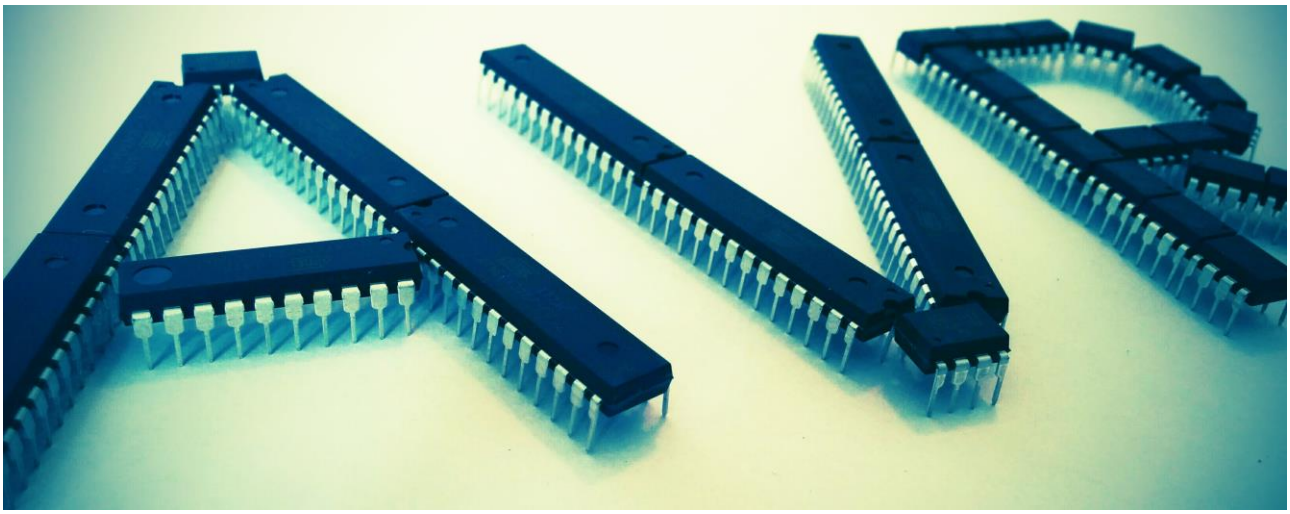
ΣΗΜΜΥ ΕΜΠ
Συστήματα Μικροϋπολογιστών / AVR lab / Team82
Πέτρος Έλληνας / el18702
Νικόλαος Πηγαδάς / el18445
2020-2021



Άσκηση 1

```
; Authors : Petros Ellhnas, Nicolas Pigadas
 ; Team82
 ; AM            18702           18445

ldi r24 , low(RAMEND)  ; Arxikopoihsh stack pointer
out SPL , r24
ldi r24 , high(RAMEND)
out SPH , r24

ser r25                ; Thetoyme mono assous ston r25
out DDRA,r25           ; Xrhsh PORTA ws output
clr r25                ; Thetoume mono mhdenika ston r25
out DDRB,r25           ; Xrhsh PORTB ws input

init:
       ldi r28,0x00000000 ; Arxikopoihsh tou PB0 me 0, afoy den einai pathmeno arxika kai
ksekinaei to rotation
       out PORTB,r28
       ldi r28,0x00000001 ; Arxikopoihsh lampakiwn sto led0
       out PORTA,r28
       ldi r29,0x00       ; Flag phgainw apo lsb se msb h antitheta?
       nop                ; ~250 ms delayn x2
       nop

loop:
       sbic PINB,0x00     ; skip the next instr if PB0=0
       rjmp stay          ; else, make no move
       rjmp cont          ; if you skipped, let's rotate

cont:
       cpi r28,0x80       ; An eimaste sto led7 pame apo ta msb sta lsb
       breq msblsb
       cpi r28,0x01       ; An eimaste sto led0 pame apo ta lsb sta msb
       breq lsbmsb
       rjmp ipologise     ; An eisai se endiameso, exontas to FLAG, kane swsto rotation

lsbmsb:
       ldi r29,0x00       ; FLAG = 0 shmainei oti exw kateythynsh apo ta lsb sta msb
    rjmp ipologise     ; Hrthes edw epeidh allakses poreia, eftiakses to flag, ksekina to
rotation

msblsb:
       ldi r29,0x01       ; FLAG = 1 shmainei oti exw kateythynsh apo ta msb sta lsb

ipologise:
       cpi r29,0x01       ; An FLAG == 1 tote kane rotation apo ta msb pros ta lsb (lsr) (****)
    breq piso
       lsl r28            ; An FLAG == 0 tote kane rotation apo ta lsb pros ta msb (lsl)
                             ; An synexiseis apo edw kai katw phgaine na anapseis to left-
rotated led

output:
        out PORTA,r28  ; Anavei to rotaded led
              nop            ; ~250 ms delay x2
              nop
              rjmp loop      ; Epanalave

piso:
       lsr r28         ; (****)
    rjmp output        ; Phgaine na anapseis to right-rotated led

stay:                  ; Enallages etiketwn loop kai stay mexri to PB0 na vrethei na mhn einai
pathmeno
       ; out PORTA, r28   ; Den xreiazetai output epeidh einai hdh anammeno to led gia thn
katastash stay kai to afhnoume ws exei
                                          rjmp loop
```

Άσκηση 2

```
/*
 * Authors : Petros Ellhnas, Nicolas Pigadas
 * Team82
 * AM            18702           18445
 */
#include <avr/io.h>
char x,A,B,C,D,F0=0x00,F1=0x00,N=0x00; // Arxikopoihsh eisodwn kai eksodwn twn logikwn
synarthsewn

int main(void)
{
            DDRB = 0xFF; // use PORTB as output
            DDRA = 0x00; // use PORTA as input

      while (1)
    {
            x = PINA & 0x0F;   // Apomonwsh 4 LSB stoixiwn ths thyras eisodou, x =
0000 WXYZ
                              // Katopin antistoixoume kathe pshfio 0-3 se kathe bit
ths thyras eisodou A-D kata antistoixia
                                        // kai olisthsh olwn sthn prwth thesh
            A = x & 0x01;   // A = 0000 000Z
            B = x & 0x02;   // B = 0000 00Y0
            B = B>>1;       // B = 0000 000Y
            C = x & 0x04;   // C = 0000 0X00
            C = C>>2;       // C = 0000 000X
            D = x & 0x08;   // D = 0000 W000
            D = D>>3;       // D = 0000 000W

            // Logikes synarthseis
            F0 = !((A & B & !C) | (C & D));

            F1 = ((A | B) & (C | D));
            F1 = F1<<1;

            // Apotelesma
            N = F0 + F1;

            // Emfanish kai katopin epanelave
            PORTB = (N);
    }
}
```

Άσκηση 3

```
/*
 * Authors : Petros Ellhnas, Nicolas Pigadas
 * Team82
 * AM            18702           18445
 */

#include <avr/io.h>

int main(void)
{
        DDRA=0XFF; //Output
        DDRC=0X00; //Input
        char Current_state=0x01; //Arxikopoihsh twn leds me anamma tou prwtou
        PORTA = Current_state;   //dhladh ths theshs 0 ths thyras eksodou PORTA

    while (1)
    {
                //H antistoixia twn diakoptwn SWx sta bit ths thyras eisodou PORTC
                //SW0 = (PINC & 0x01);
                //SW1 = (PINC & 0x02);
                //SW2 = (PINC & 0x04);
                //SW3 = (PINC & 0x08);

                //SW0
                if((PINC & 0x01)==1){                  //Elegxos pathmatos push-button SW0
                        while((PINC & 0x01)==1);         //Elegxos epanaforas push-button SW0
                                if(Current_state==0x80)     //An eimaste sto PA7, synexizoume apo to
PA0
                                        Current_state=0x01;
                                else                        //Alliws kanoume olisthsh pros ta aristera
                                        Current_state=Current_state << 1;

                                PORTA = Current_state;  //Emfanish sthn eksodo
                }

                //SW1
                if((PINC & 0x02)==2){                 //Elegxos pathmatos push-button SW1 me 2
(maska 10 sto dyadiko)
                        while((PINC & 0x02) == 2);      //Elegxos epanaforas push-button SW1
                                if(Current_state==0x01)    //An eimaste sto PA0, synexizoume apo to
PA7
                                        Current_state=0x80;
                                else                        //Alliws kanoume olisthsh pros ta deksia
                                        Current_state=Current_state >> 1;

                                PORTA = Current_state;          //Emfanish sthn eksodo
                }

                //SW2
                if((PINC & 0x04)==4){                //Elegxos pathmatos push-button SW2 me 4
(maska 100 sto dyadiko)
                        while((PINC & 0x04)==4);       //Elegxos epanaforas push-button SW2
                                Current_state=0x80;       //Metakihsh tou anammenou led sthn thesh
MSB(led7)
                                PORTA=Current_state;
                }

                //SW3
                if((PINC & 0x08)==8){                //Elegxos pathmatos push-button SW3 me 8
(maska 1000 sto dyadiko)
                        while((PINC & 0x08)==8);       //Elegxos epanaforas push-button SW3
                                Current_state=0x01;       //Metakihsh tou anammenou led sthn thesh
LSB(led0)
                                PORTA=Current_state;
                }
    }
}
```

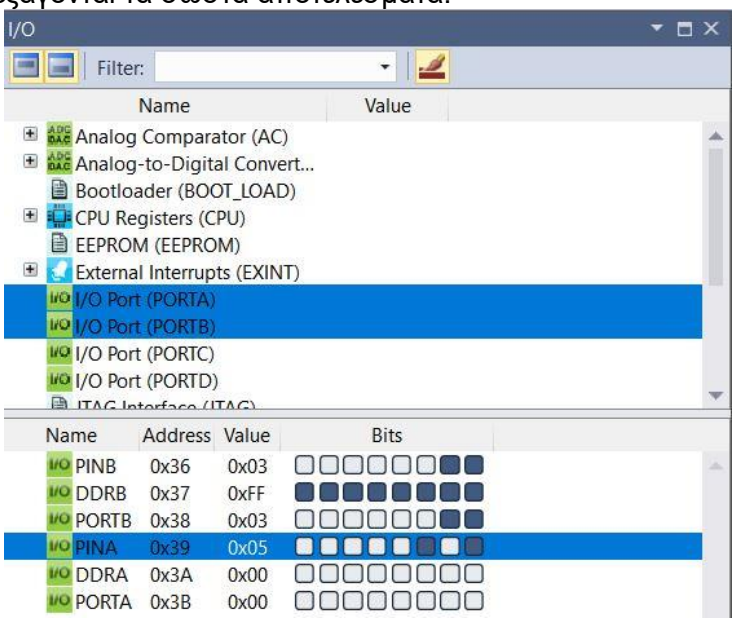
Σχόλια

Άσκηση 1

Εδώ ας αναφέρουμε ότι αρχικοποιούμε την στοίβα ως καλή τακτική για προγραμματισμό AVR σε ASSEMBLY, όπως αναφέρθηκε και στο εργαστήριο. Εδώ, όμως, δεν χρειάζεται να σώσουμε τίποτα στην στοίβα, καθώς δεν κάνουμε χρήση διακοπών ή ρουτινών. Συγκεκριμένα, δεν χρειάζεται να αποθηκεύσουμε τον program counter, δηλαδή διευθύνσεις επιστροφής, γιατί εδώ γνωρίζουμε συνεχώς την ροή εκτέλεσης του προγράμματος. Η ροή εκτέλεσης είναι σειριακή και αλλάζει μόνο με τα jumps, μετά τα οποία συνεχίζει σειριακά και δεν υπάρχει η απαίτηση να επιστρέψουμε σε προηγούμενη ετικέτα, παρά μόνο με ακόμα ένα jump.

Άσκηση 2

Εδώ αξίζει να τονιστεί το γεγονός ότι για να αναπαραστήσουμε τις θύρες κάνουμε χρήση chars. Κάναμε αυτήν την επιλογή, επειδή δεσμεύουν 1 byte οπότε κάνουμε εξοικονόμηση χώρου, ενώ ταυτόχρονα βολεύουν για την χρήση των bitwise operators σύγκρισης που χρησιμοποιήσαμε έναντι των logical. Επιλέξαμε τους πρώτους επειδή εξετάζοντας το πρόγραμμα σε πιο αυστηρά πλαίσια, οι bitwise δίνουν αποτελέσματα 0 ή 1, ενώ οι logical true ή false. Και στις δύο περιπτώσεις, όμως, εξάγονται τα σωστά αποτελέσματα.



Παράδειγμα εκτέλεσης με A = 1, B = 0, C = 1, D = 0. Τα αποτελέσματα είναι τα επιθυμητά, αφού:

F0 = (ABC'+CD)' = 1

F1 = (A+B)(C+D) = 1

Και από το σχήμα:

PB0 = 1

PB1 = 1

Άσκηση 3

Η προσομοίωση, η οποία τρέχει βήμα βήμα, είναι παραπλανητική, αφού ενώ πατάμε και αφήνουμε το LSB του input, δεν μπαίνει σε κάποιον έλεγχο αν εκείνη την στιγμή το βήμα είναι παγωμένο σε έναν άλλο έλεγχο του κώδικα. Οπότε, για να το τρέξουμε πρέπει να κάνουμε όλα τα βήματα μέχρι να φτάσουμε στον επιθυμητό έλεγχο. Στην πραγματικότητα, βέβαια, αυτό ακριβώς γίνεται, καθώς οι εντολές αυτές εκτελούνται ασύγκριτα πιο γρήγορα από το πάτημα του κουμπιού. Εδώ ας επισημανθεί ότι γίνονται συνεχώς έλεγχοι, γεγονός που ωθεί στην σπατάλη υπολογιστικής ισχύος, οπότε η χρήση διακοπών θα συνέφερε.

Ας επισημανθεί ακόμα ότι με τις εντολές `while((PINC & 0x0x)==x);` γίνεται ο έλεγχος για το αν αφέθηκε το κουμπί, ώστε να γίνει η ζητούμενη ενέργεια. Πιο αναλυτικά, αν έχουμε περάσει τον έλεγχο ότι πατήθηκε ένα από τα ενεργά κουμπιά, κάνουμε ατέρμονη επανάληψη μέχρι αυτό να αφεθεί.