Εργαστήριο Μικροϋπολογιστών

7η Εργαστηριακή Άσκηση

Άσκηση 1

```
.def temp=r16
.def given=r17
; ---- Αρχή τμήματος δεδομένων
.DSEG
	_tmp_: .byte 2
; ---- Τέλος τμήματος δεδομένων

.CSEG
	.org 0x0
	rjmp reset
	.org 0x10			; διάνυσμα διακοπής TIMER1
	rjmp ISR_TIMER1_OVF

reset:
	; Αρχικοποίηση της στοίβας.
	LDI temp, HIGH(RAMEND)	; Το άνω byte του τέλους της μνήμης
	OUT SPH, temp		; τίθεται στον stack pointer (high)
	LDI temp, LOW(RAMEND)	; κι όμοια το κάτω byte.
	OUT SPL, temp

	; Ενεργοποίηση διακοπής υπερχείλισης του TCNT1.
	ldi temp, (1<<TOIE1)
	out TIMSK, temp
	; Η συχνότητα αύξησης του TCNT1 τίθεται ίση με CLK/1024.
	ldi temp, (1<<CS12) | (0<<CS11) | (1<<CS10)
	out TCCR1B, temp

	; Αρχικοποίηση μεταβλητής _tmp_ .
	ldi temp, 0
	ldi r26, low(_tmp_)
	ldi r27, high(_tmp_)
	st X+, temp
	st X, temp
```

```
	; Αρχικοποίηση της PORTB για είσοδο συναγερμού.
	ldi temp, 0x00
	out DDRB, temp
	; Αρχικοποίηση της PORTA για  έξοδο συναγερμού.
	ldi temp, 0xff
	out DDRA, temp
	; Αρχικοποίηση της PORTC για το πληκτρολόγιο.
	ldi temp, 0xf0
	out DDRC, temp
	; Αρχικοποίηση της PORTD για την οθόνη LCD.
	ldi temp, 0xff
	out DDRD, temp
	; Αρχικοποίηση οθόνης.
	rcall lcd_init

	sei

	; 8MHz/1024Hz = 7812.5 = 1 sec

	; Άρα, για να έχουμε την ανάλογη καθυστέρηση,
	; υπολογίζουμε κάθε φορά σύμφωνα με τον τύπο 2^16-t*7812.5 (t: δευτερόλεπτα)
	; και μετατρέπουμε σε δεκαεξαδική μορφή

	; 5 sec: 2^16-5*7812.5 = 26473.5 = 0x6769


;; ΑΡΧΗ ΚΥΡΙΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ ;;
	ldi given, 1
start:
	in temp, PINB
	cpi temp, 0
	breq start          ; Όσο δεν ενεργοποιείται ο συναγερμός, επαναλαμβάνει.
	;; Καθαρίζεται η οθόνη.
	ldi given, 0              ; το given γίνεται 0
	ldi r24 ,0x01             ; καθαρισμός της οθόνης
	rcall lcd_command
	ldi r24 ,low(1530)        ; θέλει 1.53ms για να ξεσκαλώσει
	ldi r25 ,high(1530)
	rcall wait_usec
	;; Τίθεται ο μετρητής στα 5 δευτερόλεπτα.
	ldi r25, 0x67
	ldi r24, 0x69
	out TCNT1H, r25
	out TCNT1L, r24
	;; Διάβασμα του πληκτρολογίου, μέχρι να εισαχθεί ο κωδικός.
read_start:
	ldi r24 ,0x0e             ; ενεργοποίηση της οθόνης, εμφάνιση του κέρσορα
	rcall lcd_command
	ldi r24, 10                     ; Όσο δεν έχει διαβαστεί το '5'
	rcall scan_keypad_rising_edge    ; επανέρχεται εδώ.
	rcall keypad_to_ascii
```

```
	cpi r24, 0
	breq read_start
	push r24
	rcall lcd_data			; Τυπώνει μόνο όταν έχει πληκτρολογηθεί κάτι.
	pop r24
inter_start:
	cpi r24, '5'
	brne read_start
read_first:				; Έρχεται εδώ όταν έχει πληκτρολογηθεί '5' (5__).
	ldi r24, 10
	rcall scan_keypad_rising_edge
	rcall keypad_to_ascii
	cpi r24, 0
	breq read_first			; Περιμένει να πατηθεί κάποιο πλήκτρο.
	push r24
	rcall lcd_data
	pop r24
	cpi r24,'1'			; Αν είναι '1' προχωράει παρακάτω (51_),
	brne inter_start		; αλλιώς κάνει έλεγχο για '5', πριν πάει στο start.
read_second:
	ldi r24, 10
	rcall scan_keypad_rising_edge
	rcall keypad_to_ascii
	cpi r24, 0
	breq read_second		; Περιμένει να πατηθεί κάποιο πλήκτρο.
	push r24
	rcall lcd_data
	pop r24
	cpi r24, '2'			; Αν δεν είναι '2' ((512) (TO AM MOY! (LUL))),
	brne inter_start		; ελέγχει όμοια για '5'.
	; Αλλιώς "ξεκλειδώνει" το συναγερμό.

unlocked:
	ldi given, 1			; Το given τίθεται 1 για όταν χτυπήσει ο χρονιστής.
	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, 'A'			; Τύπωμα "ALARM OFF".
	rcall lcd_data
	ldi r24, 'L'
	rcall lcd_data
	ldi r24, 'A'
	rcall lcd_data
	ldi r24, 'R'
	rcall lcd_data
	ldi r24, 'M'
	rcall lcd_data
	ldi r24, ' '
```

```
       rcall lcd_data
       ldi r24, 'O'
       rcall lcd_data
       ldi r24, 'F'
       rcall lcd_data
       ldi r24, 'F'
       rcall lcd_data
       rjmp start                      ; Διαρκής επανάληψη.
;; ΤΕΛΟΣ ΚΥΡΙΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ ;;


;; ΑΡΧΗ ΡΟΥΤΙΝΑΣ ΕΞΥΠΗΡΕΤΗΣΗΣ ΧΡΟΝΙΣΤΗ ;;
;; Καλείται όταν τελειώνει ο μετρητής.
ISR_TIMER1_OVF:
; Σώσιμο των καταχωρητών.
       push temp
       in temp, SREG
       push temp

       cpi given, 1              ; Αν ο κωδικός έχει δοθεί (given=1),
       breq timoffret            ; απλώς επιστρέφει.
       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, 'A'              ; ΚΟΥΚΛΑ ΜΟΥ!!!
       rcall lcd_data
       ldi r24, 0xd7             ; Κωδικός για το Λ;
       rcall lcd_data
       ldi r24, 'E'
       rcall lcd_data
       ldi r24, 0xd8             ; Κωδικός για το Ξ;
       rcall lcd_data
       ldi r24, 'A'
       rcall lcd_data
       ldi r24, 'N'
       rcall lcd_data
       ldi r24, 0x7f             ; Κωδικός για το Δ;
       rcall lcd_data
       ldi r24, 'P'
       rcall lcd_data
       ldi r24, 'A'
       rcall lcd_data
       ldi r24, ' '
       rcall lcd_data
       ldi r24, '<'              ; ΗΘΕΛΑ ΝΑ ΦΤΙΑΞΩ ΚΑΡΔΟΥΛΑ ΜΕ CUSTOM
       rcall lcd_data            ; CHARACTER ΑΛΛΑ ΜΟΥ ΕΧΕΤΕ ΓΕΙΩΣΕΙ ΤΟΝ R/W.
       ldi r24, '3'
       rcall lcd_data
```

```
	ldi temp, 0xff		; Αρχικοποίηση σε ON
alarm_on:
	out PORTA, temp
	ldi r24 ,low(400)	; 0.4 sec ON
	ldi r25 ,high(400)
	rcall wait_msec
	com temp		; off
	out PORTA, temp
	ldi r24 ,low(100)	; 0.1 sec OFF
	ldi r25 ,high(100)
	rcall wait_msec
	com temp		; on για επαναφορά στη λούπα
	rjmp alarm_on		; Αναβοσβήνουν μόνιμα με συνολική περίοδο 0.5 sec.

; Επαναφορά των καταχωρητών και επιστροφή.
timoffret:
	pop temp
	out SREG, temp
	pop temp
	reti
;; ΤΕΛΟΣ ΡΟΥΤΙΝΑΣ ΕΞΥΠΗΡΕΤΗΣΗΣ ΧΡΟΝΙΣΤΗ ;;

;; ΑΡΧΗ ΡΟΥΤΙΝΩΝ ΟΘΟΝΗΣ ;;
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=0)
```

```
	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
	rcall wait_usec		; return home, 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
	rcall lcd_command	; διεύθυνσης  που είναι αποθηκευμένη στον μετρητή
				; διευθύνσεων και απενεργοποίηση της ολίσθησης
				; ολόκληρης της οθόνης
	ret
;; ΤΕΛΟΣ ΡΟΥΤΙΝΩΝ ΟΘΟΝΗΣ ;;
```

```
;; ΑΡΧΗ ΡΟΥΤΙΝΩΝ ΠΛΗΚΤΡΟΛΟΓΙΟΥ ;;
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
      pop r23             ; απόρριψε όσα πλήκτρα εμφανίζουν
      pop r22             ; σπινθηρισμό
      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
	ret			; είναι ‘1’) 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
;; ΤΕΛΟΣ ΡΟΥΤΙΝΩΝ ΠΛΗΚΤΡΟΛΟΓΙΟΥ ;;


;; ΑΡΧΗ ΡΟΥΤΙΝΩΝ ΧΡΟΝΟΚΑΘΥΣΤΕΡΗΣΗΣ ;;
;; Προκαλεί καθυστέρηση r25:r24 msec.
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)

;; Προκαλεί καθυστέρηση r25:r24 μsec.
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)
;; ΤΕΛΟΣ ΡΟΥΤΙΝΩΝ ΧΡΟΝΟΚΑΘΥΣΤΕΡΗΣΗΣ ;;
```

Άσκηση 2

```
.def input=r16
.def inputold=r17
.def temp=r18
.def counter=r19

	; Αρχικοποίηση της στοίβας.
	LDI temp, HIGH(RAMEND)    ; Το άνω byte του τέλους της μνήμης
	OUT SPH, temp             ; τίθεται στον stack pointer (high)
	LDI temp, LOW(RAMEND)     ; κι όμοια το κάτω byte.
	OUT SPL, temp

	; Αρχικοποίηση της PORTA για την είσοδο.
	ldi temp, 0x00
	out DDRA, temp
	; Αρχικοποίηση της PORTD για την οθόνη LCD.
	ldi temp, 0xff
	out DDRD, temp
	; Αρχικοποίηση οθόνης.
	rcall lcd_init

	ldi inputold, 0xff


;; ΑΡΧΗ ΚΥΡΙΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ ;;
start:
	in input, PINA
	cp input, inputold        ; Αν δεν έχει αλλάξει το input
	breq start                ; Άραξε
	mov inputold, input

	; Καθαρισμός της οθόνης.
	ldi r24 ,0x01
	rcall lcd_command
	ldi r24 ,low(1530)
	ldi r25 ,high(1530)
	rcall wait_usec

	; Τύπωμα των 8 ψηφίων του PINA.
	push input
	ldi counter, 8
show_loop:
	ldi r24,'0'               ; Αρχικά τοποθετείται το '0' στον r24.
	rol input
	brcc not_one              ; Αν το Carry είναι 1
	ldi r24,'1'               ; τοποθετείται το '1' στον r24.
not_one:
	rcall lcd_data            ; Τύπωμα του ψηφίου.
	dec counter
	brne show_loop            ; Επανάληψη 8 φορές (μία για κάθε ψηφίο).
```

```
	pop input

	ldi r24,'='		; Τύπωμα του '='.
	rcall lcd_data

	ldi r24, '+'		; Το r24 αρχικοποιείται σε '+'.
	sbrc input, 7
	ldi r24, '-'		; Το r24 γίνεται '-' αν είναι 1 το MSB της εισόδου.
	rcall lcd_data		; Εμφάνιση του προσήμου

	sbrc input, 7		; Αν η είσοδος ήταν αρνητική,
	com input		; παίρνουμε το συμπλήρωμα ως προς 1 (εντολή COM).

;; Παρατηρούμε ότι 00000000=+0 και 11111111=-0 (αφού το 2ο θα αντιστραφεί με COM
;; αλλά θα έχει περαστεί από πριν αρνητικό πρόσιμο), οπότε λήφθηκε υπόψην η
;; διπλή αναπαράσταση του 0.

; Ο input πλέον περιέχει την απόλυτη τιμή της εισόδου.
	ldi temp,0x00		; Ο temp θα γίνει 0xff αν τυπωθούν εκατοντάδες.
	cpi input, 100
	brlo decades
	subi input, 100
	ldi temp, 0xff		; Αποθηκεύεται στον temp ότι τυπώθηκαν εκατοντάδες.
	ldi r24, '1'		; Αν είναι μεγαλύτερο του 100,
	rcall lcd_data		; τυπώνει '1'.
decades:
	ldi r24, 0		; Στον r24 θα σχηματιστούν οι δεκάδες.
decades_loop:
	cpi input, 10
	brlo decades_pr		; Αν είναι μικρότερο από 10, πάει στο decades_pr.
	inc r24			; Αλλιώς αυξάνει τον r24 κατά 1
	subi input, 10		; και μειώνει την είσοδο κατά 10.
	rjmp decades_loop	; Επαναλαμβάνει μέχρι να ληφθούν όλες οι δεκάδες.
decades_pr:
	cpi r24, 0		; Ελέγχει αν οι δεκάδες είναι 0.
	brne decades_trpr	; Αν όχι, απλά τις τυπώνει.
	cpi temp,0x00		; Αν ναι, ελέγχει αν έχουν τυπωθεί εκατοντάδες
	breq monades		; κι αν όχι πάει κατ' ευθείαν στις μονάδες.
decades_trpr:
	ldi temp, 0x30
	add r24, temp		; Ascii-οποίηση των δεκάδων
	rcall lcd_data		; και τύπωμα τους.
monades:
	ldi r24, 0x30
	add r24, input		; Στον input έχουν απομείνει οι μονάδες,
	rcall lcd_data		; οπότε και τυπώνονται σε ASCII.

	rjmp start		; Διαρκής επανάληψη.
;; ΤΕΛΟΣ ΚΥΡΙΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ ;;
```

```
;; ΑΡΧΗ ΡΟΥΤΙΝΩΝ ΟΘΟΝΗΣ ;;
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
       andi r24 ,0xf0          ; κατάσταση 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=0)
       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
       rcall wait_usec         ; return home, που απαιτούν σημαντικά μεγαλύτερο
       ret                     ; χρονικό διάστημα.

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
       ldi r24 ,0x30           ; 8 bit
       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
	rcall lcd_command	; διεύθυνσης που είναι αποθηκευμένη στον μετρητή
				; διευθύνσεων και απενεργοποίηση της ολίσθησης
				; ολόκληρης της οθόνης
	ret
;; ΤΕΛΟΣ ΡΟΥΤΙΝΩΝ ΟΘΟΝΗΣ ;;

;; ΑΡΧΗ ΡΟΥΤΙΝΩΝ ΧΡΟΝΟΚΑΘΥΣΤΕΡΗΣΗΣ ;;
;; Προκαλεί καθυστέρηση r25:r24 msec.
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)

;; Προκαλεί καθυστέρηση r25:r24 μsec.
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)
;; ΤΕΛΟΣ ΡΟΥΤΙΝΩΝ ΧΡΟΝΟΚΑΘΥΣΤΕΡΗΣΗΣ ;;
```

Άσκηση 3

```
.def counter=r16
.def min_counter=r17
.def input=r18
.def temp=r19

reset:
      ; Αρχικοποίηση της στοίβας.
      LDI temp, HIGH(RAMEND)    ; Το άνω byte του τέλους της μνήμης
      OUT SPH, temp             ; τίθεται στον stack pointer (high)
      LDI temp, LOW(RAMEND)     ; κι όμοια το κάτω byte.
      OUT SPL, temp

      ; Αρχικοποίηση της PORTB για την είσοδο.
      ldi temp, 0x00
      out DDRA, temp
      ; Αρχικοποίηση της PORTD για την οθόνη LCD.
      ldi temp, 0xff
      out DDRD, temp
      ; Αρχικοποίηση οθόνης.
      rcall lcd_init

      clr counter               ; Αρχικοποίηση     counters
      clr min_counter

;; ΑΡΧΗ ΚΥΡΙΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ ;;
start:
;; Λεπτά
    cpi min_counter, 0x3c       ; σύγκριση με 60 *ΠΡΙΝ* την απεικόνιση,
    brne just_print             ; αφού inc-> στο προηγούμενο loop. Αν <60, συνέχισε,
    clr min_counter             ; αλλιώς μηδένισε τον min_counter (αφού πέρασε 1
                                ;΄ώρα)
just_print:
      rcall printer
wait_1_sec:
      ldi r24,low(1000)         ; Ξεκινάει τη μέτρηση του δευτερολέπτου,
      ldi r25,high(1000)        ; ελέγχοντας ταυτόχρονα,
      rcall check_wait_msec     ; αν είναι πατημένο το PB0.
      rjmp start                ; Διαρκής επανάληψη για χρονομέτρηση.
;; ΤΕΛΟΣ ΚΥΡΙΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ ;;


;; ΑΡΧΗ ΜΗ ΕΤΟΙΜΑΤΖΙΔΙΚΩΝ ΡΟΥΤΙΝΩΝ ;;
printer:
      rcall lcd_init
  mov input, min_counter
  rcall prnt_time               ; Τύπωμα λεπτών
  ldi r24, ' '
  rcall lcd_data
  ldi r24, 'M'
  rcall lcd_data
```

```
  ldi r24, 'I'
  rcall lcd_data
  ldi r24, 'N'
  rcall lcd_data
       ldi r24, ':'
       rcall lcd_data
;; Δευτερόλεπτα
       mov input, counter
       rcall prnt_time           ; Τύπωμα δευτερολέπτων και αύξηση
       inc counter               ; του μετρητή δευτερολέπτων μετά την απεικόνιση.
       cpi counter, 0x3c         ; Σύγκριση με 60 *ΜΕΤΑ* την απεικόνιση.
       brlo cont_sec             ; Αν μικρότερο, απλά συνέχισε.
       clr counter               ; Αλλιώς, μηδένισε τον counter,
       inc min_counter           ; και αύξησε τον min_counter (αφού πέρασε 1 λεπτό).
cont_sec:
  ldi r24, ' '
  rcall lcd_data
  ldi r24, 'S'
  rcall lcd_data
  ldi r24, 'E'
  rcall lcd_data
  ldi r24, 'C'
  rcall lcd_data
       ret

prnt_time:                       ; μετατρέπει την τιμή του input σε bcd και την
τυπώνει
       ldi r24, 0                ; Στον r24 θα σχηματιστούν οι δεκάδες.
       ldi temp, 0x30
decades_loop:
       cpi input, 10
       brlo print_all            ; Αν είναι μικρότερο από 10, πάει στο print_all.
       inc r24                   ; Αλλιώς αυξάνει τον r24 κατά 1,
       subi input, 10            ; και μειώνει την είσοδο κατά 10.
       rjmp decades_loop         ; Επαναλαμβάνει μέχρι να ληφθούν όλες οι δεκάδες.
print_all:
       add r24, temp             ; Ascii-οποίηση των δεκάδων
       rcall lcd_data            ; και τύπωμα τους.
       add input,temp
       mov r24, input            ; Στον input έχουν απομείνει οι μονάδες,
       rcall lcd_data            ; οπότε και τυπώνονται σε ASCII.
  ret

;; Προκαλεί καθυστέρηση r25:r24 msec, καλώντας την check_wait_2usec.
check_wait_msec:
       push r24                  ; 2 κύκλοι (0.250 μsec)
       push r25                  ; 2 κύκλοι
       ldi r24 , low(499)        ; φόρτωσε τον καταχ. r25:r24 με 998 (1 κύκλος-0.125
                                 ; μsec)
       ldi r25 , high(499)       ; 1 κύκλος (0.125 μsec)
       rcall check_wait_2usec    ; 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 check_wait_msec     ; 1 ή 2 κύκλοι (0.125 ή 0.250 μsec)
       ret                      ; 4 κύκλοι (0.500 μsec)

;; Η check_wait_msec έχει ίδια κατασκευή με την απλή wait_msec, με όρισμα όμως,
;; το μισό της απλής wait_msec. Θα δούμε γιατί.

;; Η check_wait_2usec προκαλεί καθυστέρηση, ελέγχοντας για πάτημα των PB7 & PB0.
;; Αν πατηθεί το PB7 φεύγει για το reset.
;; Αν αφεθεί το PB0, μένει στην check_wait_2usec.
;; ΣΗΜΕΙΩΣΗ: η εντολή rjmp έχει μήκος μίας λέξης, οπότε όταν σκιπάρεται
;;           και στις 2 περιπτώσεις, παρόλο που η ίδια θα έκανε 2 κύκλους,
;;           το sbrc θα μετρήσει 2 κύκλους.
;; Σύνολο 1+2+2=5 κύκλοι προστέθηκαν στο βρόγχο. Έχουμε 1 παραπάνω,οπότε δεν
;; μπορούμε να αντικαταστήσουμε απλά τις nop με τις εντολές για έλεγχο.
;; Θα εργαστούμε ως εξής: Θα διπλασιάσουμε το σύνολο των κύκλων, από 8 σε 16
;; (προσθέτοντας άλλες 3 nop (σύνολο 7 τώρα) πέρα από τις εντολές ελέγχου)
;; και θα ρυθμίσουμε την check_wait_msec που θα κατασκευάσουμε ώστε να τρέχει
;; την check_wait_2usec τις μισές φορές σε σχέση με τη η ρουτίνα που δίδεται
;; από τη θεωρία.
check_wait_2usec:
       in temp, PINB            ; Διάβασμα της θύρας B. 1 κύκλος       (0.125 μsec)
       sbrs temp , 7            ; Έλεγχος αν είναι πατημένο το PB7.    (0.250 μsec)
       rjmp cont_c
       mov temp,counter
       add temp,min_counter
       cpi temp,0
       breq check_wait_2usec
       clr counter
       clr min_counter
       rcall printer
       rjmp check_wait_2usec
cont_c:
       sbrs temp , 0            ; Έλεγχος αν είναι πατημένο το PB0.    (0.250 μsec)
       rjmp check_wait_2usec    ; Αν όχι, επανάληψη, (2 κύκλοι αν είναι πατημένο).
       sbiw r24 , 1             ; 2 κύκλοι                             (0.250 μsec)
       nop
       nop
       nop
       nop
       nop                      ; Προσθέτουμε 3 nop.
       nop                      ; Άρα σύνολο 8+5+3=16 κύκλοι στον βρόγχο.
       nop                      ; Και 15 όταν το brne γίνει false (τελευταία επαν.)
       brne check_wait_2usec    ; 1 ή 2 κύκλοι (0.125 ή 0.250 μsec)
       ret                      ; 4 κύκλοι (0.500 μsec)

;; Συνολικά η ρουτίνα θα προκαλεί καθυστέρηση
;; [16*0.125*(998-1)]+[15*0.125+0.5]  = 1996.375μs για είσοδο 998, άρα
;; [16*0.125*(499-1)]+[15*0.125+0.5] =   998.375μs για ε΄ισοδο 998/2=499.
;; ΤΕΛΟΣ ΜΗ ΕΤΟΙΜΑΤΖΙΔΙΚΩΝ ΡΟΥΤΙΝΩΝ ;;
```

```
;; ΑΡΧΗ ΡΟΥΤΙΝΩΝ ΟΘΟΝΗΣ ;;
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
      andi r24 ,0xf0          ; κατάσταση, 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=0)
      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
      rcall wait_usec         ; return home, που απαιτούν σημαντικά μεγαλύτερο
      ret                     ; χρονικό διάστημα.

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
	rcall lcd_command	; διεύθυνσης που είναι αποθηκευμένη στον μετρητή
				; διευθύνσεων και απενεργοποίηση της ολίσθησης
	ret			; ολόκληρης της οθόνης
;; ΤΕΛΟΣ ΡΟΥΤΙΝΩΝ ΟΘΟΝΗΣ ;;

;; ΑΡΧΗ ΡΟΥΤΙΝΩΝ ΧΡΟΝΟΚΑΘΥΣΤΕΡΗΣΗΣ ;;
;; Προκαλεί καθυστέρηση r25:r24 msec.
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)

;; Προκαλεί καθυστέρηση r25:r24 μsec.
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)
;; ΤΕΛΟΣ ΡΟΥΤΙΝΩΝ ΧΡΟΝΟΚΑΘΥΣΤΕΡΗΣΗΣ ;;
```