Question 1 (10 points) For each of the following, either fill in the correct answer or answer **True** or **False**, as appropriate:

1. What is the word length of the LC-3 ALU? ___16 bits______

2. How many data registers does LC-3 have? ________7_______

3. What is the word length (addressability) of LC-3 memory? ___16 bits______

4. **X** is a Condition Code Register in LC-3 (True/False) ____False______

5. What does ISA stand for? Instruction Set Architecture

6. LC3 is a hybrid CISC/RISC architecture (True/False) ______True______

7. What keeps track of the next instruction to be executed? _______PC________

8. **.END** is an LC-3 instruction to indicate end of program (True/False)______False______

9. LC-3 has two I/O devices: keyboard and display (True/False) ______True______

10. LC-3 opcodes are coded in 5 bits (True/False) ______False______

Question 2 (15 points) Write **short** answers to each of the following.
Part A: In LC-3 what are **MAR** and **MDR**? What is their function? How many bits are there in the **MAR** and **MDR**?

MAR = Memory Address Register
MDR = Memory Data Register

While reading from memory, the address of the location to be read is first placed in the MAR. Then, the data from that memory location appears in the MDR. While writing, the data to be stored is first placed in the MDR. Then the adddress where it is to be stored is placed in the MAR.

Part B: In LC-3 what are the **condition code registers**? How are they used?

N, Z, and P are the three condition codes in LC-3. Every time a data is moved some place (registers or memory), or an operation is performed, the condition codes are set depending on the value.

Part C: In LC-3 what is the purpose of the **Instruction Register**?

Contains the fetched instruction from the memory location pointed to by the PC. This is the instruction to decoded and executed.

Question 3 (10 points)
Part A: The LC-3 has 15 opcodes that define the instructions in its ISA. However, the instruction set charts of LC-3 (see page 2) lists 19 instructions. Explain.

Because instructions like ADD and AND use two different addressing modes. Also, there are two Jump instructions that use the same opcode. These four variations make up 15+4 = 19 instructions.

Part B: For each of the following categories, name **all the LC-3 instructions** that fall in that category (Select from: **ADD**, **AND**, **Branch**, **Jump**, **Load**, **NOT**, **Store**, **HALT**):

Operate Instructions	ADD, AND, NOT
Data Movement Instructions	Load, Store
Control Instructions	Branch, Jump, HALT

Question 4 (20 points) For each of the following, decode each instruction by listing the following: the opcode, all operands, the addressing mode used, and the task it performs using symbolic names (e.g. **Load**, **ADD**, **R3**, **R2**, **#11, R2 = R2 + 3**, etc.)):

A.

0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1

Opcode: __0000 (BR)__________ Operands: ___#-5_______________

Addressing Mode: __PC Relative mode__ Task: _if P then PC = PC - 5________

B.

0	0	0	1	1	1	0	1	1	0	0	0	0	1	1	0

Opcode: ___0001 (ADD)_________ Operands: __R6, R6, R6_________

Addressing Mode: __Register mode______ Task: __R6 = R6 + R6_________

C.

0	0	0	1	1	1	0	1	1	0	1	0	0	1	1	0

Opcode: ___0001 (ADD)________ Operands: __R6, R6, #6_________

Addressing Mode: Register+Immediate mode Task: ___R6 = R6 + 6___________

D.

1	1	0	1	1	0	1	1	0	1	0	0	0	1	1	0

Opcode: ___1101 (Reserved)__ Operands: _____N/A____________

Addressing Mode: ___N/A_______________ Task: _Not an instruction_________

E.

0	0	1	0	1	1	1	0	0	0	0	0	1	0	0	1

Opcode: __0010 (LD)__________ Operands: __R7, #9_____________

Addressing Mode: ___Register+PC Relative_ Task: ___R7 = M[PC+9]__________

Question 5 (15 points) Encode each of the following tasks into **an** equivalent LC-3 **machine language** instruction. Where needed, the address of the current instruction is provided.:

A. `R6 = NOT(R6)`

1001 110 110 111111

B. `x600A    R4 = M[x6000]`

0010 100 111 110 101 ; R4 = M[PC - 11]

C. `R3 = M[R0]`

0110 011 000 000000

D. `x600D  Branch if Positive x6008`

0000 001 11111 1010 ; BR if P to PC-6

E. `R5 = R5 - 3`

`0001  101  101 1 11101`

Question 6 (15 points) Write a sequence of LC-3 assembly language instructions to accomplish the tasks given (use **Question 6 (15 points) Write a sequence of LC-3 assembly language instructions** to accomplish the tasks given (use comments to indicate what each instruction does):

A. **`R7 = R3 - R0`**

```
1001  110  000 111111       NOT    R6, R0              ; R6 = NOT(R0)
0001  110  110 1 00001      ADD    R6, R6, #1          ; R6 = R6 + 1
;  Now R6 is -R0
0001  111  011 000 110      ADD    R7, R3, R6          ; R7 = R3 + R6
```

[Note: Not a good idea to change values in R3 or R0. Why?]

B. **`R7 = R6`**

```
0101  111  111  1  00000    AND    R7, R7, #0          ; First, set R7 =0
0001  111  111  000  110    ADD    R7, R7, R6          ; Add R6 to R7
```

Alternately:

```
0001   111  110 1 00000     ADD    R7, R6, #0          ; R7 = R6 + 0
```

C. **`R7 = R7 * 2`**

There is no multiply instruction. But multiplication by 2 can be achieved by adding the number to itself.

```
0001  111  111  000  111    ADD    R7, R7, R7          ; R7 = R7 + R7
```

D. Swap the contents of **R6** and **R7**.

```
; We will use R5 as temp
0001  101  110  1 00000     ADD    R5, R6, #0          ; R5 = R6 + 0
0001  110  111  1 00000     ADD    R6, R7, #0          ; R6 = R7 + 0
0001  111  101  1 00000     ADD    R7, R5, #0          ; R7 = R5 + 0
```

E. **`R7 = R1 + R2 + R3`**

```
0001  111  001  0 00 010    ADD    R7, R1, R2          ; R7 = R1 + R2
0001  111 111  0 00 011     ADD    R7, R7, R3          ; R7 = R7 + R3
```

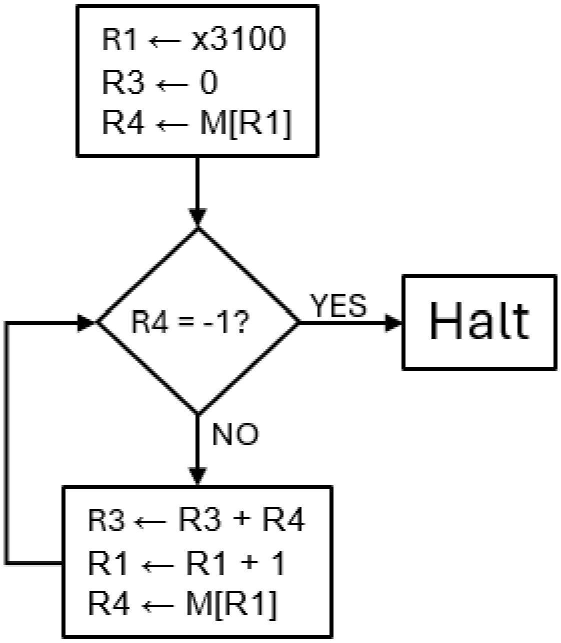
Question 7 (15 points) Write an LC-3 Assembly Language program to add a bunch of integers (quantity unknown). The integers are stored starting from address x3100. A sentinel value of -1 will indicate the end of input. Below, an algorithm, register allocations, and a flow chart for accomplishing the task are provided.

Algorithm:

```
sum ← 0
n ← first number
while n != -1 do
    sum ← sum + n
    n ← next number
```

We will use the following registers:

```
R1: starting address of data (x3100)
R3: sum
R4: n
```



The flowchart is shown on the right. **Your task is to code the flowchart, into a <u>complete LC-3 Assembly Language Program</u>.** The program should be stored starting from x3000. Continue on next page if needed.

```
        .ORIG   x3000
START   LEA     R1, DATA        ; R1 < x3100
        AND     R3, R3, #0  ; R3 <- 0
        LDR     R4, R1, #0  ; R4 <- M[R1]
; while R4 != -1
LOOP    BRn     DONE        ;   YES (R4 = -1)
; do
        ADD     R3, R3, R4  ; R3 <- R3 + R4
        ADD     R1, R1, #1  ; R1 <- R1 + 1
        LDR     R4, R1, #0  ; R4 <- M[R1]
        BR      LOOP
DONE    HALT
        .END
; Data
        .ORIG   x3100
DATA    .FILL   n1
        .FILL   n2
        … … …
        .END
```