Lab1-Report

1 Algorithm

bitwise left shift: left shift 1 bit is equal to doubling,so use **ADD** to operate.

To be concise, assuming that the 16-bit value is stored in R0, and use `<<` to represent bitwise left shift.

if R0 has the pattern '111',

then `R0 AND (R0 << 1)` will have '11' pattern, that is not equal to 0, *otherwise we can judge that R0 doesn't have '11', let alone '111', so jump to the end.*

and `(R0 << 1) and (R0 << 2)` will overlap at least one '1', that is not equal to 0 too.

We can use some registers to store the itermediate viariables.

At last, if we have found '111', reset R2 to 1 to satisfy lab's demand.

2 Codes & Comments

```
0011 0000 0000 0000   ; .ORIG x3000

0010 000 0 1111 1111  ; R0 <- M[x3100]
0001 001 000 000 000  ; R1 <- R0 << 1

;; if R0 has '11' pattern, R2 = NOT 0, otherwise R2 = 0
0101 010 000 000 001  ; R2 <- R0 & R1
0000 010 0000 00101   ; BRZ, HALT

0001 001 010 000 010  ; R1 <- R2 << 1
;; if R1 has '11' pattern (when R0 has '111' pattern), R2 = NOT 0, otherwise R2 = 0
0101 010 001 000 010  ; R2 <- R1 & R2
0000 010 0000 00010   ; BRZ, HALT
;; if R2 is NOT 0, reset R2 to 1
0101 010 010 1 00000  ; R2 <- 0
0001 010 010 1 00001  ; R2 <- R2 + 1
1111 0000 0010 0101   ; HALT
```

3 TA's Questions

3.1 Your Algorithm

The same as part 1

3.2 Why use BR instructions twice?

It's not a good idea to use BR twice, which increases the length of codes.And the fisrt BR is of no favour to the program actually.