(Matrix I/O)

(Curtis Robinson, Khondakar Mujtaba)

(ELEN 120)

(Wednesday 2:15)

Objective: Our goal throughout this lab project is to program the segmented LCD display on the DISCO board. We must know the register specifications in the STM32L4x6 advanced ARM®-based 32-bit MCUs reference manual and pre-supplied code discussed in the lab below. With the tools provided our goal is to display the text phrase "SANTA" wait one second "CLARA" wait one second and repeat indefinitely.

Procedure:

Problem 1: In the subroutine locations below we are going to utilize the let2font file to convert the chosen ASCII letter characters to their corresponding font representation.
After that, we are supposed to return the font to the output positions with the implementation of the stack to then display the correct wording on the LCD display.

Problem 2:

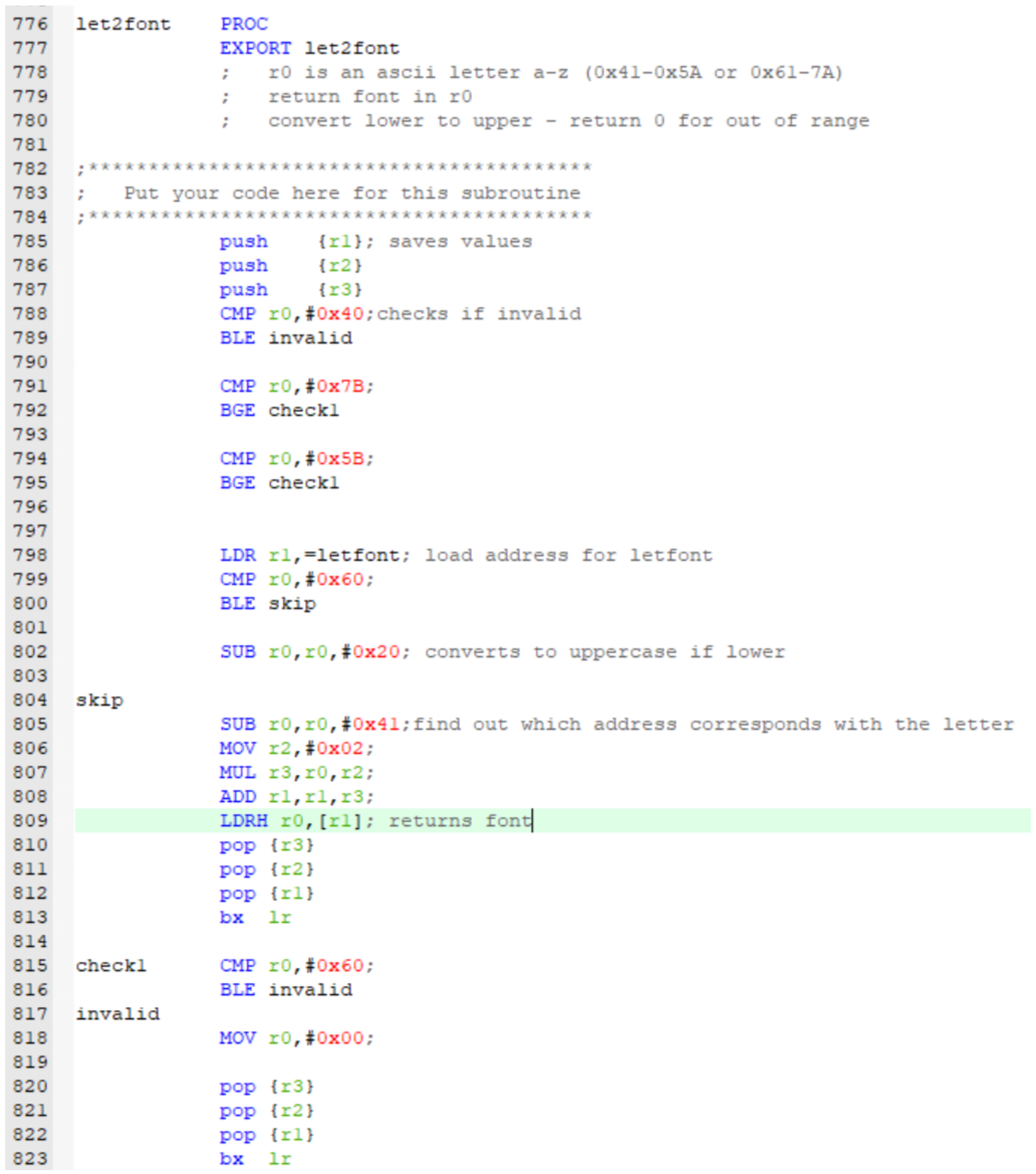
For this problem, we are supposed to carry out the same additions to the file to allow the font conversion and displaying of the numbers 1234 and 5678 on their respective positions on the display.

Problem 3:

For this problem, we add the functionality of a keypad into the program. We must determine if one of the 16 keypad buttons is pressed and display that respective input on the LCD screen. Below we implement a kpad_scan function is kpad.s that grabs the information from the input registers.

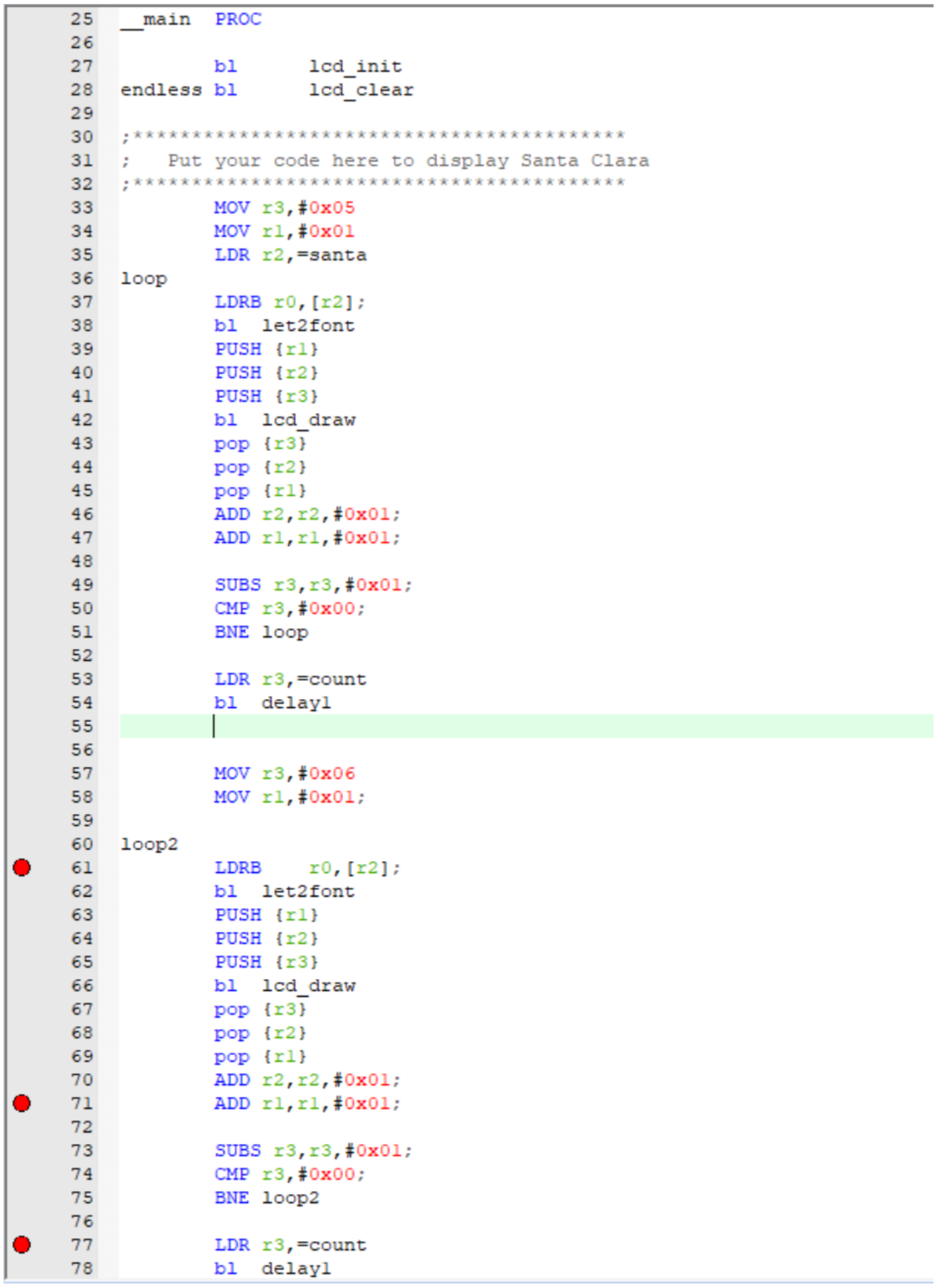
Problem 1:

let2font:

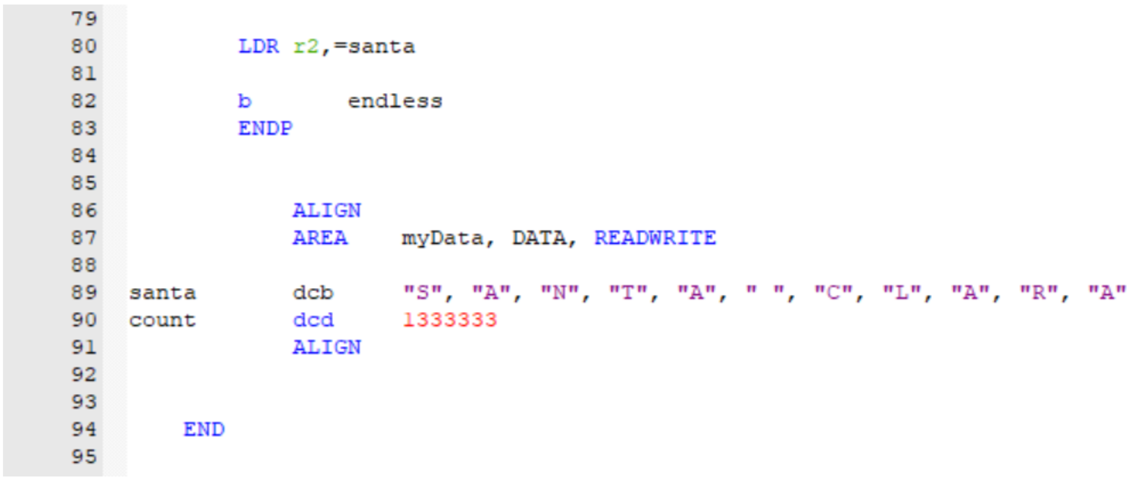


```
let2font    PROC
            EXPORT let2font
            ;   r0 is an ascii letter a-z (0x41-0x5A or 0x61-7A)
            ;   return font in r0
            ;   convert lower to upper - return 0 for out of range

;*******************************************
;   Put your code here for this subroutine
;*******************************************
            push    {r1}; saves values
            push    {r2}
            push    {r3}
            CMP r0,#0x40;checks if invalid
            BLE invalid

            CMP r0,#0x7B;
            BGE checkl

            CMP r0,#0x5B;
            BGE checkl


            LDR r1,=letfont; load address for letfont
            CMP r0,#0x60;
            BLE skip

            SUB r0,r0,#0x20; converts to uppercase if lower

skip
            SUB r0,r0,#0x41;find out which address corresponds with the letter
            MOV r2,#0x02;
            MUL r3,r0,r2;
            ADD r1,r1,r3;
            LDRH r0,[r1]; returns font
            pop {r3}
            pop {r2}
            pop {r1}
            bx  lr

checkl      CMP r0,#0x60;
            BLE invalid
invalid
            MOV r0,#0x00;

            pop {r3}
            pop {r2}
            pop {r1}
            bx  lr
```

main.s



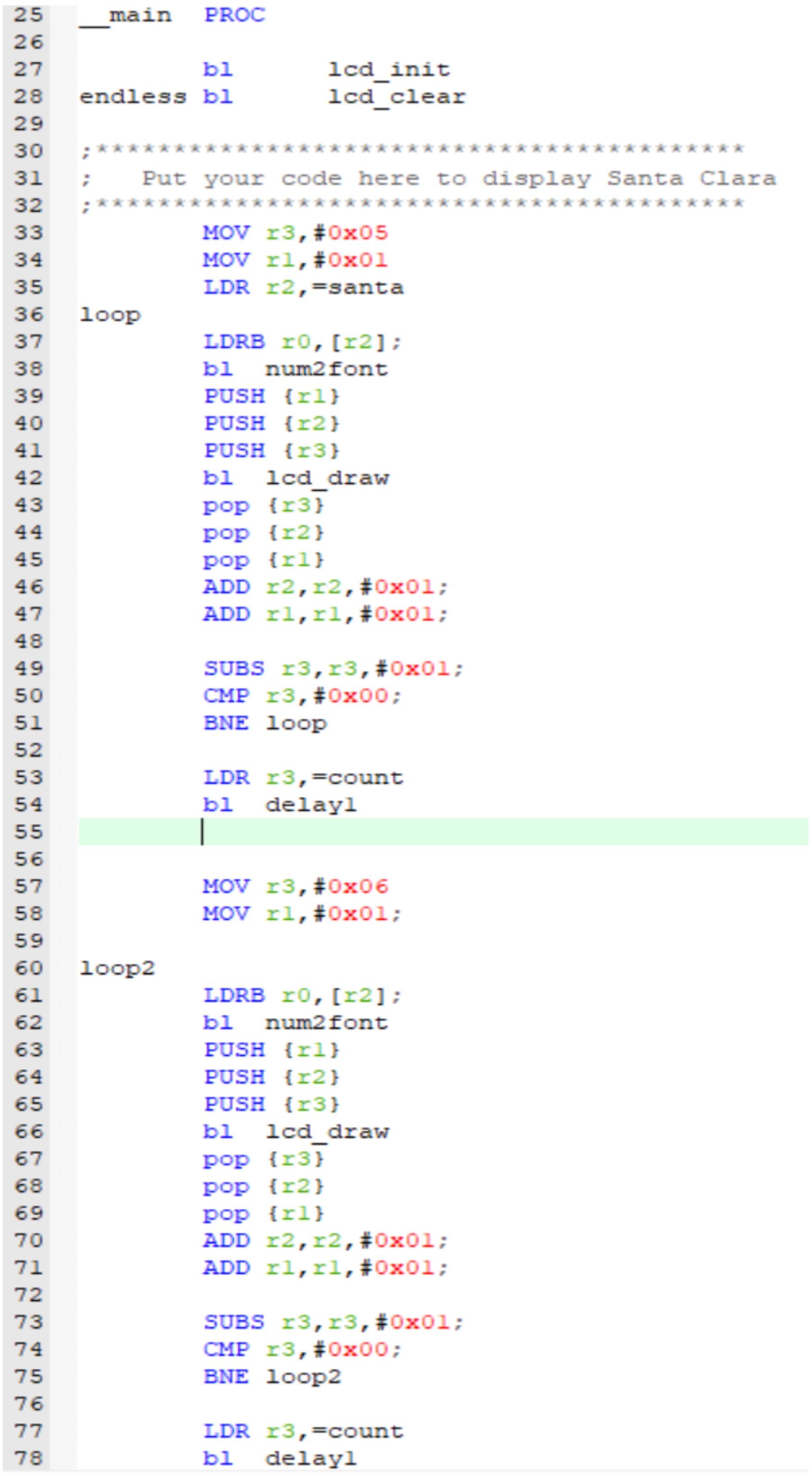
```
__main  PROC

        bl      lcd_init
endless bl      lcd_clear

;*******************************************
;   Put your code here to display Santa Clara
;*******************************************
        MOV r3,#0x05
        MOV r1,#0x01
        LDR r2,=santa
loop
        LDRB r0,[r2];
        bl  let2font
        PUSH {r1}
        PUSH {r2}
        PUSH {r3}
        bl  lcd_draw
        pop {r3}
        pop {r2}
        pop {r1}
        ADD r2,r2,#0x01;
        ADD r1,r1,#0x01;

        SUBS r3,r3,#0x01;
        CMP r3,#0x00;
        BNE loop

        LDR r3,=count
        bl  delay1


        MOV r3,#0x06
        MOV r1,#0x01;

loop2
        LDRB    r0,[r2];
        bl  let2font
        PUSH {r1}
        PUSH {r2}
        PUSH {r3}
        bl  lcd_draw
        pop {r3}
        pop {r2}
        pop {r1}
        ADD r2,r2,#0x01;
        ADD r1,r1,#0x01;

        SUBS r3,r3,#0x01;
        CMP r3,#0x00;
        BNE loop2

        LDR r3,=count
        bl  delay1
```



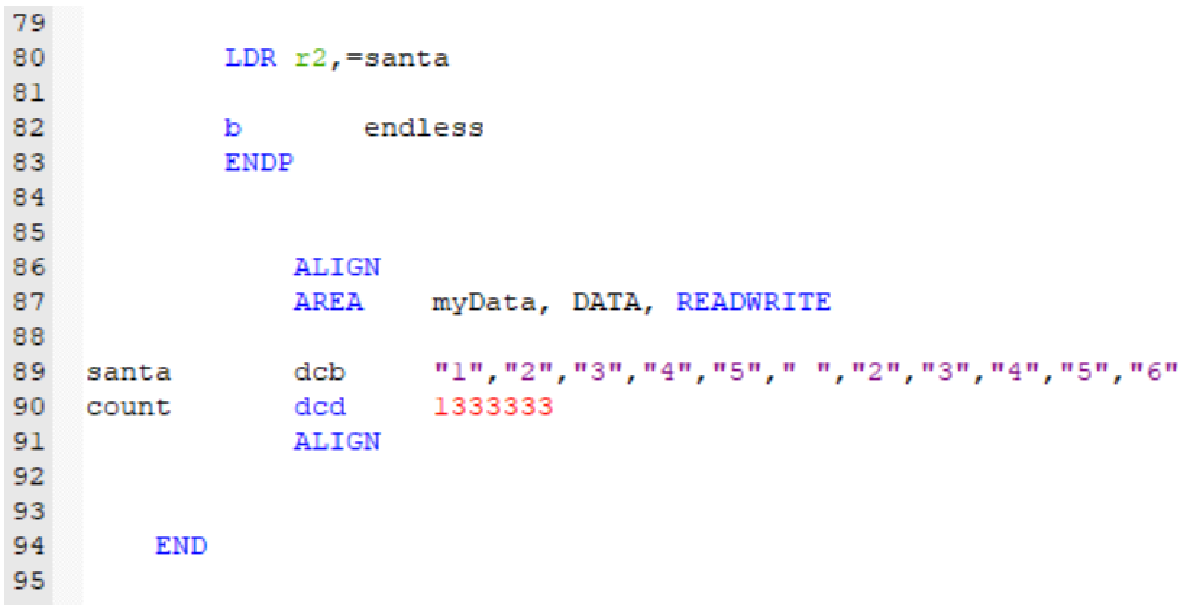
```
        LDR r2,=santa

        b       endless
        ENDP


            ALIGN
            AREA    myData, DATA, READWRITE

santa       dcb     "S", "A", "N", "T", "A", " ", "C", "L", "A", "R", "A"
count       dcd     1333333
            ALIGN


    END

```

Problem 2

Main.s:

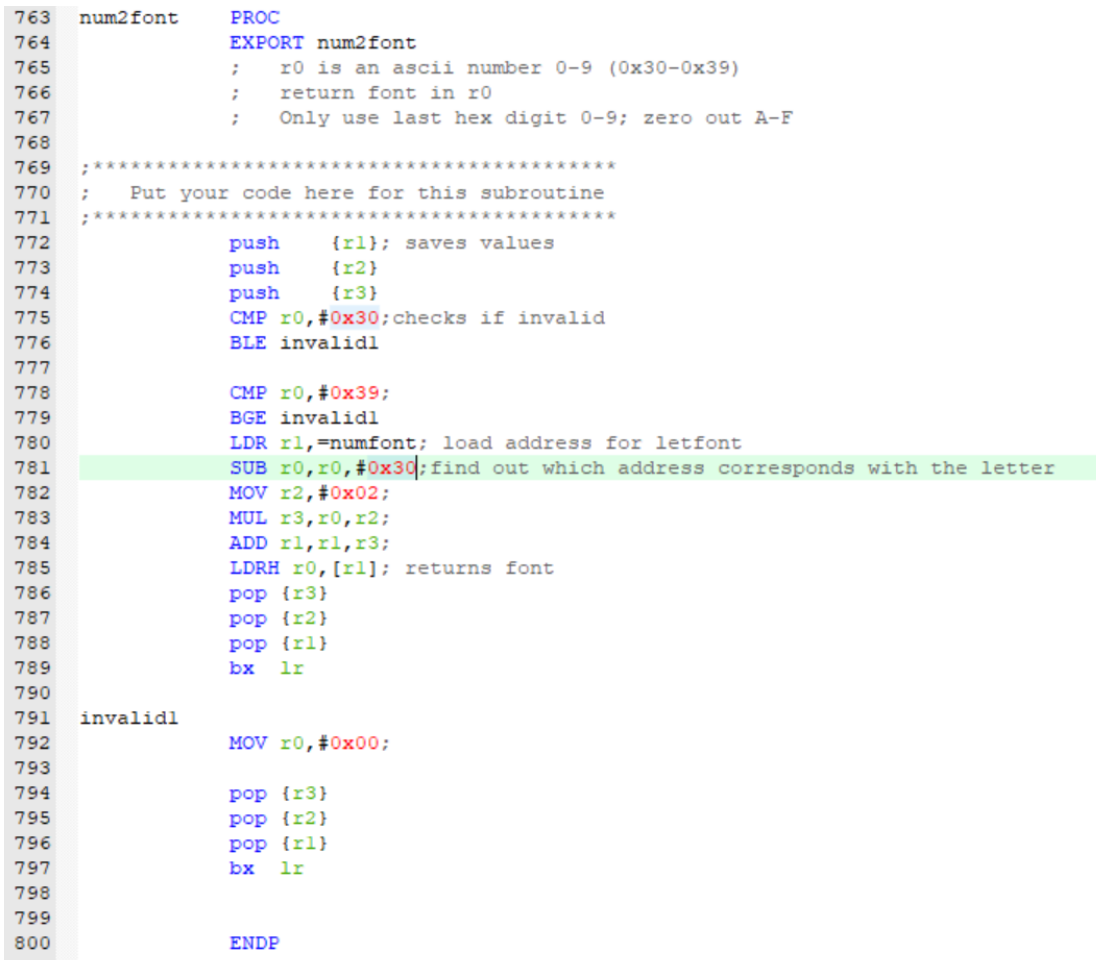


```
__main  PROC

        bl      lcd_init
endless bl      lcd_clear

;*******************************************
;    Put your code here to display Santa Clara
;*******************************************
        MOV r3,#0x05
        MOV r1,#0x01
        LDR r2,=santa
loop
        LDRB r0,[r2];
        bl  num2font
        PUSH {r1}
        PUSH {r2}
        PUSH {r3}
        bl  lcd_draw
        pop {r3}
        pop {r2}
        pop {r1}
        ADD r2,r2,#0x01;
        ADD r1,r1,#0x01;

        SUBS r3,r3,#0x01;
        CMP r3,#0x00;
        BNE loop

        LDR r3,=count
        bl  delay1


        MOV r3,#0x06
        MOV r1,#0x01;

loop2
        LDRB r0,[r2];
        bl  num2font
        PUSH {r1}
        PUSH {r2}
        PUSH {r3}
        bl  lcd_draw
        pop {r3}
        pop {r2}
        pop {r1}
        ADD r2,r2,#0x01;
        ADD r1,r1,#0x01;

        SUBS r3,r3,#0x01;
        CMP r3,#0x00;
        BNE loop2

        LDR r3,=count
        bl  delay1
```



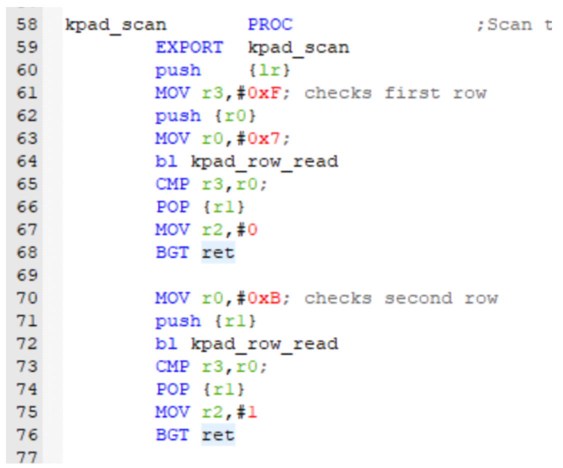
```

          LDR r2,=santa

          b        endless
          ENDP


              ALIGN
              AREA    myData, DATA, READWRITE

santa         dcb     "1","2","3","4","5"," ","2","3","4","5","6"
count         dcd     1333333
              ALIGN


     END

```

numfont2.s

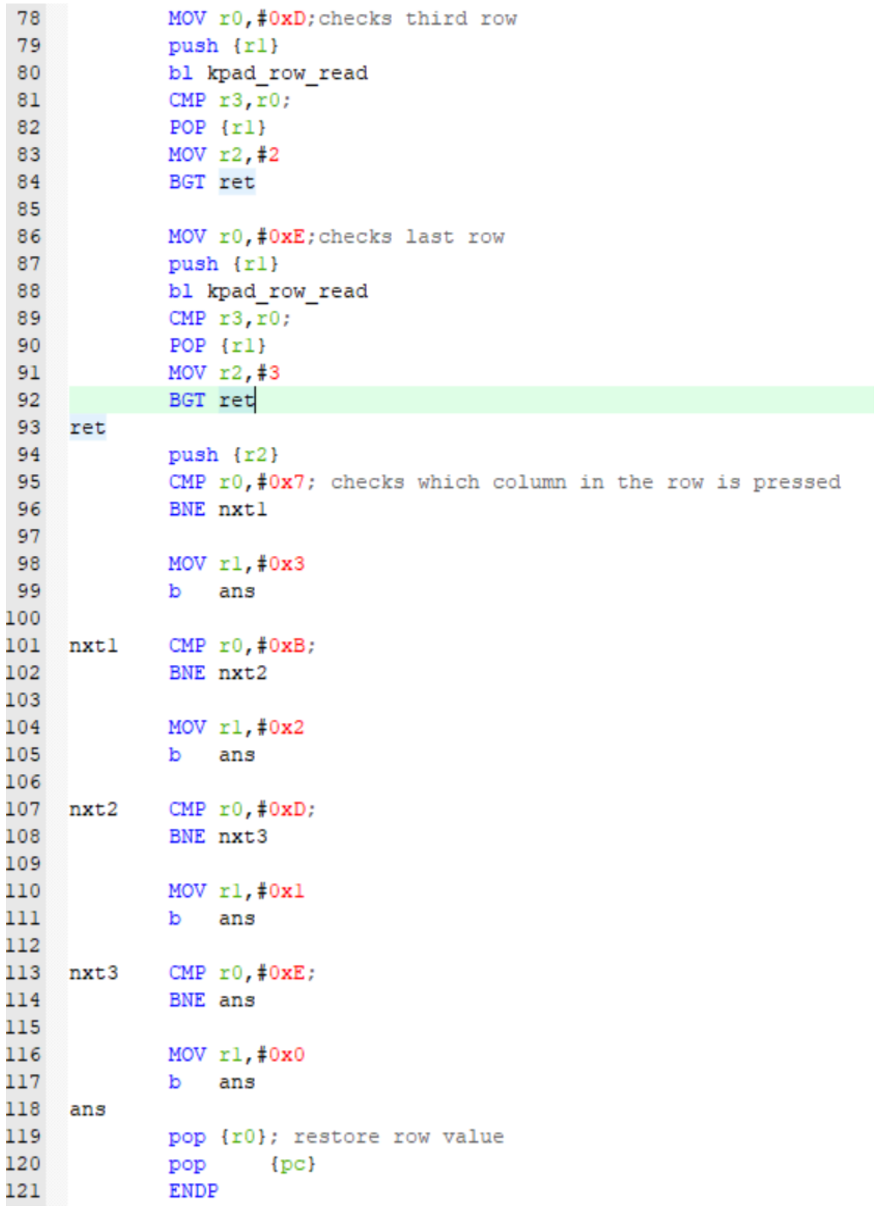


```
num2font    PROC
            EXPORT num2font
            ;   r0 is an ascii number 0-9 (0x30-0x39)
            ;   return font in r0
            ;   Only use last hex digit 0-9; zero out A-F

;*******************************************
;   Put your code here for this subroutine
;*******************************************
            push    {r1}; saves values
            push    {r2}
            push    {r3}
            CMP r0,#0x30;checks if invalid
            BLE invalidl

            CMP r0,#0x39;
            BGE invalidl
            LDR r1,=numfont; load address for letfont
            SUB r0,r0,#0x30;find out which address corresponds with the letter
            MOV r2,#0x02;
            MUL r3,r0,r2;
            ADD r1,r1,r3;
            LDRH r0,[r1]; returns font
            pop {r3}
            pop {r2}
            pop {r1}
            bx  lr

invalidl
            MOV r0,#0x00;

            pop {r3}
            pop {r2}
            pop {r1}
            bx  lr


            ENDP
```

Problem 3:



```
kpad_scan       PROC                ;Scan t
        EXPORT  kpad_scan
        push    {lr}
        MOV r3,#0xF; checks first row
        push {r0}
        MOV r0,#0x7;
        bl kpad_row_read
        CMP r3,r0;
        POP {r1}
        MOV r2,#0
        BGT ret

        MOV r0,#0xB; checks second row
        push {r1}
        bl kpad_row_read
        CMP r3,r0;
        POP {r1}
        MOV r2,#1
        BGT ret

```



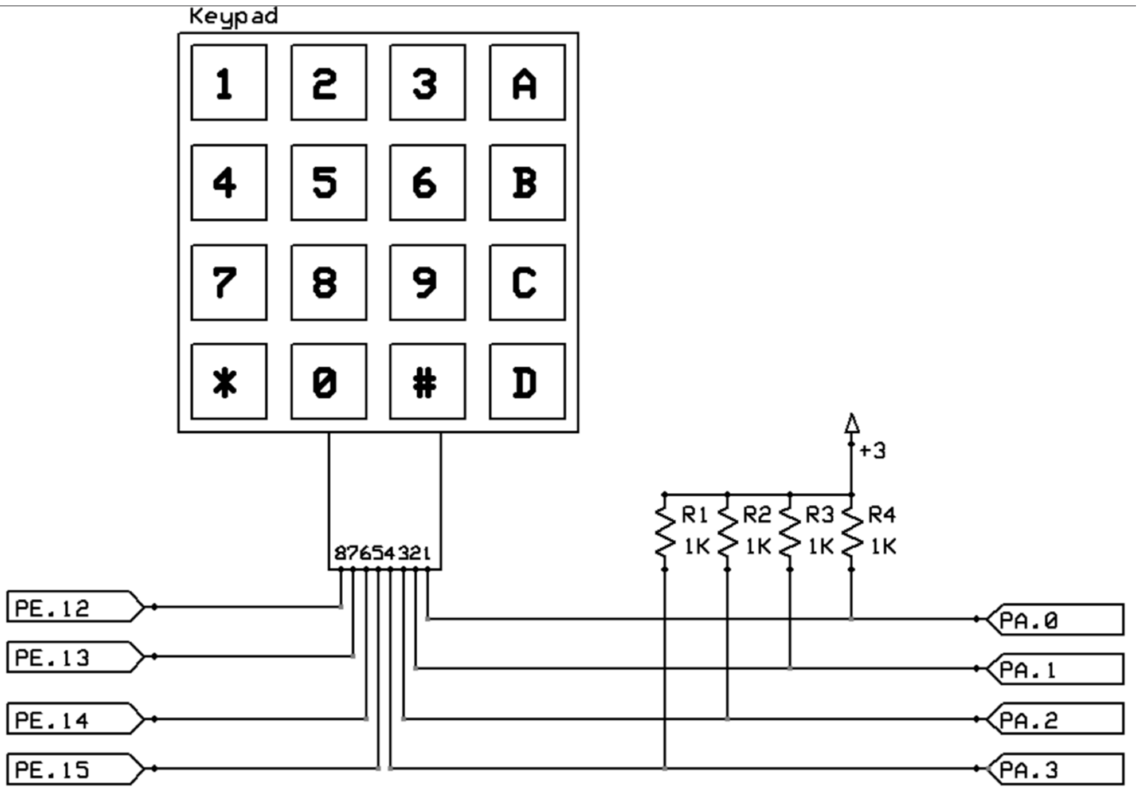
```
        MOV r0,#0xD;checks third row
        push {r1}
        bl kpad_row_read
        CMP r3,r0;
        POP {r1}
        MOV r2,#2
        BGT ret

        MOV r0,#0xE;checks last row
        push {r1}
        bl kpad_row_read
        CMP r3,r0;
        POP {r1}
        MOV r2,#3
        BGT ret
ret
        push {r2}
        CMP r0,#0x7; checks which column in the row is pressed
        BNE nxt1

        MOV r1,#0x3
        b   ans

nxt1    CMP r0,#0xB;
        BNE nxt2

        MOV r1,#0x2
        b   ans

nxt2    CMP r0,#0xD;
        BNE nxt3

        MOV r1,#0x1
        b   ans

nxt3    CMP r0,#0xE;
        BNE ans

        MOV r1,#0x0
        b   ans
ans
        pop {r0}; restore row value
        pop     {pc}
        ENDP
```

Demo Videos:

Problem 1:
https://drive.google.com/file/d/1vbmB37prATh0jty4OchS_PwtPw3hdaXD/view?usp=sharing

Problem 2:
https://drive.google.com/file/d/1G4euDIDRwhTT0hbo3z7dyvxv_Gcbi2uM/view?usp=sharing

Problem 3:
https://drive.google.com/file/d/1Vh36kh6vAKjRmibtZV6D2az8wxruCEEY/view?usp=sharing


Keypad
1
2
3
A
4
5
6
B
7
8
9
C
*
0
#
D
87654321
+3
R1 1K
R2 1K
R3 1K
R4 1K
PE.12
PE.13
PE.14
PE.15
PA.0
PA.1
PA.2
PA.3