The 1st lab report

(Due on Jul.10th)

1. Program function:

Detect whether a 16-bit value has at least three consecutives '1'.

2.Program algorithm (expressed in C):

```
#include <stdio.h>
int main()
{
                                    //initialization
    int R2 = M[x3100];
    int R1 = 3;
    int R3 = 1;                     //0000 0000 0000 0001
    int R7;
    int R6,R4;
    while(R3 != 0)                  //When R3 overflows, stop the loop
    {
        R6 = R3 & R2;               //if corresponding bit is 1, R6 = 1, otherwise R6 = 0
        R7 = ~R3 + 1;
        R4 = R6 + R7;               //the same as R6 - R3, if R4 != 0,which means R6 == 0
        R1--;                       //finding 1, R1-- and won't be reset
        if(R1 == 0) break;          //finding 3 consecutive 1, then jump out of the circulation
        if (R4 != 0)
            R1 = 3;                 //if finding 0, which is a breaking point, then reset R1
        R3 << 1;                    //left-moving one bit
    }
    if(R3 != 0) printf("Y");        //break halfway means find 3 consecutive 1
    else printf("N");
    return 0;
}
```

3.Brief Explanations

In this program, R2 stores the value loaded from x3100. R1 serves as a counter. If a 1-bit is found, R1 will subtract 1; if a 0-bit is found, which is a "breaking point", R1 will be reset to 3 and restart the searching and counting.

Once R1 equals 0, which means we have found 3 consecutive 1-bits, then the program will jump out of the circulation and prints "Y". if R1 doesn't equal 0 and we haven't had all bits examined yet, the circulation will continue.

Register R3, which is initially set to 1(0000000000000001), serves as a "mask" to examine every bit of R2. Every bit of R3 will move left for 1 bit to examine the next bit of R2 before the next circulation begins. If R3 finally overflows (1000 0000 0000 0000

* 2 == 0000 0000 0000 0000) but R1 still doesn't equal 0 at that time, there can't be 3 consecutive 1-bits. Then the circulation will be at an end and the program prints "N".

4.Source Code

```
0011 0000 0000 0000;      (ORIG x3000)
0010 010 011111111;       (LD R2, x3100)
0101 001 001 1 00000;    (AND R1, R1, #0)
0001 001 001 1 00011;    (ADD R1, R1, #3)
0101 011 011 1 00000;    (AND R3, R3, #0)
0001 011 011 1 00001;    (ADD R3, R3, #1)
0101 110 011 0 00 010; (AND R6, R3, R2)
1001 111 011 111111;      (NOT R7, R3)
0001 111 111 1 00001;    (ADD R7, R7, #1)
0001 100 110 0 00 111; (ADD R4, R6, R7)
0000 101 000000011;       (BRNP x300D)
0001 001 001 1 11111;    (ADD R1, R1, #-1)
0000 010 000000110;       (BRZ x3012)
0000 111 000000010;       (BRNZP x300F)
0101 001 001 1 00000;    (AND R1, R1, #0)
0001 001 001 1 00011;    (ADD R1, R1, #3)
0001 011 011 0 00 011; (ADD R3, R3, R3)
0000 010 000000100;       (BRZ x3015)
0000 111 111110011;       (BRMZP x300F)

0010 000 000000101;       (LD RO, x3018)
1111 0000 00100001;       (TRAP OUT)
0000 111 000000010;       (BRMZP x3017)
0010 000 000000011;       (LD R0, x3019)
1111 0000 00100001;       (TRAP OUT)
1111 0000 0010 0101;      (TRAP HALT)
0000 0000 0101 1001;      (STORE 'Y' 'S ASCII)
0000 0000 0100 1110;      (STORE 'N''S ASCII)
```