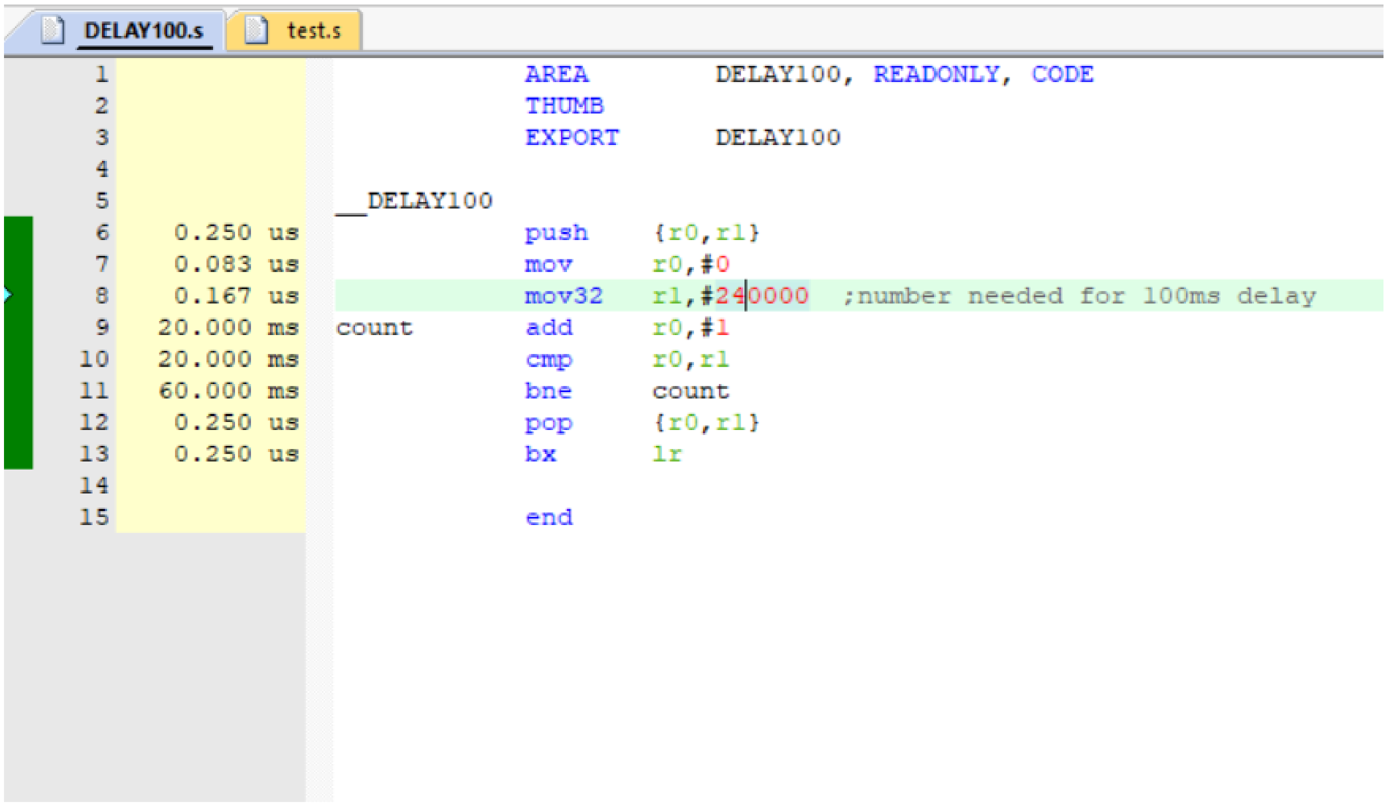
Mehmet Ataş

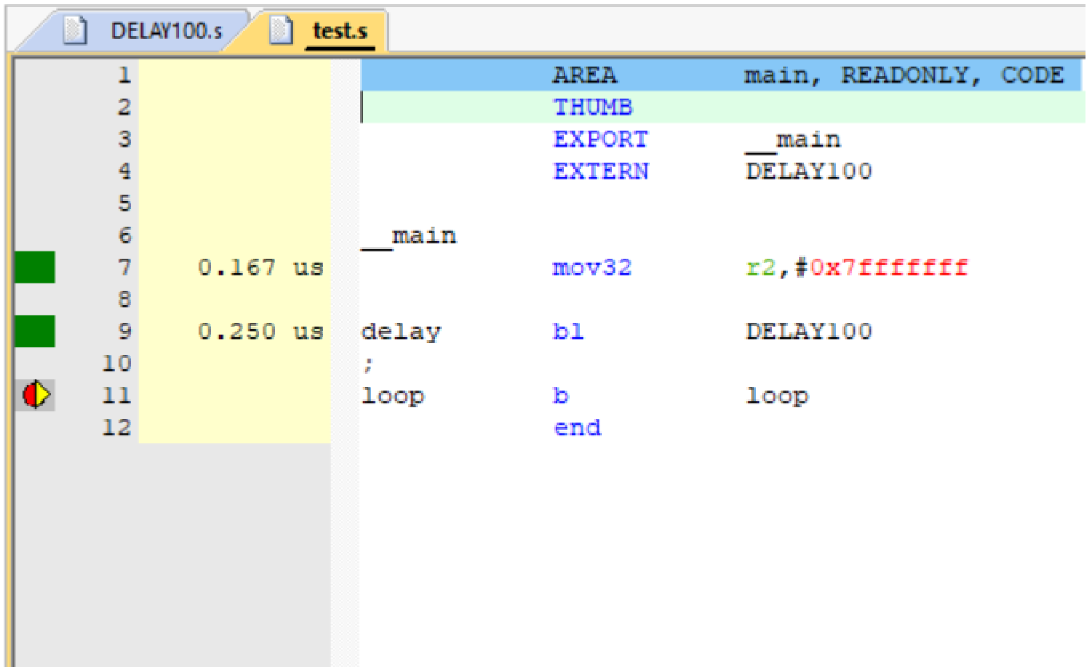
2304020

Preliminary Work for Exp. #2

1) I write a subroutine, DELAY100, that causes approximately 100 msec delay upon calling.

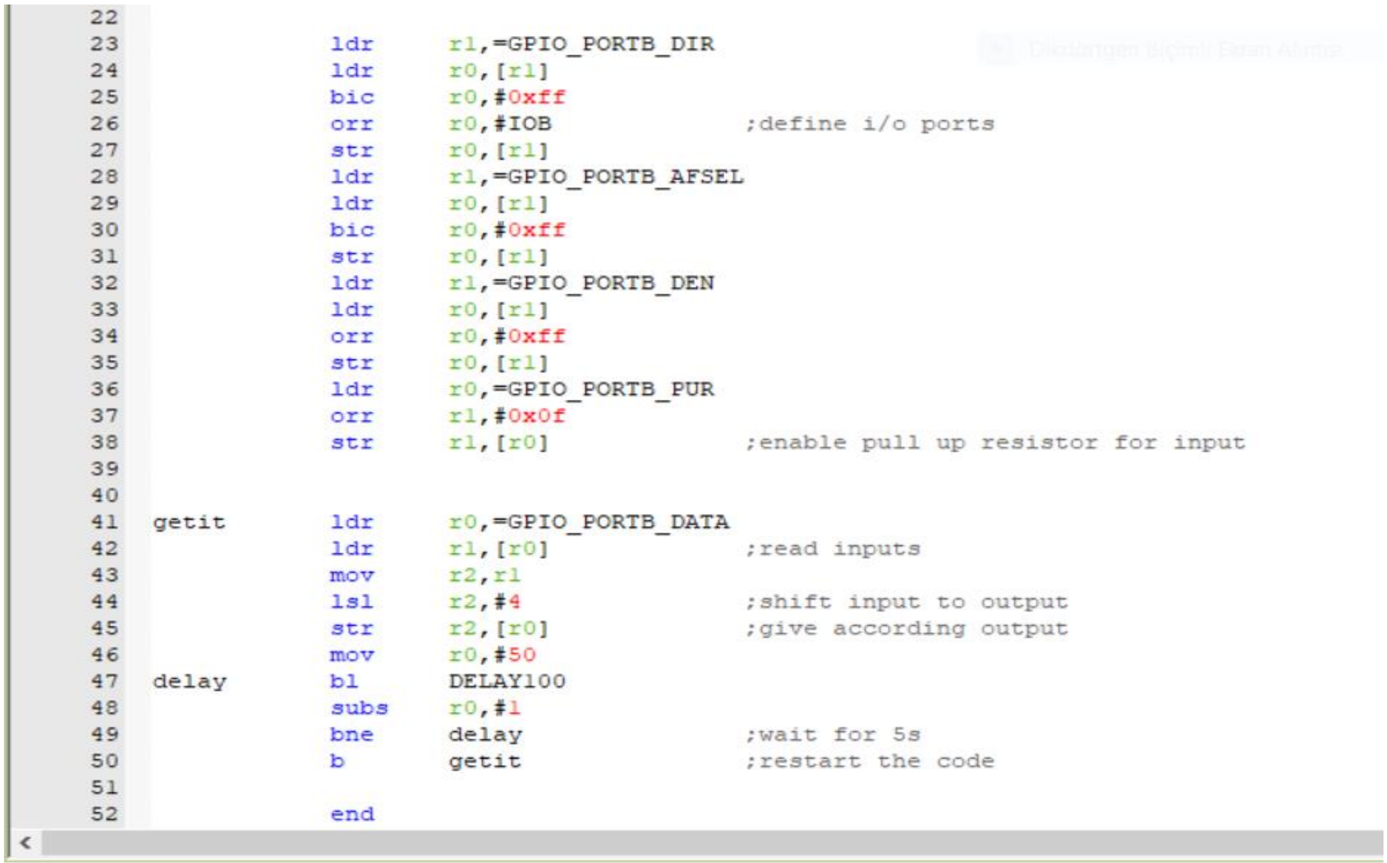


```
                AREA        DELAY100, READONLY, CODE
                THUMB
                EXPORT      DELAY100

__DELAY100
0.250 us        push    {r0,r1}
0.083 us        mov     r0,#0
0.167 us        mov32   r1,#240000  ;number needed for 100ms delay
20.000 ms count add     r0,#1
20.000 ms       cmp     r0,r1
60.000 ms       bne     count
0.250 us        pop     {r0,r1}
0.250 us        bx      lr

                end
```

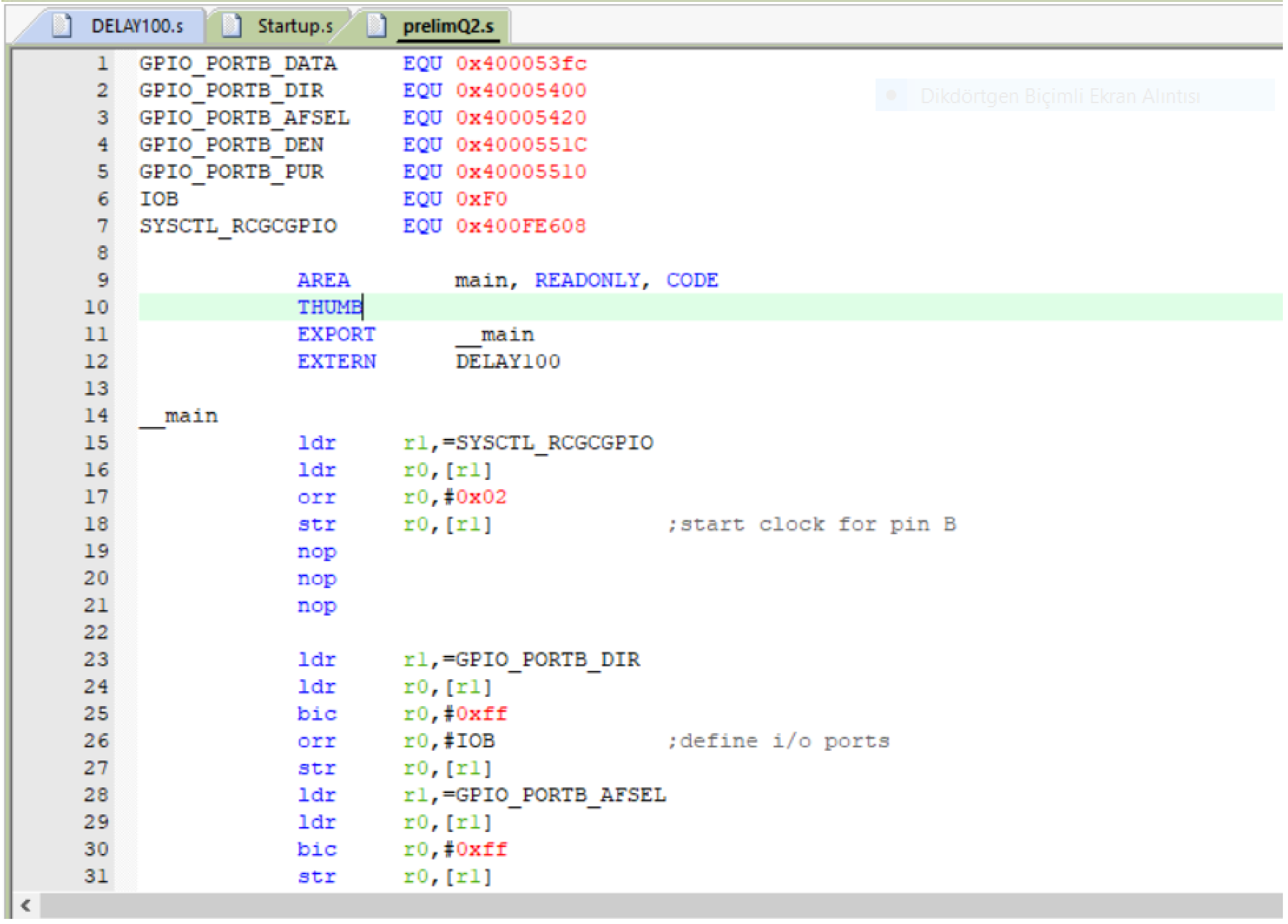


```
                AREA        main, READONLY, CODE
                THUMB
                EXPORT      __main
                EXTERN      DELAY100

__main
0.167 us        mov32       r2,#0x7fffffff

0.250 us delay  bl          DELAY100
;
loop            b           loop
                end
```

2) I write a program that continuously detects which key is pressed and outputs the ID of the key through Termite Window after the key is released.



```

            ldr     r1,=GPIO_PORTB_DIR
            ldr     r0,[r1]
            bic     r0,#0xff
            orr     r0,#IOB                 ;define i/o ports
            str     r0,[r1]
            ldr     r1,=GPIO_PORTB_AFSEL
            ldr     r0,[r1]
            bic     r0,#0xff
            str     r0,[r1]
            ldr     r1,=GPIO_PORTB_DEN
            ldr     r0,[r1]
            orr     r0,#0xff
            str     r0,[r1]
            ldr     r0,=GPIO_PORTB_PUR
            orr     r1,#0x0f
            str     r1,[r0]                 ;enable pull up resistor for input


getit       ldr     r0,=GPIO_PORTB_DATA
            ldr     r1,[r0]                 ;read inputs
            mov     r2,r1
            lsl     r2,#4                   ;shift input to output
            str     r2,[r0]                 ;give according output
            mov     r0,#50
delay       bl      DELAY100
            subs    r0,#1
            bne     delay                   ;wait for 5s
            b       getit                   ;restart the code

            end
```



```
GPIO_PORTB_DATA     EQU 0x400053fc
GPIO_PORTB_DIR      EQU 0x40005400
GPIO_PORTB_AFSEL    EQU 0x40005420
GPIO_PORTB_DEN      EQU 0x4000551C
GPIO_PORTB_PUR      EQU 0x40005510
IOB                 EQU 0xF0
SYSCTL_RCGCGPIO     EQU 0x400FE608

            AREA        main, READONLY, CODE
            THUMB
            EXPORT      __main
            EXTERN      DELAY100

__main
            ldr     r1,=SYSCTL_RCGCGPIO
            ldr     r0,[r1]
            orr     r0,#0x02
            str     r0,[r1]                 ;start clock for pin B
            nop
            nop
            nop

            ldr     r1,=GPIO_PORTB_DIR
            ldr     r0,[r1]
            bic     r0,#0xff
            orr     r0,#IOB                 ;define i/o ports
            str     r0,[r1]
            ldr     r1,=GPIO_PORTB_AFSEL
            ldr     r0,[r1]
            bic     r0,#0xff
            str     r0,[r1]
```

3) While considering the interface of the 4x4 keypad introduced in Chapter 2, I write a program that constantly detects which key is pressed and gives the key's ID via the Termite Window after the key is released. I can only assume that one key will be pressed at a time and that no other keys can be pressed before releasing a key. The program is resistant to possible bouncing effects during both pressing and releasing.

a. How can you detect whether any key is pressed?

We can check for inputs, if any is LOW then a key is pressed.

b. How can you detect whether a pressed key is released?

We can check the inputs after a key is pressed, if none of the inputs are LOW then the key is released.

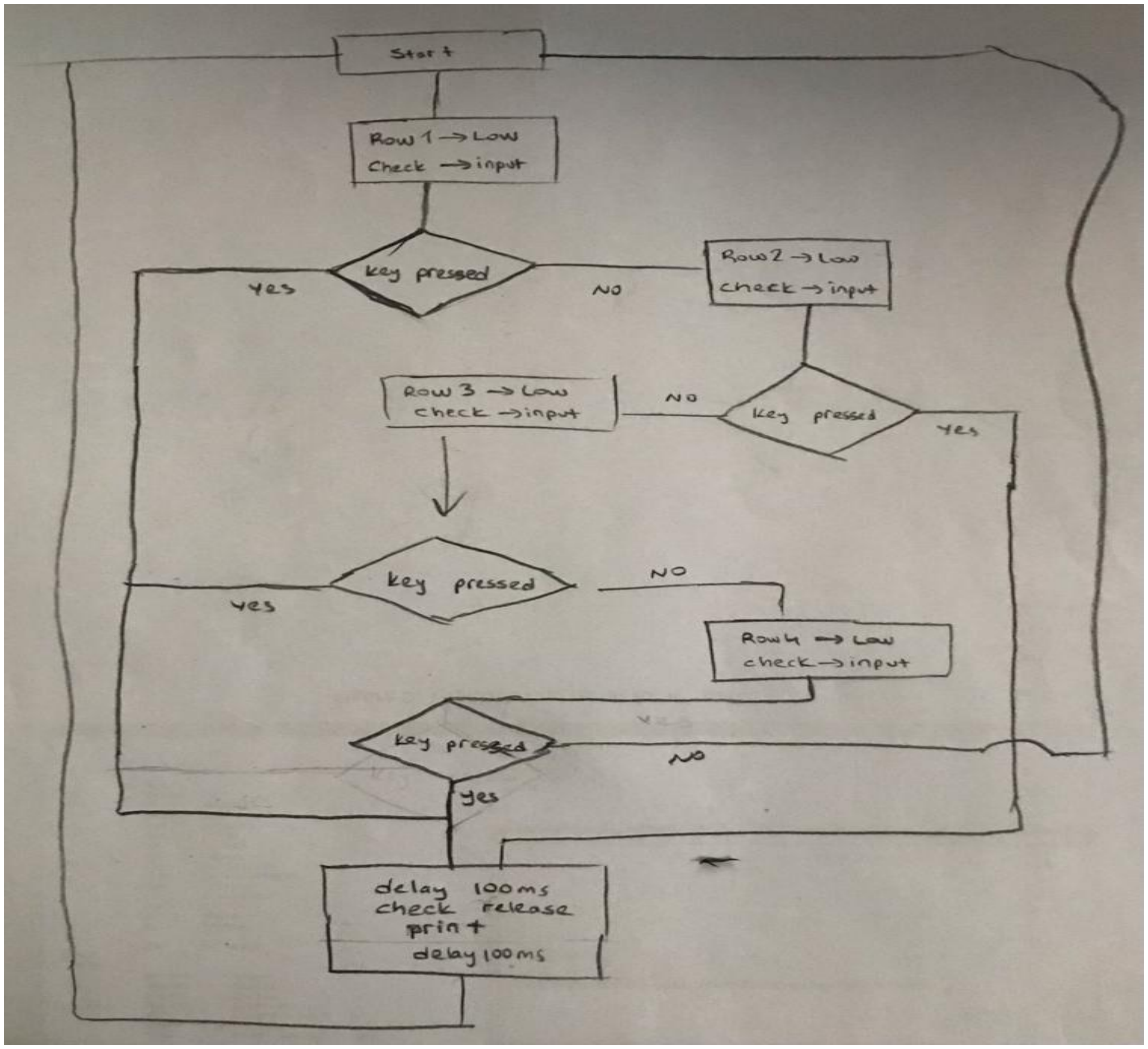
c. Assuming that you have detected that a key is pressed. Explain your algorithm to determine which one is pressed.

We can make one row's output LOW at a time and check the inputs. If any key is pressed in that row than we can see the column from the inputs. So, we can locate any key.

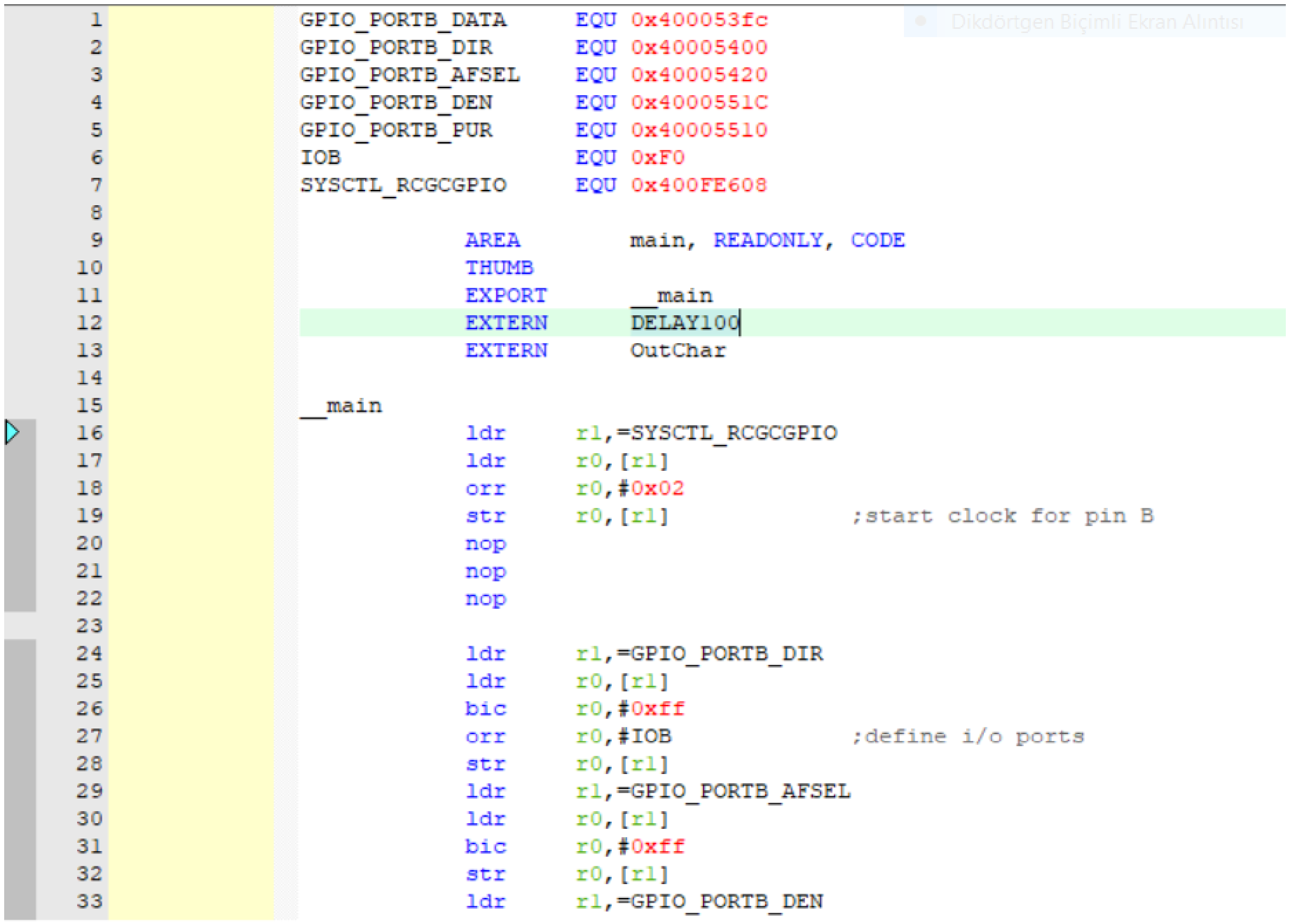
d. Discuss what can happen due to bouncing. How can you avoid bouncing effects?

If we check the output immediately after it is pressed, we can see that the output is HIGH even if the key is not yet released due to bouncing effect. We can wait 100ms before reading the input, so that bouncing effect disappears.

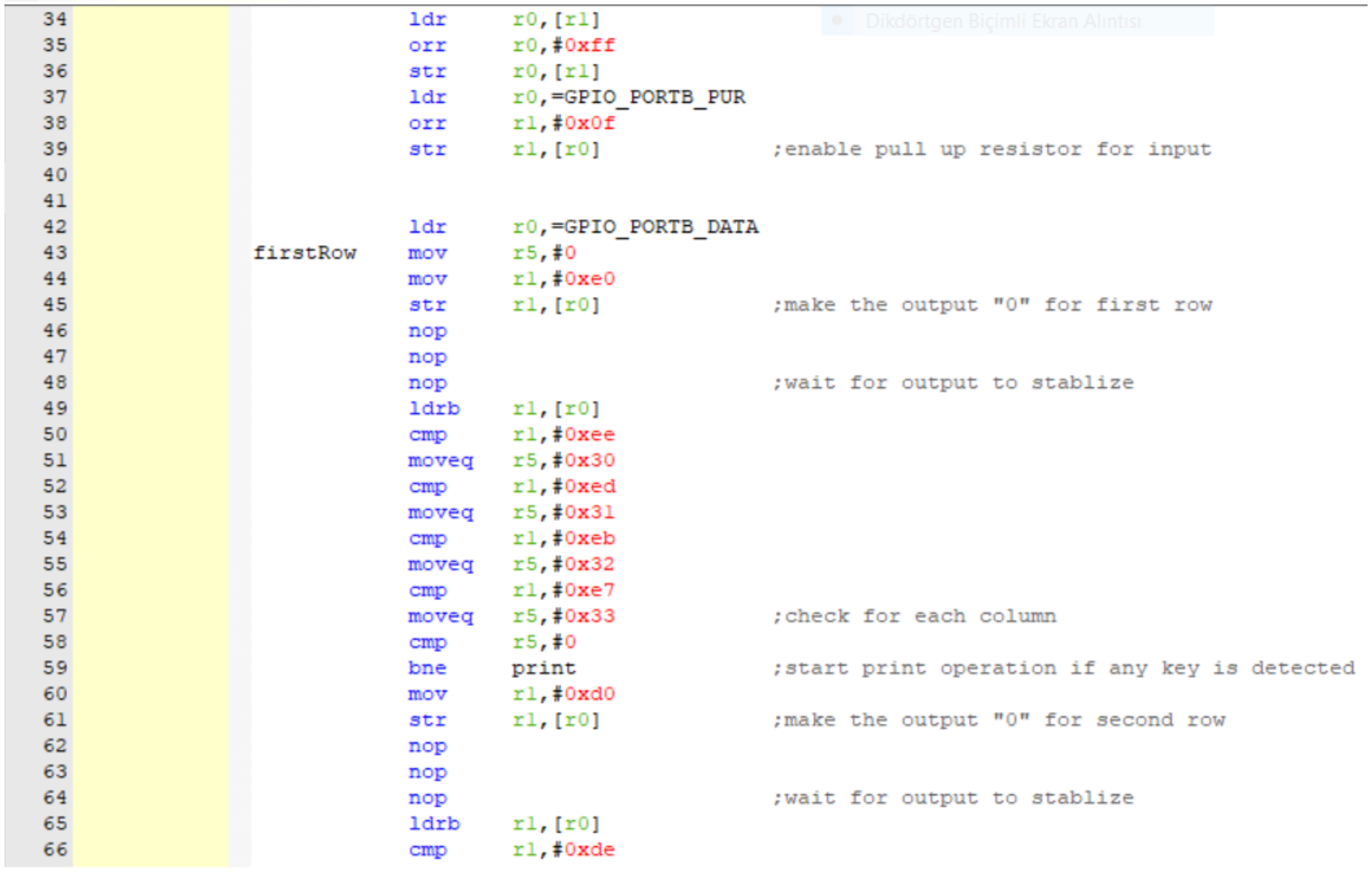
e. Now, develop your overall end-to-end algorithm that outputs ID of the pressed key to the terminal window and draw its flow chart.



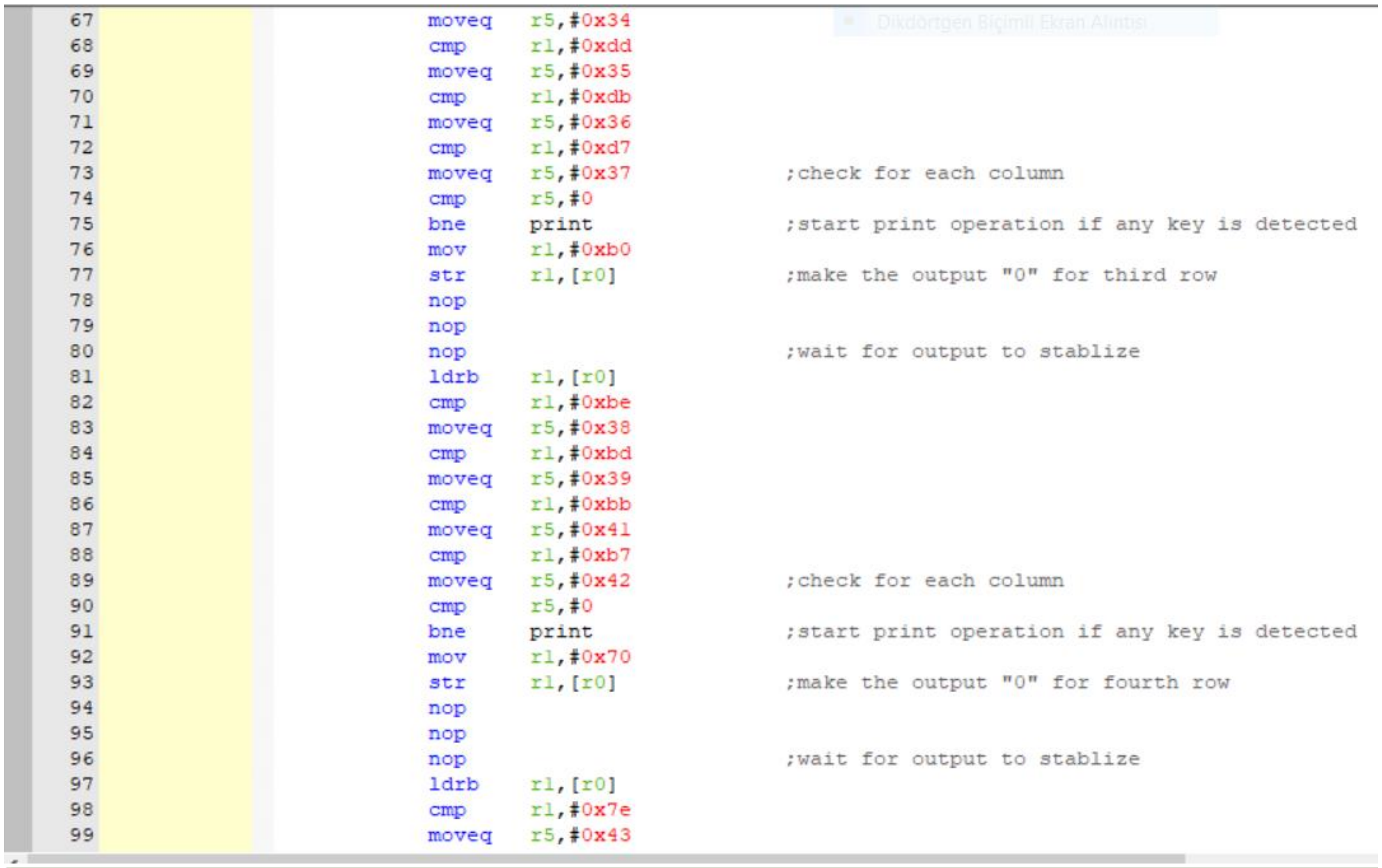
f. Implement the developed algorithm in part-e by using assembly language.



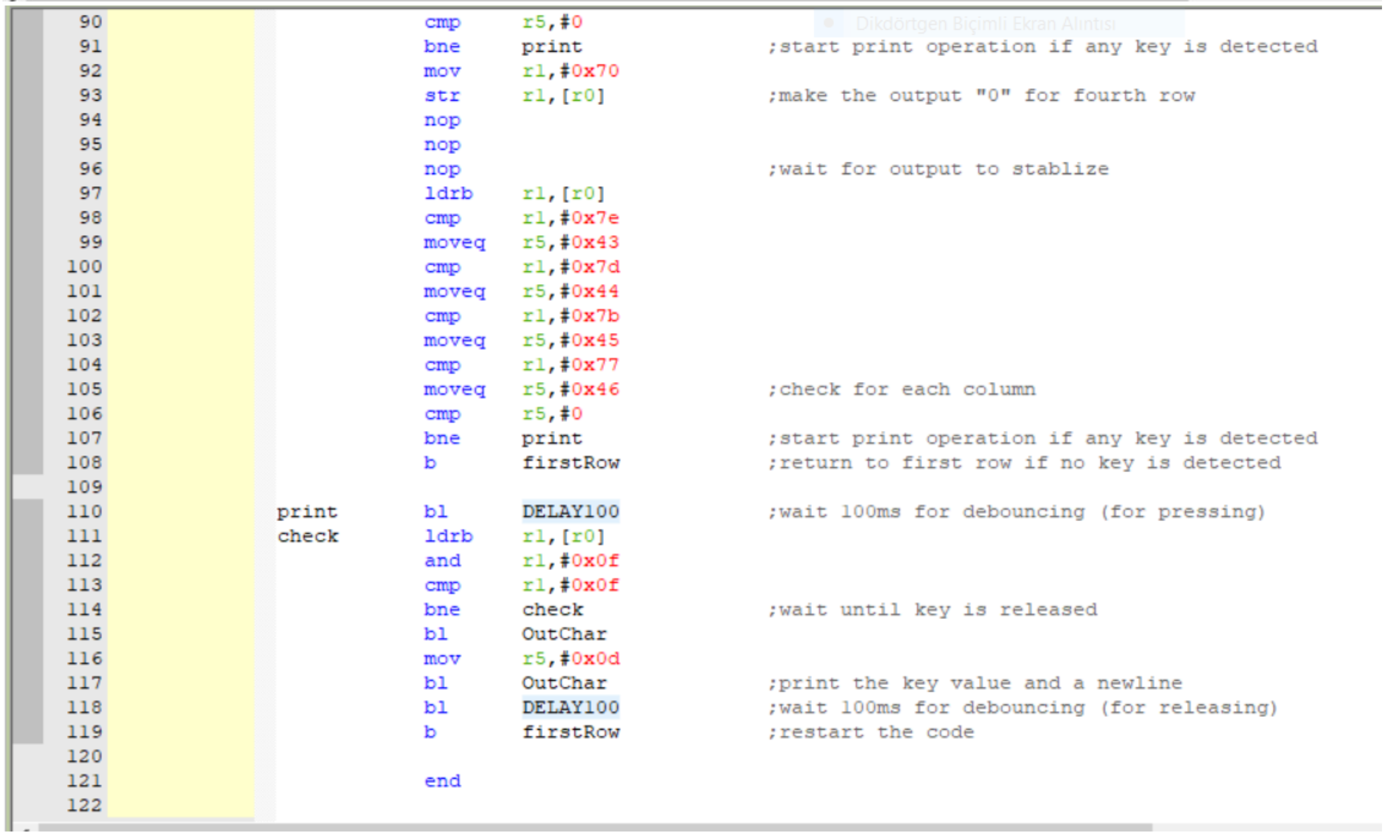
```
GPIO_PORTB_DATA     EQU 0x400053fc
GPIO_PORTB_DIR      EQU 0x40005400
GPIO_PORTB_AFSEL    EQU 0x40005420
GPIO_PORTB_DEN      EQU 0x4000551C
GPIO_PORTB_PUR      EQU 0x40005510
IOB                 EQU 0xF0
SYSCTL_RCGCGPIO     EQU 0x400FE608

            AREA        main, READONLY, CODE
            THUMB
            EXPORT      __main
            EXTERN      DELAY100
            EXTERN      OutChar

__main
            ldr     r1,=SYSCTL_RCGCGPIO
            ldr     r0,[r1]
            orr     r0,#0x02
            str     r0,[r1]             ;start clock for pin B
            nop
            nop
            nop

            ldr     r1,=GPIO_PORTB_DIR
            ldr     r0,[r1]
            bic     r0,#0xff
            orr     r0,#IOB             ;define i/o ports
            str     r0,[r1]
            ldr     r1,=GPIO_PORTB_AFSEL
            ldr     r0,[r1]
            bic     r0,#0xff
            str     r0,[r1]
            ldr     r1,=GPIO_PORTB_DEN
```



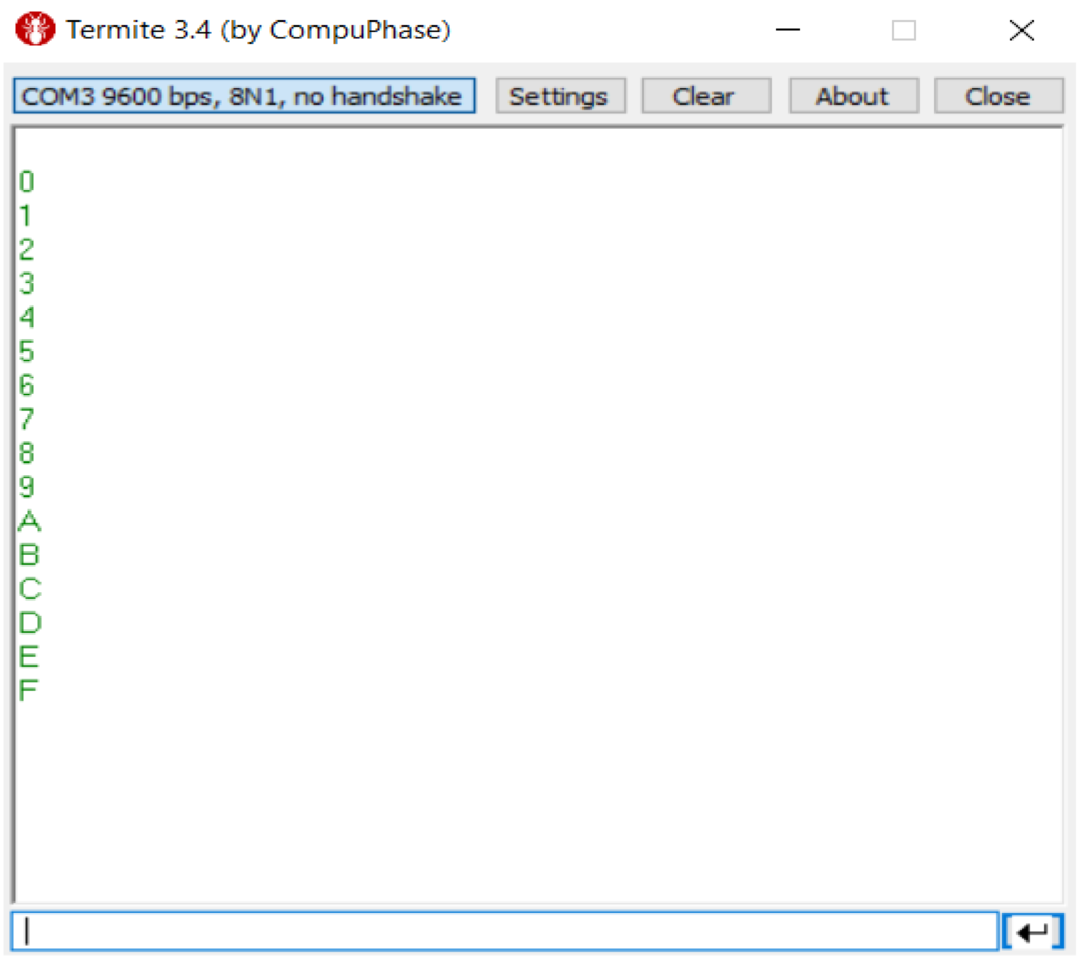
```
            ldr     r0,[r1]
            orr     r0,#0xff
            str     r0,[r1]
            ldr     r0,=GPIO_PORTB_PUR
            orr     r1,#0x0f
            str     r1,[r0]             ;enable pull up resistor for input


            ldr     r0,=GPIO_PORTB_DATA
firstRow    mov     r5,#0
            mov     r1,#0xe0
            str     r1,[r0]             ;make the output "0" for first row
            nop
            nop
            nop                         ;wait for output to stablize
            ldrb    r1,[r0]
            cmp     r1,#0xee
            moveq   r5,#0x30
            cmp     r1,#0xed
            moveq   r5,#0x31
            cmp     r1,#0xeb
            moveq   r5,#0x32
            cmp     r1,#0xe7
            moveq   r5,#0x33            ;check for each column
            cmp     r5,#0
            bne     print               ;start print operation if any key is detected
            mov     r1,#0xd0
            str     r1,[r0]             ;make the output "0" for second row
            nop
            nop
            nop                         ;wait for output to stablize
            ldrb    r1,[r0]
            cmp     r1,#0xde
```



```
                moveq   r5,#0x34
                cmp     r1,#0xdd
                moveq   r5,#0x35
                cmp     r1,#0xdb
                moveq   r5,#0x36
                cmp     r1,#0xd7
                moveq   r5,#0x37            ;check for each column
                cmp     r5,#0
                bne     print               ;start print operation if any key is detected
                mov     r1,#0xb0
                str     r1,[r0]             ;make the output "0" for third row
                nop
                nop
                nop                         ;wait for output to stablize
                ldrb    r1,[r0]
                cmp     r1,#0xbe
                moveq   r5,#0x38
                cmp     r1,#0xbd
                moveq   r5,#0x39
                cmp     r1,#0xbb
                moveq   r5,#0x41
                cmp     r1,#0xb7
                moveq   r5,#0x42            ;check for each column
                cmp     r5,#0
                bne     print               ;start print operation if any key is detected
                mov     r1,#0x70
                str     r1,[r0]             ;make the output "0" for fourth row
                nop
                nop
                nop                         ;wait for output to stablize
                ldrb    r1,[r0]
                cmp     r1,#0x7e
                moveq   r5,#0x43
```



```
                cmp     r5,#0
                bne     print               ;start print operation if any key is detected
                mov     r1,#0x70
                str     r1,[r0]             ;make the output "0" for fourth row
                nop
                nop
                nop                         ;wait for output to stablize
                ldrb    r1,[r0]
                cmp     r1,#0x7e
                moveq   r5,#0x43
                cmp     r1,#0x7d
                moveq   r5,#0x44
                cmp     r1,#0x7b
                moveq   r5,#0x45
                cmp     r1,#0x77
                moveq   r5,#0x46            ;check for each column
                cmp     r5,#0
                bne     print               ;start print operation if any key is detected
                b       firstRow            ;return to first row if no key is detected

print           bl      DELAY100            ;wait 100ms for debouncing (for pressing)
check           ldrb    r1,[r0]
                and     r1,#0x0f
                cmp     r1,#0x0f
                bne     check               ;wait until key is released
                bl      OutChar
                mov     r5,#0x0d
                bl      OutChar             ;print the key value and a newline
                bl      DELAY100            ;wait 100ms for debouncing (for releasing)
                b       firstRow            ;restart the code

                end
```



```
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
```

	L1	L2	L3	L4	
R1	ee	ed	eb	e7	0123
R2	de	dd	db	d7	4567
R3	be	bd	bb	b7	89AB
R4	7e	7d	7b	77	CDEF

B0->R1

B1->R2

B2->R3

B3->R4

B4->L1

B5->L2

B6->L3

B7->L4