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

1. Name of the basic circuit we have learned that is used to store 1-bit of information. Any pulse controlled latch

2. The unique number associated with each word in memory is called by this name. Address

3. The addressability of LC-3 is 2 bytes. (True/False) True

4. Write the size of the address space of LC-3’s memory. 32768

5. The Gated D-Latch is a sequential logic circuit. (True/False) False

6. Finite State Machines are implemented using Combinational Logic Circuits. (True/False) False

7. The Finite State Machine of LC-3 is an asynchronous Finite State Machine False

8. Flip-Flops help to synchronize with the system clock. (True/False) True

9. What is another name for the Program Counter (or PC). Instruction Pointer

10. What is the word length of the LC-3 ALU? 16 bits
Question 2 (15 points) Write short answers to each of the following.

Part A: In LC-3 what are MAR and MDR? What are they used for? 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 address 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 be 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):

<table>
<thead>
<tr>
<th>Operate Instructions</th>
<th>ADD, AND, NOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Movement Instructions</td>
<td>Load, Store</td>
</tr>
<tr>
<td>Control Instructions</td>
<td>Branch, Jump, HALT</td>
</tr>
</tbody>
</table>

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.

\[
\begin{array}{cccccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1
\end{array}
\]

Opcode: 0000 (BR)________ Operands: #5____________
Addressing Mode: PC Relative mode____ Task: if P then PC = PC - 5_____

B.

\[
\begin{array}{cccccccccccccc}
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0
\end{array}
\]

Opcode: 0001 (ADD)_______ Operands: R6, R6, R6_______
Addressing Mode: Register mode____ Task: R6 = R6 + R6_______

C.

\[
\begin{array}{cccccccccccccc}
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0
\end{array}
\]
D. Opcode: \(_{0001} (ADD)\) Operands: \(_{R6, R6, #6} \) Addressing Mode: Register+Immediate mode Task: \(_{R6 = R6 + 6} \)
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. \( R_7 = R_3 - R_0 \)

\[
\begin{align*}
&1001 \ 110 \ 000 \ 111111 \quad \text{NOT} \quad R_6, R_0 ; \ R_6 = \text{NOT}(R_0) \\
&0001 \ 110 \ 110 \ 1 \ 00001 \quad \text{ADD} \quad R_6, R_6, #1 \quad ; \ R_6 = R_6 + 1 \\
&\quad ; \ \text{Now} \ R_6 \ \text{is} \ -R_0 \\
&0001 \ 111 \ 011 \ 000 \ 110 \quad \text{ADD} \quad R_7, R_3, R_6 \quad ; \ R_7 = R_3 + R_6
\end{align*}
\]

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

B. \( R_7 = R_6 \)

\[
\begin{align*}
&0101 \ 111 \ 111 \ 1 \ 00000 \quad \text{AND} \quad R_7, R_7, #0 \quad ; \ \text{First, set} \ R_7 = 0 \\
&0001 \ 111 \ 111 \ 000 \ 110 \quad \text{ADD} \quad R_7, R_7, R_6 \quad ; \ \text{Add} \ R_6 \ \text{to} \ R_7 \\
&\quad \text{Alternately:} \\
&0001 \ 111 \ 110 \ 1 \ 00000 \quad \text{ADD} \quad R_7, R_6, #0 \quad ; \ R_7 = R_6 + 0
\end{align*}
\]

C. \( R_7 = R_7 \times 2 \)

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

\[
\begin{align*}
&0001 \ 111 \ 111 \ 000 \ 111 \quad \text{ADD} \quad R_7, R_7, R_7 \quad ; \ R_7 = R_7 + R_7
\end{align*}
\]

D. Swap the contents of \( R_6 \) and \( R_7 \).

\[
\begin{align*}
&; \ \text{We will use} \ R_5 \ \text{as temp} \\
&0001 \ 101 \ 110 \ 1 \ 00000 \quad \text{ADD} \quad R_5, R_6, #0 \quad ; \ R_5 = R_6 + 0 \\
&0001 \ 110 \ 111 \ 1 \ 00000 \quad \text{ADD} \quad R_6, R_7, #0 \quad ; \ R_6 = R_7 + 0 \\
&0001 \ 111 \ 101 \ 1 \ 00000 \quad \text{ADD} \quad R_7, R_5, #0 \quad ; \ R_7 = R_5 + 0
\end{align*}
\]

E. \( R_7 = R_1 + R_2 + R_3 \)

\[
\begin{align*}
&0001 \ 111 \ 001 \ 0 \ 0010 \quad \text{ADD} \quad R_7, R_1, R_2 \quad ; \ R_7 = R_1 + R_2 \\
&0001 \ 111 \ 111 \ 0 \ 00011 \quad \text{ADD} \quad R_7, R_7, R_3 \quad ; \ R_7 = R_7 + R_3
\end{align*}
\]
**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:**

- \( \text{sum} \leftarrow 0 \)
- \( \text{n} \leftarrow \text{first number} \)
- while \( \text{n} \neq -1 \) do
  - \( \text{sum} \leftarrow \text{sum} + \text{n} \)
  - \( \text{n} \leftarrow \text{next number} \)

We will use the following registers:

- \( \text{R1}: \) starting address of data (x3100)
- \( \text{R3}: \) sum
- \( \text{R4}: \) n

The flowchart is shown on the right.

**Your task is to code the flowchart, into a complete LC-3 Assembly Language Program.** 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
```