Report for LAB-1

1 Algorithm: How to solve the problem?

1.1 Basic: Does the number end with `111` ?

1. **Extract the last 3 bits.**

   We can do this with mask `0000 0000 0000 0111`.
   By applying a bit-wise `AND` operation, we focus on the last 3 bits (and ignore the rest part).

2. **Are these 3 bits all `1` s?**
   The simplest way is to judge whether `(num & mask) == mask`, but:

   - There is no `EQ` insctruction in LC-3.
   - There would be a BR instruction right after the comparison, whose action depends on *Condition Code* , we should modify the value of one of thoese registers.

   We actually judge whether `(num & mask) - mask == 0` here.

1.2 Iterate all posible cases: what about `111` in the middle?

We can handle this by using mask `0000 0000 0000 1110`, `0000 0000 0001 1100` , ... , `1110 0000 0000 0000` .

The list of mask could be obtained by left-shift, which could also achieve by double (`mask << 1 == 2 * mask`) .

By using loop structure, we can simply iterate all posible cases.

1.3 Solution (in C)

```c
int Judge() {
  int data = Memory[x3100];
  int iter = 14;
  int mask = x0007;
  while (iter--, iter != 0) {
    int tmp = data & mask;
    if (tmp == mask) return 1; // True
    mask = mask << 1;
  }
  return 0;
}
```

2 Code (with comments)

```
0011 0000 0000 0000   ; .ORIG x3000                      the program start at x3000
; Initializations
0010 100  011111111   ;  LD  R4, x00ff                   load input -> R4 = Memory[x3100]
0101 001  001 1 00000 ;  AND R1, R1, #0
0001 001  001 1 01111 ;  ADD R1, R1, #15                 iteration counter -> R1 = 15
0101 010  010 1 00000 ;  AND R2, R2, #0                  result -> R2 = 0
0101 011  011 1 00000 ;  AND R3, R3, #0
0001 011  011 1 00111 ;  ADD R3, R3, #7                  mask -> R3 = x0007
; Loop Begin
0001 001  001 1 11111 ;  ADD R1, R1, #-1 (Label Loop)    loop condition -> R1--
0000 010  000001000;     BRz +9 lines    (Go Fin)        if (R1==0) go Fin
0101 101  100 0 00 011;  AND R5, R4, R3                  R5 = input & mask
1001 110  011 111111  ;  NOT R6, R3
0001 110  110 1 00001 ;  ADD R6, R6, #1                  R6 = -mask
0001 101  101 0 00 110;  ADD R5, R5, R6                  R5 = (input & mask) + (-mask)
0000 010  000000010;     BRz +3 lines   (Go Success)     if (R5 == 0) go Success
0001 011  011 0 00 011;  ADD R3, R3, R3                  mask << 1
0000 111  111110111;     BRnzp -9 lines (Go Loop)
; Success
0001 010  010 1 00001 ;  ADD R2, R2, #1 (Label Success)  update return value -> R2 = 1
; Halt(the end)
1111 0000 00100101;      HALT           (Label Fin)
```

3 Q & A

1. How to solve the problem?
   Shown in section 1 & 2.
2. Which part should we modify if we want to detect four consequtive 1's?
   - Mask: `x0007` -> `x000F`
     - Iteration Counter: `15` -> `14`