



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΜ&ΜΥ
Εργαστήριο Μικροϋπολογιστών

5^η Εργαστηριακή Άσκηση
Ακ. έτος 2011-2012

Ομάδα C07:

Ελένη Ευαγγελάτου	A.M.: 03108050
Γρηγόρης Λύρας	A.M.: 03109687
Βασιλεία Φραγκιαδάκη	A.M.: 03108026

29 Ιανουαρίου 2012

Άσκηση (i)

Το πρόγραμμά μας απεικονίζει ένα αναμμένο led, το οποίο έχει την δυνατότητα να μετακινείται πάνω στα led PA0-PA7. Η φορά κίνησης είναι από τα αριστερά προς τα δεξιά, και αντίστροφα για την περίπτωση που φτάσει σε ένα από τα δύο ακραία leds (PA0 ή PA7). Η περίπτωση αυτή εξετάζεται στα σημεία cpi r26, 128 (το οποίο είναι 2^7) και τότε πάμε στο up_lim και cpi r26, 1 απότε πάμε στην label down_lim. Η μετακίνηση για το επόμενο led είναι στο next, όπου άμα γίνεται, πηγαίνουμε δεξιά ή αριστερά το αναμμένο led. (Labels right και left αντίστοιχα). Η διάρκεια ανάμματος του led είναι 0.5 sec και αυτό επιτυγχάνεται βάζοντας στην wait_msec "όρισμα" 500, στους καταχωρητές δηλαδή r24 και r25, ώστε να έχουμε καθυστέρηση $500 * 1 \text{ msec} = 0.5 \text{ sec}$. Η κίνηση είναι ελεγχόμενη από το push button PB0, που άμα πατηθεί σταματάει την κίνηση πάνω στα leds. Ο συγκεκριμένος έλεγχος γίνεται μέσα στην wait_msec, όπου διαβάζουμε από το PINB, και ελέγχουμε αν έχουμε carry, οπότε και πρέπει να πάμε στην wait_msec. Κυρίως κώδικας:

```
1  #include <avr/io.h>
2  #define __SFR_OFFSET 0
3
4  .global main
5  main:
6  reset:
7      ldi r24,lo8(RAMEND)
8      out SPL,r24
9      ldi r24,hi8(RAMEND)
10     out SPH,r24
11     ser r24
12     out DDRA,r24
13     clr r26
14     clr r27
15     out DDRB,r27
16     clr r28
17     inc r26
18  calc:
19     out PORTA, r26
20     ldi r24,lo8(500)
21     ldi r25,hi8(500)
22     rcall wait_msec
23
24     cpi r26,128
25     breq up_lim
26     cpi r26,1
27     breq down_lim
28     rjmp next
29
30  up_lim:
31     ldi r28,1
32     rjmp next
33  down_lim:
34     ldi r28,0
35     rjmp next
36
37  next:
38     cpi r28,0
39     breq left
40     rjmp right
41  right:
42     ror r26
43     rjmp calc
44  left:
45     rol r26
46     rjmp calc
47
48  wait_usec:
49     sbiw r24,1
50     nop
51     brne wait_usec
52     ret
53
54  wait_msec:
55     in r27,PINB
56     ror r27
57     brcs wait_msec
58     push r24
```

```

59     push r25
60     ldi r24,lo8(998)
61     ldi r25,hi8(998)
62     rcall wait_usec
63     pop r25
64     pop r24
65     sbiw r24,1
66     brne wait_msec
67     ret

```

Άσκηση (ii)

Σε αυτό το μέρος ο σκοπός μας είναι να αναβοσβήνουν τα leds PA0 έως PA7 αλλά με συγκεκριμένη καθυστέρηση, την οποία δίνει ο χρήστης μέσω των dip switches B του AVR. Για τον σκοπό αυτό με την ρουτίνα `read_delay_A` διαβάζουμε και παίρνουμε τα 4 least significant bits και έπειτα προσθέτουμε 1 και και πολλαπλασιάζουμε επί 100 κατά την δοθείσα σχέση $Delay = (1 + x) * 100$. Για να διαβάσουμε την καθυστέρηση για το σβήσιμο διαβάζουμε όπως πριν μόνο που κάνουμε shift τέσσερις θέσεις αριστερά ώστε να απομονώσουμε τα επιθυμητά bits και να εφαρμόσουμε την ίδια σχέση. Η κλήση για τις καθυστερήσεις είναι φυσικά στο κυρίο πρόγραμμα, αμέσως μετά την κλήση για άναμμα ή σβήσιμο αντίστοιχα. Κυρίως κώδικας:

```

1  /*
2   * AVRAssembler1.asm
3   *
4   * Created: 23/1/2012 10:02:18 ??
5   * Author: Eleni
6   */
7
8  ;.INCLUDE "m16def.inc"
9  #include <avr/io.h>
10 #define __SFR_OFFSET 0
11
12 .global main
13
14 main:
15     ldi r24,lo8(RAMEND)
16     out SPL,r24
17     ldi r24,hi8(RAMEND)
18     out SPH,r24
19
20     ser r26                ;arxikopoihsh ths PORTA ; ser = set register
21     out DDRA, r26          ;gia eksodo
22     ;arxikopoihsh gia eisodo:
23     clr r25                ;bawz mhdenika gia na exoume eisodo s ola ta bits
24     out DDRB, r25          ;giati selw apo ta B switches
25
26 flash:
27     in r28, PINB           ;diabazei kia ta grafei ston r28
28
29     rcall on                ;anapse ta LEDS
30     rcall delay1            ;diabazw delay gia anamma
31     rcall wait_msec
32
33
34     rcall off               ;sbhse ta LEDS
35     rcall delay2            ;diabazw delay gia sbhsimo
36     rcall wait_msec
37
38     rjmp flash              ; do it again
39
40
41 ;Yporoutina gia na anaboun ta leds
42
43 on: ser r26                ;8ese thn 8ura eksodou tw n LED
44     out PORTA, r26
45     ret                    ;gurise sto kurio programma
46
47 ;Yporoutina gia na sbhnoun ta leds
48
49 off: clr r26               ;mhdenise thn 8ura eksodou tw n LED
50     out PORTA, r26
51     ret                    ;gurise sto kuriws programma
52
53 ;===== delay1 gia anamma=====

```

```

54
55 delay1:
56     ldi r24, 0x0F
57     and r24, r28           ;gia na parw ta lsbs
58     ldi r23, 01           ;gia na pros8esw 1
59     add r24, r23
60     ldi r23, 0x64         ;gia na pollaplasiasw epi 100
61     mul r24, r23          ;kanw ton pollaplasiasmo k exw etoimh thn ka8usterhsh
62                             ;ston diplo r0:r1
63     mov r25, r1           ;metaferw ta msbs ston r25 gia na ta parei wait_msec
64     mov r24, r0           ;metaferw ta lsbs ston r24 gia na ta parei wait_msec
65
66     ret
67
68 ;=====delay2 gia sbhsimo=====
69
70 delay2:
71     ldi r24, 0xF0
72     and r24, r28           ;gia na parw ta msbs
73     lsr r24                ;4 shift pros ta aristera gia na pane stis swstes 8eseis
74     lsr r24
75     lsr r24
76     lsr r24
77
78
79     ldi r23, 0x01         ;gia na pros8esw 1
80     add r24, r23
81     ldi r23, 0x64         ;gia na pollaplasiasw epi 100
82     mul r24, r23          ;kanw ton pollaplasiasmo k exw etoimh thn ka8usterhsh ston diplo r0:r1
83     mov r25, r1           ;metaferw ta msbs ston r25 gia na ta parei wait_msec
84     mov r24, r0           ;metaferw ta lsbs ston r24 gia na ta parei wait_msec
85
86     ret
87
88 ;=====wait_msec kai wait_usec (etoimh apo thn ekfwnhsh)=====
89
90 wait_msec:
91     push r24
92     push r25
93     ldi r24, lo8(998)
94     ldi r25, hi8(998)
95     rcall wait_usec
96     pop r25
97     pop r24
98     sbiw r24, 1
99     brne wait_msec
100
101     ret
102
103 wait_usec:
104     sbiw r24,1
105     nop
106     nop
107     nop
108     nop
109     brne wait_usec
110
111     ret


```

Άσκηση (iii)

Ζητείται να γράψουμε ένα πρόγραμμα σε C που να ανάβει το led0 που είναι συνδεδεμένο στο bit0 της θύρας εξόδου PORTB το οποίο να μετακινείται κατάλληλα ανάλογα με τα push buttons που πατάμε (SW0-4 της PORTD). Μεγαλύτερη προτεραιότητα έχουν τα MSB buttons. Γι' αυτό το σκοπό περιμένουμε να γίνει 1 κάποιο button και στη συνέχεια μπαίνει σε loop μέχρι να γίνει μηδέν. Μετά εκτελεί τη μετακίνηση που αντιστοιχεί σε αυτό. Εκτός αν πατηθεί ωστόσο button υψηλότερης προτεραιότητας οπότε και πάμε στο αντίστοιχο loop που περιμένουμε να γίνει 0. Για τις διάφορες μετακινήσεις κάναμε και τις απαραίτητες διορθώσεις στην τιμή του led, ώστε να δείχνει την τιμή που περιμένουμε μετά τις διάφορες ολισθήσεις.

Κυρίως κώδικας:

```

1   /*
2  * part3.c
3  *

```

```

4  * Created: 22/1/2012 7:55:21
5  * Author: Valia
6  */
7
8  #include <avr/io.h>
9
10 int main(void)
11 {
12     //while(1)
13     //{
14         //TODO:: Please write your application code
15         DDRB = 0xff; //output
16         DDRD = 0x00; //input
17         PORTB = 0x01;
18         while(1){
19             if ((PIND & 1) == 0x01){
20                 while (((PIND & 1) != 0x00) && (PIND <= 0x01 )) ; //perimenei mexri na ginei 0 h feugei an
21                 if ((PIND & 1) == 0x00){ //button ypsilo
22                     if ((PORTB << 1) == 0x100) PORTB = 0x01;
23                     else PORTB = PORTB << 1;
24                 }
25             }
26
27             if ((PIND & 2) == 0x02){
28                 while (((PIND & 2) != 0x00) && (PIND <= 0x03 )) ; //perimenei
29                 if ((PIND & 2) == 0x00){
30                     if ((PORTB >> 1) == 0)
31                         PORTB = 0x80; //128 dec
32                     else PORTB = PORTB >> 1;
33                 }
34             }
35
36             if ((PIND & 4) == 0x04){
37                 while (((PIND & 4) != 0x00) && (PIND <= 0x07 )) ; //perimenei
38                 if ((PIND & 4) == 0x00){
39                     if ((PORTB << 1) == 0x100)
40                         PORTB = 0x02; //dior8wseis
41                     else if ((PORTB << 2) == 0x100) PORTB = 0x01;
42                     else PORTB = PORTB << 2;
43                 }
44             }
45
46             if ((PIND & 8) == 0x08){
47                 while (((PIND & 8) != 0x00) && (PIND <= 0x0F )) ; //perimenei
48                 if ((PIND & 8) == 0){
49                     if ((PORTB >> 1) == 0)
50                         PORTB = 0x40;
51                     else if ((PORTB >> 2) == 0)
52                         PORTB = 0x80;
53                     else PORTB = PORTB >> 2; //dior8wseis
54                 }
55             }
56
57             if ((PIND & 16) == 16){
58                 while ((PIND & 16) != 0x00) ; //perimenei
59                 PORTB = 0x01;
60             }
61         }
62     return 0;
63 }

```