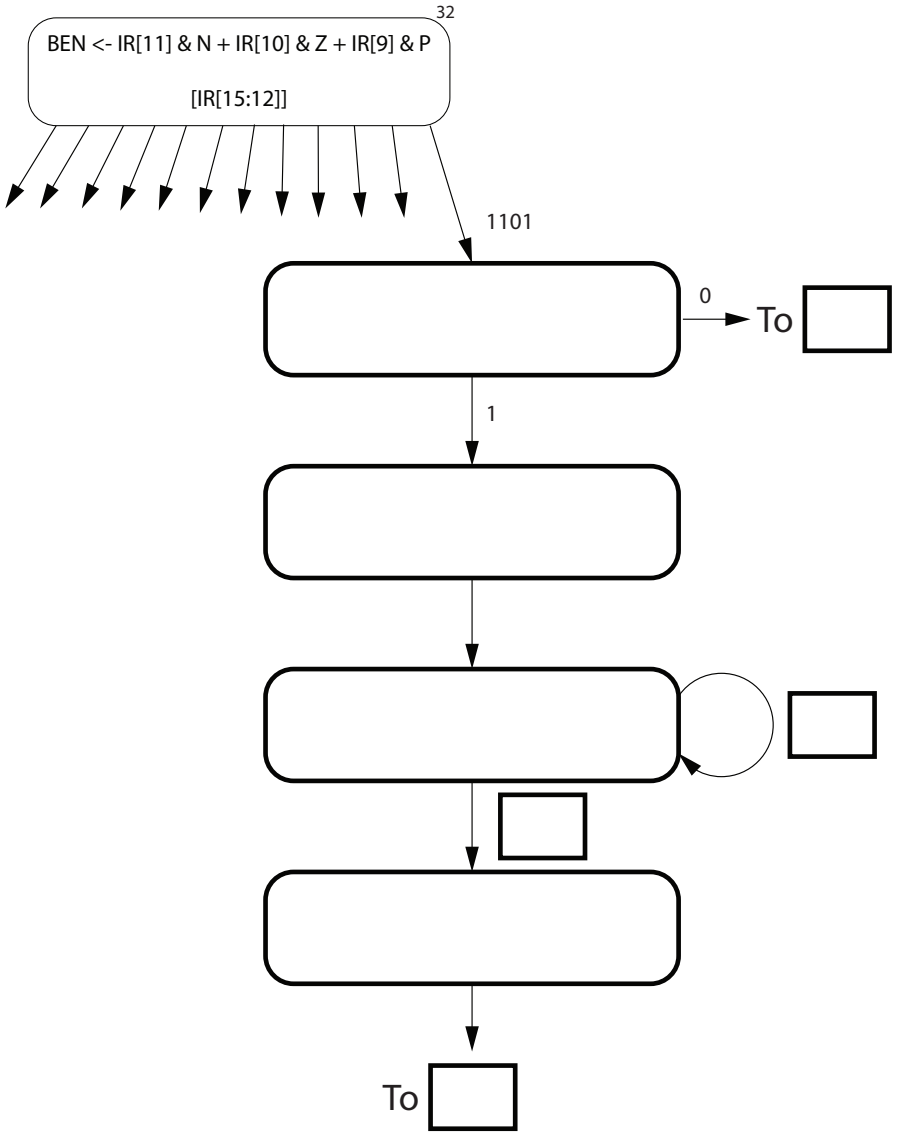
On exam 1, we introduced a conditional load instruction (LDC) which is similar to the conditional branch instruction. In both cases, the condition codes specified in bits [11:9] are tested. If none of the condition codes tested are set, the instruction acts as a no-op. In the case of LDC, if any of the condition codes tested are set, the PC-offset is added to the incremented PC, and the value in the resulting memory location is loaded into DR. **Condition codes are set based on the value loaded into DR. Note that this instruction does not produce Access Control Violations (ACVs)**.

LDC uses the unused opcode and has the following format:



To implement LDC, 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 4. (15 points):

Consider the following nonsense program written in LC-3 assembly language.

```
        .ORIG x3000
VALUE   .FILL xEFFF
        LEA R0, ARRAY
        LD R1, N
        ADD R0, R0, R1
        LD R2, VALUE
LOOP    STR R2, R0, #0
        ADD R0, R0, #-1
        ADD R2, R2, #2
        ADD R1, R1, #-1
        BRzp LOOP
ARRAY   .BLKW #20
N       .FILL #19
        .END
```

Note: Assume that memory locations x3000 through xFDFF all contain x0000 before this program was loaded into the LC-3.

Part a. (4 points): Complete the symbol table generated while assembling the program.

Symbol	Address
VALUE	
LOOP	
ARRAY	
N	

PROBLEM CONTINUES ON NEXT PAGE

Part b. (4 points): Before the program is run, the registers contain the following values.

R0	x1D24
R1	x3C32
R2	x3C5C
R3	x4D5D
R4	xD296
R5	x7BBB
R6	x18C2
R7	x4B4F

Recall that a breakpoint can be placed at a memory location to pause execution of the program when the PC hits that memory location (i.e. the instruction at that location will not have executed yet). We set a breakpoint a x3002 and then run the program.

Your job: Fill in the contents of the registers when the breakpoint at x3002 is hit. Only fill in the registers that change.

R0	
R1	
R2	
R3	
R4	
R5	
R6	
R7	

Part c. (7 points): The breakpoint at x3002 is removed, a breakpoint is placed at x3100, and the program is restarted. Does the breakpoint at x3100 get hit? Explain why or why not in 25 words or fewer.

Name:_____

Problem 5. (20 points):

A palindrome is a word, phrase, or sequence that reads the same, backward and forward. Some examples are *madam, mom,* and *racecar*.

We wrote a program (see next page) in LC-3 assembly language that determines whether a single word is a palindrome. Some of the instructions are missing.

Specifically, the program inputs a word from the user, one typed character at a time, followed by the ENTER key, then determines if that word is a palindrome. If the word is a palindrome, a 1 is stored in the location labeled OUTPUT. If not, a 0 is stored in OUTPUT. Note that the ASCII code for ENTER is #10.

The program uses a stack and a queue, using subroutines PUSH, POP, INSERT, and REMOVE. R6 is the stack pointer; R3 and R4 point to the front and rear of the queue. After PUSH is executed, the value in R0 will be pushed on to the stack. After POP is executed, R0 will contain the value popped from the stack. After INSERT is executed, the value in R0 will be inserted at the rear of the queue. After REMOVE is executed, R0 will contain the value removed from the front of the queue. For all subroutines, if the subroutine executes successfully, then R5 will contain 0. If not, R5 will contain 1.

Assume the stack and queue are initially empty, and that both have enough space allocated to fit the input string without overflowing.

Your Job: Fill in the missing instructions.

```
          .ORIG x3000
          LD R6, SP
          LD R3, QUEUE
          ADD R4, R3, #0

          LD R1, NENTER
AGAIN     TRAP x23
          [                    ]
          BRz PHASE2
          JSR PUSH
          JSR INSERT
          BRnzp AGAIN

PHASE2    JSR POP
          [                    ]
          BRnp A
          ADD R1, R0, #0
          JSR REMOVE
          [                    ]
          [                    ]
          [                    ]
          BRz PHASE2
          AND R0, R0, #0
          BRnzp DONE
A         AND R0, R0, #0
          ADD R0, R0, #1
DONE      ST R0, OUTPUT
          HALT

NENTER    .FILL #-10
SP        .FILL xFE00
QUEUE     .FILL x8000
OUTPUT    .BLKW #1
          .END
```

Name:_____

Problem 6. (25 points):

A program is executed in the LC-3. The following table shows the contents of the MAR and MDR for the first ten memory accesses made during the execution of the program. Note that several entries in the table have been left blank.

Your Job: Fill in the missing entries.

MAR	**MDR**
	xA610
x480C	
x4801	x40C0
	x1DBF
	x7180
x7FFF	
x4A20	xC1C0
	x7E04
x5008	