Preliminary work

EE 447: Lab #2

Parallel Input/Output and Keyboard Interface

Berkay İPEK

2304814 – Sec.2

Question 1)

D:\OKUL\ee 4ün 1i\Lab-447\LabThree\delay\programming_directive.s

```
 1
 2                AREA        main, READONLY, CODE
 3                THUMB
 4                EXTERN      OutChar
 5                EXPORT      DELAY100
 6
 7   DELAY100     PROC;
 8                PUSH        {R8}            ;To keep data in register8 same
 9   ;Since its clock is 16MHz. Each cycle will take 0.06us (1/16M sec) to operate
10   ;To take 100ms (0.1 sec = 10^5 us),
11   ;It should take (0.1sec)/((1/16M)sec) =1.6M cycle
12                MOV32       R8,#400000      ;Since each loop takes 4 cycles. # of iteration should be
     1.600.000/4 = 400000.
13
14   delaying     NOP                         ;Taking 1 cycle
15                SUBS        R8,#1           ;Taking 1 cycle
16                BNE         delaying        ;Taking 2 cycle
17                POP         {R8}
18                BX          LR
19                ALIGN
20                ENDP
```

Question 2)

D:\OKUL\ee 4ün 1i\Lab-447\LabThree\questiontwo\programming_directive.s

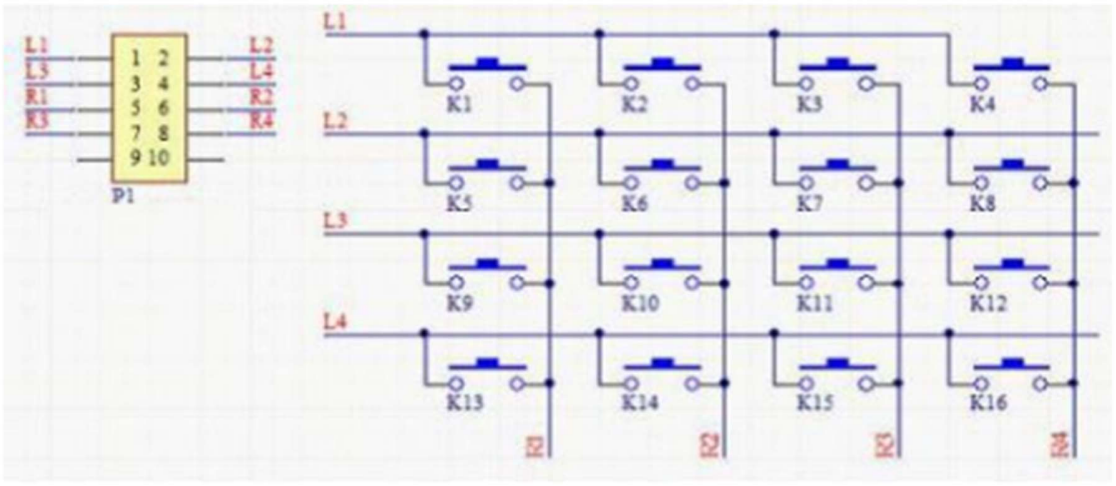
```
PB_INP              EQU 0x4000503C
PB_OUT              EQU 0x400053C0
GPIO_PORTB_DATA     EQU 0x400053FC ; data a d d r e s s t o a l l pi n s
GPIO_PORTB_DIR      EQU 0x40005400
GPIO_PORTB_AFSEL    EQU 0x40005420
GPIO_PORTB_DEN      EQU 0x4000551C
GPIO_PORTB_PDR      EQU 0x40005514
IOB                 EQU 0xF0
SYSCTL_RCGCGPIO     EQU 0x400FE608


                    AREA        main, READONLY, CODE
                    THUMB
                    EXTERN      DELAY100
                    EXPORT      __main

__main              PROC;
Start               LDR         R1,=SYSCTL_RCGCGPIO
                    LDR         R0,[R1]
                    ORR         R0,R0,#0x12
                    STR         R0,[R1]
                    NOP
                    NOP
                    NOP
                    LDR         R1,=GPIO_PORTB_DIR
                    LDR         R0,[R1]
                    MOV         R0,#0xF0
                    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         R1,=GPIO_PORTB_PDR
                    LDR         R0,[R1]
                    ORR         R0,#0xF0
                    STR         R0,[R1]

                    MOV         R2,#0
nanInp              LDR         R1,=PB_INP
                    LDR         R0,[R1]
                    CMP         R0,#0xF
                    BEQ         nanInp
                    BL          DELAY100
                    LDR         R3,[R1]
                    CMP         R3,R0
                    BNE         nanInp

                    CMP         R0,#0x0E
                    BEQ         LED1
                    CMP         R0,#0x0D
                    BEQ         LED2
                    CMP         R0,#0x0B
                    BEQ         LED3
                    CMP         R0,#0x07
                    BEQ         LED4
                    B           nanInp

LED1                LDR         R4,[R1]
                    CMP         R4,R0
                    BEQ         LED1
                    EOR         R2,R2,#0x10
                    LDR         R1,=PB_OUT
                    STR         R2,[R1]
                    B           nanInp

LED2                LDR         R4,[R1]
                    CMP         R4,R0
                    BEQ         LED2
                    EOR         R2,R2,#0x20
                    LDR         R1,=PB_OUT
                    STR         R2,[R1]
                    B           nanInp

```

```
78  LED3                LDR         R4,[R1]
79                      CMP         R4,R0
80                      BEQ         LED3
81                      EOR         R2,R2,#0x40
82                      LDR         R1,=PB_OUT
83                      STR         R2,[R1]
84                      B           nanInp
85
86  LED4                LDR         R4,[R1]
87                      CMP         R4,R0
88                      BEQ         LED4
89                      EOR         R2,R2,#0x80
90                      LDR         R1,=PB_OUT
91                      STR         R2,[R1]
92                      B           nanInp
93
94                      ALIGN
95                      ENDP
96                      END
97
```

Question 3)

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

The answer is between line 25-35 in the main function (programming_directives.s file). If there is a push in any keys. The output data will be changed accordingly. I connected L1-4 to PB0-3, R1-4 to PB 4-7. I set PB0-3 as outputs and PB4-7 as inputs. Also, I gave high voltages to these outputs one by one in a loop. Therefore, when there is pushed button, the input pins (PB4-7) should change. If there is no change in PB4-7, i.e. R2 is 0x00, that means there is no pushed button. If there is a pushed in a button, there will be a connection between L1-4 and R1-4. Therefore, PB4-7 values will be changed, and input data will be no longer zero. (It can be 0x10, 0x20, 0x40 or 0x80) I implemented this algorithm by comparing input data with possible data values(0x10, 0x20, 0x40, 0x80). If there is no matching , then it will go into the start point in a loop without calling OutChar function. If there is a matching, then it will go into the line COL1, COL2, COL3, or COL4.



b) How can you detect whether any key is released?

The answer is in line 38-40, 54-56, 70-72, 86-88 in the main function (programming_directives.s file). If there is a button pressed it will jump into COL1, COL2, COL3, COL4. After this jump, input will be read again, and it will be compared with previous read data. If they are equal, it will go into COL1 again. If they are not equal, that means key is released.

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

I gave high voltages into L1, L2, L3, L4 respectively and check R1, R2, R3, R4 status. If there is an R pin with high voltages, that means there is a pressed key. To determine which one is pressed, I store data for which L pin and R pin are high. To explain algorithm better,

At time t=0, L1 is high. Checks any R is also high.

At time t=t, L2 is high. Checks any R is also high.

At time t=2t, L3 is high. Checks any R is also high.

At time t=3t, L4 is high. Checks any R is also high.

By checking the register that stores input pin data, we can determine key in which column is pressed. For example, if K4 is pressed, then input register will be 1000.0000(0x80) when L1 is high. If K6 is pressed, then input register will be 0010.0000(0x20) when L2 is high. This input register will determine the column for the pressed key. That's why it will compare input register with

Input registers:

0000.0000 (0x00) -> there is no pushed button

0001.0000 (0x10) -> there is a pushed button in column 1

0010.0000 (0x20) -> there is a pushed button in column 2

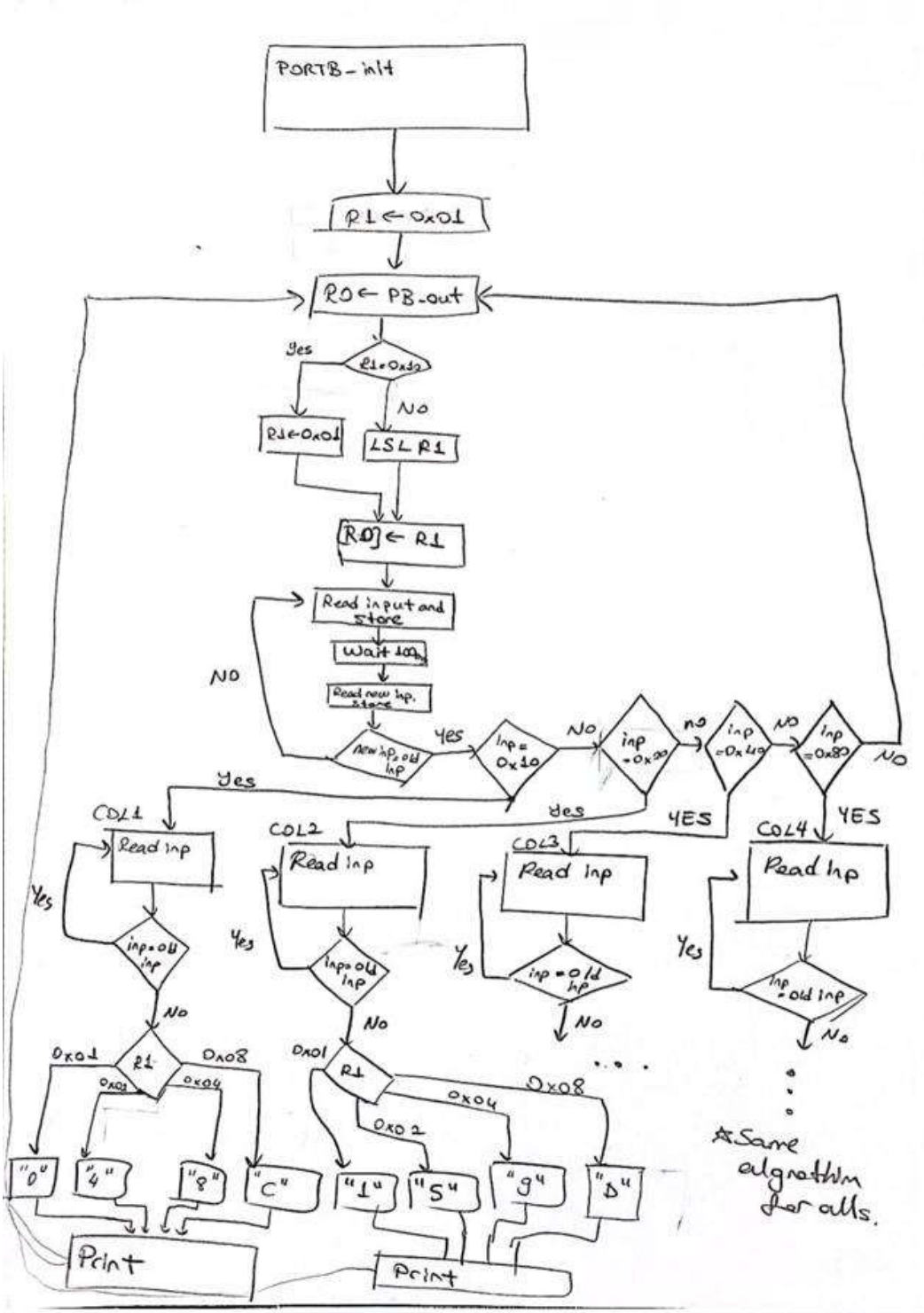
0100.0000 (0x40) -> there is a pushed button in column 3

1000.0000 (0x80) -> there is a pushed button in column 4

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

With bouncing effect, the result will not be acceptable. When we pushed a button, and if we do not put any delay, the result will be undetermined. It will do it for maybe lots of time since its value will be changed '1' and '0' very often. However, by putting little bit delay, bouncing effect can be omitted. After the first read in the code, it will go into DELAY100 subroutine. After that, it will read again and check if there is a change input data. If there is not any change it will do the rest of code. If there is, it will go into the first read part again. It can be seen in lines 17-23 in main function (programming_directives.s file).

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.


PORTB - init
R1 ← 0x01
R0 ← PB-out
Yes
R1=0x10
NO
R1←0x01
LSL R1
[R0] ← R1
Read input and store
Wait 100ms
Read new inp. store
NO
new inp = old inp
Yes
inp = 0x10
NO
inp = 0x20
NO
inp = 0x40
NO
inp = 0x80
NO
Yes
Yes
YES
YES
COL1
COL2
COL3
COL4
Read inp
Read inp
Read inp
Read inp
Yes
Yes
Yes
Yes
inp = old inp
inp = old inp
inp = old inp
inp = old inp
NO
NO
NO
NO
0x01
R1
0x08
0x02
0x04
0x01
R1
0x08
0x04
0x02
"0"
"4"
"8"
"C"
"1"
"5"
"9"
"D"
Print
Print
*Same algorithm for alls.

f) Implemented the developed algorithm in part-e by using assembly language

D:\OKUL\ee 4ün 1i\Lab-447\LabThree\questionthree\programming_directive.s

```
1   PB_OUT              EQU 0x4000503C          ;Address to output data
2   PB_INP              EQU 0x400053C0          ;Address to output data
3                       AREA        main, READONLY, CODE
4                       THUMB
5                       EXTERN      PORTB_INIT  ;To initialize port B
6                       EXTERN      OutChar     ;To print spesific Characters
7                       EXTERN      DELAY100    ;To create delay 100 msec
8                       EXPORT      __main      ;To make this script available
9
10  __main              PROC
11                      BL          PORTB_INIT
12                      MOV         R1,#0x01    ;R1 register determines which pin will be given high
    voltage
13  TRY                 LDR         R0,=PB_OUT
14                      CMP         R1,#0x10    ;When all 4 pins are given high voltages, it should
    return back into the original
15                      LSLNE       R1,#1       ;If it is not finished (NE), then it will go into next pin
16                      MOVEQ       R1,#0x1     ;If it is finished (EQ), then it will go into the
    starter pin
17  waitInp             STR         R1,[R0]     ;Give necessary voltage settings
18                      LDR         R0,=PB_INP
19                      LDR         R2,[R0]     ;Taking data from keys
20                      BL          DELAY100    ;Wait about 100 msec
21                      LDR         R3,[R0]     ;Taking another data from keys
22                      CMP         R3,R2       ;Then compare
23                      BNE         waitInp     ;If they are not equal, then go into waiting input stage
24
25                      ;If a key is preseed, R2 will be 0x10,0x20,0x40, or 0x80.
26                      CMP         R2,#0x10
27                      BEQ         COL1
28                      CMP         R2,#0x20
29                      BEQ         COL2
30                      CMP         R2,#0x40
31                      BEQ         COL3
32                      CMP         R2,#0x80
33                      BEQ         COL4
34                      ;If a key is not preseed in that corresponding rows, it will go into the next
    rows (SEE LINE 14)
35                      B           TRY
36
37  ;A KEY IN COLUMN1 HAS BEEN PRESSED
38  COL1                LDR         R4,[R0]     ;Take input data
39                      CMP         R2,R4       ;Compare with previous one
40                      BEQ         COL1        ;If it is still same, it means it is not released yet
41                      ;This part will determine which key is pressed by checking which pin is high
    voltage
42                      CMP         R1,#0x01    ;That means the first row (column 1)
43                      MOVEQ       R5,#0x30
44                      CMP         R1,#0x02    ;That means the second row (column 1)
45                      MOVEQ       R5,#0x34
46                      CMP         R1,#0x04    ;That means the third row (column 1)
47                      MOVEQ       R5,#0x38
48                      CMP         R1,#0x08    ;That means the forth row (column 1)
49                      MOVEQ       R5,#0x43
50                      BL          OutChar     ;Print the assigned value
51                      B           TRY         ;Go into the next iteration
52
53  ;A KEY IN COLUMN2 HAS BEEN PRESSED
54  COL2                LDR         R4,[R0]     ;Take input data
55                      CMP         R2,R4       ;Compare with previous one
56                      BEQ         COL2        ;If it is still same, it means it is not released yet
57                      ;This part will determine which key is pressed by checking which pin is high
    voltage
58                      CMP         R1,#0x01    ;That means the first row (column 2)
59                      MOVEQ       R5,#0x31
60                      CMP         R1,#0x02    ;That means the second row (column 2)
61                      MOVEQ       R5,#0x35
62                      CMP         R1,#0x04    ;That means the third row (column 2)
63                      MOVEQ       R5,#0x39
64                      CMP         R1,#0x08    ;That means the forth row (column 2)
65                      MOVEQ       R5,#0x44
66                      BL          OutChar     ;Print the assigned value
67                      B           TRY         ;Go to next iteration
68
69  ;A KEY IN COLUMN3 HAS BEEN PRESSED
70  COL3                LDR         R4,[R0]     ;Take input data
71                      CMP         R2,R4       ;Compare with previous one
```

```
 72                      BEQ         COL3        ;If it is still same, it means it is not released yet
 73                      ;This part will determine which key is pressed by checking which pin is high
    voltage
 74                      CMP         R1,#0x01    ;That means the first row (column 3)
 75                      MOVEQ       R5,#0x32
 76                      CMP         R1,#0x02    ;That means the second row (column 3)
 77                      MOVEQ       R5,#0x36
 78                      CMP         R1,#0x04    ;That means the third row (column 3)
 79                      MOVEQ       R5,#0x41
 80                      CMP         R1,#0x08    ;That means the forth row (column 3)
 81                      MOVEQ       R5,#0x45
 82                      BL          OutChar     ;Print the assigned value
 83                      B           TRY         ;Go to next iteration
 84
 85  ;A KEY IN COLUMN4 HAS BEEN PRESSED
 86  COL4                LDR         R4,[R0]     ;Take input data
 87                      CMP         R2,R4       ;Compare with previous one
 88                      BEQ         COL4        ;If it is still same, it means it is not released yet
 89                      ;This part will determine which key is pressed by checking which pin is high
    voltage
 90                      CMP         R1,#0x01    ;That means the first row (column 4)
 91                      MOVEQ       R5,#0x33
 92                      CMP         R1,#0x02    ;That means the second row (column 4)
 93                      MOVEQ       R5,#0x37
 94                      CMP         R1,#0x04    ;That means the third row (column 4)
 95                      MOVEQ       R5,#0x42
 96                      CMP         R1,#0x08    ;That means the forth row (column 4)
 97                      MOVEQ       R5,#0x46
 98                      BL          OutChar     ;Print the assigned value
 99                      B           TRY         ;Go to next iteration
100
101                     ALIGN
102                     ENDP
```

```
1   GPIO_PORTB_DIR      EQU 0x40005400      ;Data address to direction register
2   GPIO_PORTB_AFSEL    EQU 0x40005420
3   GPIO_PORTB_DEN      EQU 0x4000551C
4   GPIO_PORTB_PUR      EQU 0x40005510
5   GPIO_PORTB_PDR      EQU 0x40005514
6   SYSCTL_RCGCGPIO     EQU 0x400FE608
7                       AREA        main, READONLY, CODE
8                       THUMB
9                       EXTERN      DELAY100
10                      EXPORT      PORTB_INIT
11
12  PORTB_INIT          PROC;
13  Start               PUSH        {R0,R1}
14                      LDR         R1,=SYSCTL_RCGCGPIO
15                      LDR         R0,[R1]
16                      ORR         R0,R0,#0x02         ;Clock initializer for PORT B
17                      STR         R0,[R1]
18                      NOP
19                      NOP
20                      NOP                             ;let GPIO clock stabilize
21                      LDR         R1,=GPIO_PORTB_DIR  ;config for direction register 1 means output 0
    means input
22                      LDR         R0,[R1]
23                      MOV         R0,#0x0F            ;For pb0-3 = Output(L1,L2,L3,L4),
    pb4-7=Input(R1,R2,R3,R4)
24                      STR         R0,[R1]
25                      LDR         R1,=GPIO_PORTB_AFSEL
26                      LDR         R0,[R1]
27                      BIC         R0,#0xFF            ;No AFSEL for all pins
28                      STR         R0,[R1]
29                      LDR         R1,=GPIO_PORTB_DEN
30                      LDR         R0,[R1]
31                      ORR         R0,#0xFF            ;Digital enables for all pins
32                      STR         R0,[R1]
33                      LDR         R1,=GPIO_PORTB_PDR
34                      LDR         R0,[R1]
35                      ORR         R0,#0xF0            ;Pull down registers for Output pins
36                      STR         R0,[R1]
37                      POP         {R0,R1}
38                      BX          LR
39                      ENDP
40
```

After running these files, I pushed keys from KEY1 to KEY16. The results are shown below.