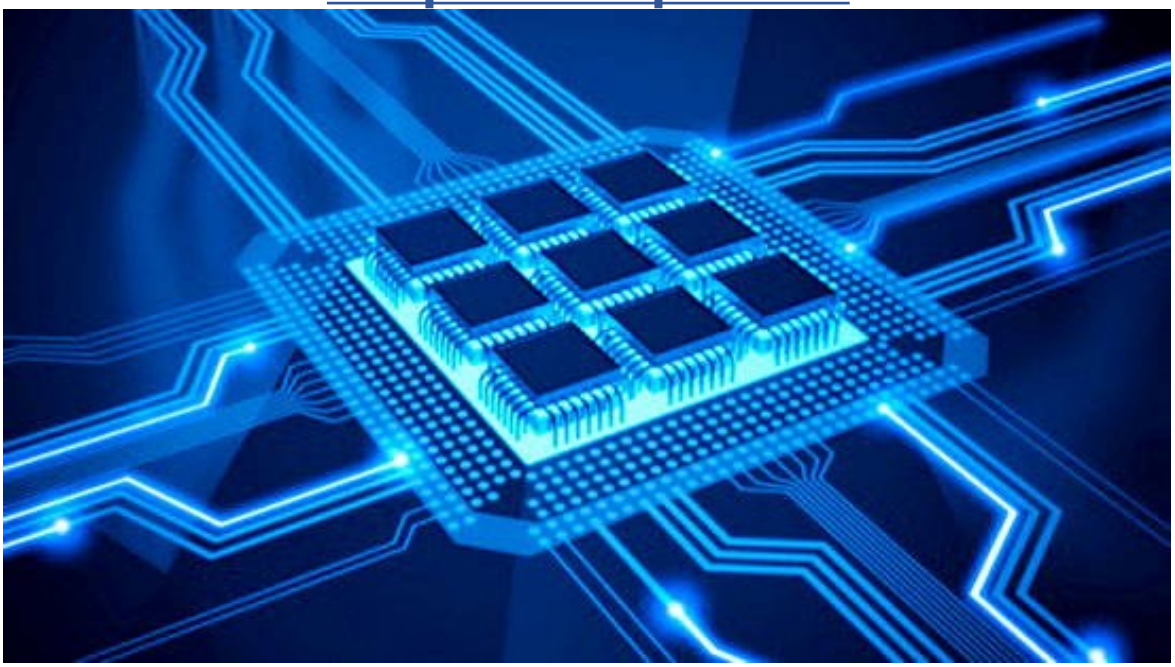
6η ΕΡΓΑΣΤΗΡΙΑΚΗ ΆΣΚΗΣΗ ΓΙΑ ΤΟ ΜΑΘΗΜΑ "Εργαστήριο Μικροϋπολογιστών"



Ομάδα: 16
Ημερομηνία επίδειξης: 29/11/2017
Μέλη:
Κερασιώτης Ιωάννης, Α.Μ.:03114951
Πευκιανάκης Κωνσταντίνος, Α.Μ.: 03114897
Ραφτόπουλος Ευάγγελος, Α.Μ.:03114743

ΑΣΚΗΣΗ 1:

Αρχικά ορίζουμε τις θύρες A,C ως εισόδους και την B ως έξοδο, αφού από την θύρα A θα διαβάζονται τα δεδομένα, η C θα αλλάζει την ένδειξη των LEDs και στις θέσεις 0-3 της B θα εμφανίζονται τα αποτελέσματα. Έπειτα διαβάζουμε την είσοδο A και σχηματίζουμε τις μεταβλητές χ0,χ1....χ7,y0,y1,y2,y3 με ολίσθηση της εισόδου και με την εφαρμογή κατάλληλης μάσκας.
Έτσι έχουμε τις μεταβλητές σε μορφή _ _ _ _ _ _ _ x όπου το χ είναι το λογικό 1 ή 0.
στην συνέχεια γίνονται οι λογικές πράξεις. Πρώτα γίνεται η PA0 XOR PA1. Στο αποτέλεσμα εφαρμόζουμε την μάσκα 0χ01 και συνεχίζουμε με τις υπόλοιπες λογικές πράξεις. Όταν σχηματίσουμε τα αποτελέσματα y0,y1,y2,y3 τότε θα πρέπει να τα διαμορφώσουμε κατάλληλα
ώστε να εμφανιστούν στις σωστές θέσεις της θύρας εξόδου. Έτσι θα κάνουμε τις κατάλληλες ολισθήσεις στα y0,y1,y2,y3 και θα τα συνδυάσουμε με ένα λογικό OR ως προς bit για να σχηματιστεί τελικά η έξοδος στην μορφή _ _ _ _ y3 y2 y1 y0.
Για την αντιστροφή της ένδειξης κάνουμε μια λογική πράξη XOR μεταξύ του αποτελέσματος και της εισόδου C. Έτσι θα αντιστραφούν τα LEDs στα οποία είναι πατημένο το αντίστοιχο κουμπί της εισόδου C.

Ο κώδικας της άσκησης δίνεται παρακάτω:

```
.include "m16def.inc"
.def x0=r16                ;arxikopoihsh ton kataxoriton pou xreisimopoioume
.def x1=r17
.def x2=r18
.def x3=r19
.def x4=r20
.def x5=r21
.def x6=r22
.def x7=r23
.def y0=r24
.def y1=r25
.def y2=r26
.def y3=r27
.def temp1=r28
.def temp=r29


ser temp                   ;h B os eksodos
out DDRB,temp
clr temp                   ;h A,C os eisodoi
out DDRA,temp
out DDRC,temp

loop:
clr temp                   ;eisagogh ton timon stis metablhtes
in temp,PINA               ;diabasma ths eisodou apo thn eisodo A
mov x0,temp                ;sto x0 bazo thn eisodo
andi x0,0x01               ;kai me thn maska 00000001 pairno to LSB
ror temp                   ;deksia olisthisi ths eisodou
mov x1,temp                ;omoios kai gia ta epomena
andi x1,0x01
ror temp
mov x2,temp
andi x2,0x01
ror temp
mov x3,temp
andi x3,0x01
ror temp
mov x4,temp
andi x4,0x01
ror temp
mov x5,temp
andi x5,0x01
ror temp
mov x6,temp
andi x6,0x01
ror temp
mov x7,temp
andi x7,0x01

eor x0,x1              ;ginetai h logikh praksh x0 = x0 xor x1, to apotelesma sto x0
```

```
andi x0,0x01          ;apomononoume to LSB
or x2,x3              ;x2 = x2 or x3
and x0,x2             ;x0= (x0 xor x1) and (x2 or x3)
mov y0,x0             ;to apotelesma sto y0
mov y1,x2             ;to x2 sto y1 to opoio theloume na emfanish
or x4,x5              ;x4=x4 or x5
com x4                ;x4= x4 nor x5
andi x4,0x01
mov y2,x4             ;to apotelesma sto y2
andi y2,0x01
eor x6,x7             ;x6=x6 xor x7
andi x6,0x01
com x6                ;x6=x6 nxor x7
andi x6,0x01
mov y3,x6             ;sto y3 to apotelesma
andi y3,0x01
bclr 0                ;h shmaia c=0
rol y1                ;mia olisthish gia na paei to y1 sthn sosth thesh
rol y2                ;dio olisthiseis gia na paei sthn sosth thesh to y2
rol y2
rol y3                ;tris olisthiseis gia ton idio logo
rol y3
rol y3
or y0,y1              ;sxhmatismos ths eksodou y0= y0 | y1 | y2
or y0,y2
or y0,y3
in temp,PINC          ;diabasma ths eisodou C

eor y0,temp           ;PINC xor y0, oste na antistrafei h endeiksh ton leds pou stis
                      ;antistoixes theseis  einai 1
out PORTB, y0         ;emfanish ths eksodou
jmp loop
```

ΑΣΚΗΣΗ 2:

Στην άσκηση αυτή ορίζουμε την θύρα A ως είσοδο και την C ως έξοδο. Έπειτα μπαίνει σε έναν ατέρμον βρόγχο όπου εκτελεί τις λογικές πράξεις. Πιο συγκεκριμένα, σχηματίζουμε τις μεταβλητές A,B,C,D,E από την είσοδο με κατάλληλη εφαρμογή μάσκας. Η A για παράδειγμα θα είναι της μορφής _ _ _ _ _ _ _ A, η B της μορφής _ _ _ _ _ _ B _ κλπ. Στην συνέχεια γίνονται οι λογικές πράξεις ανά λέξη (και όχι να bit) που ζητάει η άσκηση. Εδώ τονίζουμε ότι η πράξη & κάνει λογική πράξη AND ανά bit ενώ η && κάνει λογική πράξη AND ανά λέξη. Άρα το πρώτο μέρος της F0 θα είναι 1 μόνο όταν τα A,B,C είναι τα ίδια και 0 σε κάθε άλλη περίπτωση. Αφού σχηματιστεί το F0 (το οποίο θα έχει την τιμή 1 ή 0 στο LSB) θα το ολισθήσουμε 5 θέσεις αριστερά ώστε να πάει στην σωστή θέση της θύρας εξόδου που θέλουμε. Ομοίως γίνονται και οι υπόλοιπες πράξεις και ολισθήσεις. Τέλος όπως και στην άσκηση 1 κάνουμε το λογικό OR ανά bit μεταξύ των F0,F1,F2 ώστε να σχηματιστεί η έξοδος την οποία και εμφανίζουμε.

Ο κώδικας αυτής της άσκησης είναι ο εξής:

```
#include <avr/io.h>
```

```
int main(void) {

unsigned char input;
unsigned char A;
unsigned char B;
unsigned char C;
unsigned char D;
unsigned char E;
unsigned char F0;
unsigned char F1;
unsigned char F2;



DDRA = 0x00;              //orizoume thn thura A os eisodo
PORTA = 0x00;             //ta pull-up ths thuras eisodou A (den xreiazontai aparaithta)
DDRC = 0xFF;              //orizoume thn thura C os eksodo

while(1) {                 //atermon epanalhpsh
input = (PINA & 0x1F); //sto input exoume thn eisodo me maska 00011111 afou mas
                       //endiaferoun ta 5 prota bits
A= (input & 0x01);        //A=000000x
B= (input & 0x02);        //B=00000x0
C= (input & 0x04);        //C=0000x00
D= (input & 0x08);        //D=000x000
E= (input & 0x10);         //E=00x0000, opou x 0 h 1

F0 = ((A && B && C )) || (C && D) || (D && E); //h logikh praksh gia thn F0. otan exoume
                                               //&& kanei thn sugkrisi leksis. to F0 einai 1 h 0

F0 = !F0;                              //sumbhroma tou F0
F0 = F0 << 5;                           //5 deksies olisthiseis oste na bgei sthn eksodo
                                        //00x00000


F1=(A && B && C) || (!D && !E);   //h logikh praksh gia thn F1.
F1=F1<<6;                         //6 deksies olisthiseis oste na bgei sthn eksodo 0x000000

F2 = F0 || F1;                    //h logikh praksh gia thn F2.
F2=F2<<7;                         //7 deksies olisthiseis oste na bgei sthn eksodo x0000000
PORTC= F0 | F1 | F2;               //logikh praksh OR ana bit metaksi ton F0,F1,F2 oste na
                                   //emfanistei h eksodos sthn morfh F2 F1 F0 0 0 0 0 0
}
return 0;
}
```

ΑΣΚΗΣΗ 3:

Αφού ορίσουμε τις θύρες εισόδου, εξόδου και αρχικοποιήσουμε τον καταχωρητή του πληκτρολογίου, μπαίνουμε σε μία επανάληψη όπου περιμένουμε να πατηθεί πρώτα το 1ο κουμπί και μετά το 2ο. Για τον έλεγχο του αν πατήθηκε η ομάδα μας λειτουργούμε ως εξής. Αρχικοποιούμε μια σημαία flag με την τιμή FF. Αν δεν πατήθηκε το 1ο κουμπί της ομάδας μας τότε η σημαία γίνεται 0. αν στην συνέχεια το 2ο κουμπί δεν είναι της ομάδας μας τότε πάλι γίνεται 0. Αρα η μόνη περίπτωση η σημαία να παραμείνει 1 μετά το πάτημα των δύο κουμπιών είναι να έχει πατηθεί ο αριθμός της ομάδας μας. Στην συνέχεια αν πατήθηκε ο αριθμός της ομάδας μας τότε ανάβουμε τα LEDs και καλούμε μια χρονοκαθυστέρηση για 4 sec και μετά τα σβήνουμε. Αν όμως δεν πατήθηκε ο αριθμός της ομάδας μας τότε μπαίνουμε σε ένα βρόγχο με 8 επαναλήψεις όπου ανάβουμε τα LEDs για 0.25 sec και μετά τα σβήνουμε για 0.25 sec. Αυτό θα επαναληφθεί 4 φορές.

Έχουμε χρησιμοποιήσει τις ρουτίνες που μας δίνονται για την χρήση του πληκτρολογίου και χρονοκαθυστέρησης. Επιπλέον έχουμε ορίσει και μια άλλη wait_for_keypad η οποία περιμένει να πατηθεί κάποιο κουμπί για να συνεχίσει και να επιστρέψει ως έξοδο το κουμπί που πατήθηκε.

Ο κώδικας της άσκησης δίνεται παρακάτω:

```
.include "m16def.inc"
.def temp = r16
.def counter = r17
.def flag = r18

.DSEG
_tmp_: .byte 2

.CSEG
rjmp main


main:
        ldi temp, low(RAMEND)           ;orosmos tou deikth ths stoibas afou tha ginei klhsh
                                        ;routinon
        out spl, temp
        ldi temp, high(RAMEND)
        out sph, temp
        ldi temp, (1<<PC7)|(1<<PC6)|(1<<PC5)|(1<<PC4) ;eksodoi ta 4 MSB, eisodoi ta 4
                                                      ;LSB
        out DDRC, temp
        clr temp
        out PORTC, temp                 ; apenergopoihsh pull-up antistasewn
        ser temp
        out DDRB, temp                  ; PORTB os eksodos
        rcall scan_keypad_rising_edge   ; klhsh gia thn arxikopoihsh tou _tmp_ se 0000

eternal_loop:
        ser flag                        ; flag <- FF

        rcall wait_for_keypad           ; if first key pressed is 1 then flag <- 00
        sbrs r25, 4
```

```
        clr flag

        rcall wait_for_keypad                   ; if second key pressed is 6 then flag <- 00
        sbrs r25, 2
        clr flag

        tst flag                        ; if flag=00, den patithike o arithmos ths omadas mas
                                        ;(16) allios patithike
        breq incorrect_password

correct_password:
        rcall leds_on                   ; anamma twn leds ths thyras PORTB

        ldi r24, low(4000)              ; xronokathisterisi 4 seconds = 4000ms
        ldi r25, high(4000)
        rcall wait_msec

        rcall leds_off
        rjmp eternal_loop               ; synexis leitourgeia

incorrect_password:
        ldi counter, 8                  ; orismos plithous epanalipsewn sto 8

blink_loop:
        ldi r24, low(250)
        ldi r25, high(250)
        rcall leds_on                   ; opote, ektelountai 8 epanalhpseis pou h
                                        ;kathemia
        rcall wait_msec                 ; krataei 0.25+0.25=0.5 sec, ara synolikos xronos
                                                ;8*0.5=4 sec


        rcall leds_off
        ldi r24, low(250)
        ldi r25, high(250)
        rcall wait_msec
        dec counter
        brne blink_loop

        rjmp eternal_loop               ; synexis leitourgeia

wait_usec:
        sbiw r24 ,1
        nop
        nop
        nop
        nop
        brne wait_usec
        ret

wait_msec:
        push r24
        push r25
```

```
        ldi r24 , low(998)
        ldi r25 , high(998)
        rcall wait_usec
        pop r25
        pop r24
        sbiw r24 , 1
        brne wait_msec
        ret

scan_row:
        ldi r25 , 0x08
back_:
        lsl r25
        dec r24
        brne back_
        out PORTC , r25
        nop
        nop
        in r24 , PINC
        andi r24 ,0x0f
        ret

scan_keypad:
        ldi r24 , 0x01
        rcall scan_row
        swap r24
        mov r27 , r24
        ldi r24 ,0x02
        rcall scan_row
        add r27 , r24
        ldi r24 , 0x03
        rcall scan_row
        swap r24
        mov r26 , r24
        ldi r24 ,0x04
        rcall scan_row
        add r26 , r24
        movw r24 , r26
        ret

scan_keypad_rising_edge:
        mov r22 ,r24
        rcall scan_keypad
        push r24
        push r25
        mov r24 ,r22
        ldi r25 ,0
        rcall wait_msec
        rcall scan_keypad
        pop r23
        pop r22
        and r24 ,r22
```

```
        and r25 ,r23
        ldi r26 ,low(_tmp_)
        ldi r27 ,high(_tmp_)
        ld r23 ,X+
        ld r22 ,X
        st X ,r24
        st -X ,r25
        com r23
        com r22
        and r24 ,r22
        and r25 ,r23
        ret

leds_on:
        ser temp
        out PORTB, temp
        ret
leds_off:
        clr temp
        out PORTB, temp
        ret

wait_for_keypad:
        ldi r24, 20
        rcall scan_keypad_rising_edge
        tst r24
        brne next1
        tst r25
        breq wait_for_keypad
next1:
        ret
```