Beste Öztop 2375624
Deniz Karakay 2443307

14.11.2022

EE447 Introduction to Microprocessors

Laboratory-2 Preliminary Work

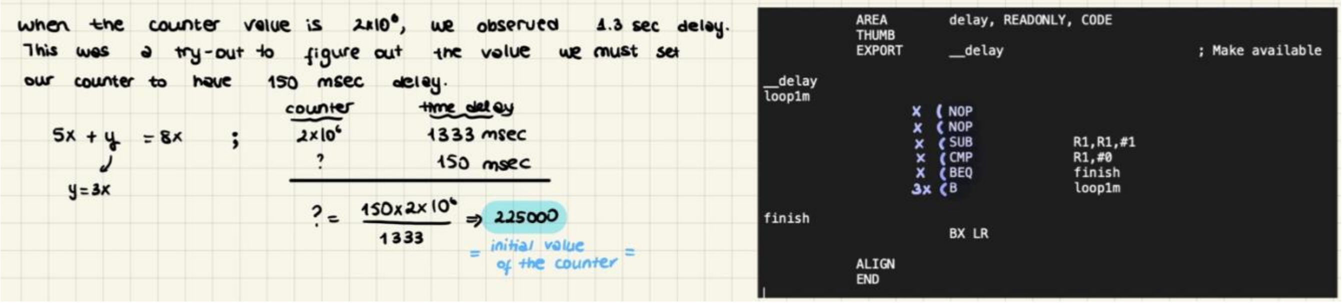
Question-1

Write a subroutine, DELAY150, that causes approximately 150 msec delay upon calling.

For the DELAY150 subroutine, we have utilized the following section of code with a main code down below.

```
            AREA        main, READONLY, CODE
            THUMB
               EXTERN       __delay      ; Make available
            EXPORT     __main      ; Make available

__main
                       MOV32          R1, #225000
                       BL                    __delay

            ALIGN
            END

//Subroutine

            AREA        delay, READONLY, CODE
            THUMB
            EXPORT      __delay                              ; Make available

__delay
loop1m
                       NOP
                       NOP
                       SUB            R1,R1,#1
                       CMP            R1,#0
                       BEQ            finish
                       B              loop1m

finish
                       BX LR

            ALIGN
            END
```

In the main code, we have a counter value kept in the memory address located in R11 which was initially set to $2\ x\ 10^6$. Then, from the toolbar, we have seen that the delay obtained with this counter value is approximately 1300 msec (1.3sec). We have also observed that the following time delays have x-x-x-x-x-3x proportionality among them. Therefore, at the end, 8x equals 1.3sec. With this logic, from a direct proportionality, we obtained that the counter should be set to 225000 to have 150 msec delay. The exact calculations can be found down below. The result with 150 msec delay can be found in Figure 1.



when the counter value is 2×10^{6}, we observed 1.3 sec delay. This was a try-out to figure out the value we must set our counter to have 150 msec delay.

$$5x + y = 8x \quad ; \qquad y = 3x$$

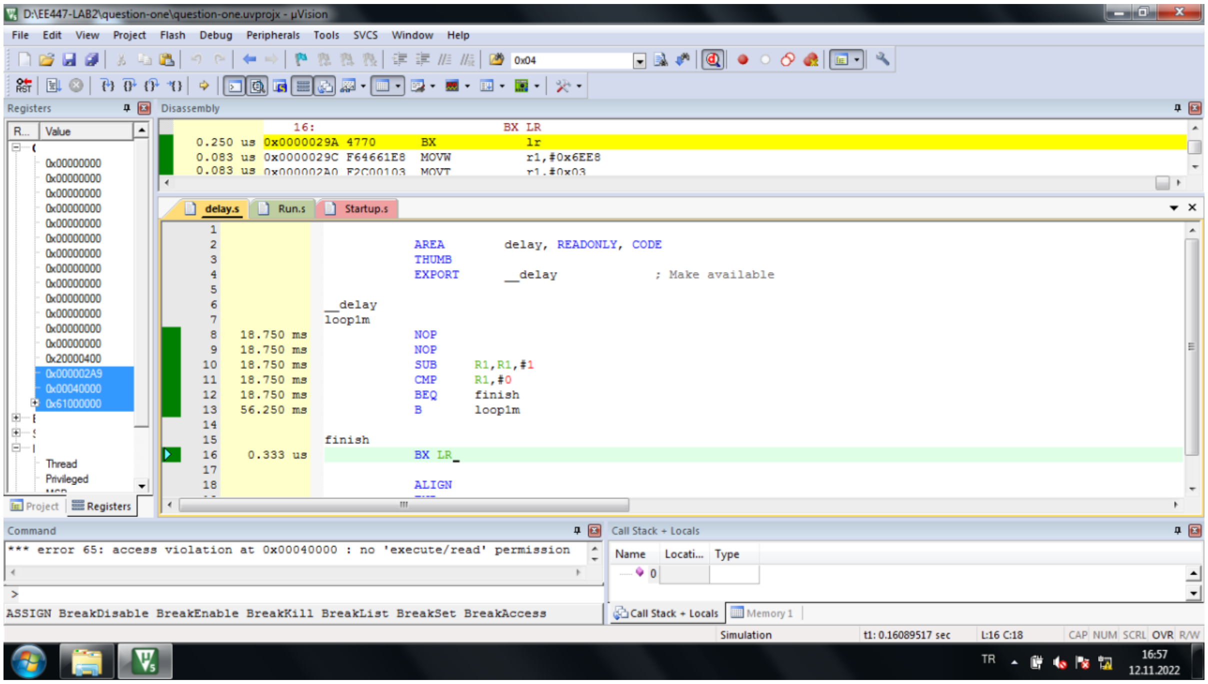
counter	time delay
2×10^6	1333 msec
?	150 msec

$$? = \frac{150\times2\times10^6}{1333} \Rightarrow \boxed{225000}$$

= initial value of the counter =

```
            AREA        delay, READONLY, CODE
            THUMB
            EXPORT      __delay                       ; Make available

__delay
loop1m
                x  ( NOP
                x  ( NOP
                x  ( SUB             R1,R1,#1
                x  ( CMP             R1,#0
                x  ( BEQ             finish
               3x  ( B               loop1m

finish
                       BX LR

            ALIGN
            END
```

Beste Öztop 2375624
Deniz Karakay 2443307 14.11.2022



```
                16:                     BX LR
  0.250 us 0x0000029A 4770      BX          lr
  0.083 us 0x0000029C F64661E8  MOVW        r1,#0x6EE8
  0.083 us 0x000002A0 F2C00103  MOVT        r1,#0x03
```

```
                                AREA        delay, READONLY, CODE
                                THUMB
                                EXPORT      __delay             ; Make available

                        __delay
                        loop1m
            18.750 ms           NOP
            18.750 ms           NOP
            18.750 ms           SUB     R1,R1,#1
            18.750 ms           CMP     R1,#0
            18.750 ms           BEQ     finish
            56.250 ms           B       loop1m

                        finish
             0.333 us           BX LR

                                ALIGN
```

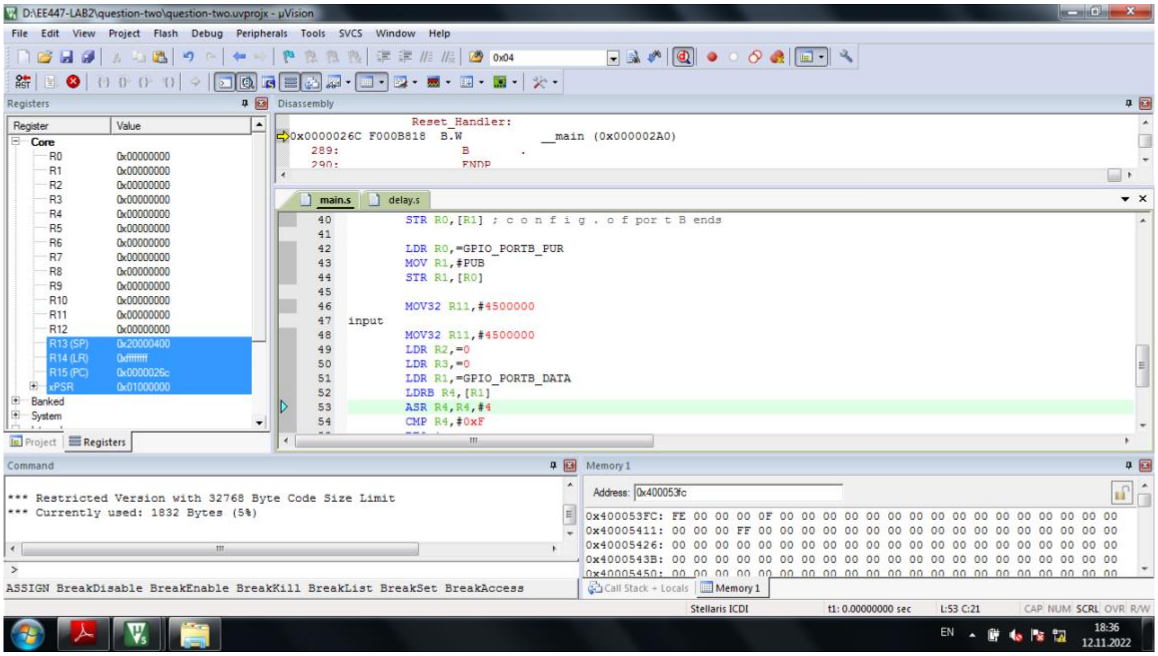
```
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x20000400
0x000002A9
0x00040000
0x61000000
```

```
*** error 65: access violation at 0x00040000 : no 'execute/read' permission
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
```

Figure 1: *DELAY150 (delay.s) Subroutine with 150 msec time delay*

Question-2

Write a program for a simple data transfer scheme. You are required to take inputs from push buttons and reflect the status of the buttons to the LEDs that are connected to the output port for approximately every 3 seconds. Namely, an input should be read for every 3 seconds and the status of that reading should remain at the output until the next reading. The status of a pressed button is 1 and the status of a released button is 0. You should use low nibble of Port B (B_3 to B_0) for outputs, and high nibble of Port B (B_7 to B_4) for inputs.



Register	Value
Core	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000400
R14 (LR)	0xffffffff
R15 (PC)	0x0000026c
xPSR	0x01000000

```
                Reset_Handler:
0x0000026C F000B818  B.W              __main (0x000002A0)
   289:                    B        .
   290:                    ENDP
```

```
        STR R0,[R1] ; c o n f i g . o f por t B ends

        LDR R0,=GPIO_PORTB_PUR
        MOV R1,#PUB
        STR R1,[R0]

        MOV32 R11,#4500000
input
        MOV32 R11,#4500000
        LDR R2,=0
        LDR R3,=0
        LDR R1,=GPIO_PORTB_DATA
        LDRB R4,[R1]
        ASR R4,R4,#4
        CMP R4,#0xF
```

```
*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 1832 Bytes (5%)
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
```

```
Address: 0x400053fc
0x400053FC: FE 00 00 00 0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40005411: 00 00 00 FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40005426: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000543B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40005450: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 2: *Input taking from the keyboard and showing on the LEDs , 3sec delay*

Beste Öztop 2375624
Deniz Karakay 2443307 14.11.2022

Please refer to this video to see our working code results.

Question-3

Consider the interface of the 4x4 keypad introduced in Section 2. You are required to 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. The IDs can be sequential numbers. For instance, the ID of the key at row 1 and column 1 can be 0 and the key at row 4 and column 4 can be F (0:F total 16 IDs). You may assume that only one key is to be pressed at a time and no other key can be pressed before releasing a key. Your program should be robust to possible bouncing effect during both pressing and releasing. Please attempt this problem by answering the following items:

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

When any key is pressed, using the algorithm we developed in the second question, we observe that the value stored in R5 (right shifted value showing the column entry) is always 0xF. Therefore, it is known that any key is pressed when this value is different than 0xF. By using the following section of code, we compare the value taken from the buttons (Port B) with 0xF.

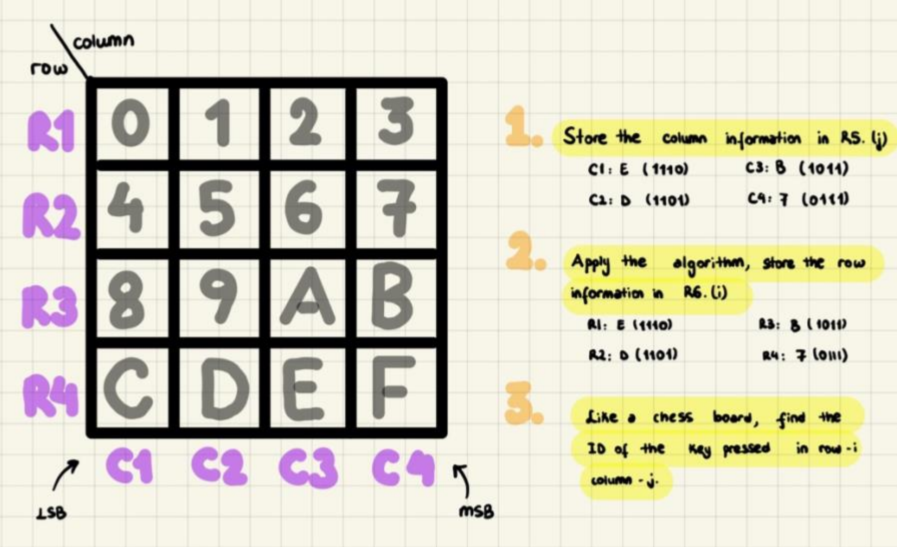
```
CMP R5,#0xF
BEQ input //Go and check again
```

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

With the same logic we applied in part-a i.e. to detect whether any key is pressed or not, now we check if the row value kept in R6 is rather different than 0xF or not. If so, we realize that the pressed key is released. Thus, before putting a new value on Termite, we need to print the current one. Therefore, the loop is sent to `print`. From the question description, it is suitable top rint the ID of the pressed key once it is released.

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

Firstly, we determine the column to which the pressed key belongs to. This value is then stored in R5. Then, we check the output observed in the columns by applying different row inputs. Each time, one of the rows is connected to the ground (i.e. set to 0) with the help of the pull down resistors. Then, if the output observed in the columns is different from 0xF (i.e. we reach the corresponding column), we have the exact address of the pressed key. These addresses kept in R4 and are determined as follows:



row \ column	C1	C2	C3	C4
R1	0	1	2	3
R2	4	5	6	7
R3	8	9	A	B
R4	C	D	E	F

1. Store the column information in R5. (j)

C1: E (1110)	C3: B (1011)
C2: D (1101)	C4: 7 (0111)

2. Apply the algorithm, store the row information in R6. (i)

R1: E (1110)	R3: B (1011)
R2: D (1101)	R4: 7 (0111)

3. Like a chess board, find the ID of the key pressed in row - i column - j.

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

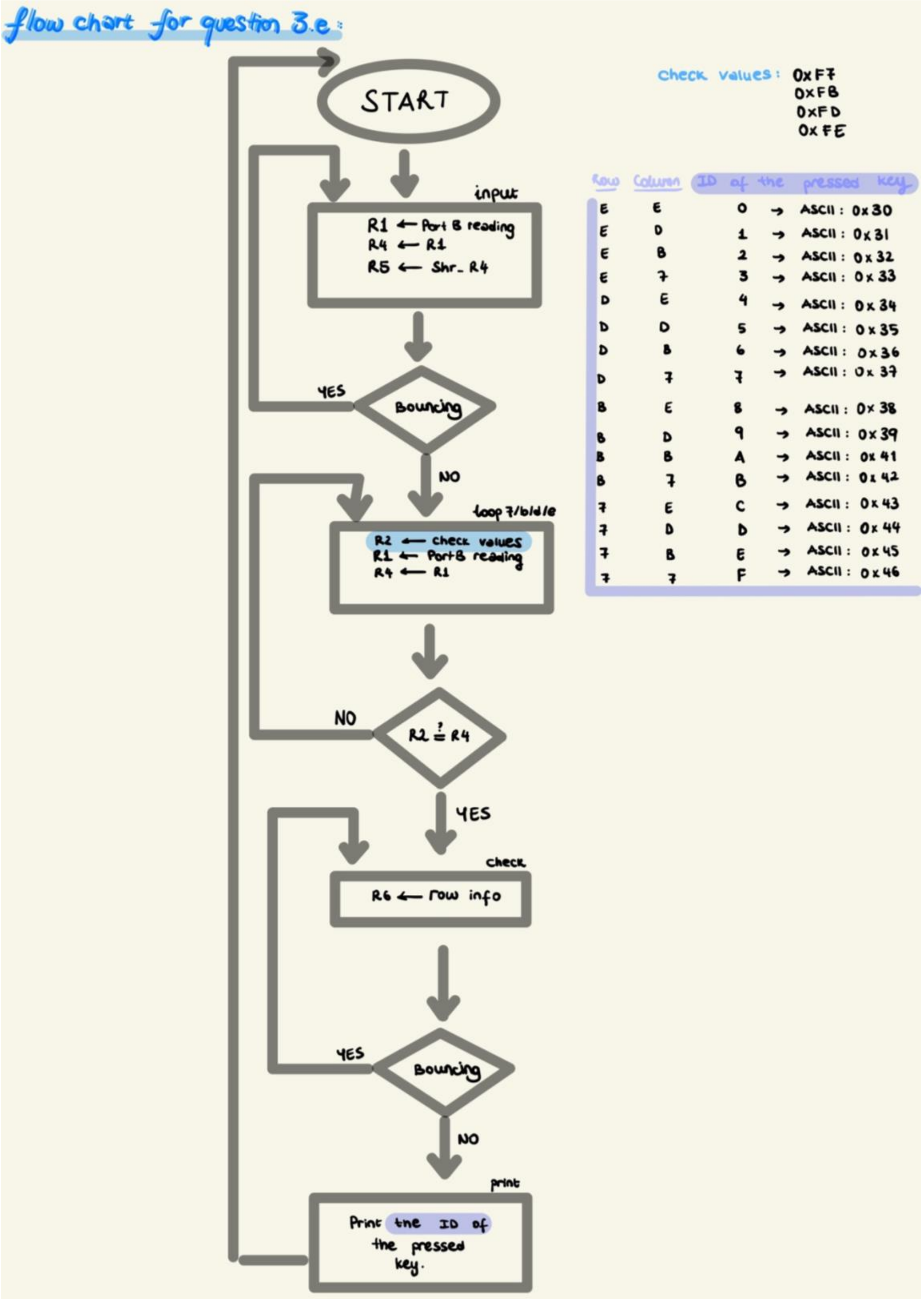
Due to bouncing, in fact, we observed some unwanted behavior in our code before applying the delay. The fact about bouncing is that if for some reason one key is pressed mistakenly before or after pressing the wanted key, there can be unwanted outputs in the termite. To avoid this behavior, we have used 50 msec of delay (with counter set to 75000) after taking the column information and also when we check whether a key is released or not. In other words, bouncing can affecct the working conditions when the key is pressed or released. The main principle in applying 50 msec delay to avoid bouncing effects is to compare the values taken from R1 (port-B) with 50msec of time difference. These values are stored in other register to be used in the compare operation. You can check the below section of code fort his comparison.

```
        ;Key is pressed, bouncing check

            LDR R1,=GPIO_PORTB_DATA
            LDRB R4,[R1]
            ASR   R3,R4,#4 ; R5 Column

            BL __delay

            LDR R1,=GPIO_PORTB_DATA
            LDRB R4,[R1]
            ASR   R5,R4,#4 ; R5 Column

            CMP R3, R5
            BNE input  ;Return to the input loop! You have
bouncing and you need to have another input from the port.
```

```
            ; Checking bouncing after the key is released
            SUB R6, R2,#0xF0 ;row
            MOV32 R11,#75000

            LDR R1,=GPIO_PORTB_DATA
            LDRB R8,[R1]
            ASR   R8,R8,#4 ; R5 Column

            BL __delay

            LDR R1,=GPIO_PORTB_DATA
            LDRB R9,[R1]
            ASR   R9,R9,#4 ; R5 Column

            CMP R8,R9
            BNE check ;Return to the check loop, you have
bouncing!
```

Beste Öztop 2375624
Deniz Karakay 2443307 14.11.2022

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.


flow chart for question 3.e:

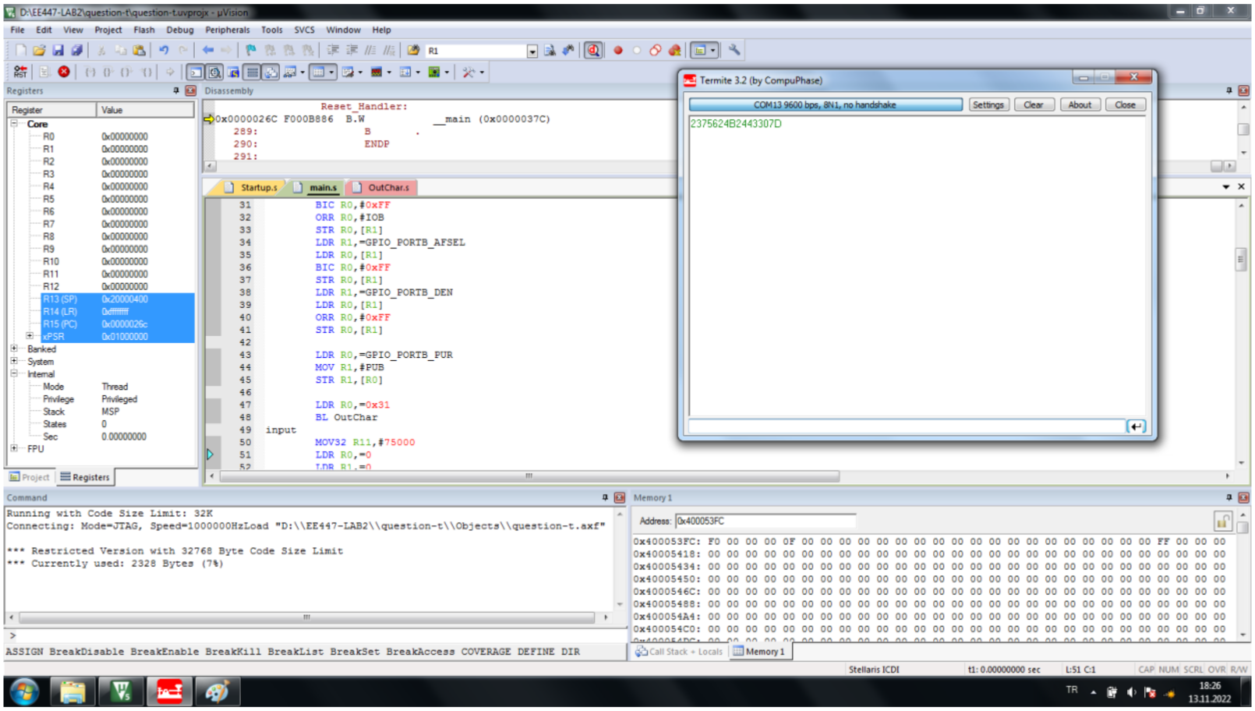
check values: 0xF7
0xFB
0xFD
0xFE

Row	Column	ID of the pressed key
E	E	0 → ASCII : 0x30
E	D	1 → ASCII : 0x31
E	B	2 → ASCII : 0x32
E	7	3 → ASCII : 0x33
D	E	4 → ASCII : 0x34
D	D	5 → ASCII : 0x35
D	B	6 → ASCII : 0x36
D	7	7 → ASCII : 0x37
B	E	8 → ASCII : 0x38
B	D	9 → ASCII : 0x39
B	B	A → ASCII : 0x41
B	7	B → ASCII : 0x42
7	E	C → ASCII : 0x43
7	D	D → ASCII : 0x44
7	B	E → ASCII : 0x45
7	7	F → ASCII : 0x46

Beste Öztop 2375624
Deniz Karakay 2443307 14.11.2022

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

The algorithm is implemented. The result can be checked from Figure 3.



D:\EE447-LAB2\question-t\question-t.uvprojx - µVision

Register	Value
Core	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000400
R14 (LR)	0xffffffff
R15 (PC)	0x0000026c
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	0
Sec	0.00000000
FPU	

```
                 Reset_Handler:
0x0000026C F000B886  B.W           __main (0x0000037C)
   289:                 B       .
   290:                 ENDP
   291:
```

```
            BIC R0,#0xFF
            ORR R0,#IOB
            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
            MOV R1,#PUB
            STR R1,[R0]

            LDR R0,=0x31
            BL OutChar
input
            MOV32 R11,#75000
            LDR R0,=0
            LDR R1,=0
```

```
Running with Code Size Limit: 32K
Connecting: Mode=JTAG, Speed=1000000HzLoad "D:\\EE447-LAB2\\question-t\\Objects\\question-t.axf"

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 2328 Bytes (7%)
```

```
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR
```

Address: 0x400053FC

```
0x400053FC: F0 00 00 00 0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 00 00 00
0x40005418: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40005434: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40005450: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000546C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40005488: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x400054A4: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x400054C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Termite 3.2 (by CompuPhase)

COM13 9600 bps, 8N1, no handshake

```
2375624B2443307D
```

Stellaris ICDI t1: 0.00000000 sec L:51 C:1

Figure 3: *The result of the algorithm printing the ID of the pressed key*