T1

The program counter contains the address of an `LDR` instruction. In order for the LC-3 to process that instruction, how many memory accesses must be made? Repeat this task for `STI` and `TRAP`.

T2

We would like to have an instruction that does nothing. Many ISAs actually have an opcode devoted to doing nothing. It is usually called NOP, for NO OPERATION. the instruction still goes through the six phases of the instruction cycle but the execution phase is to do nothing! Please provide a machine instruction using `ADD`, `AND` and `BR` respectively that could be used for `NOP` and have the program still work correctly. You can use the register 1.

T3

The PC contains x3010. The following memory locations contain values as shown:

```
x3050:    x70A4
x70A2:    x70A3
x70A3:    xFFFF
x70A4:    x123B
```

The following three LC-3 instructions are then executed, causing a value to be loaded into R6. What is that value?

```
x3010    1110 0110 0011 1111
x3011    0110 1000 1100 0000
x3012    0110 1101 0000 0000
```

We could replace the three-instruction sequence with a single instruction. What is it?

T4

What is the difference between the following LC-3 instructions A and B? How are they similar? How are they different? A: `0000 111 101010101` B: `0100 1 11101010101`

T5

Please explain the addressing range of PC-relative mode, regarding the current instruction address as the origin.

T6

State the contents of R0, R1, R2 and R3 after the program starting at location x3001 halts.

Adress	Data
x3001	1110 001 111111101
x3002	0001 011 001 100011
x3003	0101 010 001 000011
x3004	0011 010 000000001
x3005	1001 001 001 11111
x3006	1001 011 010 11111
x3007	1111 0000 0010 0101
x3008	
x3009	0000 1101 0011 0001

T7

When solving a complex problem, it is often necessary to break it down into several subtasks, implement them first, and then integrate them in a specific structure to achieve the overall goal. The most common fundamental structures are **sequence, condition, and iteration**.

1. Using the **sequential** construct, write a LC-3 machine language routine that calculates the 4th term of the Fibonacci sequence `F4`. Store the result in `R2`.
2. Using the **iterative** construct, write a LC-3 machine language routine that calculates the 10th term of the Fibonacci sequence `F10`. Store the result in `R2`.

T8

The LC-3 does not have an opcode for the logical function `OR`. That is, there is no instruction in the LC-3 ISA that performs the `OR` operation. However, we can write a sequence of instructions to implement the `OR` operation. The following four-instruction sequence performs the `OR` of the contents of register 1 and register 2 and puts the result in register 3. Fill in the two missing instructions so that the four-instruction sequence will do the job.

(1): `1001 100 001 111111`

(2):

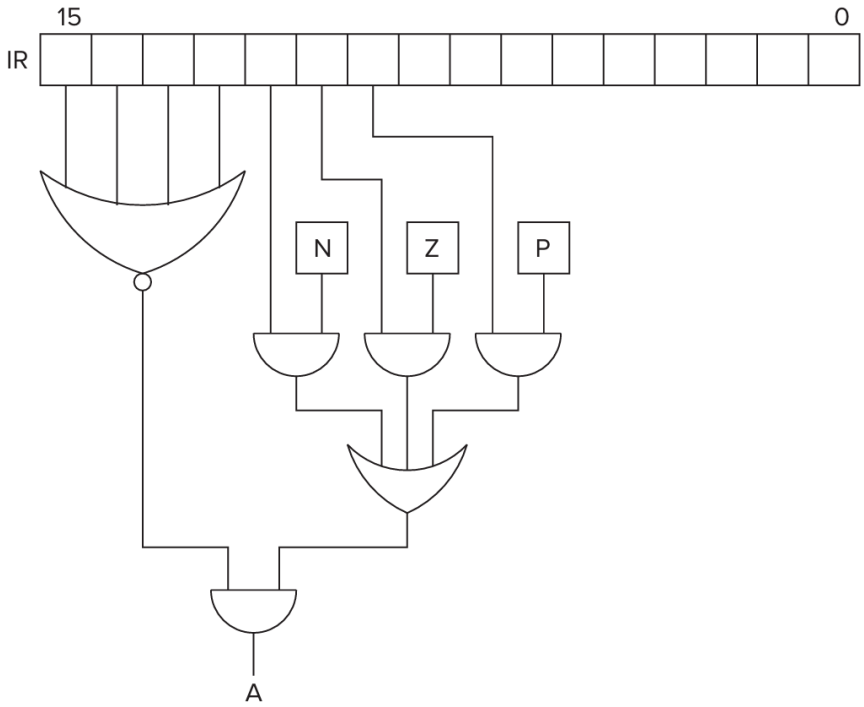
(3): `0101 110 100 000 101`

(4):

T9

The following logic diagram shows part of the control structure of the LC-3 machine. What is the purpose of the signal labeled A?

5.40 The following logic diagram shows part of the control structure of the LC-3 machine. What is the purpose of the signal labeled A?



T10

The code below is an program to multiply two positive integers in `R1` and `R2` respectively. The result is written into `R0`.

Adress	Data	Operation
x3000	0101 0000 0010 0000	`AND R0 <- R0, #0`
x3001	0001 0010 0111 1111	`ADD R1 <- R1, #-1`
x3002	0000 1000 0000 0010	`BRn x3005`
x3003	0001 0000 1000 0000	`ADD R0 <- R2, R0`
x3004	0000 1111 1111 1100	`BRnp x3001`
x3005	1111 0000 0010 0101	`HALT`

What results will be stored in `R0` if we replace the instruction in x3003 with **`ADD R0 <- R0, R1`**? Use `R1` or `R2` to represent your answer.