Εργαστήριο Μικροϋπολογιστών

9η εργαστηριακή άσκηση

Τμήμα: Β Ομάδα: 15

Συνεργάτες: Μαρουφίδης Ιωάννης (03113506),

Περράκης Γεώργιος (03113511) ,

Σοφιανίδης Γεώργιος (03113179)

1η Άσκηση

```
.include "m16def.inc"

.dseg
_tmp_: .byte 2
.def temp = r18
.cseg

start:
	ldi r24, low(RAMEND)

	out SPL, r24
	ldi r24, high(RAMEND)
	out SPH, r24
	ldi r24, (1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
                                            ; set 4 MSBs as output
	out DDRC, r24
                                                                        ;
in PORTC
	ldi r24, (1 << PD7) | (1 << PD6) | (1 << PD5) | (1 << PD4) | (1 << PD3)
| (1 << PD2)        ; set 6 MSBs as output
	out DDRD, r24
                                                                        ;
in PORTD
	call lcd_init
                                                                        ;
initialize lcd screen with "NONE"
	ldi r24,'N'
	call lcd_data
	ldi r24,'O'
	call lcd_data
	ldi r24,'N'
	call lcd_data
	ldi r24,'E'
	call lcd_data

loop:
	ldi r24, 20                                     ; set 20ms bouncing
delay
	call scan_keypad_rising_edge      ; read keyboard
```

```
        call keypad_to_ascii
        cpi r24, 0                              ; if no buttons are
pressed, read again
        breq loop
        mov temp, r24                           ; temp has the number that was
pressed

        ldi r24, 0x02                           ; clear screen
        rcall lcd_command

        mov r24, temp
        call lcd_data                           ; print number to the screen
        ldi r24,' '
        call lcd_data
        ldi r24,' '
        call lcd_data
        ldi r24,' '
        call lcd_data
        jmp loop


wait_usec:
        sbiw r24 ,1                             ; 2 κύκλοι (0.250 μsec)
        nop                                     ; 1 κύκλος (0.125
μsec)
        nop                                     ; 1 κύκλος (0.125
μsec)
        nop                                     ; 1 κύκλος (0.125
μsec)
        nop                                     ; 1 κύκλος (0.125
μsec)
        brne wait_usec                          ; 1 ή 2 κύκλοι (0.125 ή
0.250 μsec)
        ret                                     ; 4 κύκλοι (0.500
μsec)


wait_msec:
        push r24                                ; 2 κύκλοι (0.250 μsec)
        push r25                                ; 2 κύκλοι
        ldi r24 , low(998)                      ; φόρτωσε τον καταχ. r25:r24 με
998 (1 κύκλος - 0.125 μsec)
        ldi r25 , high(998)                     ; 1 κύκλος (0.125 μsec)
        rcall wait_usec                         ; 3 κύκλοι (0.375 μsec),
προκαλεί συνολικά καθυστέρηση 998.375 μsec
        pop r25                                 ; 2 κύκλοι (0.250
μsec)
        pop r24                                 ; 2 κύκλοι
        sbiw r24 , 1                            ; 2 κύκλοι
        brne wait_msec                          ; 1 ή 2 κύκλοι (0.125 ή
0.250 μsec)
        ret                                     ; 4 κύκλοι (0.500
μsec)


lcd_data:
        sbi PORTD ,PD2                          ; επιλογή του καταχωρήτη
δεδομένων (PD2=1)
        rcall write_2_nibbles                   ; αποστολή του byte
        ldi r24 ,43                             ; αναμονή 43μsec μέχρι
να ολοκληρωθεί η λήψη
```

```
	ldi r25 ,0 ; των δεδομένων από τον
ελεγκτή της lcd
	rcall wait_usec
	ret


lcd_command:
	cbi PORTD ,PD2 ; επιλογή του καταχωρητή
εντολών (PD2=1)
	rcall write_2_nibbles ; αποστολή της εντολής και
αναμονή 39μsec
	ldi r24 ,39 ; για την ολοκλήρωση της
εκτέλεσης της από τον ελεγκτή της lcd.
	ldi r25 ,0 ; ΣΗΜ.: υπάρχουν δύο
εντολές, οι clear display και return home,
	rcall wait_usec ; που απαιτούν σημαντικά
μεγαλύτερο χρονικό διάστημα.
	ret


write_2_nibbles:
	push r24 ; στέλνει τα 4 MSB
	in r25 ,PIND ; διαβάζονται τα 4 LSB και τα
ξαναστέλνουμε
	andi r25 ,0x0f ; για να μην χαλάσουμε
την όποια προηγούμενη κατάσταση
	andi r24 ,0xf0 ; απομονώνονται τα 4 MSB
και
	add r24 ,r25 ; συνδυάζονται με τα
προϋπάρχοντα 4 LSB
	out PORTD ,r24 ; και δίνονται στην
έξοδο
	sbi PORTD ,PD3 ; δημιουργείται παλμός
Enable στον ακροδέκτη PD3
	cbi PORTD ,PD3 ; PD3=1 και μετά PD3=0
	pop r24 ; στέλνει τα 4
LSB. Ανακτάται το byte.
	swap r24 ; εναλλάσσονται τα 4 MSB
με τα 4 LSB
	andi r24 ,0xf0 ; που με την σειρά τους
αποστέλλονται
	add r24 ,r25
	out PORTD ,r24
	sbi PORTD ,PD3 ; Νέος παλμός Enable
	cbi PORTD ,PD3
	ret


lcd_init:
	ldi r24 ,40 ; Όταν ο ελεγκτής της
lcd τροφοδοτείται με
	ldi r25 ,0 ; ρεύμα εκτελεί την δική
του αρχικοποίηση.
	rcall wait_msec ; Αναμονή 40 msec μέχρι
αυτή να ολοκληρωθεί.
	ldi r24 ,0x30 ; εντολή μετάβασης σε 8 bit
mode
	out PORTD ,r24 ; επειδή δεν μπορούμε να
είμαστε βέβαιοι
	sbi PORTD ,PD3 ; για τη διαμόρφωση
εισόδου του ελεγκτή
```

```
	cbi PORTD ,PD3			; της οθόνης, η εντολή
αποστέλλεται δύο φορές
	ldi r24 ,39
	ldi r25 ,0			; εάν ο ελεγκτής της
οθόνης βρίσκεται σε 8-bit mode
	rcall wait_usec			; δεν θα συμβεί τίποτα,
αλλά αν ο ελεγκτής έχει διαμόρφωση
						; εισόδου 4 bit
θα μεταβεί σε διαμόρφωση 8 bit
	ldi r24 ,0x30
	out PORTD ,r24
	sbi PORTD ,PD3
	cbi PORTD ,PD3
	ldi r24 ,39
	ldi r25 ,0
	rcall wait_usec
	ldi r24 ,0x20			; αλλαγή σε 4-bit mode
	out PORTD ,r24
	sbi PORTD ,PD3
	cbi PORTD ,PD3
	ldi r24 ,39
	ldi r25 ,0
	rcall wait_usec
	ldi r24 ,0x28			; επιλογή χαρακτήρων μεγέθους
5x8 κουκίδων
	rcall lcd_command		; και εμφάνιση δύο γραμμών στην
οθόνη
	ldi r24 ,0x0c			; ενεργοποίηση της οθόνης,
απόκρυψη του κέρσορα
	rcall lcd_command
	ldi r24 ,0x01			; καθαρισμός της οθόνης
	rcall lcd_command
	ldi r24 ,low(1530)
	ldi r25 ,high(1530)
	rcall wait_usec
	ldi r24 ,0x06			; ενεργοποίηση αυτόματης
αύξησης κατά 1 της διεύθυνσης
	rcall lcd_command		; που είναι αποθηκευμένη στον
μετρητή διευθύνσεων και
						; απενεργοποίηση
της ολίσθησης ολόκληρης της οθόνης
	ret


scan_row:
	ldi r25 ,0x08			; αρχικοποίηση με ‘0000 1000’
back_: lsl r25				; αριστερή ολίσθηση του
‘1’ τόσες θέσεις
	dec r24					; όσος είναι ο
αριθμός της γραμμής
	brne back_
	out PORTC ,r25			; η αντίστοιχη γραμμή
τίθεται στο λογικό ‘1’
	nop
	nop					; καθυστέρηση για
να προλάβει να γίνει η αλλαγή κατάστασης
	in r24 ,PINC			; επιστρέφουν οι θέσεις
(στήλες) των διακοπτών που είναι πιεσμένοι
	andi r24 ,0x0f			; απομονώνονται τα 4 LSB
όπου τα ‘1’ δείχνουν που είναι πατημένοι
	ret					; οι διακόπτες.
```

```
scan_keypad:
	ldi r24 ,0x01 ; έλεγξε την πρώτη γραμμή του
πληκτρολογίου
	rcall scan_row
	swap r24 ; αποθήκευσε το
αποτέλεσμα
	mov r27 ,r24 ; στα 4 msb του r27
	ldi r24 ,0x02 ; έλεγξε τη δεύτερη γραμμή του
πληκτρολογίου
	rcall scan_row
	add r27 ,r24 ; αποθήκευσε το αποτέλεσμα στα
4 lsb του r27
	ldi r24 ,0x03 ; έλεγξε την τρίτη γραμμή του
πληκτρολογίου
	rcall scan_row
	swap r24 ; αποθήκευσε το
αποτέλεσμα
	mov r26 ,r24 ; στα 4 msb του r26
	ldi r24 ,0x04 ; έλεγξε την τέταρτη γραμμή του
πληκτρολογίου
	rcall scan_row
	add r26 ,r24 ; αποθήκευσε το αποτέλεσμα στα
4 lsb του r26
	movw r24 ,r26 ; μετέφερε το αποτέλεσμα στους
καταχωρητές r25:r24
	ret


scan_keypad_rising_edge:
	mov r22 ,r24 ; αποθήκευσε το χρόνο
σπινθηρισμού στον r22
	rcall scan_keypad ; έλεγξε το πληκτρολόγιο για
πιεσμένους διακόπτες
	push r24 ; και αποθήκευσε το
αποτέλεσμα
	push r25
	mov r24 ,r22 ; καθυστέρησε r22 ms (τυπικές
τιμές 10-20 msec που καθορίζεται από τον
	ldi r25 ,0 ; κατασκευαστή του
πληκτρολογίου – χρονοδιάρκεια σπινθηρισμών)
	rcall wait_msec
	rcall scan_keypad ; έλεγξε το πληκτρολόγιο ξανά
και απόρριψε
	pop r23 ; όσα πλήκτρα
εμφανίζουν σπινθηρισμό
	pop r22
	and r24 ,r22
	and r25 ,r23
	ldi r26 ,low(_tmp_) ; φόρτωσε την κατάσταση των
διακοπτών στην
	ldi r27 ,high(_tmp_) ; προηγούμενη κλήση της ρουτίνας στους
r27:r26
	ld r23 ,X+
	ld r22 ,X
	st X ,r24 ; αποθήκευσε στη RAM τη
νέα κατάσταση
	st -X ,r25 ; των διακοπτών
	com r23
	com r22 ; βρες τους
διακόπτες που έχουν «μόλις» πατηθεί
	and r24 ,r22
```

```
        and r25 ,r23
        ret


keypad_to_ascii:                                ; λογικό ‘1’ στις θέσεις του
καταχωρητή r26 δηλώνουν
        movw r26 ,r24                           ; τα παρακάτω σύμβολα και
αριθμούς
        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                                     ; αν δεν είναι
‘1’παρακάμπτει την ret, αλλιώς (αν είναι ‘1’)
        ret                                                     ; επιστρέφει με
τον καταχωρητή r24 την ASCII τιμή του D.
        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'                            ; λογικό ‘1’ στις θέσεις του
καταχωρητή r27 δηλώνουν
        sbrc r27 ,0                                     ; τα παρακάτω σύμβολα
και αριθμούς
        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
```

Στην παραπάνω άσκηση, απεικονίζουμε στην οθόνη LCD τον τελευταίο χαρακτήρα που δίνουμε από το πληκτρολόγιο. Αρχικά η ένδειξη της οθόνης είναι “NONE”. Για το σκοπό της άσκησης χρησιμοποιούμε τις ειδικές συναρτήσεις για τη χρήση της οθόνης και του πληκτρολογίου.

2η Άσκηση

```
.include "m16def.inc"

.dseg
_tmp_: .byte 2
.cseg

.def temp = r17
.def temp1 = r16
.def flag = r18
.def leds = r19
.def count = r21

.org 0x00
jmp start
.org 0x10
jmp ISR_TIMER1_OVF

start:
	ldi r24,LOW(RAMEND)
                                                                        ;
initialize stack
	out SPL,r24
	ldi r25,HIGH(RAMEND)
	out SPH,r25

	clr r24
	out DDRA ,r24
                                                                        ;
PORTA as input (sensors)
	ldi r24, (1 << PD7) | (1 << PD6) | (1 << PD5) | (1 << PD4) | (1 << PD3)
| (1 << PD2)	; set 6 MSBs as output (monitor)
	out DDRD, r24

	ser r24
	out DDRB ,r24
                                                                        ;
PORTB as output (alarm leds)
	ldi r24, (1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
                                                ; PORTD as output (keyboard)

	out DDRC ,r24

	call lcd_init
                                                                        ;
initialize LCD Monitor
	ldi flag,0xFF
	ldi count, 0
                                                                        ;
if flag == 0xFF password is incorrect (we start with this value)
```

```
sensor_loop:
                                                                        ;
we wait till one of the PORTA buttons is pressed
       in temp,PINA
                                                                        ;
which driggers the alarm
       cpi temp,0x00
       breq sensor_loop

       ldi temp,(1<<TOIE1)
                                                                        ;
enable overflow interrupt of register TCNT1
       out TIMSK,temp

       ; for timer1
       ldi temp,(1<<CS12) | (0<<CS11) | (1<<CS10)
                                                              ; CK/1024 =
8MHz/1024 = 7812.5Hz
    out TCCR1B,temp
                                                                        ;
7812.5 * 4sec = 31250 cycles
       ldi temp,0x85
                                                                        ;
MAX - 31250 = 65536 - 31250 = 34286 cycles = 0x85EE
       out TCNT1H,temp

       ; we set TCNT1 to overflow after 4 seconds
       ldi temp,0xEE
       out TCNT1L,temp
       sei

       ldi r24,0x0f
                                                                        ;
we display cursor
       rcall lcd_command

password_loop:
       ldi r24, 20
       call scan_keypad_rising_edge

       call keypad_to_ascii
       mov r20, r24
       cpi r20, 0
       breq password_loop
       cpi r20, 66

       ; we check if 'B' is pressed
       brne password_incorrect
       call lcd_data
       inc count
first_digit_correct:
       ldi r24, 20
       call scan_keypad_rising_edge

       call keypad_to_ascii
       mov r20, r24
       cpi r20, 0
       breq first_digit_correct
       cpi r20, 49

       ; we check if '1' is pressed
       brne password_incorrect
```

```
        call lcd_data
        inc count
second_digit_correct:
        ldi r24, 20
        call scan_keypad_rising_edge

        call keypad_to_ascii
        mov r20, r24
        cpi r20, 0
        breq second_digit_correct
        cpi r20, 53

        ; we check if '5' is pressed
        brne password_incorrect
        call lcd_data

        ldi flag,0x00
                                                                        ;
if yes flag is set accordingly

password_correct:
        ldi r24,0x0c
                                                                        ;
we remove cursor
        rcall lcd_command
        ldi r24, 0x02
                                                                        ;
return home
        rcall lcd_command
        ldi r24,'A'

        ; display message for alarm off
        call lcd_data
        ldi r24,'L'
        call lcd_data
        ldi r24,'A'
        call lcd_data
        ldi r24,'R'
        call lcd_data
        ldi r24,'M'
        call lcd_data
        ldi r24,' '
        call lcd_data
        ldi r24,'O'
        call lcd_data
        ldi r24,'F'
        call lcd_data
        ldi r24,'F'
        call lcd_data
password_correct1:
                                                                        ;
and wait till timer is triggered
        jmp password_correct1

password_incorrect:
                                                                        ;
if password was incorrect we wait till 4 sec time ends
        call lcd_data
        inc count
inner_loop:
        cpi count, 3
```

```
        breq label0

        ; while waiting user can write and see pressed digits
        ldi r24, 20
        call scan_keypad_rising_edge
        call keypad_to_ascii
        cpi r24, 0
        breq inner_loop
        inc count
        call lcd_data
        jmp inner_loop

label0:
        ldi temp,(1<<TOIE1)
                                                                        ;
enable overflow interrupt of register TCNT1
        out TIMSK,temp

        ; for timer1
        ldi temp,(1<<CS12) | (0<<CS11) | (1<<CS10)
                                                                ; CK/1024 =
8MHz/1024 = 7812.5Hz
    out TCCR1B,temp
                                                                        ;
7812.5 * 4sec = 31250 cycles
        ldi temp,0x00
                                                                        ;
MAX - 31250 = 65536 - 31250 = 34286 cycles = 0x85EE
        out TCNT1H,temp

        ; we set TCNT1 to overflow after 4 seconds
        ldi temp,0x00
        out TCNT1L,temp

ISR_TIMER1_OVF:
        ldi r24,low(200)
        ldi r25,high(200)
        call wait_msec
        cpi flag,0x00
                                                                        ;
if 4 seconds pass we check if password was given correctly
        breq end_loop
                                                                        ;
if yes we skip all
        ldi r24,0x0c
                                                                        ;
we hide
        rcall lcd_command
        ldi r24, 0x02
                                                                        ;
return home
        rcall lcd_command
        ldi r24,'A'

        ; display message for alarm on
        call lcd_data
        ldi r24,'L'
        call lcd_data
        ldi r24,'A'
        call lcd_data
        ldi r24,'R'
        call lcd_data
```

```
        ldi r24,'M'
        call lcd_data
        ldi r24,' '
        call lcd_data
        ldi r24,'O'
        call lcd_data
        ldi r24,'N'
        call lcd_data
leds_loop:

        ; we set on the alarm
        ser leds
        out PORTB,leds

        ; leds on for 0.2 sec
        ldi r24,low(200)
        ldi r25,high(200)
        call wait_msec
        clr leds
        out PORTB,leds

        ; leds off for 0.2 sec
        ldi r24,low(200)
        ldi r25,high(200)
        call wait_msec
        jmp leds_loop
end_loop:
        reti


scan_row:
        ldi r25 ,0x08
                                ; αρχικοποίηση με ‘0000 1000’
back_: lsl r25
                                        ; αριστερή ολίσθηση του ‘1’ τόσες θέσεις
        dec r24
                                                ; όσος είναι ο αριθμός της γραμμής
        brne back_
        out PORTC ,r25
                                        ; η αντίστοιχη γραμμή τίθεται στο λογικό ‘1’
        nop
        nop
                                                ; καθυστέρηση για να προλάβει να γίνει
η αλλαγή κατάστασης
        in r24 ,PINC
                                ; επιστρέφουν οι θέσεις (στήλες) των διακοπτών που
είναι πιεσμένοι
        andi r24 ,0x0f
                                        ; απομονώνονται τα 4 LSB όπου τα ‘1’ δείχνουν
που είναι πατημένοι
        ret
                                                ; οι διακόπτες.


scan_keypad:
        ldi r24 ,0x01
                                ; έλεγξε την πρώτη γραμμή του πληκτρολογίου
        rcall scan_row
        swap r24
                                        ; αποθήκευσε το αποτέλεσμα
        mov r27 ,r24
                                ; στα 4 msb του r27
```

```
        ldi r24 ,0x02
                            ; έλεγξε τη δεύτερη γραμμή του πληκτρολογίου
        rcall scan_row
        add r27 ,r24
                            ; αποθήκευσε το αποτέλεσμα στα 4 lsb του r27
        ldi r24 ,0x03
                            ; έλεγξε την τρίτη γραμμή του πληκτρολογίου
        rcall scan_row
        swap r24
                                    ; αποθήκευσε το αποτέλεσμα
        mov r26 ,r24
                            ; στα 4 msb του r26
        ldi r24 ,0x04
                            ; έλεγξε την τέταρτη γραμμή του πληκτρολογίου
        rcall scan_row
        add r26 ,r24
                            ; αποθήκευσε το αποτέλεσμα στα 4 lsb του r26
        movw r24 ,r26
                            ; μετέφερε το αποτέλεσμα στους καταχωρητές r25:r24
        ret

scan_keypad_rising_edge:
        mov r22 ,r24
                            ; αποθήκευσε το χρόνο σπινθηρισμού στον r22
        rcall scan_keypad
                            ; έλεγξε το πληκτρολόγιο για πιεσμένους διακόπτες
        push r24
                                    ; και αποθήκευσε το αποτέλεσμα
        push r25
        mov r24 ,r22
                            ; καθυστέρησε r22 ms (τυπικές τιμές 10-20 msec που
καθορίζεται από τον
        ldi r25 ,0
                                    ; κατασκευαστή του πληκτρολογίου –
χρονοδιάρκεια σπινθηρισμών)
        rcall wait_msec
        rcall scan_keypad
                            ; έλεγξε το πληκτρολόγιο ξανά και απόρριψε
        pop r23
                                        ; όσα πλήκτρα εμφανίζουν σπινθηρισμό
        pop r22
        and r24 ,r22
        and r25 ,r23
        ldi r26 ,low(_tmp_)
                            ; φόρτωσε την κατάσταση των διακοπτών στην
        ldi r27 ,high(_tmp_)
                        ; προηγούμενη κλήση της ρουτίνας στους r27:r26
        ld r23 ,X+
        ld r22 ,X
        st X ,r24
                                    ; αποθήκευσε στη RAM τη νέα κατάσταση
        st -X ,r25
                                    ; των διακοπτών
        com r23
        com r22
                                        ; βρες τους διακόπτες που έχουν
«μόλις» πατηθεί
        and r24 ,r22
        and r25 ,r23
        ret
```

```
keypad_to_ascii:
                        ; λογικό ‘1’ στις θέσεις του καταχωρητή r26
δηλώνουν
        movw r26 ,r24
                        ; τα παρακάτω σύμβολα και αριθμούς
        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
                            ; αν δεν είναι ‘1’παρακάμπτει την ret, αλλιώς
(αν είναι ‘1’)
        ret
                                ; επιστρέφει με τον καταχωρητή r24 την
ASCII τιμή του D.
        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'
                        ; λογικό ‘1’ στις θέσεις του καταχωρητή r27
δηλώνουν
        sbrc r27 ,0
                            ; τα παρακάτω σύμβολα και αριθμούς
        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
```

```
write_2_nibbles:
       push r24                        ; στέλνει τα 4 MSB
       in r25 ,PIND                  ; διαβάζονται τα 4 LSB και τα ξαναστέλνουμε
       andi r25 ,0x0f                  ; για να μην χαλάσουμε την όποια
προηγούμενη κατάσταση
       andi r24 ,0xf0                  ; απομονώνονται τα 4 MSB και
       add r24 ,r25                  ; συνδυάζονται με τα προϋπάρχοντα 4 LSB
       out PORTD ,r24                  ; και δίνονται στην έξοδο
       sbi PORTD ,PD3                  ; δημιουργείται παλμός Enable στον
ακροδέκτη PD3
       cbi PORTD ,PD3                  ; PD3=1 και μετά PD3=0
       pop r24                               ; στέλνει τα 4 LSB. Ανακτάται
το byte.
       swap r24                        ; εναλλάσσονται τα 4 MSB με τα 4 LSB
       andi r24 ,0xf0                  ; που με την σειρά τους αποστέλλονται
       add r24 ,r25
       out PORTD ,r24
       sbi PORTD ,PD3                  ; Νέος παλμός Enable
       cbi PORTD ,PD3
       ret


lcd_data:
       sbi PORTD ,PD2                  ; επιλογή του καταχωρήτη δεδομένων
(PD2=1)
       rcall write_2_nibbles       ; αποστολή του byte
       ldi r24 ,43                     ; αναμονή 43μsec μέχρι να ολοκληρωθεί
η λήψη
       ldi r25 ,0                      ; των δεδομένων από τον ελεγκτή της
lcd
       rcall wait_usec
       ret


lcd_command:
       cbi PORTD ,PD2                  ; επιλογή του καταχωρητή εντολών
(PD2=1)
       rcall write_2_nibbles       ; αποστολή της εντολής και αναμονή 39μsec
       ldi r24 ,39                     ; για την ολοκλήρωση της εκτέλεσης της
από τον ελεγκτή της lcd.
       ldi r25 ,0                      ; ΣΗΜ.: υπάρχουν δύο εντολές, οι clear
display και return home,
       rcall wait_usec                 ; που απαιτούν σημαντικά μεγαλύτερο
χρονικό διάστημα.
       ret


lcd_init:
       ldi r24 ,40                     ; Όταν ο ελεγκτής της lcd
τροφοδοτείται με
       ldi r25 ,0                      ; ρεύμα εκτελεί την δική του
αρχικοποίηση.
       rcall wait_msec                 ; Αναμονή 40 msec μέχρι αυτή να
ολοκληρωθεί.
       ldi r24 ,0x30               ; εντολή μετάβασης σε 8 bit mode
       out PORTD ,r24                  ; επειδή δεν μπορούμε να είμαστε
βέβαιοι
       sbi PORTD ,PD3                  ; για τη διαμόρφωση εισόδου του
ελεγκτή
       cbi PORTD ,PD3                  ; της οθόνης, η εντολή αποστέλλεται
δύο φορές
```

```
       ldi r24 ,39
       ldi r25 ,0                      ; εάν ο ελεγκτής της οθόνης βρίσκεται
σε 8-bit mode
       rcall wait_usec                 ; δεν θα συμβεί τίποτα, αλλά αν ο
ελεγκτής έχει διαμόρφωση
                                              ; εισόδου 4 bit θα μεταβεί σε
διαμόρφωση 8 bit
       ldi r24 ,0x30
       out PORTD ,r24
       sbi PORTD ,PD3
       cbi PORTD ,PD3
       ldi r24 ,39
       ldi r25 ,0
       rcall wait_usec
       ldi r24 ,0x20             ; αλλαγή σε 4-bit mode
       out PORTD ,r24
       sbi PORTD ,PD3
       cbi PORTD ,PD3
       ldi r24 ,39
       ldi r25 ,0
       rcall wait_usec
       ldi r24 ,0x28             ; επιλογή χαρακτήρων μεγέθους 5x8 κουκίδων
       rcall lcd_command         ; και εμφάνιση δύο γραμμών στην οθόνη
       ldi r24 ,0x0c             ; ενεργοποίηση της οθόνης, απόκρυψη του
κέρσορα
       rcall lcd_command
       ldi r24 ,0x01             ; καθαρισμός της οθόνης
       rcall lcd_command
       ldi r24 ,low(1530)
       ldi r25 ,high(1530)
       rcall wait_usec
       ldi r24 ,0x06             ; ενεργοποίηση αυτόματης αύξησης κατά 1 της
διεύθυνσης
       rcall lcd_command         ; που είναι αποθηκευμένη στον μετρητή
διευθύνσεων και
                                              ; απενεργοποίηση της ολίσθησης
ολόκληρης της οθόνης
       ret


wait_usec:
   sbiw r24 ,1
                                 ; 2 κύκλοι (0.250 μsec)
   nop
                                 ; 1 κύκλος (0.125 μsec)
   nop
                                 ; 1 κύκλος (0.125 μsec)
   nop
                                 ; 1 κύκλος (0.125 μsec)
   nop
                                 ; 1 κύκλος (0.125 μsec)
   brne wait_usec
                           ; 1 ή 2 κύκλοι (0.125 ή 0.250 μsec)
   ret
                                 ; 4 κύκλοι (0.500 μsec)


wait_msec:
   push r24
                                 ; 2 κύκλοι (0.250 μsec)
   push r25
                                 ; 2 κύκλοι
```

```
   ldi r24 , low(998)
                           ; φόρτωσε τον καταχ. r25:r24 με 998 (1 κύκλος -
0.125 μsec)
   ldi r25 , high(998)
                           ; 1 κύκλος (0.125 μsec)
   rcall wait_usec
                           ; 3 κύκλοι (0.375 μsec), προκαλεί συνολικά
καθυστέρηση 998.375 μsec
   pop r25
                                 ; 2 κύκλοι (0.250 μsec)
   pop r24
                                 ; 2 κύκλοι
   sbiw r24 , 1
                                 ; 2 κύκλοι
   brne wait_msec
                           ; 1 ή 2 κύκλοι (0.125 ή 0.250 μsec)
   ret
                                 ; 4 κύκλοι (0.500 μsec)
```

Στην παραπάνω άσκηση εξομοιώνουμε ένα σύστημα συναγερμού. Για το σκοπό αυτό χρησιμοποιούμε χρονιστές, ως επίσης και τις ειδικές συναρτήσεις για τη χρήση της οθόνης και του πληκτρολογίου.

3η Άσκηση

```
.include "m16def.inc"

.def temp = r18
.def ekat = r19
.def dek = r20
.def mon = r21
.def msb = r22
.def flag = r23
.def temp1 = r17

start:
       ldi r24, low(RAMEND)

       out SPL, r24
       ldi r24, high(RAMEND)
       out SPH, r24

       clr temp

       ; set PORTA as input
       out DDRB, temp

       ldi r24, (1 << PD7) | (1 << PD6) | (1 << PD5) | (1 << PD4) | (1 << PD3)
| (1 << PD2)        ; set 6 MSBs as output (for monitor)
       out DDRD, r24


start1:
       call lcd_init
                                                                          ;
initialization
```

```
        in temp, PINB

        ldi r24, 48

        ; show the binary on screen
        sbrc temp, 7
        inc r24
        call lcd_data
        ldi r24, 48
        sbrc temp, 6
        inc r24
        call lcd_data
        ldi r24, 48
        sbrc temp, 5
        inc r24
        call lcd_data
        ldi r24, 48
        sbrc temp, 4
        inc r24
        call lcd_data
        ldi r24, 48
        sbrc temp, 3
        inc r24
        call lcd_data
        ldi r24, 48
        sbrc temp, 2
        inc r24
        call lcd_data
        ldi r24, 48
        sbrc temp, 1
        inc r24
        call lcd_data
        ldi r24, 48
        sbrc temp, 0
        inc r24
        call lcd_data

        ldi r24, '='
        call lcd_data
        ldi msb, 0
        sbrc temp , 0x07
                                                                        ;
check for negative number
        ldi msb,1

        ; msb is for sign
        sbrc temp , 0x07
        neg temp

        ; if msb = 1 , temp = temp' + 1
        cpi msb,0x00
        breq positive
        ldi r24,'-'

        ; show sign on screen
        call lcd_data
                                                                        ;
'-' for negative number
        jmp continue
positive:
```

```
        ldi r24, '+'                                                    ;
'+' for positive number
        call lcd_data

continue:
        ldi temp1, 1
        ldi ekat,0

        ; ekat = dec = mon = 0
        ldi dek,0
        ldi mon,0
        cpi temp,100
        brlo to_dec

        ldi ekat, 1

        ; if temp >= 100 , ekat = 1
        subi temp,100                                                   ;
and temp = temp - 100
to_dec:
        cpi temp,10
        brlo to_mon

        ; if temp < 10 move to define mon
        add dek,temp1                                                   ;
else dec = dec + 1 and temp = temp - 10
        subi temp,10
        jmp to_dec
to_mon:
        mov mon,temp                                                    ;
mon = temp

        ldi flag, 1

        ; if flag = 1 no hundreds
        mov r24,ekat
        cpi r24, 0
        breq no_ekat
        ldi r24, 48
        add r24, ekat
        call lcd_data                                                   ;
show ekat
        ldi flag, 0
no_ekat:
        cpi dek, 0
        breq check
label:

        ; show decades
        ldi r24, 48
        add r24, dek
        call lcd_data
        jmp no_dec
check:
        cpi flag, 0
        breq label
no_dec:
```

```
        ldi r24, 48
        add r24, mon
                                                                        ;
show mon
        call lcd_data
        jmp start1


wait_usec:
        sbiw r24 ,1                             ; 2 κύκλοι (0.250 μsec)
        nop                                             ; 1 κύκλος (0.125
μsec)
        nop                                             ; 1 κύκλος (0.125
μsec)
        nop                                             ; 1 κύκλος (0.125
μsec)
        nop                                             ; 1 κύκλος (0.125
μsec)
        brne wait_usec                          ; 1 ή 2 κύκλοι (0.125 ή
0.250 μsec)
        ret                                             ; 4 κύκλοι (0.500
μsec)


wait_msec:
        push r24                                        ; 2 κύκλοι (0.250 μsec)
        push r25                                        ; 2 κύκλοι
        ldi r24 , low(998)                      ; φόρτωσε τον καταχ. r25:r24 με
998 (1 κύκλος - 0.125 μsec)
        ldi r25 , high(998)                     ; 1 κύκλος (0.125 μsec)
        rcall wait_usec                                 ; 3 κύκλοι (0.375 μsec),
προκαλεί συνολικά καθυστέρηση 998.375 μsec
        pop r25                                         ; 2 κύκλοι (0.250
μsec)
        pop r24                                         ; 2 κύκλοι
        sbiw r24 , 1                            ; 2 κύκλοι
        brne wait_msec                                  ; 1 ή 2 κύκλοι (0.125 ή
0.250 μsec)
        ret                                             ; 4 κύκλοι (0.500
μsec)


lcd_data:
        sbi PORTD ,PD2                                  ; επιλογή του καταχωρήτη
δεδομένων (PD2=1)
        rcall write_2_nibbles                   ; αποστολή του byte
        ldi r24 ,43                                     ; αναμονή 43μsec μέχρι
να ολοκληρωθεί η λήψη
        ldi r25 ,0                                      ; των δεδομένων από τον
ελεγκτή της lcd
        rcall wait_usec
        ret


lcd_command:
        cbi PORTD ,PD2                                  ; επιλογή του καταχωρητή
εντολών (PD2=1)
        rcall write_2_nibbles                   ; αποστολή της εντολής και
αναμονή 39μsec
        ldi r24 ,39                                     ; για την ολοκλήρωση της
εκτέλεσης της από τον ελεγκτή της lcd.
```

```
	ldi r25 ,0 ; ΣΗΜ.: υπάρχουν δύο
εντολές, οι clear display και return home,
	rcall wait_usec ; που απαιτούν σημαντικά
μεγαλύτερο χρονικό διάστημα.
	ret


write_2_nibbles:
	push r24 ; στέλνει τα 4 MSB
	in r25 ,PIND ; διαβάζονται τα 4 LSB και τα
ξαναστέλνουμε
	andi r25 ,0x0f ; για να μην χαλάσουμε
την όποια προηγούμενη κατάσταση
	andi r24 ,0xf0 ; απομονώνονται τα 4 MSB
και
	add r24 ,r25 ; συνδυάζονται με τα
προϋπάρχοντα 4 LSB
	out PORTD ,r24 ; και δίνονται στην
έξοδο
	sbi PORTD ,PD3 ; δημιουργείται παλμός
Enable στον ακροδέκτη PD3
	cbi PORTD ,PD3 ; PD3=1 και μετά PD3=0
	pop r24 ; στέλνει τα 4
LSB. Ανακτάται το byte.
	swap r24 ; εναλλάσσονται τα 4 MSB
με τα 4 LSB
	andi r24 ,0xf0 ; που με την σειρά τους
αποστέλλονται
	add r24 ,r25
	out PORTD ,r24
	sbi PORTD ,PD3 ; Νέος παλμός Enable
	cbi PORTD ,PD3
	ret


lcd_init:
	ldi r24 ,40 ; Όταν ο ελεγκτής της
lcd τροφοδοτείται με
	ldi r25 ,0 ; ρεύμα εκτελεί την δική
του αρχικοποίηση.
	rcall wait_msec ; Αναμονή 40 msec μέχρι
αυτή να ολοκληρωθεί.
	ldi r24 ,0x30 ; εντολή μετάβασης σε 8 bit
mode
	out PORTD ,r24 ; επειδή δεν μπορούμε να
είμαστε βέβαιοι
	sbi PORTD ,PD3 ; για τη διαμόρφωση
εισόδου του ελεγκτή
	cbi PORTD ,PD3 ; της οθόνης, η εντολή
αποστέλλεται δύο φορές
	ldi r24 ,39
	ldi r25 ,0 ; εάν ο ελεγκτής της
οθόνης βρίσκεται σε 8-bit mode
	rcall wait_usec ; δεν θα συμβεί τίποτα,
αλλά αν ο ελεγκτής έχει διαμόρφωση
	; εισόδου 4 bit
θα μεταβεί σε διαμόρφωση 8 bit
	ldi r24 ,0x30
	out PORTD ,r24
	sbi PORTD ,PD3
	cbi PORTD ,PD3
	ldi r24 ,39
```

```
	ldi r25 ,0
	rcall wait_usec
	ldi r24 ,0x20                          ; αλλαγή σε 4-bit mode
	out PORTD ,r24
	sbi PORTD ,PD3
	cbi PORTD ,PD3
	ldi r24 ,39
	ldi r25 ,0
	rcall wait_usec
	ldi r24 ,0x28                          ; επιλογή χαρακτήρων μεγέθους
5x8 κουκίδων
	rcall lcd_command                      ; και εμφάνιση δύο γραμμών στην
οθόνη
	ldi r24 ,0x0c                          ; ενεργοποίηση της οθόνης,
απόκρυψη του κέρσορα
	rcall lcd_command
	ldi r24 ,0x01                          ; καθαρισμός της οθόνης
	rcall lcd_command
	ldi r24 ,low(1530)
	ldi r25 ,high(1530)
	rcall wait_usec
	ldi r24 ,0x06                          ; ενεργοποίηση αυτόματης
αύξησης κατά 1 της διεύθυνσης
	rcall lcd_command                      ; που είναι αποθηκευμένη στον
μετρητή διευθύνσεων και
                                                   ; απενεργοποίηση
της ολίσθησης ολόκληρης της οθόνης
	ret
```

Στην παραπάνω άσκηση μετατρέπουμε δυαδικούς αριθμούς σε μορφή συμπληρώματος ως προς δύο σε ισοδύναμους δεκαδικούς αριθμούς με πρόσημο ακολουθώντας το δοσμένο διάγραμμα ροής.

4η Άσκηση

```
.include "m16def.inc"

.def temp = r18
.def minutes = r19
.def seconds = r20
.def minutes1 = r21
.def seconds1 = r22
.def ascii = r23

start:
	ldi r24, low(RAMEND)

	out SPL, r24
	ldi r24, high(RAMEND)
	out SPH, r24

	clr temp                                   ; set PORTA as input
	out DDRA, temp

	ldi r24, (1 << PD7) | (1 << PD6) | (1 << PD5) | (1 << PD4) | (1 << PD3)
| (1 << PD2)        ; set 6 MSBs as output
```

```
	out DDRD, r24
                                                                                ;
in PORTD

	call lcd_init                           ; initialize lcd screen
	ldi r24,'0'

	call lcd_data                           ; 00 MIN:00 SEC
	ldi r24,'0'
	call lcd_data
	ldi r24,' '
	call lcd_data
	ldi r24,'M'
	call lcd_data
	ldi r24,'I'
	call lcd_data
	ldi r24,'N'
	call lcd_data
	ldi r24,':'
	call lcd_data
	ldi r24,'0'
	call lcd_data
	ldi r24,'0'
	call lcd_data
	ldi r24,' '
	call lcd_data
	ldi r24,'S'
	call lcd_data
	ldi r24,'E'
	call lcd_data
	ldi r24,'C'
	call lcd_data

	ldi ascii, 0x30

loop:
	in temp, PINA
	sbrs temp, 0x00                                 ; check PA0

	jmp loop                                        ; if it's pressed then
skip instruction and start the stopwatch

beginning:                                              ; set minutes = seconds
= 0
	clr minutes
	ldi seconds, 1
	clr minutes1
	clr seconds1
	ldi r24, 0x02                           ; clear screen
	rcall lcd_command
	ldi r24,'0'

	call lcd_data                           ; 00 MIN:00 SEC
	ldi r24,'0'
	call lcd_data
	ldi r24,' '
	call lcd_data
	ldi r24,'M'
	call lcd_data
	ldi r24,'I'
	call lcd_data
	ldi r24,'N'
```

```
        call lcd_data
        ldi r24,':'
        call lcd_data
        ldi r24,'0'
        call lcd_data
        ldi r24,'0'
        call lcd_data
        ldi r24,' '
        call lcd_data
        ldi r24,'S'
        call lcd_data
        ldi r24,'E'
        call lcd_data
        ldi r24,'C'
        call lcd_data
        ldi r24, low(1000)                      ; delay 1 second
        ldi r25, high(1000)
        rcall wait_msec

loop1:
        in temp, PINA
        sbrc temp, 0x07                             ; check PA7
        jmp restart                                 ; if it's pressed then
restart the stopwatch
        sbrs temp, 0x00                             ; check PA0

        jmp loop1                                   ; if it's pressed then
skip instruction and start the stopwatch

        ldi r24, 0x02                           ; clear screen
        rcall lcd_command

        mov temp, minutes1
        add minutes1, ascii
        mov r24, minutes1
        mov minutes1, temp
        call lcd_data                           ; print first digit of minutes
to the screen

        mov temp, minutes
        add minutes, ascii
        mov r24, minutes
        mov minutes, temp
        call lcd_data                           ; print second digit of minutes
to the screen

        ldi r24,' '
        call lcd_data
        ldi r24,'M'
        call lcd_data
        ldi r24,'I'
        call lcd_data                           ; print "MIN:" to the screen
        ldi r24,'N'
        call lcd_data
        ldi r24,':'
        call lcd_data

        mov temp, seconds1                      ; print first digit of seconds
to the screen
        add seconds1, ascii
        mov r24, seconds1
        mov seconds1, temp
```

```
	call lcd_data

	mov temp, seconds			; print second digit of seconds
to the screen
	add seconds, ascii
	mov r24, seconds
	mov seconds, temp
	call lcd_data

	ldi r24,' '
	call lcd_data
	ldi r24,'S'
	call lcd_data				; print " SEC" to the screen
	ldi r24,'E'
	call lcd_data
	ldi r24,'C'
	call lcd_data

	ldi r24, low(1000)			; delay 1 second
	ldi r25, high(1000)
	rcall wait_msec

	cpi seconds, 9				; if second digit is 9
increase first digit and clear seconds
	brlo label0
	inc seconds1				; else just increase seconds
	ldi seconds, -1

label0:
	inc seconds

	cpi seconds1, 6				; check if 60 sec passed
	breq label1
	jmp loop1

label1:						; if 60 sec
passed increase second digit of minutes
	cpi minutes, 9
	brlo label2				; if second digit is 9
increase first digit and clear minutes
	inc minutes1
	ldi minutes, -1

label2:
	inc minutes

	clr seconds
	clr seconds1
	cpi minutes1, 6				; check if 60 minutes
passed
	breq label3				; if yes restart the
stopwatch
	jmp loop1

label3:
	jmp beginning

restart:
	in temp, PINA
	sbrs temp, 0x07				; check if PA7 was
pressed
	jmp beginning			; if not restart the stopwatch
```

```
	jmp restart 				; if yes wait


wait_usec:
	sbiw r24 ,1 				; 2 κύκλοι (0.250 μsec)
	nop 					; 1 κύκλος (0.125
μsec)
	nop 					; 1 κύκλος (0.125
μsec)
	nop 					; 1 κύκλος (0.125
μsec)
	nop 					; 1 κύκλος (0.125
μsec)
	brne wait_usec 				; 1 ή 2 κύκλοι (0.125 ή
0.250 μsec)
	ret 					; 4 κύκλοι (0.500
μsec)


wait_msec:
	push r24 				; 2 κύκλοι (0.250 μsec)
	push r25 				; 2 κύκλοι
	ldi r24 , low(998) 			; φόρτωσε τον καταχ. r25:r24 με
998 (1 κύκλος - 0.125 μsec)
	ldi r25 , high(998) 			; 1 κύκλος (0.125 μsec)
	rcall wait_usec 			; 3 κύκλοι (0.375 μsec),
προκαλεί συνολικά καθυστέρηση 998.375 μsec
	pop r25 				; 2 κύκλοι (0.250
μsec)
	pop r24 				; 2 κύκλοι
	sbiw r24 , 1 				; 2 κύκλοι
	brne wait_msec 				; 1 ή 2 κύκλοι (0.125 ή
0.250 μsec)
	ret 					; 4 κύκλοι (0.500
μsec)


lcd_data:
	sbi PORTD ,PD2 				; επιλογή του καταχωρήτη
δεδομένων (PD2=1)
	rcall write_2_nibbles 			; αποστολή του byte
	ldi r24 ,43 				; αναμονή 43μsec μέχρι
να ολοκληρωθεί η λήψη
	ldi r25 ,0 				; των δεδομένων από τον
ελεγκτή της lcd
	rcall wait_usec
	ret


lcd_command:
	cbi PORTD ,PD2 				; επιλογή του καταχωρητή
εντολών (PD2=1)
	rcall write_2_nibbles 			; αποστολή της εντολής και
αναμονή 39μsec
	ldi r24 ,39 				; για την ολοκλήρωση της
εκτέλεσης της από τον ελεγκτή της lcd.
	ldi r25 ,0 				; ΣΗΜ.: υπάρχουν δύο
εντολές, οι clear display και return home,
	rcall wait_usec 			; που απαιτούν σημαντικά
μεγαλύτερο χρονικό διάστημα.
	ret
```

```
write_2_nibbles:
       push r24                                ; στέλνει τα 4 MSB
       in r25 ,PIND                            ; διαβάζονται τα 4 LSB και τα
ξαναστέλνουμε
       andi r25 ,0x0f                          ; για να μην χαλάσουμε
την όποια προηγούμενη κατάσταση
       andi r24 ,0xf0                          ; απομονώνονται τα 4 MSB
και
       add r24 ,r25                            ; συνδυάζονται με τα
προϋπάρχοντα 4 LSB
       out PORTD ,r24                          ; και δίνονται στην
έξοδο
       sbi PORTD ,PD3                          ; δημιουργείται παλμός
Enable στον ακροδέκτη PD3
       cbi PORTD ,PD3                          ; PD3=1 και μετά PD3=0
       pop r24                                       ; στέλνει τα 4
LSB. Ανακτάται το byte.
       swap r24                                ; εναλλάσσονται τα 4 MSB
με τα 4 LSB
       andi r24 ,0xf0                          ; που με την σειρά τους
αποστέλλονται
       add r24 ,r25
       out PORTD ,r24
       sbi PORTD ,PD3                          ; Νέος παλμός Enable
       cbi PORTD ,PD3
       ret


lcd_init:
       ldi r24 ,40                             ; Όταν ο ελεγκτής της
lcd τροφοδοτείται με
       ldi r25 ,0                              ; ρεύμα εκτελεί την δική
του αρχικοποίηση.
       rcall wait_msec                         ; Αναμονή 40 msec μέχρι
αυτή να ολοκληρωθεί.
       ldi r24 ,0x30                           ; εντολή μετάβασης σε 8 bit
mode
       out PORTD ,r24                          ; επειδή δεν μπορούμε να
είμαστε βέβαιοι
       sbi PORTD ,PD3                          ; για τη διαμόρφωση
εισόδου του ελεγκτή
       cbi PORTD ,PD3                          ; της οθόνης, η εντολή
αποστέλλεται δύο φορές
       ldi r24 ,39
       ldi r25 ,0                              ; εάν ο ελεγκτής της
οθόνης βρίσκεται σε 8-bit mode
       rcall wait_usec                         ; δεν θα συμβεί τίποτα,
αλλά αν ο ελεγκτής έχει διαμόρφωση
                                                      ; εισόδου 4 bit

θα μεταβεί σε διαμόρφωση 8 bit
       ldi r24 ,0x30
       out PORTD ,r24
       sbi PORTD ,PD3
       cbi PORTD ,PD3
       ldi r24 ,39
       ldi r25 ,0
       rcall wait_usec
       ldi r24 ,0x20                           ; αλλαγή σε 4-bit mode
       out PORTD ,r24
       sbi PORTD ,PD3
       cbi PORTD ,PD3
```

```
       ldi r24 ,39
       ldi r25 ,0
       rcall wait_usec
       ldi r24 ,0x28                          ; επιλογή χαρακτήρων μεγέθους
5x8 κουκίδων
       rcall lcd_command                      ; και εμφάνιση δύο γραμμών στην
οθόνη
       ldi r24 ,0x0c                          ; ενεργοποίηση της οθόνης,
απόκρυψη του κέρσορα
       rcall lcd_command
       ldi r24 ,0x01                          ; καθαρισμός της οθόνης
       rcall lcd_command
       ldi r24 ,low(1530)
       ldi r25 ,high(1530)
       rcall wait_usec
       ldi r24 ,0x06                          ; ενεργοποίηση αυτόματης
αύξησης κατά 1 της διεύθυνσης
       rcall lcd_command                      ; που είναι αποθηκευμένη στον
μετρητή διευθύνσεων και
                                                      ; απενεργοποίηση

της ολίσθησης ολόκληρης της οθόνης
       ret
```

Στην τελευταία άσκηση υλοποιούμε ένα ψηφιακό χρονόμετρο με τη βοήθεια μετρητών και συναρτήσεων για την απεικόνιση χαρακτήρων στην οθόνη LCD.