Report of lab 1

Algorithm explanation

- In order to check if the number fits the condition, I set a basic mask at `R1`. The mask has only four continuous `1`s which **should be found** as a substring of the input if the input is a *F-word*
- And I will program a loop to **shift it left** step by step (from `0000 0000 0000 1111` to `1111 0000 0000 0000`), that will iterate all the smallest possible mask .
- That is, if one of those possible masks satisfies { (`input` `AND` `mask`) equal to (`mask`) }, then we can tell that the number is a *F-word*. Conversely, if none of the 13 masks is suitable, then we can tell that the number is not a *F-word*.

Essential parts of your code with sufficient comments

```
0011 0000 0000 0000   ; set program at x3000
; ---------------------------------------------------------------------------- ;
; Read a word at x3100.                                                        ;
; Check if it contains 4 continuous 1, and put the answer to R2 (1 or 0).      ;
;                                                                              ;
; Special define of register:                                                  ;
; R0: store input                                                              ;
; R1: mask                                                                     ;
; R2: output and zero                                                          ;
; R3: result of <(mask AND input) - mask>                                      ;
; R4:                                                                          ;
; R5: -1                                                                       ;
; R6: -mask                                                                    ;
; R7: rest cnt of loop (start from 13)                                         ;
; ---------------------------------------------------------------------------- ;

; Init:
0010 000 011111111      ; LD:   R0 <- mem[PC+x00ff] (expect x3100) ; Get input.
0101 010 010 1 00000    ; AND:  R2 <- 0000B ; Set zero.
0001 101 010 1 11111    ; ADD:  R7 <- NOT R7 ; Set -1.
0001 111 010 1 01101    ; ADD:  R7 <- R2 + 1101B ; Set loop cnt = 13. (16-4+1=13)
0001 001 010 1 01111    ; ADD:  R1 <- R2 + 1111B ; Set first mask.

; Begin loop:

; Calculate -mask:
1001 110 001 111111     ; NOT:  R6 <- NOT R1 ;          Calculating -mask.
0001 110 110 1 00001    ; ADD:  R6 <= R6 + 0001B ;      Calculate -mask.
; Caculate diff between mask and result:
0101 011 000 0 00 001   ; AND:  R3 <- R0 & R1 ;                 Calculating diff
between mask and result.
0001 011 011 0 00 110   ; ADD:  R3 <- R3 + R6 = R3 -mask ;      Calculate diff
between mask and result.
; Checking if it fits mask.
```

```
0000 1 0 1 000000010    ; BR:   If result isn't zero, [+2lines]exit with R2 = 1,
or continue to loop.
; [false] is zero:
0001 010 010 1 00001    ; ADD:  R2 <- 0001B ; Set output true.
1111 0000 00100101      ; halt: ; Exit.
; [true] isn't zero:
0001 001 001 0 00 001   ; ADD:  R1 <- R1 + R1 ; Double R1. (i.e., shift left
mask)
0001 111 111 0 00 101   ; ADD:  R7 <- R7 + R5 ; Note that one loop has finished.
; Check if the loop should end.
0000 1 0 1 111110110    ; BR:   If result isn't zero, [-8 lines]continue to loop,
or exit with R2 = 0(R2 is zero already).

1111 0000 00100101      ; halt
```

Questions TA asked you and your answer in Check

Q: explain the algorithm.

A: see subtitle above.