Assembly Homework 2 #Procedure

Department: CSIE 2-B

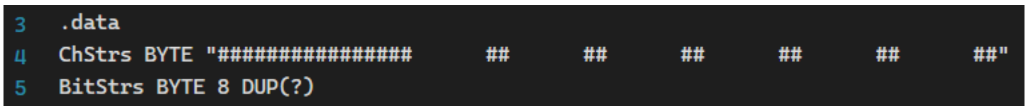
Student Number: 110502567

Name: 蔡淵丞 Vincent

DATA SECTION:

ChStrs: the pattern ‘7’ in 8x8 grid

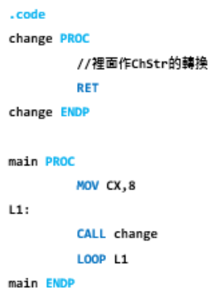
BitStrs: the 1X8 line we want to print to the console each time



```
.data
ChStrs BYTE "################      ##      ##      ##      ##      ##      ##"
BitStrs BYTE 8 DUP(?)
```

THOUGHT:

Given the recommended structure of the code, it is easy to see that we treat ‘change PROC’ as a function which we want to call 8 time, for each we print a 1X8 line of our 8X8 pattern to the screen. Therefore, we just have to consider what we are going to perform to each line of the pattern.



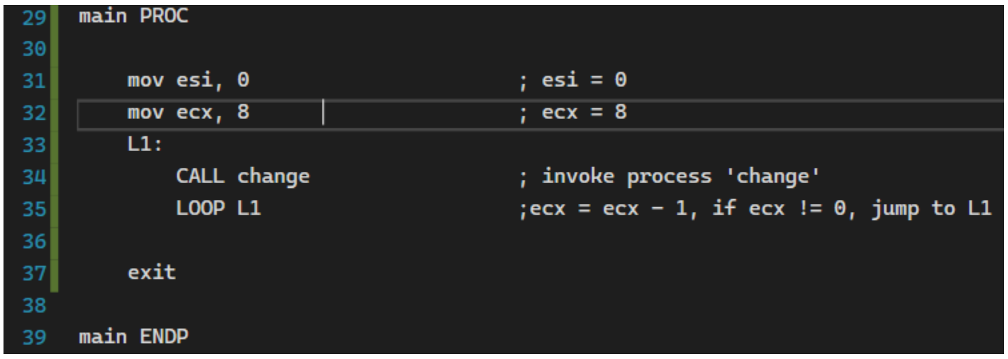
```
.code
change PROC
        //裡面作ChStr的轉換
        RET
change ENDP

main PROC
        MOV CX,8
L1:
        CALL change
        LOOP L1
main ENDP
```

MAIN PROC:

Esi: to store the current index of ChrStrs

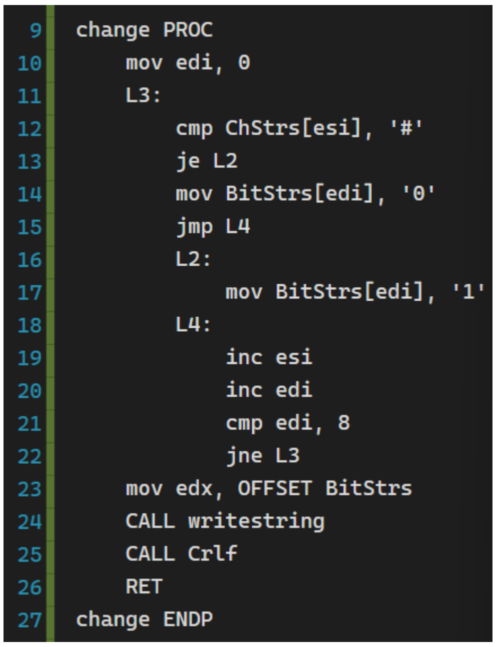
Ecx: loop count

L1: call ‘change’ 8 times



```
main PROC

    mov esi, 0                    ; esi = 0
    mov ecx, 8                    ; ecx = 8
    L1:
        CALL change               ; invoke process 'change'
        LOOP L1                   ;ecx = ecx - 1, if ecx != 0, jump to L1

    exit

main ENDP
```

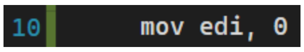
CHANGE PROC:



```
 9  change PROC
10      mov edi, 0
11      L3:
12          cmp ChStrs[esi], '#'
13          je L2
14          mov BitStrs[edi], '0'
15          jmp L4
16          L2:
17              mov BitStrs[edi], '1'
18          L4:
19              inc esi
20              inc edi
21              cmp edi, 8
22              jne L3
23      mov edx, OFFSET BitStrs
24      CALL writestring
25      CALL Crlf
26      RET
27  change ENDP
```

Line 10:

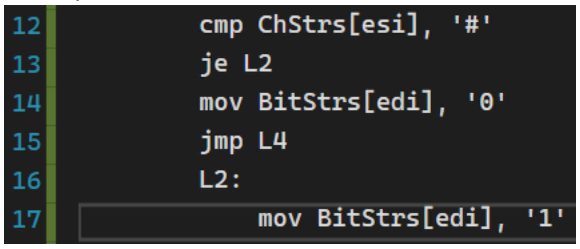
I use ***edi*** as loop count of L3 and the current index of BitStrs.



```
10      mov edi, 0
```

Line 12-17:

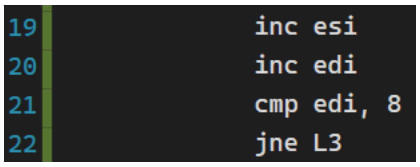
Compare character in ChStrs. If it is ‘#’ then store ‘1’ in BitStrs, else store’0’.



```
12          cmp ChStrs[esi], '#'
13          je L2
14          mov BitStrs[edi], '0'
15          jmp L4
16          L2:
17              mov BitStrs[edi], '1'
```

Line 19-22:

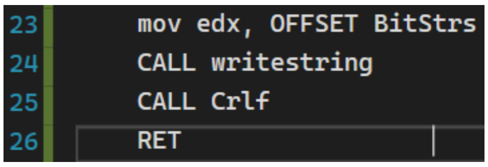
Increase ***esi, edi***. If ***edi*** equals 8, the line is complete, finish the loop.



```
19              inc esi
20              inc edi
21              cmp edi, 8
22              jne L3
```

Line 23-26:

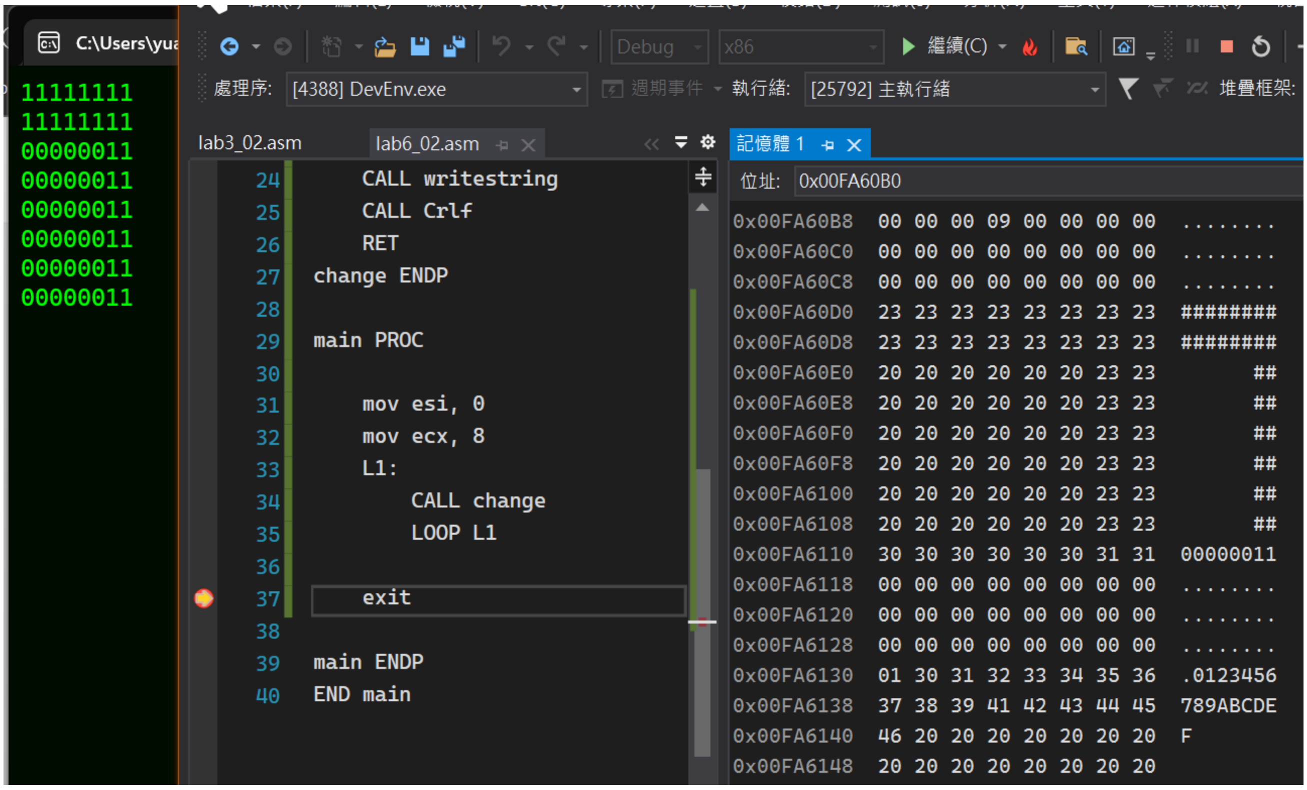
Write the line (BitStrs) to the console and return.



```
23      mov edx, OFFSET BitStrs
24      CALL writestring
25      CALL Crlf
26      RET
```

Before returning out of ‘change’ we will have printed a 1X8 line of our pattern to the console.

RESULT:



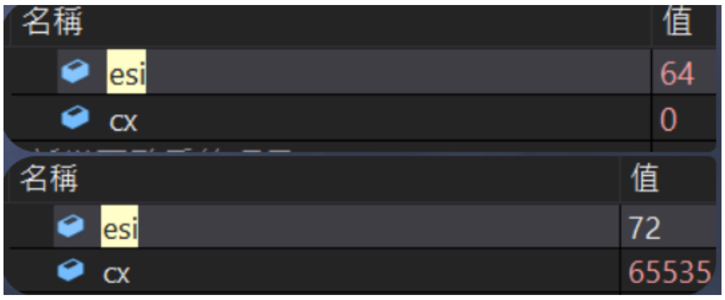
```
11111111
11111111
00000011
00000011
00000011
00000011
00000011
00000011
```

```
        CALL writestring
        CALL Crlf
        RET
change ENDP

main PROC

        mov esi, 0
        mov ecx, 8
        L1:
            CALL change
            LOOP L1

        exit

main ENDP
END main
```

```
0x00FA60B8  00 00 00 09 00 00 00 00  ........
0x00FA60C0  00 00 00 00 00 00 00 00  ........
0x00FA60C8  00 00 00 00 00 00 00 00  ........
0x00FA60D0  23 23 23 23 23 23 23 23  ########
0x00FA60D8  23 23 23 23 23 23 23 23  ########
0x00FA60E0  20 20 20 20 20 20 23 23        ##
0x00FA60E8  20 20 20 20 20 20 23 23        ##
0x00FA60F0  20 20 20 20 20 20 23 23        ##
0x00FA60F8  20 20 20 20 20 20 23 23        ##
0x00FA6100  20 20 20 20 20 20 23 23        ##
0x00FA6108  20 20 20 20 20 20 23 23        ##
0x00FA6110  30 30 30 30 30 30 31 31  00000011
0x00FA6118  00 00 00 00 00 00 00 00  ........
0x00FA6120  00 00 00 00 00 00 00 00  ........
0x00FA6128  00 00 00 00 00 00 00 00  ........
0x00FA6130  01 30 31 32 33 34 35 36  .0123456
0x00FA6138  37 38 39 41 42 43 44 45  789ABCDE
0x00FA6140  46 20 20 20 20 20 20 20  F
0x00FA6148  20 20 20 20 20 20 20 20
```

REVIEW:

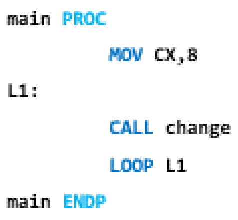
I encountered a bug during my homework. The loop L1 doesn't finish when the ***cx*** counts down to 0, and then my friend told me to make sure the entire ***ecx*** is 0, not until which I realized that I didn't move 0 to the entire register ***ecx***.



名稱	值
esi	64
cx	0

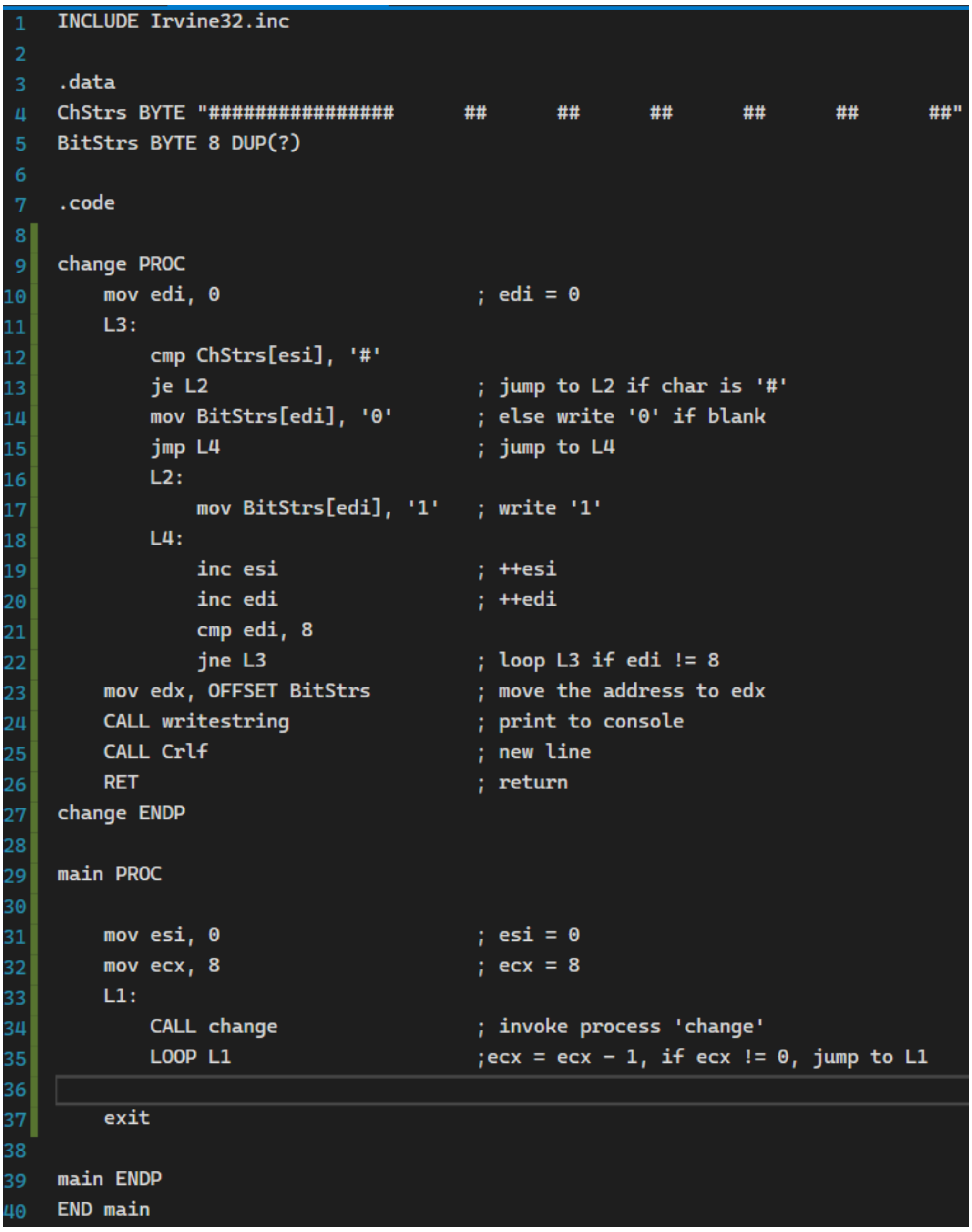
名稱	值
esi	72
cx	65535

the code beside is misleading



```
main PROC
        MOV CX,8
L1:
        CALL change
        LOOP L1
main ENDP
```

FULL CODE:



```
INCLUDE Irvine32.inc

.data
ChStrs BYTE "################      ##      ##      ##      ##      ##      ##"
BitStrs BYTE 8 DUP(?)

.code

change PROC
    mov edi, 0                      ; edi = 0
    L3:
        cmp ChStrs[esi], '#'
        je L2                       ; jump to L2 if char is '#'
        mov BitStrs[edi], '0'       ; else write '0' if blank
        jmp L4                      ; jump to L4
        L2:
            mov BitStrs[edi], '1'   ; write '1'
        L4:
            inc esi                 ; ++esi
            inc edi                 ; ++edi
            cmp edi, 8
            jne L3                  ; loop L3 if edi != 8
    mov edx, OFFSET BitStrs         ; move the address to edx
    CALL writestring                ; print to console
    CALL Crlf                       ; new line
    RET                             ; return
change ENDP

main PROC

    mov esi, 0                      ; esi = 0
    mov ecx, 8                      ; ecx = 8
    L1:
        CALL change                 ; invoke process 'change'
        LOOP L1                     ;ecx = ecx - 1, if ecx != 0, jump to L1

    exit

main ENDP
END main
```