ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΜ&ΜΥ
Εργαστήριο Μικροϋπολογιστών

7η Εργαστηριακή Άσκηση
Ακ. έτος 2011-2012

Ομάδα C07:

Ελένη Ευαγγελάτου Α.Μ.: 03108050
Γρηγόρης Λύρας Α.Μ.: 03109687
Βασιλεία Φραγκιαδάκη Α.Μ.: 03108026

18 Φεβρουαρίου 2012

Άσκηση (i)

Σε αυτή την άσκηση ζητείται να υλοποιήσουμε πέντε λογικές πύλες διαβάζοντας την είσοδο από την πόρτα Α ενώ η έξοδος φαίνεται στα τέσσερα LSB leds της πόρτας Β. Για το σκοπό αυτό διαβάζουμε ανά δύο τα bits της εισόδου και υλοποιούμε τις αντίστοιχες λογικές συναρτήσεις αποθηκεύοντάς 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 Α, 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:

```
/*Includes for compatibility with GNU toolchain*/
#define __SFR_OFFSET 0
#include <avr/io.h>
#include <avr/interrupt.h>
.global main
/*
 * AVR_3h_ask1.asm
 *
 *  Created: 10/2/2012 10:01:52 ??
 *   Author: Valia
 */


 /*
 * AVR_ask1.asm
 *
 *  Created: 10/2/2012 9:24:48 ??
 *   Author: Valia
 */

#define temp1  r24
#define input  r28
#define output  r27
#define temp2  r25
#define regC  r29
#define tempo  r26
#define output1  r30

main:
    ldi temp1, hi8(RAMEND)
    out SPH, temp1
    ldi temp1, lo8(RAMEND)
    out SPL, temp1

    ser temp1
    out DDRB, temp1

    clr temp1
    out DDRD, temp1

    clr regC
    out DDRC, regC
start:
    clr output1
    in regC, PINC

    lsr regC
    brcc syn1
etik1:      ;//perimenei mexri na ginei 0 (button)
    in tempo, PINC
    lsr tempo
    ;brcc syn1
    ori output1, 1

syn1:
    lsr regC
    brcc syn2
etik2:      ;//perimenei mexri na ginei 0 (button)
    in tempo, PINC
    lsr tempo
    lsr tempo
```

```
    ;brcc syn2
    ori output1, 2

syn2:
    lsr regC
    brcc syn3
etik3:      ;//perimenei mexri na ginei 0 (button)
    in tempo, PINC
    lsr tempo
    lsr tempo
    lsr tempo
    ;brcc syn3
    ori output1, 4

 syn3:
    lsr regC
    brcc syn4
etik4:      ;//perimenei mexri na ginei 0 (button)
    in tempo, PINC
    lsr tempo
    lsr tempo
    lsr tempo
    lsr tempo
    ;brcc syn4
    ori output1, 8

 syn4:
    lsr regC
    brcc syn5
etik5:      ;//perimenei mexri na ginei 0 (button)
    in tempo, PINC
    lsl tempo
    lsl tempo
    lsl tempo
    lsl tempo
    ;brcs etik5
    ori output1, 0x010

syn5:
    lsr regC
    brcc syn6
etik6:      ;//perimenei mexri na ginei 0 (button)
    in tempo, PINC
    lsl tempo
    lsl tempo
    lsl tempo
    ;brcs etik6
    ori output1, 0x020

 syn6:
    lsr regC
    brcc syn7
 etik7:      ;//perimenei mexri na ginei 0 (button)
    in tempo, PINC
    lsl tempo
    lsl tempo
    ;brcs etik7
    ori output1, 0x040

syn7:
    lsr regC
    brcc synexise
 etik8:      ;//perimenei mexri na ginei 0 (button)
    in tempo, PINC
    lsl tempo
    ;brcs etik8
    ori output1, 0x080

 synexise:

    in input, PIND
    clr temp2   ;//mhdenismos exodwn
    clr output

gate_1:
```

```
    lsr input


        brcc exit1_gate1  ;//an to lsb = 0 feugoume
    lsr input
    brcc exit_gate1  ;//an to 2o lsb = 0 feugoume
    inc temp2
        rjmp exit_gate1

exit1_gate1:
        lsr input;//dior8wsh
exit_gate1:

gate_2:
    lsr input
    brcc exit1_gate2 ;//an to 3o lsb = 0 feugoume
    lsr input
    brcc exit_gate2  ;//an to 4o lsb = 0 feugoume
    ori output, 3
        rjmp exit_gate2
exit1_gate2:
        lsr input
exit_gate2:
    or output, temp2    ;//exodos sta 2 prwta lsb bit

gate_3:
    lsr input
    brcs exit1_gate3 ;//an to 5o lsb = 1 feugoume
    lsr input
    brcs exit_gate3 ;//an to 6o lsb = 1 feugoume
    ori output, 4  ;// or me 100
        rjmp exit_gate3
exit1_gate3:
        lsr input
exit_gate3:

gate_4:
    lsr input
    brcs  one   ;//an einai 1 psaxnoumt an kai to allo einai 1
    lsr input   ;//an einai to 1o 0, tsekaroyme an kai to 2o einai 0
    brcs exit_gate4
    rjmp setting

one:
    lsr input
    brcc exit_gate4 ;// an to allo einai 0 feugoyme

setting:
    ori output,8

exit_gate4:
   eor output, output1  ;//xor
    out PORTB, output
    rjmp start
```

Άσκηση (ii)

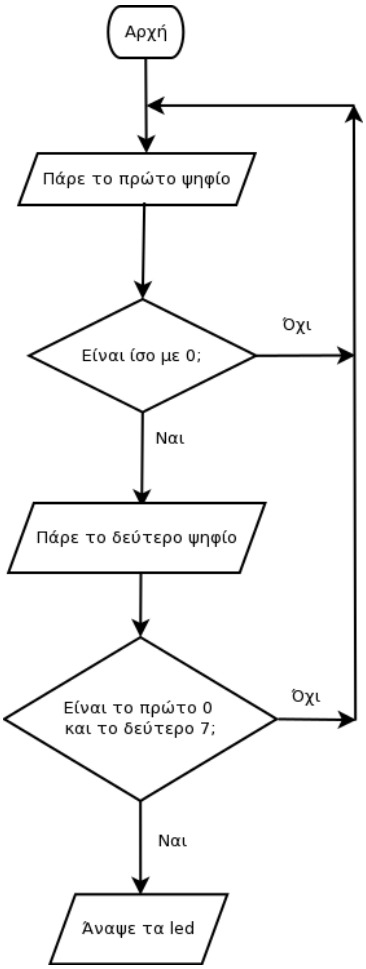
Στο μέρος αυτό υλοποιούμε τις συναρτήσεις F0 = (AB+BC+CD+DE)' , F1= ABCD+E , F2 = F0+F1. Χρησιμοποιώντας τις bitwise λογικές πράξεις της C, έχουμε τα εξής: Για την F0 θέλουμε οποιαδήποτε 2 συνεχόμενα bits της θύρας εισόδου να είναι στο λογικό 1. Για τον σκοπό αυτό κάνουμε την πράξη and με μάσκα binary 11 = dec 3, και την οποιά κάνουμε shift προς τα δεξιά κατά μία θέση (c»=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 11111 = 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 ολισθίσεις.
Κυρίως κώδικας:

```
/* -.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.

* File Name : part2.c

* Purpose :
```

```

* Creation Date : 07-02-2012

* Last Modified : Wed 08 Feb 2012 12:09:50 AM EET

* Created By : Greg Liras <gregliras@gmail.com>

_._._._._._._._._._._._._._._._._._._._._.*/


#include<avr/io.h>

int f0 ( int c )
{
    for( ; c > 0 ; c >>=1 )
        if ( ( c & 3 ) == 3 )
            return 0;
    return 1;
}
int f1 ( int c )
{
    if (  c <= 15 || c == 31 )
        return 1;
    return 0;
}
int main(void)
{
    DDRA = 0xff;
    DDRC = 0x00;
    int f_0;
    int f_1;
    int c;
    for( ;; )
    {
        c = PINC & 31;
        f_0 = f0( c );
        f_1 = f1( c );
        PORTA = f_0 | ( f_1 << 1) | ( ( f_0 | f_1 ) << 2);
    }
}
```

Άσκηση (iii)

Στην άσκηση αυτή θέλουμε να διαβάζουμε 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 ελεχουμε 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:

```
#define __SFR_OFFSET 0
#include <avr/io.h>

.data
    _tmp_: .byte 2
.text
.global main
main:
reset:
    ldi r24,lo8(RAMEND)
    out SPL,r24
    ldi r24,hi8(RAMEND)
    out SPH,r24
    ;// PORTA output
    ser r24
    out DDRA,r24
    ;// 4x4 pad input
    ldi r24 ,(1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
    out DDRC ,r24
    ;// initialize
    rcall scan_keypad_rising_edge
lock:
    ;//reset the memory
    ldi r24,0x00
    ldi r25,0x00
    ldi r26 ,lo8(_tmp_)
    ldi r27 ,hi8(_tmp_)
    st X+ ,r24
    st X ,r25
first_key:
    ldi r24, 0x14
    rcall scan_keypad_rising_edge
```

```
    cpi r24,0x00
    breq first_key
    cpi r24, 0x02
    ;//first test failed
    brne lock

    ldi r24,0x00
    ldi r25,0x00
    ldi r26 ,lo8(_tmp_)
    ldi r27 ,hi8(_tmp_)
    st X+ ,r24
    st X ,r25

second_key:
    ldi r24, 0x14
    rcall scan_keypad_rising_edge
    cpi r24,0x00
    breq second_key
    cpi r24, 0x10
    ;//second test failed
    brne lock

pass:
    ldi r24,0xff
    out PORTA, r24
    ldi r24,lo8(2000)
    ldi r25,hi8(2000)
    rcall wait_msec
    ;//reset the leds
    ldi r24,0x00
    out PORTA, r24
    rjmp lock


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 ,lo8(_tmp_)
    ldi r27 ,hi8(_tmp_)
    ld r23 ,X+
    ld r22 ,X
    st X ,r24
    st -X ,r25
    com r23
    com r22
    and r24 ,r22
    and r25 ,r23
    ret



wait_usec:
    sbiw r24,1
    nop
    brne wait_usec
    ret

wait_msec:
    in r27,PINB
    ror r27
    brcs wait_msec
    push r24
    push r25
    ldi r24,lo8(998)
    ldi r25,hi8(998)
    rcall wait_usec
    pop r25
    pop r24
    sbiw r24,1
    brne wait_msec
    ret
```

Άσκηση (iv)

Κυρίως κώδικας:

```
/*Includes for compatibility with GNU toolchain*/
#define __SFR_OFFSET 0
#include <avr/io.h>
#include <avr/interrupt.h>
.global main
/*
 * AVRAsk4.asm
 *
 *  Created: 12/2/2012 10:31:36 ??
 *   Author: Valia
 */
 .data
 _tmp_ :.byte 2

 .text
 #define temp1  r24
 #define temp2  r25

 main:

    ser r24
    out DDRD,r24
    ldi r24 ,(1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
    out DDRC ,r24

    rcall lcd_init ;//initialize



    ldi r24,'T'  ;//'T'
```

```
    rcall lcd_data
    ldi r24,'E' ;//'E'
    rcall lcd_data
    ldi r24,'A' ;//'E'
    rcall lcd_data
    ldi r24,'M' ;//'E'
    rcall lcd_data
    ldi r24,' ' ;//'E'
    rcall lcd_data
    ldi r24,'0' ;//'E'
    rcall lcd_data
    ldi r24,'7' ;//'E'
    rcall lcd_data

start:
    ldi r24,0x14
    rcall scan_keypad_rising_edge ;//apotelesma sto r24:r25
    rcall keypad_to_ascii
    cpi r24,0x0
    breq start

    push r24
    ldi r24,0x01     ;//freskarisma
    rcall lcd_command
    ldi r24,lo8(1530)
    ldi r25,hi8(1530)
    rcall wait_usec
    pop r24

    rcall lcd_data  ;//emfanish sthn o8onh

    ldi r26,lo8(_tmp_)  ;//ka8arisma mnhmhs
    ldi r27,hi8(_tmp_)
    ldi r24,0x0
    st  X+,r24
    st  X,r24

    rjmp start

keypad_to_ascii:
    movw r26,r24
    ldi r24,'*'
    sbrc r26,0
    ret
    ldi r24,'0'
    sbrc r26,1
    ret
    ldi r24,'#'
    sbrc r26,2
    ret
    ldi r24,'D'
    sbrc r26,3
    ret
    ldi r24,'7'
    sbrc r26,4
    ret
    ldi r24,'8'
    sbrc r26,5
    ret
    ldi r24,'9'
    sbrc r26,6
    ret
    ldi r24,'C'
    sbrc r26,7
    ret
    ldi r24,'4'
    sbrc r27,0
    ret
    ldi r24,'5'
    sbrc r27,1
    ret
    ldi r24,'6'
    sbrc r27,2
    ret
    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 emfanizoyn 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
```

```
    nop
    in r24,PINC
    andi r24,0x0f
    ret

    wait_msec:
    push temp1
    push temp2
    ldi temp1, lo8(998)
    ldi temp2, hi8(998)
    rcall wait_usec
    pop temp2
    pop temp1
    sbiw temp1, 1
    brne wait_msec

    ret

    wait_usec:
    sbiw temp1,1
    nop
    nop
    nop
    nop
    brne wait_usec

    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:

```
#define __SFR_OFFSET 0
#include <avr/io.h>
#include <avr/interrupt.h>

.data
    _tmp_ :.byte 2

.text

.global main
.org 0x000
    rjmp main
.org 0x010
    rjmp ovf_int_rout
    reti

main:
reset:
    ldi r24,lo8(RAMEND)
    out SPL,r24
    ldi r24,hi8(RAMEND)
    out SPH,r24

    ser r24
    out DDRA,r24
    out DDRD,r24

    clr r24
    out DDRB,r24

    ldi r24 ,(1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
    out DDRC ,r24

```

```
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,'O'
177     rcall lcd_data
178     ldi r24,'F'
179     rcall lcd_data
180     ldi r24,'F'
181     rcall lcd_data
182     ret
183
```

```
clear_tmp_:
    push r24
    push r26
    ldi r26,lo8(_tmp_)  ;//ka8arisma mnhmhs
    ldi r27,hi8(_tmp_)
    ldi r24,0x0
    st  X+,r24
    st  X,r24
    pop r26
    pop r24
    ret


set_timer_off:
    push r24
    ldi r24,0x0
    out TCCR1A,r24
    out TCCR1B,r24
    ret

keypad_to_ascii:
    movw r26,r24
    ldi r24,'*'
    sbrc r26,0
    ret
    ldi r24,'0'
    sbrc r26,1
    ret
    ldi r24,'#'
    sbrc r26,2
    ret
    ldi r24,'D'
    sbrc r26,3
    ret
    ldi r24,'7'
    sbrc r26,4
    ret
    ldi r24,'8'
    sbrc r26,5
    ret
    ldi r24,'9'
    sbrc r26,6
    ret
    ldi r24,'C'
    sbrc r26,7
    ret
    ldi r24,'4'
    sbrc r27,0
    ret
    ldi r24,'5'
    sbrc r27,1
    ret
    ldi r24,'6'
    sbrc r27,2
    ret
    ldi r24,'B'
    sbrc r27,3
    ret
    ldi r24,'1'
    sbrc r27,4
    ret
    ldi r24,'2'
    sbrc r27,5
    ret
    ldi r24,'3'
    sbrc r27,6
    ret
    ldi r24,'A'
    sbrc r27,7
    ret
    clr r24
    ret

scan_keypad_rising_edge:
    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 emfanizoyn 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 συπμλήρωμά 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 αποθηκευουμε 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             ;//bazw mhdenika gia eisodo sthn A
```

```
    out DDRB, r25

    ser r25                             ;//bazw 1 gia eksodo sthn D
    out DDRD, r25



        rcall lcd_init

start:



    in input, PINB
    mov tempo, input
    lsl tempo
    brcc positive               ;//an carry = 0 pame sto positive

negative:                               ;//edw einai arnhtikos o ari8mos
    ;// bazw to arnhtiko proshmo
    ldi sign, '-'
    neg input                   ;//to neg dinei kateu8eian to sumplhrwma ws pros 2
    rjmp calculation

positive:
    ;//an einai 8etikos bazw to + kai proxwrw sto calculation gia display
    ldi sign, '+'

calculation:
    clr ekat
    clr dek
    clr mon

    cpi input, 0x64             ;//sugkrish me to 100
    brlo count_dek              ;//branch if less sto metrhma dekadwn

    ;//alliws exoume 1 ekatontada
count_ekat:
    ldi ekat, 1                 ;//giati mia ekatontada to polu 8a exoume
    subi input, 0x64    ;//afairw 100 epeidh 8elw na sunexisw

count_dek:
    cpi input, 10               ;//sugkrish me to 10
    brlo count_mon              ;//an einai mikrotero tou 10 metrame tis monades
    ;//alliws
    subi input, 10              ;//afairw 10
    inc dek
        rjmp count_dek

    ;//edw ston input exoun meinei pleon oi monades
count_mon:
    mov mon, input
    rjmp print_to_lcd

print_to_lcd:
        ;//tupwnw to proshmo
lcd_clear:
        ldi r24, 0x01           ;//ka8arismos o8onhs
    rcall lcd_command
    ldi r24, lo8(1530)
    ldi r25, hi8(1530)
    rcall wait_usec     ;//telos ka8arismos o8onhs
    mov quantum, sign
    rcall lcd_data

    ;//elegxos gia an exw ekatontada
    cpi ekat, 1
    brlo check_dekades  ;//an <100 koitaw mhpws einai kai dekades0
    subi ekat, -48              ;//pros8etw 48 gia na parw ton ascii
    mov quantum, ekat
    rcall lcd_data

check_dekades:
    cpi dek, 0
    breq check_monades  ;//an einai iso me 0
```

```
119     subi dek, -48               ;//alliws.
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
```