



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΜ&ΜΥ
Εργαστήριο Μικροϋπολογιστών

7^η Εργαστηριακή Άσκηση
Ακ. έτος 2011-2012

Ομάδα C07:

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

18 Φεβρουαρίου 2012

Άσκηση (i)

Σε αυτή την άσκηση ζητείται να υλοποιήσουμε πέντε λογικές πύλες διαβάζοντας την είσοδο από την πόρτα A ενώ η έξοδος φαίνεται στα τέσσερα LSB leds της πόρτας B. Για το σκοπό αυτό διαβάζουμε ανά δύο τα bits της εισόδου και υλοποιούμε τις αντίστοιχες λογικές συναρτήσεις αποθηκευόντάς τα παράλληλα σε ένα καταχωρητή. Ακόμη ζητείται να αντιστρέφονται τα αντίστοιχα leds με τα push-buttons PC0-7. Γι' αυτό το σκοπό, πριν διαβάσουμε την είσοδο από την πόρτα A, διαβάζουμε το PINC και ελέγχουμε κάθε φορά ποια push-buttons πατήθηκαν και τα “κρατάμε” σε έναν καταχωρητή. Στο τέλος, πριν εμφανίσουμε το αποτέλεσμα στην έξοδο, κάνουμε λογικό xor μεταξύ των δύο καταχωρητών ώστε να αντιστρέψουμε τις τιμές των leds που ενεργοποιήθηκαν και να αφήσουμε ως έχουν τα υπόλοιπα.

Κυρίως κώδικας:

```
1  /*Includes for compatibility with GNU toolchain*/
2  #define __SFR_OFFSET 0
3  #include <avr/io.h>
4  #include <avr/interrupt.h>
5  .global main
6  /*
7   * AVR_3h_ask1.asm
8   *
9   * Created: 10/2/2012 10:01:52 ??
10  * Author: Valia
11  */
12
13
14  /*
15  * AVR_ask1.asm
16  *
17  * Created: 10/2/2012 9:24:48 ??
18  * Author: Valia
19  */
20
21  #define temp1 r24
22  #define input r28
23  #define output r27
24  #define temp2 r25
25  #define regC r29
26  #define tempo r26
27  #define output1 r30
28
29  main:
30      ldi temp1, hi8(RAMEND)
31      out SPH, temp1
32      ldi temp1, lo8(RAMEND)
33      out SPL, temp1
34
35      ser temp1
36      out DDRB, temp1
37
38      clr temp1
39      out DDRD, temp1
40
41      clr regC
42      out DDRC, regC
43  start:
44      clr output1
45      in regC, PINC
46
47      lsr regC
48      brcc syn1
49  etik1:    ;//perimenei mexri na ginei 0 (button)
50      in tempo, PINC
51      lsr tempo
52      ;brcc syn1
53      ori output1, 1
54
55  syn1:
56      lsr regC
57      brcc syn2
58  etik2:    ;//perimenei mexri na ginei 0 (button)
59      in tempo, PINC
60      lsr tempo
61      lsr tempo
```

```

62     ;brcc syn2
63     ori output1, 2
64
65 syn2:
66     lsr regC
67     brcc syn3
68 etik3:    ;//perimenei meaxri na ginei 0 (button)
69     in tempo, PINC
70     lsr tempo
71     lsr tempo
72     lsr tempo
73     ;brcc syn3
74     ori output1, 4
75
76 syn3:
77     lsr regC
78     brcc syn4
79 etik4:    ;//perimenei meaxri na ginei 0 (button)
80     in tempo, PINC
81     lsr tempo
82     lsr tempo
83     lsr tempo
84     lsr tempo
85     ;brcc syn4
86     ori output1, 8
87
88 syn4:
89     lsr regC
90     brcc syn5
91 etik5:    ;//perimenei meaxri na ginei 0 (button)
92     in tempo, PINC
93     lsl tempo
94     lsl tempo
95     lsl tempo
96     lsl tempo
97     ;brcc etik5
98     ori output1, 0x010
99
100 syn5:
101     lsr regC
102     brcc syn6
103 etik6:    ;//perimenei meaxri na ginei 0 (button)
104     in tempo, PINC
105     lsl tempo
106     lsl tempo
107     lsl tempo
108     ;brcc etik6
109     ori output1, 0x020
110
111 syn6:
112     lsr regC
113     brcc syn7
114 etik7:    ;//perimenei meaxri na ginei 0 (button)
115     in tempo, PINC
116     lsl tempo
117     lsl tempo
118     ;brcc etik7
119     ori output1, 0x040
120
121 syn7:
122     lsr regC
123     brcc synexise
124 etik8:    ;//perimenei meaxri na ginei 0 (button)
125     in tempo, PINC
126     lsl tempo
127     ;brcc etik8
128     ori output1, 0x080
129
130 synexise:
131
132     in input, PIND
133     clr temp2 ;//mhdenismos ezodun
134     clr output
135
136 gate_1:

```

```
1  /* .....
2
3  * File Name : part2.c
4
5  * Purpose :
```

Στο μέρος αυτό υλοποιούμε τις συναρτήσεις $F0 = (AB+BC+CD+DE)$, $F1 = ABCD+E$, $F2 = F0+F1$. Χρησιμοποιώντας τις bitwise λογικές πράξεις της C, έχουμε τα εξής: Για την $F0$ θέλουμε οποιαδήποτε 2 συνεχόμενα bits της θύρας εισόδου να είναι στο λογικό 1. Για τον σκοπό αυτό κάνουμε την πράξη and με μάσκα binary $11 = \text{dec } 3$, και την οποία κάνουμε shift προς τα δεξιά κατά μία θέση ($c \gg 1$) μέσα στο loop όσο το c είναι θετικό. Αν βρούμε δύο ίσα επιστρέφουμε 0. Σε άλλη περίπτωση επιστρέφουμε 1. Για την $F1$ θέλουμε είτε όλα τα 4 πρώτα bits να είναι 1 είτε να έχουμε 0 το bit E. Το bit E θα είναι 0 αν ο αριθμός μας είναι μικρότερος ή ίσος του 15. Ακόμη για να έχουμε τα ABCD όλα 1 θα πρέπει να έχουμε μάσκα binary $1111 = \text{dec } 31$. Τέλος το $F2$ είναι αληθές είτε $F1$ είτε $F2$. Και περνάμε το αποτέλεσμα στην έξοδο στο σωστό bit όπως ζητάει η εκώνηση με διαδοχικές ολισθήσεις.

Κυρίως κώδικας:

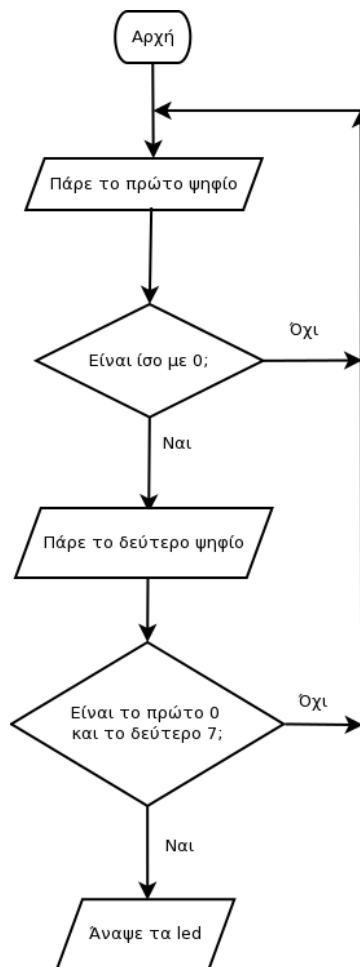
```

6
7  * Creation Date : 07-02-2012
8
9  * Last Modified : Wed 08 Feb 2012 12:09:50 AM EET
10
11 * Created By : Greg Liras <gregliras@gmail.com>
12
13 .....*/
14
15
16 #include<avr/io.h>
17
18 int f0 ( int c )
19 {
20     for( ; c > 0 ; c >>=1 )
21         if ( ( c & 3 ) == 3 )
22             return 0;
23     return 1;
24 }
25 int f1 ( int c )
26 {
27     if ( c <= 15 || c == 31 )
28         return 1;
29     return 0;
30 }
31 int main(void)
32 {
33     DDRA = 0xff;
34     DDRC = 0x00;
35     int f_0;
36     int f_1;
37     int c;
38     for( ;; )
39     {
40         c = PINC & 31;
41         f_0 = f0( c );
42         f_1 = f1( c );
43         PORTA = f_0 | ( f_1 << 1 ) | ( ( f_0 | f_1 ) << 2);
44     }
45 }

```

Άσκηση (iii)

Στην άσκηση αυτή θέλουμε να διαβάσουμε 2 πλήκτρα από το πληκτρολόγιο (τα 0 και 7) και μόνο τότε να ανάβουμε τα leds PA0-7, ανεξάρτητα από την χρονική διάρκεια που έμεινε πατημένο το κάθε πλήκτρο . Ουσιαστικά πραγματοποιούμε μία ηλεκτρονική κλειδαριά. Εν προκειμένω, καταρχάς διαβάζουμε για να δούμε αν έχει έρθει χαρακτήρας από το πληκτολόγιο. Αν όχι περιμένουμε έως ότου διαβάσουμε. Σε περίπτωση που διαβάσουμε ελέγχουμε αν είναι ο σωστός χαρακτήρας, καταρχάς το '0' του πληκτρολογίου, το οποίο αντιστοιχεί στο hex 2. Αν δεν είναι τότε πρέπει να περιμένουμε από την αρχή να διαβάσουμε πάλι το '0'. Σε περίπτωση που το διαβάσουμε τότε ελεγχουμε για το αν διαβάσαμε '7', το οποίο αντιστοιχεί στο hex 10. Αν δεν διαβάσουμε το '7', τότε πάμε πάλι στην αρχή και περιμένουμε να διαβάσουμε πάλι το first_key (δηλαδή το '0'). Αν διαβάσουμε επιτυχώς και τα 2 κλειδιά τότε ανάβουμε τα leds PA0-7. Παρακάτω φαίνεται το διάγραμμα ροής και ο κώδικας.



Σχήμα 1: Διάγραμμα ροής

Κυρίως κώδικας:

```

1  #define __SFR_OFFSET 0
2  #include <avr/io.h>
3
4  .data
5      _tmp_: .byte 2
6  .text
7  .global main
8  main:
9  reset:
10     ldi r24,lo8(RAMEND)
11     out SPL,r24
12     ldi r24,hi8(RAMEND)
13     out SPH,r24
14     ;// PORTA output
15     ser r24
16     out DDRA,r24
17     ;// 4x4 pad input
18     ldi r24,(1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
19     out DDRC ,r24
20     ;// initialize
21     rcall scan_keypad_rising_edge
22  lock:
23     ;//reset the memory
24     ldi r24,0x00
25     ldi r25,0x00
26     ldi r26 ,lo8(_tmp_)
27     ldi r27 ,hi8(_tmp_)
28     st X+ ,r24
29     st X ,r25
30  first_key:
31     ldi r24, 0x14
32     rcall scan_keypad_rising_edge
  
```

```

33     cpi r24,0x00
34     breq first_key
35     cpi r24, 0x02
36     ;//first test failed
37     brne lock
38
39     ldi r24,0x00
40     ldi r25,0x00
41     ldi r26 ,lo8(_tmp_)
42     ldi r27 ,hi8(_tmp_)
43     st X+ ,r24
44     st X ,r25
45
46     second_key:
47     ldi r24, 0x14
48     rcall scan_keypad_rising_edge
49     cpi r24,0x00
50     breq second_key
51     cpi r24, 0x10
52     ;//second test failed
53     brne lock
54
55     pass:
56     ldi r24,0xff
57     out PORTA, r24
58     ldi r24,lo8(2000)
59     ldi r25,hi8(2000)
60     rcall wait_msec
61     ;//reset the leds
62     ldi r24,0x00
63     out PORTA, r24
64     rjmp lock
65
66
67     scan_row:
68     ldi r25 ,0x08
69     back_: lsl r25
70     dec r24
71     brne back_
72     out PORTC ,r25
73     nop
74     nop
75     in r24 ,PINC
76     andi r24 ,0x0f
77     ret
78
79
80
81     scan_keypad:
82     ldi r24 ,0x01
83     rcall scan_row
84     swap r24
85     mov r27 ,r24
86     ldi r24 ,0x02
87     rcall scan_row
88     add r27 ,r24
89     ldi r24 ,0x03
90     rcall scan_row
91     swap r24
92     mov r26 ,r24
93     ldi r24 ,0x04
94     rcall scan_row
95     add r26 ,r24
96     movw r24 ,r26
97     ret
98
99
100    scan_keypad_rising_edge:
101    mov r22 ,r24
102    rcall scan_keypad
103    push r24
104    push r25
105    mov r24 ,r22
106    ldi r25 ,0
107    rcall wait_msec

```

```

108     rcall scan_keypad
109     pop r23
110     pop r22
111     and r24 ,r22
112     and r25 ,r23
113     ldi r26 ,lo8(_tmp_)
114     ldi r27 ,hi8(_tmp_)
115     ld r23 ,X+
116     ld r22 ,X
117     st X ,r24
118     st -X ,r25
119     com r23
120     com r22
121     and r24 ,r22
122     and r25 ,r23
123     ret
124
125
126
127 wait_usec:
128     sbiw r24,1
129     nop
130     brne wait_usec
131     ret
132
133 wait_msec:
134     in r27,PINB
135     ror r27
136     brcs wait_msec
137     push r24
138     push r25
139     ldi r24,lo8(998)
140     ldi r25,hi8(998)
141     rcall wait_usec
142     pop r25
143     pop r24
144     sbiw r24,1
145     brne wait_msec
146     ret

```

Άσκηση (iv)

Σε αυτή την άσκηση ζητείται να απεικονίσουμε στην lcd οθόνη που συνδέουμε στον AVR αρχικά το μήνυμα TEAM07 (ο αριθμός της ομάδας μας) και στη συνέχεια να απεικονίζεται το πλήκτρο που πατήθηκε τελευταίο στο keypad 4x4. Επίσης ο κέρσορας δεν πρέπει να είναι ορατός. Για το σκοπό αυτό, αρχικοποιούμε την οθόνη, διαβάζουμε το πληκτρολόγιο και περνάμε εντολές και δεδομένα στην οθόνη με τις κατάλληλες ρουτίνες που δίνονται έτοιμες στο pdf. Επίσης, σετάρουμε για είσοδο απ' το keypad την PORTC και για έξοδο στην οθόνη την PORTD. Κάθε φορά καλούμε την keypad_rising_edge αφού έχουμε βάλει στον r24 καθυστέρηση 20msec για τους σπινθηρισμούς, στη συνέχεια φρεσκάρουμε την οθόνη και καλούμε την lcd_data για απεικόνιση. Έπειτα, καθαρίζουμε τη μνήμη ώστε να έχει μόνο το τελευταίο πλήκτρο που πατήθηκε και διαβάζουμε εκ νέου απ' το πληκτρολόγιο.

Κυρίως κώδικας:

```

1  /*Includes for compatibility with GNU toolchain*/
2  #define __SFR_OFFSET 0
3  #include <avr/io.h>
4  #include <avr/interrupt.h>
5  .global main
6  /*
7   * AVRAsk4.asm
8   *
9   * Created: 12/2/2012 10:31:36 ??
10  * Author: Valia
11  */
12  .data
13  _tmp_ :.byte 2
14
15  .text
16  #define temp1 r24
17  #define temp2 r25
18
19  main:
20

```



```

21     ser r24
22     out DDRD,r24
23     ldi r24 ,(1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
24     out DDRC ,r24
25
26     rcall lcd_init ;//initialize
27
28
29
30     ldi r24,'T' ;// 'T'
31     rcall lcd_data
32     ldi r24,'E' ;// 'E'
33     rcall lcd_data
34     ldi r24,'A' ;// 'E'
35     rcall lcd_data
36     ldi r24,'M' ;// 'E'
37     rcall lcd_data
38     ldi r24,' ' ;// 'E'
39     rcall lcd_data
40     ldi r24,'0' ;// 'E'
41     rcall lcd_data
42     ldi r24,'7' ;// 'E'
43     rcall lcd_data
44
45     start:
46     ldi r24,0x14
47     rcall scan_keypad_rising_edge ;//apotelesma sto r24:r25
48     rcall keypad_to_ascii
49     cpi r24,0x0
50     breq start
51
52     push r24
53     ldi r24,0x01 ;//freskarisma
54     rcall lcd_command
55     ldi r24,lo8(1530)
56     ldi r25,hi8(1530)
57     rcall wait_usec
58     pop r24
59
60     rcall lcd_data ;//emfanish sthn o8onh
61
62     ldi r26,lo8(_tmp_) ;//ka8arisma mmhmhs
63     ldi r27,hi8(_tmp_)
64     ldi r24,0x0
65     st X+,r24
66     st X,r24
67
68     rjmp start
69
70     keypad_to_ascii:
71     movw r26,r24
72     ldi r24,'*'
73     sbrc r26,0
74     ret
75     ldi r24,'0'
76     sbrc r26,1
77     ret
78     ldi r24,'#'
79     sbrc r26,2
80     ret
81     ldi r24,'D'
82     sbrc r26,3
83     ret
84     ldi r24,'7'
85     sbrc r26,4
86     ret
87     ldi r24,'8'
88     sbrc r26,5
89     ret
90     ldi r24,'9'
91     sbrc r26,6
92     ret
93     ldi r24,'C'
94     sbrc r26,7
95     ret

```

```

96     ldi r24,'4'
97     sbrc r27,0
98     ret
99     ldi r24,'5'
100    sbrc r27,1
101    ret
102    ldi r24,'6'
103    sbrc r27,2
104    ret
105    ldi r24,'B'
106    sbrc r27,3
107    ret
108    ldi r24,'1'
109    sbrc r27,4
110    ret
111    ldi r24,'2'
112    sbrc r27,5
113    ret
114    ldi r24,'3'
115    sbrc r27,6
116    ret
117    ldi r24,'A'
118    sbrc r27,7
119    ret
120    clr r24
121    ret
122
123    scan_keypad_rising_edge:
124        mov r22,r24
125        rcall scan_keypad
126
127        push r24 ;//apo8ikeush apotelesmatos
128        push r25
129        mov r24,r22 ;//ka8ysterhsh r22msec (10-20 msec)
130        ldi r25,0
131        rcall wait_msec
132
133        rcall scan_keypad ;//aporrispe osa plhktra emfanizoyrn spin8hrismo
134        pop r23
135        pop r22
136        and r24,r22
137        and r25,r23
138        ldi r26,lo8(_tmp_)
139        ldi r27,hi8(_tmp_)
140        ld r23,X+
141        ld r22,X
142        st X,r24
143        st -X,r25
144        com r23
145        com r22
146        and r24,r22
147        and r25,r22
148        ret
149
150    lcd_init:
151        ldi r24,40
152        ldi r25,0
153        rcall wait_msec
154
155        ldi r24,0x30
156        out PORTD,r24
157        sbi PORTD,PD3
158        cbi PORTD,PD3
159        ldi r24,39
160        ldi r25,0
161        rcall wait_usec
162
163        ldi r24,0x30
164        out PORTD,r24
165        sbi PORTD,PD3
166        cbi PORTD,PD3
167        ldi r24,39
168        ldi r25,0
169        rcall wait_usec
170

```

```

171     ldi r24,0x20
172     out PORTD,r24
173     sbi PORTD,PD3
174     cbi PORTD,PD3
175     ldi r24,39
176     ldi r25,0
177     rcall wait_usec
178
179     ldi r24,0x28
180     rcall lcd_command
181
182     ldi r24,0x0c
183     rcall lcd_command
184
185     ldi r24,0x01
186     rcall lcd_command
187     ldi r24,lo8(1530)
188     ldi r25,hi8(1530)
189     rcall wait_usec
190
191     ldi r24,0x06
192     rcall lcd_command
193
194     ret
195
196     lcd_data:
197         sbi PORTD,PD2
198         rcall write_2_nibbles
199         ldi r24,43
200         ldi r25,0
201         rcall wait_usec
202         ret
203
204     lcd_command:
205         cbi PORTD,PD2
206         rcall write_2_nibbles
207         ldi r24,39
208         ldi r25,0
209         rcall wait_usec
210         ret
211
212     write_2_nibbles:
213         push r24
214         in r25,PIND
215         andi r25,0x0f
216         andi r24,0xf0
217         add r24,r25
218         out PORTD,r24
219         sbi PORTD,PD3
220         cbi PORTD,PD3
221         pop r24
222         swap r24
223         andi r24,0xf0
224         add r24,r25
225         out PORTD,r24
226         sbi PORTD,PD3
227         cbi PORTD,PD3
228         ret
229
230     scan_keypad:
231         ldi r24,0x01
232         rcall scan_row
233         swap r24
234         mov r27,r24
235         ldi r24,0x02
236         rcall scan_row
237         add r27,r24      ;//1h kai 2h grammh
238         ldi r24,0x03
239         rcall scan_row
240         swap r24
241         mov r26,r24
242         ldi r24,0x04
243         rcall scan_row
244         add r26,r24      ;//3h kai 4h grammh
245         movw r24,r26     ;//r25:r24 to apotelesma

```

```

246     ret
247
248     scan_row:
249         ldi r25,0x08
250     back_:
251         lsl r25
252         dec r24
253         brne back_
254         out PORTC,r25
255         nop
256         nop
257         in r24,PINC
258         andi r24,0x0f
259         ret
260
261     wait_msec:
262         push temp1
263         push temp2
264         ldi temp1, lo8(998)
265         ldi temp2, hi8(998)
266         rcall wait_usec
267         pop temp2
268         pop temp1
269         sbiw temp1, 1
270         brne wait_msec
271
272     ret
273
274     wait_usec:
275         sbiw temp1,1
276         nop
277         nop
278         nop
279         nop
280         brne wait_usec
281
282     ret

```

Άσκηση (v)

Σε αυτήν την άσκηση προσομοιώνουμε την λειτουργία ενός συστήματος συναγερμού. Θέτουμε την θύρα B ως είσοδο και έπειτα ελέγχουμε συνεχώς αν έχει γίνει trigger στους αισθητήρες. Άπαξ και έχει γίνει, θέτουμε τον timer για να αρχίσει να χρονομετρά και καλούμε την συνάρτηση getpass για να διαβάσουμε τον κωδικό. Ελέγχουμε τα πλήκτρα που εισάγονται με παρόμοιο τρόπο όπως στην άσκηση 3, με την διαφορά ότι άμα έχουμε λάθος πλήκτρο τότε πηγαίνουμε στην ετικέτα alarm_on, όπου θέτουμε τον συναγερμό κατά τα ζητούμενα της άσκησης. Σε περίπτωση που δοθεί ο σωστός κωδικός (τον οποίο έχουμε βάλει κατά σύμβαση 9807) απενεργοποιούμε τον συναγερμό (label alarm_off), δηλαδή απενεργοποιούμε τον timer και γραφουμε στην lcd οθόνη alarm off.

Κυρίως κώδικας:

```

1  #define __SFR_OFFSET 0
2  #include <avr/io.h>
3  #include <avr/interrupt.h>
4
5  .data
6      _tmp_ :.byte 2
7
8  .text
9
10 .global main
11 .org 0x000
12     rjmp main
13 .org 0x010
14     rjmp ovf_int_rout
15     reti
16
17 main:
18 reset:
19     ldi r24,lo8(RAMEND)
20     out SPL,r24
21     ldi r24,hi8(RAMEND)
22     out SPH,r24
23

```

```

24     ser r24
25     out DDRA,r24
26     out DDRD,r24
27
28     clr r24
29     out DDRB,r24
30
31     ldi r24 ,(1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
32     out DDRC ,r24
33
34
35     rcall lcd_init ;//initialize
36
37 loop:
38     in r24,PINB
39     cpi r24,0x0
40     breq loop
41
42
43     rcall set_timer
44     rcall getpass
45     rjmp loop
46
47 set_timer:
48     push r24
49     ;activate timer1
50     ;2 is TOIE1
51     ;avrStudio doesn't know about it
52     ldi r24,(1 << 2)
53     out TIMSK,r24
54     ldi r24,(1 << CS12)|(0<<CS11)|(1<<CS10)
55     out TCCR1B,r24
56
57     ldi r24,0xFF
58     sts TCNT1H,r24
59     ldi r24,0xFF
60     sts TCNT1L,r24
61
62     pop r24
63     sei
64     ret
65
66 ovf_int_rout:
67     rcall alarm_on
68     sei
69     reti
70
71 cls:
72     push r24
73     ldi r24,0x01
74     rcall lcd_command
75     ldi r24,lo8(1530)
76     ldi r25,hi8(1530)
77     rcall wait_usec
78     pop r24
79     ret
80
81
82 getpass:
83     rcall cls
84     rcall clear_tmp_
85     clr r24
86 dig0:
87     ldi r24,0x14
88     rcall scan_keypad_rising_edge
89
90     cpi r24,0x0
91     breq dig0
92     cpi r24,0x80
93     brne alarm_on
94
95
96     rcall clear_tmp_
97     ldi r24,0x14
98 dig1:

```

```

99     ldi r24,0x14
100    rcall scan_keypad_rising_edge
101    cpi r24,0x0
102    breq dig1
103    cpi r24,0x40
104    brne alarm_on
105
106    rcall clear_tmp_
107    ldi r24,0x14
108    dig2:
109    rcall scan_keypad_rising_edge
110    cpi r24,0x0
111    breq dig2
112    cpi r24,0x02
113    brne alarm_on
114
115    rcall clear_tmp_
116    ldi r24,0x14
117    dig3:
118    rcall scan_keypad_rising_edge
119    cpi r24,0x0
120    breq dig3
121    cpi r24,0x10
122    brne alarm_on
123    rcall alarm_off
124    ret
125
126    alarm_on:
127    push r24
128    ldi r24,0xff
129    out PORTA,r24
130    pop r24
131
132    write_alarm_on:
133    rcall lcd_init
134    ;rcall cls
135    ldi r24,'A'
136    rcall lcd_data
137    ldi r24,'L'
138    rcall lcd_data
139    ldi r24,'A'
140    rcall lcd_data
141    ldi r24,'R'
142    rcall lcd_data
143    ldi r24,'M'
144    rcall lcd_data
145    ldi r24,' '
146    rcall lcd_data
147    ldi r24,'O'
148    rcall lcd_data
149    ldi r24,'N'
150    rcall lcd_data
151    ret
152
153    alarm_off:
154    push r24
155    rcall write_alarm_off
156    ldi r24,0x00
157    out PORTA,r24
158    rcall set_timer_off
159    pop r24
160    ret
161
162    write_alarm_off:
163    rcall cls
164    ldi r24,'A'
165    rcall lcd_data
166    ldi r24,'L'
167    rcall lcd_data
168    ldi r24,'A'
169    rcall lcd_data
170    ldi r24,'R'
171    rcall lcd_data
172    ldi r24,'M'
173    rcall lcd_data

```

```

174     ldi r24, ' '
175     rcall lcd_data
176     ldi r24, '0'
177     rcall lcd_data
178     ldi r24, 'F'
179     rcall lcd_data
180     ldi r24, 'F'
181     rcall lcd_data
182     ret
183
184     clear_tmp_:
185     push r24
186     push r26
187     ldi r26, lo8(_tmp_) ;//kasarisma mmhms
188     ldi r27, hi8(_tmp_)
189     ldi r24, 0x0
190     st X+, r24
191     st X, r24
192     pop r26
193     pop r24
194     ret
195
196
197     set_timer_off:
198     push r24
199     ldi r24, 0x0
200     out TCCR1A, r24
201     out TCCR1B, r24
202     ret
203
204     keypad_to_ascii:
205     movw r26, r24
206     ldi r24, '*'
207     sbrc r26, 0
208     ret
209     ldi r24, '0'
210     sbrc r26, 1
211     ret
212     ldi r24, '#'
213     sbrc r26, 2
214     ret
215     ldi r24, 'D'
216     sbrc r26, 3
217     ret
218     ldi r24, '7'
219     sbrc r26, 4
220     ret
221     ldi r24, '8'
222     sbrc r26, 5
223     ret
224     ldi r24, '9'
225     sbrc r26, 6
226     ret
227     ldi r24, 'C'
228     sbrc r26, 7
229     ret
230     ldi r24, '4'
231     sbrc r27, 0
232     ret
233     ldi r24, '5'
234     sbrc r27, 1
235     ret
236     ldi r24, '6'
237     sbrc r27, 2
238     ret
239     ldi r24, 'B'
240     sbrc r27, 3
241     ret
242     ldi r24, '1'
243     sbrc r27, 4
244     ret
245     ldi r24, '2'
246     sbrc r27, 5
247     ret
248     ldi r24, '3'

```

```

249     sbrnc r27,6
250     ret
251     ldi r24,'A'
252     sbrnc r27,7
253     ret
254     clr r24
255     ret
256
257     scan_keypad_rising_edge:
258     mov r22,r24
259     rcall scan_keypad
260
261     push r24 ;//apo8ikeush apotelesmatos
262     push r25
263     mov r24,r22 ;//ka8ysterhsh r22msec (10-20 msec)
264     ldi r25,0
265     rcall wait_msec
266
267     rcall scan_keypad ;//aporrispe osa plhktra emfanizoyrn spin8hrismo
268     pop r23
269     pop r22
270     and r24,r22
271     and r25,r23
272     ldi r26,lo8(_tmp_)
273     ldi r27,hi8(_tmp_)
274     ld r23,X+
275     ld r22,X
276     st X,r24
277     st -X,r25
278     com r23
279     com r22
280     and r24,r22
281     and r25,r22
282     ret
283
284     lcd_init:
285     ldi r24,40
286     ldi r25,0
287     rcall wait_msec
288
289     ldi r24,0x30
290     out PORTD,r24
291     sbi PORTD,PD3
292     cbi PORTD,PD3
293     ldi r24,39
294     ldi r25,0
295     rcall wait_usec
296
297     ldi r24,0x30
298     out PORTD,r24
299     sbi PORTD,PD3
300     cbi PORTD,PD3
301     ldi r24,39
302     ldi r25,0
303     rcall wait_usec
304
305     ldi r24,0x20
306     out PORTD,r24
307     sbi PORTD,PD3
308     cbi PORTD,PD3
309     ldi r24,39
310     ldi r25,0
311     rcall wait_usec
312
313     ldi r24,0x28
314     rcall lcd_command
315
316     ldi r24,0x0c
317     rcall lcd_command
318
319     ldi r24,0x01
320     rcall lcd_command
321     ldi r24,lo8(1530)
322     ldi r25,hi8(1530)
323     rcall wait_usec

```



```

324
325     ldi r24,0x06
326     rcall lcd_command
327
328     ret
329
330 lcd_data:
331     sbi PORTD,PD2
332     rcall write_2_nibbles
333     ldi r24,43
334     ldi r25,0
335     rcall wait_usec
336     ret
337
338 lcd_command:
339     cbi PORTD,PD2
340     rcall write_2_nibbles
341     ldi r24,39
342     ldi r25,0
343     rcall wait_usec
344     ret
345
346 write_2_nibbles:
347     push r24
348     in r25,PINB
349     andi r25,0x0f
350     andi r24,0xf0
351     add r24,r25
352     out PORTD,r24
353     sbi PORTD,PD3
354     cbi PORTD,PD3
355     pop r24
356     swap r24
357     andi r24,0xf0
358     add r24,r25
359     out PORTD,r24
360     sbi PORTD,PD3
361     cbi PORTD,PD3
362     ret
363
364 scan_keypad:
365     ldi r24,0x01
366     rcall scan_row
367     swap r24
368     mov r27,r24
369     ldi r24,0x02
370     rcall scan_row
371     add r27,r24    ;//1h kai 2h grammh
372     ldi r24,0x03
373     rcall scan_row
374     swap r24
375     mov r26,r24
376     ldi r24,0x04
377     rcall scan_row
378     add r26,r24    ;//3h kai 4h grammh
379     movw r24,r26   ;//r25:r24 to apotelesma
380     ret
381
382 scan_row:
383     ldi r25,0x08
384 back_:
385     lsl r25
386     dec r24
387     brne back_
388     out PORTC,r25
389     nop
390     nop
391     in r24,PINC
392     andi r24,0x0f
393     ret
394
395 wait_msec:
396     push r24
397     push r25
398     ldi r24, lo8(998)

```

```

399     ldi r25, hi8(998)
400     rcall wait_usec
401     pop r25
402     pop r24
403     sbiw r24, 1
404     brne wait_msec
405
406     ret
407
408     wait_usec:
409     sbiw r24,1
410     nop
411     nop
412     nop
413     nop
414     brne wait_usec
415
416     ret

```

Άσκηση (vi)

Ζητούμενο της άσκησης αυτής είναι να διαβάσουμε ένα δεκαεξαδικό αριθμό από την PORTA σε μορφή δυαδικού συμπληρώματος ως προς 2 και να γράψουμε την δεκαδική του αναπαράσταση στην lcd οθόνη. Η λογική που ακολουθούμε είναι η εξής: διαβαζουμε το input και κάνουμε shift μια θέση δεξιά ώστε να δούμε αν θα έχουμε 0 ή 1 για θετικό ή αρνητικό πρόσημο αντίστοιχα. Το πρόσημο αποθηκεύεται στον καταχωρητή sign. Έπειτα ανάλογα αν είναι ο αριθμός αρνητικός παίρνουμε το συμπλήρωμά του ως προς 2, αλλιώς προχωρούμε στον υπολογισμό κατευθείαν. Αν είναι μεγαλύτερος του 100 τότε βάζουμε στον καταχωρητή ekat την μία εκατοντάδα. Αλλιώς, και έπειτα, ελέγχουμε στο loop count_dek το πόσες δεκάδες έχουμε και αποθηκεύουμε το αποτέλεσμα στον καταχωρητή dek. Αυτό που μας μένει είναι οι μονάδες και το οποίο αποθηκεύουμε στον καταχωρητή mon. Έπειτα συνεχίζουμε στο να τα δείξουμε στην lcd οθόνη. Κάθε "κβάντο" πληροφορίας που στέλνουμε αποθηκεύεται στον καταχωρητή r24 (τον οποίο έχουμε ονομάσει quantum). Για την αποστολή βέβαια των εκατοντάδων και των δεκάδων βέβαια φροντίζουμε να στέλνουμε τον ascii κωδικό των χαρακτήρων, προσθέτοντάς τους το 48 (dec). Ακόμη έχουμε φροντίσει να ελέγχουμε αν είναι μηδενικές για να μην εκτυπώνεται για παράδειγμα +009 αλλά +9. Πριν την αποστολή του αριθμού στην οθόνη Έπειτα στέλνουμε διαδοχικά το sign,(αν υπάρχουν)τις εκατοντάδες, (αν υπάρχουν) τις δεκάδες και τέλος τις μονάδες.

Κυρίως κώδικας:

```

1  /*Includes for compatibility with GNU toolchain*/
2  #define __SFR_OFFSET 0
3  #include <avr/io.h>
4  #include <avr/interrupt.h>
5  .global main
6  /*
7   * AVRAssembler3_a.asm
8   *
9   * Created: 14/2/2012 1:23:46 ??
10  * Author: Eleni
11  */
12  /*
13  * AVRAssembler3.asm
14  *
15  * Created: 13/2/2012 9:07:06 ??
16  * Author: Eleni
17  */
18
19  /*
20  * bonus_part_2.asm
21  *
22  * Created: 12/2/2012 1:08:30 ??
23  * Author: Eleni
24  */
25
26  #define input r18
27  #define sign r21
28  #define tempo r20
29  #define ekat r22
30  #define dek r23
31  #define mon r19
32  #define quantum r24
33  #define temp1 r24

```

```

34  #define temp2 r25
35
36  main:
37
38      ldi r24 ,lo8(RAMEND)
39      out SPL, r24
40      ldi r24, hi8(RAMEND)
41      out SPH, r24
42
43      clr r25                ;//bawz mhdenika gia eisodo sthn A
44      out DDRB, r25
45
46      ser r25                ;//bawz 1 gia eksodo sthn D
47      out DDRD, r25
48
49
50
51      rcall lcd_init
52
53  start:
54
55
56
57      in input, PINB
58      mov tempo, input
59      lsl tempo
60      brcc positive          ;//an carry = 0 pame sto positive
61
62  negative:                  ;//edw einai arnhtikos o ari8mos
63      ;// bawz to arnhtiko proshmo
64      ldi sign, '-'
65      neg input              ;//to neg dinei kateu8eian to sumplhrwma ws pros 2
66      rjmp calculation
67
68  positive:
69      ;//an einai 8etikos bawz to + kai pro8wurw sto calculation gia display
70      ldi sign, '+'
71
72  calculation:
73      clr ekat
74      clr dek
75      clr mon
76
77      cpi input, 0x64          ;//sugkrish me to 100
78      brlo count_dek          ;//branch if less sto metrhma dekadwn
79
80      ;//alliw8 exoume 1 ekatontada
81  count_ekate:
82      ldi ekate, 1            ;//giati mia ekatontada to polu 8a exoume
83      subi input, 0x64        ;//afairw 100 epeidh 8elw na sunexisw
84
85  count_dek:
86      cpi input, 10           ;//sugkrish me to 10
87      brlo count_mon          ;//an einai mikrotero tou 10 metrame tis monades
88      ;//alliw8
89      subi input, 10          ;//afairw 10
90      inc dek
91      rjmp count_dek
92
93      ;//edw ston input exoun meinei pleon oi monades
94  count_mon:
95      mov mon, input
96      rjmp print_to_lcd
97
98  print_to_lcd:
99      ;//tupwnw to proshmo
100  lcd_clear:
101      ldi r24, 0x01           ;//ka8arismos o8onh8
102      rcall lcd_command
103      ldi r24, lo8(1530)
104      ldi r25, hi8(1530)
105      rcall wait_usec         ;//telos ka8arismos o8onh8
106      mov quantum, sign
107      rcall lcd_data
108

```

```

109 ;//elegxos gia an exw ekatontada
110 cpi ekat, 1
111 brlo check_dekades ;//an <100 koitaw mhpws einai kai dekades0
112 subi ekat, -48 ;//pros8etw 48 gia na parw ton ascii
113 mov quantum, ekat
114 rcall lcd_data
115
116 check_dekades:
117 cpi dek, 0
118 breq check_monades ;//an einai iso me 0
119 subi dek, -48 ;//allws.
120 mov quantum, dek
121 rcall lcd_data
122
123 check_monades:
124 subi mon, -48
125 mov quantum, mon ;//oi monades o,ti kai na nai tis deixnw, giati exw sumplhrwma ws pros 2. Opote 8a deiksw gia 0 to +0.
126 rcall lcd_data
127
128 ldi r24,0xAA
129 ldi r25,0x00
130 rcall wait_msec
131
132
133 rjmp start
134
135
136 ;//=====routines gia thn o8onh=====
137 lcd_init:
138 ldi r24,40
139 ldi r25,0
140 rcall wait_msec
141
142 ldi r24,0x30
143 out PORTD,r24
144 sbi PORTD,PD3
145 cbi PORTD,PD3
146 ldi r24,39
147 ldi r25,0
148 rcall wait_usec
149
150 ldi r24,0x30
151 out PORTD,r24
152 sbi PORTD,PD3
153 cbi PORTD,PD3
154 ldi r24,39
155 ldi r25,0
156 rcall wait_usec
157
158 ldi r24,0x20
159 out PORTD,r24
160 sbi PORTD,PD3
161 cbi PORTD,PD3
162 ldi r24,39
163 ldi r25,0
164 rcall wait_usec
165
166 ldi r24,0x28
167 rcall lcd_command
168
169 ldi r24,0x0c
170 rcall lcd_command
171
172 ldi r24,0x01
173 rcall lcd_command
174 ldi r24,lo8(1530)
175 ldi r25,hi8(1530)
176 rcall wait_usec
177
178 ldi r24,0x06
179 rcall lcd_command
180
181 ret
182
183 lcd_data:

```

```

184     sbi PORTD,PD2
185     rcall write_2_nibbles
186     ldi r24,43
187     ldi r25,0
188     rcall wait_usec
189     ret
190
191     lcd_command:
192     cbi PORTD,PD2
193     rcall write_2_nibbles
194     ldi r24,39
195     ldi r25,0
196     rcall wait_usec
197     ret
198
199     write_2_nibbles:
200     push r24
201     in r25,PIND
202     andi r25,0x0f
203     andi r24,0xf0
204     add r24,r25
205     out PORTD,r24
206     sbi PORTD,PD3
207     cbi PORTD,PD3
208     pop r24
209     swap r24
210     andi r24,0xf0
211     add r24,r25
212     out PORTD,r24
213     sbi PORTD,PD3
214     cbi PORTD,PD3
215     ret
216
217     ;//=====wait routines=====
218     wait_msec:
219     push temp1
220     push temp2
221     ldi temp1, lo8(998)
222     ldi temp2, hi8(998)
223     rcall wait_usec
224     pop temp2
225     pop temp1
226     sbiw temp1, 1
227     brne wait_msec
228
229     ret
230
231     wait_usec:
232     sbiw temp1,1
233     nop
234     nop
235     nop
236     nop
237     brne wait_usec
238
239     ret

```