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,
Final Exam
Dec. 8th, 2023

Name and EID: ____

Part A:

Problem 1 (10 points): _______

Problem 2 (10 points): _______

Problem 3 (10 points): _______

Problem 4 (10 points): _______

Problem 5 (10 points): _______

Part A Total: _______________

Part B:

Problem 6 (20 points): _______

Problem 7 (20 points): _______

Problem 8 (20 points): _______

Problem 9 (20 points): _______

Part B Total: _______________

Total (130 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.

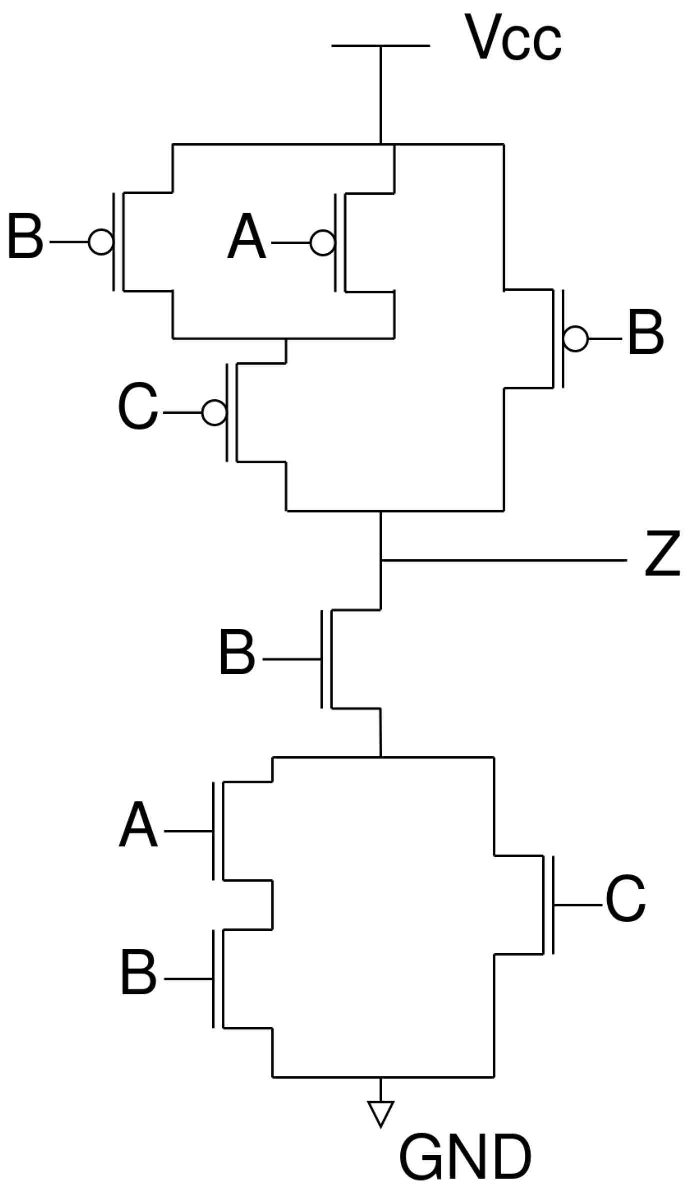
Signature: ____

GOOD LUCK!
(Have a good winter break)

Name: ______________________________________

Question 1 (10 Points): Shown below is a transistor circuit with three inputs (A, B, C) and one output (Z). Also shown is the truth table for this circuit. The outputs of the truth table are not shown.

Your job: Complete the truth table. Every input combination produces an output of either 0 or 1.



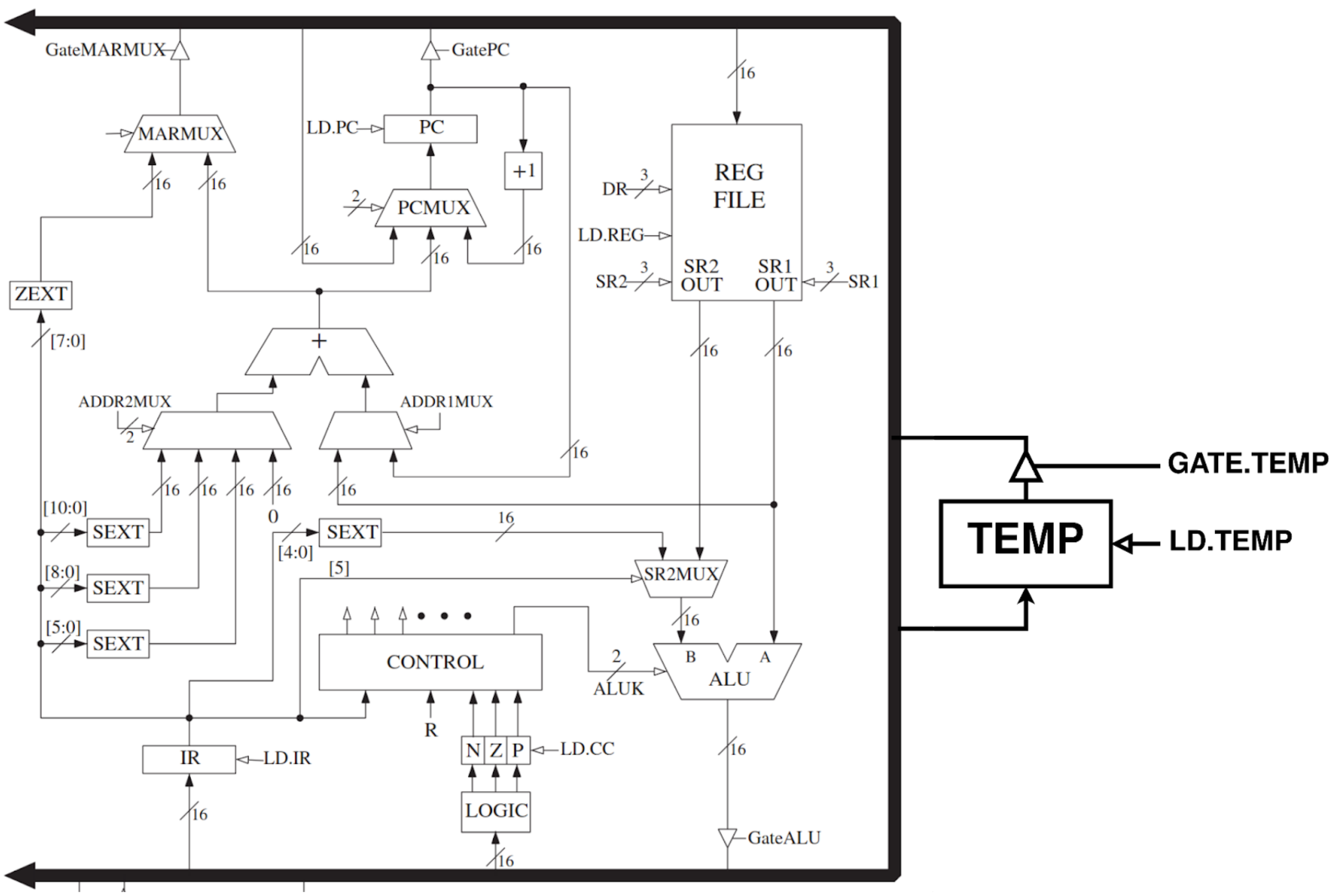
A	B	C	Z
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Name: ___________________________________

Question 2 (10 Points): We wish to implement a new instruction that swaps 2 values, one in a register and one in memory. The instruction's format is shown below. SR1 contains a value to be swapped and SR2 contains an address to a location in memory that has the other value to be swapped.

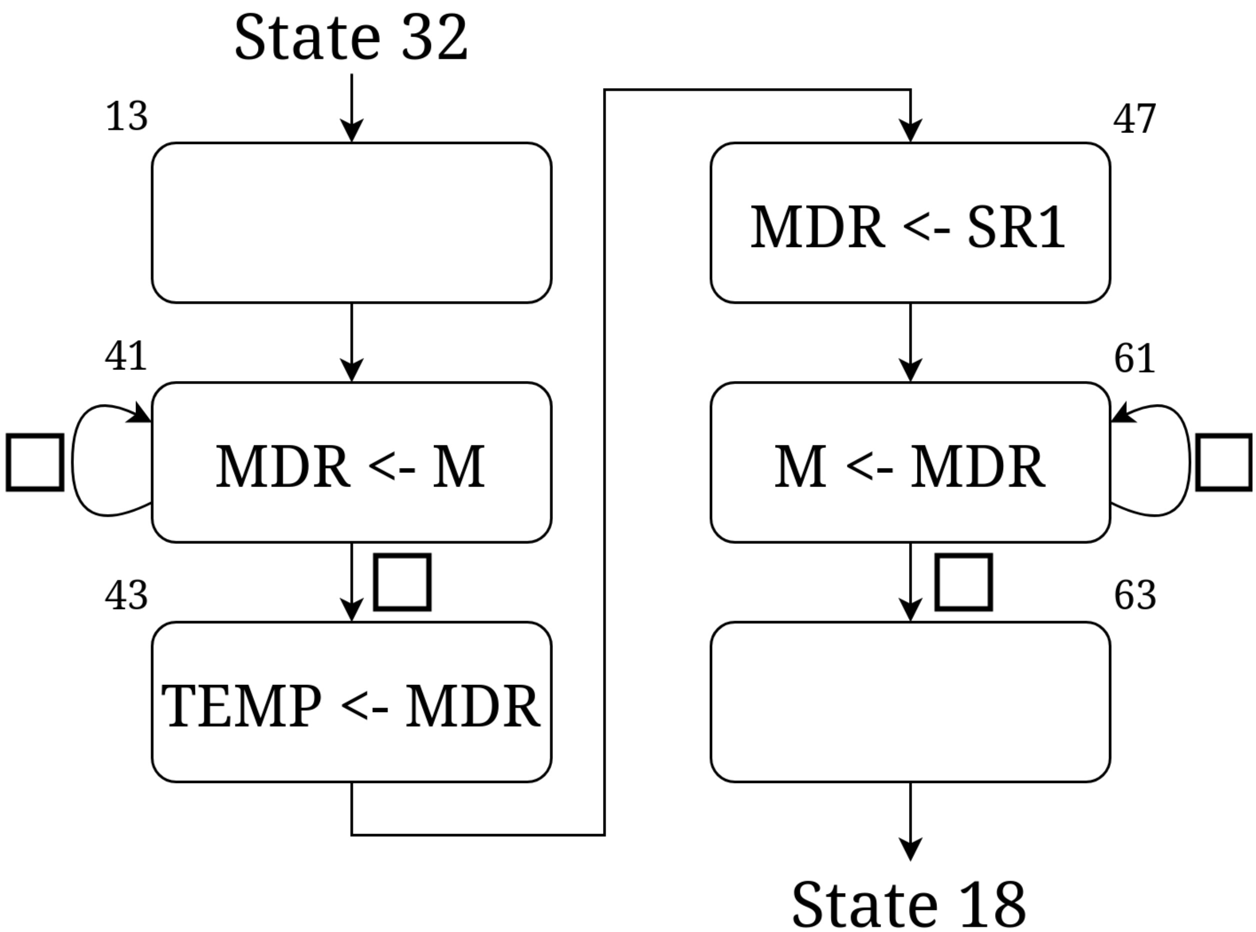
15 12	11 9	8 6	5 0
1 1 0 1	S R 1	S R 2	0 0 0 0 0 0

The changes to the datapath are bolded in the diagram below.



Name: ___________________________________

Part A: Complete the state machine for the swap instruction below.



Part B: Fill out the control signals for the states listed in the table below:

State	J	GATE.MDR	GATE.TEMP	LD.TEMP	LD.MDR	LD.MAR
41						
43						

Name: ______________________________________

Question 3 (10 Points): We want to design the state machine for an automatic bird feeder. It has two types of feed: millets and sunflower seeds. The system takes 1 bit of input, and outputs 2 bits. The input and output are described below.

Sensor Readings (inputs)

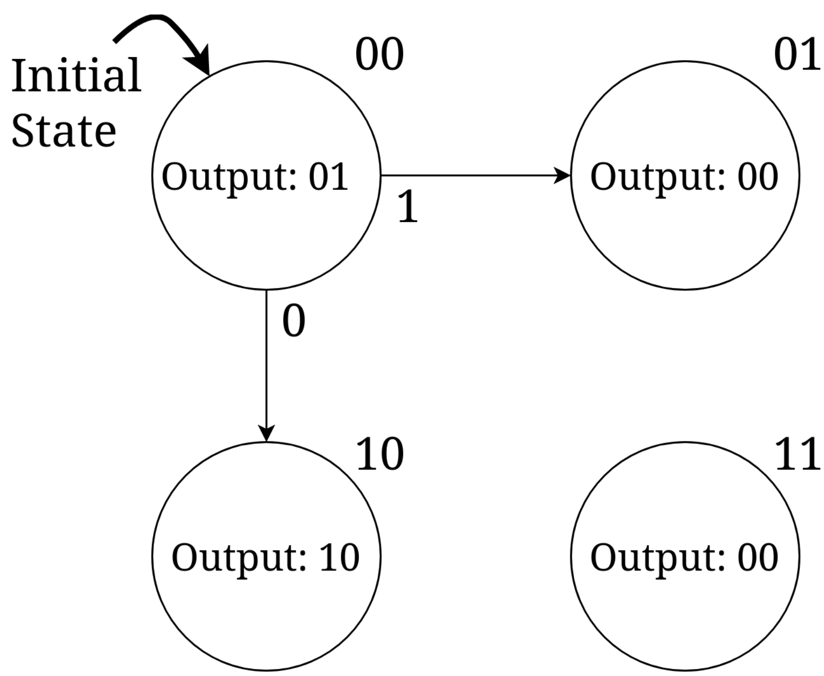
Input	Description
0	Empty
1	Not Empty

Bird Feeder Actions (outputs)

Out[1]	Out[0]	Description
0	0	Do nothing
0	1	Fill with millets
1	0	Fill with sunflower seeds

Birds need a varied diet so our bird feeder alternates between sunflower seeds and millet. **Every hour** the feeder will read the sensor. If it is empty, it will fill with the opposite food choice. If it is not empty, it will do nothing. The feeder starts empty, and should be filled with millet.

Part A: Complete the finite state machine below that allows the bird feeder to operate as described. The 2-bit number on the top-right corner of each state is the state number for that state.



Name: ___________________________________

Part B: Complete the truth table corresponding to the state machine from part A.

State[1]	State[0]	Input	Out[1]	Out[0]	Next State[1]	Next State[0]
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Part C: Draw the logic gate implementation for Out[1]. Use only AND, OR, and NOT gates.

Name: ___________________________________

Question 4 (10 Points): What value is stored in the location labeled RESULT when the following program terminates?

```
            .ORIG x3000
            AND   R1,   R1,   #0
            LEA   R0,   LABEL1
LOOP        LDR   R0,   R0,   #0
            BRz   DONE
            LDR   R2,   R0,   #1
            ADD   R1,   R1,   R2
            BRnzp LOOP
DONE        ST    R1,   RESULT
            HALT
RESULT      .BLKW  #1
LABEL1      .FILL x6000
            .END

            .ORIG       x6000
            .FILL       x5010
            .FILL       x0300
            .END
            .ORIG       x5010
            .FILL       x4020
            .FILL       xFFFF
            .END
            .ORIG       x4020
            .FILL       x0000
            .FILL       x0007
            .END
```

What kind of data structure does this program operate on?

Name: ___________________________________

Question 5 (10 Points): A student wrote a program to multiply two positive numbers X and Y, and store the product in memory location x4002. M[x4000] contains the value X. M[x4001] contains the value Y. Unfortunately she spends too much time watching football, so there is a bug in her code.

```
           .ORIG x3000
           LDI   R1,   X
           LDI   R2,   Y
           AND   R0,   R0,   #0
LOOP       ADD   R0,   R0,   R2
           ADD   R1,   R1,   #-1
           BRz   LOOP
           STI   R0,   RESULT
           HALT
X          .FILL x4000
Y          .FILL x4001
RESULT     .FILL x4002
           .END
```

Your job: Circle the instruction causing the bug, and replace it in the box below with the correct instruction.

Name: ____________________________________

Question 6 (20 Points): You are given a program that takes 2 inputs from memory and stores the result of execution in an unknown memory location. During the execution of the program, we took 7 snapshots of the LC-3 microarchitecture. The first snapshot was taken during the first clock cycle of the program execution. Subsequent snapshots are shown in the table below in the order in which they are taken. Each snapshot contains the state number, the value on the bus during the clock cycle, and the control signals necessary to execute the state.

Part A: Fill in the blanks in the code and table. Use X if a control signal value doesn't matter.

```
            .ORIG x3000
            LD    R0,   OUTPUT
            LDR   R1,   R0    #16   ;INPUT 1
            BRz   DONE
            LDR   R2,   R0,   #18   ;INPUT 2
LOOP        JSR   SUBTRACT
            ADD   R1,   R0,   #0
            BRzp  LOOP
            ADD   R1,   R0,   R2
DONE        STI   R1,   OUTPUT
            HALT
OUTPUT      .FILL ________
SUBTRACT    NOT R0, R2
            ADD R0, R0, #1
            ADD R0, R1, R0
            RET
            .END
```

(The first row of the table has been given to you as an example.)

State	BUS	Control Signals
18	x3000	LD.MAR = 1, GatePC = 1, LD.PC = 1, PCMUX = PC+1
6	x4257	LD.MAR = 1, ALUK = _____________, SR1MUX =_____________
6	x4259	LD.MAR = 1, ALUK = _____________, SR1MUX =_____________
9	xFFF5	LD.REG = 1, DR = IR[11:9], ALUK = NOT, SR1MUX = IR[8:6]
21	______	LD.REG = 1, DRMUX = _________, ADDR2MUX = ____________
16	-	LD.MDR = ____, MIO.EN = ____, R.W = ____, LD.MAR = _____
_______	-	LD.BEN = ______, J = _____________________, IRD = 1

Part B: Given the above table, and the fact that the RET instruction was executed 7 times, what were the inputs if R1 has value x0000 when the program terminates?

Input 1: ___________ Input 2: ___________

Name: ______________________________________

Question 7 (20 Points): An programmer did a poor job implementing some important features in the operating system. Thankfully, he did manage to implement TRAPs. The table below shows how the memory in the system region is configured at the start of the program. Recall the exception vector for privilege mode exception is x00, for illegal opcode is x01, and for access control violation is x02.

Memory Address	Data	Comments
…		
x0100	x1500	
x0101	x1510	
x0102	x1520	
…		
x1500	x8000	
…		
x1510	x8000	
…		
x1520	xD025	
…		

Table below shows the user program to be executed. The user program starts executing at x3000 with priority 0.

Memory Address	Data	Assembly	Comments
x3000	x5020	`AND R0, R1, #0`	; Start of User Program
x3001	x103F	`ADD R0, R0, #-1`	
x3002	xA003	`C LDI R0, A`	
x3003	x07FE	`BRzp C`	
x3004	xA002	`LDI R0, B`	
x3005	xF025	`TRAP x25`	
x3006	x0E00	`A .FILL x0E00`	
x3007	x0E01	`B .FILL x0E01`	

Name: ___________________________________

Part A: During the first 300 cycles of executing this program, what exception(s) occur if any? Assume that each memory access takes 5 cycles. Also, use the state diagram and the datapath that handles interrupts and exceptions.

Part B: What is in memory locations x2FFC~x2FFF after 300 cycles of execution?

Memory Address	Content
x2FFC	
x2FFD	
x2FFE	
x2FFF	

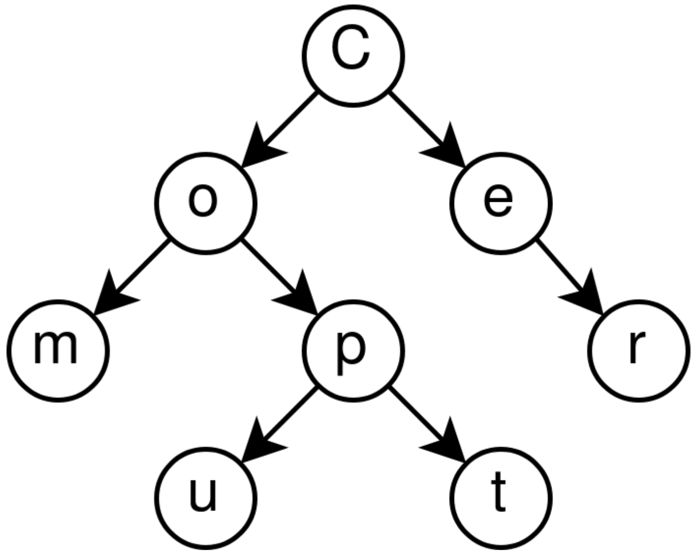
Part C: Suppose the operating system engineering changes the access control violation service routine as follows.

```
            ST    R0,   SAVE_R0
            LDR   R0,   R6,   #0
            ADD   R0,   R0,   #1
            STR   R0,   R6,   #0
            LDR   R0,   R0,   #-1
            ST    R0,   LABEL
            LD    R0,   SAVE_R0
LABEL       .BLKW #1
            RTI
SAVE_R0     .BLKW #1
```

He hopes to fix the exception by executing the faulting instruction in privileged mode. Unfortunately, the code above does not do what he wishes for the user program on the previous page. Why?

Name: ___________________________________

Question 8 (20 Points): A binary tree is a common data structure, used to represent (among other things) family trees and organization charts. A binary tree consists of nodes, connected by links as shown in the example below.



If this tree were an organization chart, C would be the CEO and o and e employees who report directly to him. If the tree were a family tree, C would be the patriarch.

Each link connects a **"parent" node** to one of its possible **"child" nodes**. For example, node o has two children nodes, m and p. Node r has no children. Every node except one has a single parent. The node that does not have a parent is called the "**root**." Each node can have up to two children. The children are the roots of subtrees. For example, the entire tree has a root (node C), a left subtree (consisting of nodes o, m, p, u, t) and a right subtree (consisting of node e and r). Node o is the root of a subtree having a left subtree (consisting of node m) and a right subtree (consisting of nodes p, u, and t).

Unlike a linked list which has only one order in which all the nodes of the link list can be visited, the nodes of a tree can be visited in several different orders. One of the orders is called "pre-order traversal," in which the root is visited first, then all the nodes of its left subtree, then all the nodes of its right subtree. In the tree above the pre-order traversal visits C then o, m, p, u, t, e, and r in that order.

If we identify the data of each node as the ASCII code of a typed character, each node in the tree above occupies three words of LC-3 memory as shown below: a link to the left child, a link to the right child, and the data (ascii code). If the node does not have a left (or right) subtree, the corresponding link is x0000.

Left Child Pointer
Right Child Pointer
ASCII Character

Name: ___________________________________

Below is a program which prints all the characters in the tree in the order described above. **For each node it visits, it keeps track of the right child (if there is one), and then follows the left child. When it visits a node with no left child, the program goes through the right children in reverse order.** Assume the tree is already in memory with the root at address x4000.

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

```
             .ORIG x3000
             LD R6, STACK
             LD R1, ____________
LOOP         ___________________
             TRAP x21               ; OUT
             LDR R0, R1, #1
             BRz SKIP
             ___________________
             ___________________
SKIP         LDR R1, R1, #0
             BR___ LOOP
             LD R2, ____________
             ADD R0, R2, R6
             BRz DONE
             ___________________
             ___________________
             BR___ LOOP
DONE         HALT

TREE_ROOT    .FILL    x4000
DONE_VAL     .FILL    x0200 ;xFE00+x0200=x0000
STACK        .FILL    xFE00
             .END
```

Name: ___________________________________

Question 9 (20 Points): In this question, **we modify how JSR/JSRR and RET works in LC3.** JSR/JSRR will use the stack as the linkage instead of R7 by pushing the return PC onto the stack. Similarly, RET will load the PC with the value popped from the top of the stack.

In memory, there are 10 subroutines to print the 10 digits. For example, the subroutine to print the number "5" consists of the following instructions:

```
Print5    LD R0 Label5
          OUT
          RET
Label5    .FILL x35
```

Since each subroutine requires 4 memory locations, the starting address of each subroutine is always 4 greater than the starting address of the previous subroutine. The starting address of the subroutine Print0 is x5000. Therefore, the starting address of the subroutine Print1 is x5004, the starting address of the subroutine Print2 is x5008, and so on.

A student wishes to surprise Dr. Patt by printing our course number "306" with a program that does not use any JSR/JSRR instructions nor I/O trap service routines in the main program.
Your Job: Fill in the blanks in the main program, so that executing this program will result in printing the digits "306" on the console, and **then halt the machine**.

Hint: What happens when a RET is executed without a JSR being executed beforehand?

```
          .ORIG x3000
          LD   R6,      _______
          LD   R0,      _______
          ADD  ____________________________
          STR  R0,      R6,       #0
          LD   R0,      _______
          ADD  ____________________________
          STR  R0,      R6,       #0
          LD   R0,      _______
          ADD  ____________________________
          STR  R0,      R6,       #0
          LD   R0,      _______
          ADD  ____________________________
          STR  R0,      R6,       #0
          RET
______________________________________________
______________________________________________
______________________________________________
______________________________________________
          HALT
SP_INIT   .FILL    xFE00
          .END
```