Εργαστήριο Μικροϋπολογιστών
8η Εργαστηριακή άσκηση

Ομάδα Γ04
Συνεργάτες:
- **Σκούλος Ραφαήλ Α.Μ: 03112404**
- **Αναστάσης Σταθόπουλος Α.Μ: 03112140**
- **Τζίνης Ευθύμιος Α.Μ: 03112007**

Άσκηση 1

Σε αυτή την άσκηση υλοποιήσαμε μία ρουτίνα η οποία κάθε φορά διαβάζει απο το keypad έναν χαρακτήρα από τους 16 και το εμφανίζει στην LCD οθόνη.

```
.dseg
_tmp_: .byte 2

.cseg
.include "m16def.inc"
.org 0x00
rjmp start

start:

ldi r26,low(RAMEND)           ;initialize stack pointer
out SPL,r26
ldi r26,high(RAMEND)
out SPH,r26
ldi r24 ,(1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
out DDRC ,r24                 ; 4x4 keypad output initiallization
ldi r24,0x00
ldi r25,0x00
ldi r26 ,low(_tmp_)           ;r26:r27 makes thw register X
ldi r27 ,high(_tmp_)
st X+ ,r24                    ;store in the address of X zero
st X ,r25
ser r24
out DDRD ,r24
rcall lcd_init                ;initialize the screen
ldi r24,'N'
rcall lcd_data
ldi r24,'O'
rcall lcd_data
ldi r24,'N'
rcall lcd_data
ldi r24,'E'
rcall lcd_data

reading:

ldi r24, 0x14                 ;r24 has the bouncing time lets say 20ms
rcall scan_keypad_rising_edge ;read the first num from keyb
rcall keypad_to_ascii         ;turn it to ascii and get the result in r24
cpi r24, 0x00                 ;r24 has the result, if r24==0 then we have no change
breq reading                  ;then check again // else we have to change the value that
we are showing
                              ;(r24 has the character that we want)
mov r27,r24
ldi r24 ,0x01                 ; clear display
rcall lcd_command

ldi r24 ,low(1530)
ldi r25 ,high(1530)
rcall wait_usec
mov r24,r27
rcall lcd_data                ;r24 has the proper data value
rjmp reading                  ;continuous functionality
```

```
;=====================================LCDROUTINES=====================================

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



ascii_to_lcd:                ;result in ascii in r24
ldi r25, 0b00101010
cpi r24 ,'*'
breq exit_show_char

ldi r25, 0b00100011
cpi r24 ,'#'
breq exit_show_char

ldi r25, 0b01000100
cpi r24 ,'D'
breq exit_show_char

ldi r25, 0b00110111
cpi r24 ,'7'
breq exit_show_char

ldi r25, 0b00111000
cpi r24 ,'8'
breq exit_show_char

ldi r25, 0b00111001
cpi r24 ,'9'
breq exit_show_char

ldi r25, 0b01000011
cpi r24 ,'C'
breq exit_show_char

ldi r25, 0b00110101
cpi r24 ,'5'
breq exit_show_char

ldi r25, 0b00110110
cpi r24 ,'6'
breq exit_show_char

ldi r25, 0b01000010
cpi r24 ,'B'
breq exit_show_char

ldi r25, 0b00110001
cpi r24 ,'1'
breq exit_show_char

ldi r25, 0b00110010
cpi r24 ,'2'
breq exit_show_char

ldi r25, 0b00110011
cpi r24 ,'3'
breq exit_show_char
```

```
ldi r25, 0b01000001
cpi r24 ,'A'
breq exit_show_char

ldi r25, 0b00110000
cpi r24 ,'0'
breq exit_show_char

ldi r25, 0b00110100
cpi r24 ,'4'
breq exit_show_char

ldi r25, 0b00000000
exit_show_char:

ret

;show_char:

;====================================KEYPAD=============================================
scan_row:

;nop
ldi r25 ,0x08
back_:

lsl r25
dec r24
brne back_
out PORTC ,r25
nop
nop
in r24 ,PINC
andi r24 ,0x0f
ret
scan_keypad:

;nop
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:

;nop
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 ,low(_tmp_)
ldi r27 ,high(_tmp_)
```

```
ld r23 ,X+
ld r22 ,X
st X ,r24
st -X ,r25
com r23
com r22
and r24 ,r22
and r25 ,r23
ret

keypad_to_ascii:              ;routine to tranduce the result in r24 to an ascii code
movw r26 ,r24                 ;in order to compare it afterwards
ldi r24 ,'*'
sbrc r26 ,0
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 ,'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
ldi r24 ,'0'
sbrc r26 ,1
ret
ldi r24 ,'4'
sbrc r27 ,0
ret
clr r24
ret

;==============================DELAYROUTINES=============================================
wait_usec:

sbiw r24 ,1                   ;2 cycle (0.250 micro sec)
nop                           ;1 cycle (0.125 micro sec)
nop
```

```
nop
nop
brne wait_usec                ;1 or 2 cycles (0.125 or 0.250 micro sec)
ret                           ;4 cycles (0.500 micro sec)

wait_msec:

push r24
push r25
ldi r24 , low(998)
ldi r25 , high(998)
rcall wait_usec
pop r25
pop r24
sbiw r24 , 1
brne wait_msec
ret
```

Άσκηση 2

Σε αυτή την άσκηση υλοποιήσαμε μία ρουτίνα η οποία κάθε φορά διαβάζει απο τους αισθητήρες A0-A7 αν υπάρχει έστω και ένας άσσος και μέσα σε 4 δευτερόλεπτα αν ισχύει η πρώτη συνθήκη πρέπει να εισαψθεί στο keypad ο κωδικός C04 για να εμφανιστεί στην οθόνη ALARM OFF ενώ αν δεν προλάβουμε να εισάγουμε αυτό τον κωδικό ανάβει στην οθόνη το ALARM ON μαζί με αναβόσβημα των leds PB0-PB7.

```
.dseg
_tmp_: .byte 2

.cseg
.include "m16def.inc"
                              ;suxnotita tou EasyAVR6 = 8MHz ara auksisis tou TCNT1
8MHz/1024=7812.5Hz
.equ Hvalue=high(34286)       ;thelw yperxeilish meta apo 4 sec ara 4*7812.5=31250
kyklous
.equ Lvalue=low(34286)        ;ara arxikh timh 65536-31250 = 34286
.def flag=r17                 ;flag if we have have entered the right code
.def alarm=r16                ;flag if we have triggered the alarm

.org 0x00
rjmp start
reti
.org 0x010
rjmp TIMER1                   ;routine if timer1 overflows
reti

start:

ldi r26,low(RAMEND)           ;initialize stack pointer
out SPL,r26
ldi r26,high(RAMEND)
out SPH,r26

clr r24
out DDRA,r24                  ; PORTA input
ser r24
out DDRB,r24                  ;PORTB=output
ldi r24,0xCF
out DDRD,r24

ldi r23 ,(1<<TOIE1)           ;ennable interrupt of overflowing TCNT1
out TIMSK ,r23
ldi r23 ,(1<<CS12) | (0<<CS11) | (1<<CS10) ;timer frequency counting CLK/1024
out TCCR1B ,r23

ser r24
ldi r24 ,(1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
out DDRC ,r24                 ; 4x4 keypad output initiallization
ldi r24,0x00
ldi r25,0x00
ldi r26 ,low(_tmp_)           ;r26:r27 makes thw register X
```

```
ldi r27 ,high(_tmp_)
st X+ ,r24                    ;store in the address of X zero
st X ,r25

call lcd_init                 ;initialize the screen



siesta:

in r27,PINA
cpi r27,0
breq siesta                   ;if we dont have an incoming person then wait again and
check

incoming:

ldi r23,Hvalue                ;Initialize TCNT1
out TCNT1H,r23                ;overflowing in 4sec
ldi r23,Lvalue
out TCNT1L,r23                ;we have to enter the code C04 in 4 seconds or else we have
an intruder
sei
enter_again:

ldi r24 ,0x0F                 ; display on ,cursor on, blinking on
rcall lcd_command
ldi flag,0                    ;in the beginning we have not entered the right code
ldi alarm,0                   ;alarm is not triggered
;sei
check_C:

ldi r24, 0x14
rcall scan_keypad_rising_edge ;read the first num from keyb
rcall keypad_to_ascii         ;turn it to ascii and get the result in r24
cpi r24, 0x00                 ;r24 has the result, if r24 == 0 means no input so loop
again
breq check_C                  ;else check again
cpi r24,'C'
brne show_first               ;if we have not entered C we can show it and check the next
one
mov flag,1                    ;else we update the flag and put it 1 so first bit is OK

show_first:

rcall lcd_data

check_0:

ldi r24, 0x14
rcall scan_keypad_rising_edge ;read the first num from keyb
rcall keypad_to_ascii         ;turn it to ascii and get the result in r24
cpi r24, 0x00                 ;r24 has the result, if r24 == 0 means no input so loop
again
breq check_0                  ;else check again
cpi r24,'0'
brne show_second              ;if we have not entered 0 we can show it and check the next
one
ori flag,2                    ;else we update the flag and put 1 in flag bit 01

show_second:

rcall lcd_data

check_4:

ldi r24, 0x14
rcall scan_keypad_rising_edge ;read the first num from keyb
rcall keypad_to_ascii         ;turn it to ascii and get the result in r24
cpi r24, 0x00                 ;r24 has the result, if r24 == 0 means no input so loop
again
```

```
breq check_4                ;else check again
cpi r24,'4'
brne show_third             ;if we have not entered 4 we can show it and check the next
one
ori flag,4                  ;else we update the flag and put 1 in flag bit 02

show_third:

rcall lcd_data

check_the_code:

mov r24,flag
cpi r24,7                   ;if we have entered the code right we show ALARM OFF
breq welcome
code_again:

rcall lcd_init
rjmp enter_again
welcome:

call alarm_off
rjmp siesta


;=====================================LCDROUTINES=====================================

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

;==================ALARMROUTINES============================================
alarm_on:

mov r24,alarm               ;r24<--flag of the alarm
cpi r24,1
breq exit_alarm             ;if we have triggered it before dont do it again
                            ;else trigger the alarm
ldi r24,'A'
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,'N'
rcall lcd_data
ldi r24 ,0x0C               ; display on ,cursor on, blinking on
rcall lcd_command


loop_alarm_on:

ser r26                     ;turn on and off the lights with period ~0.2sec
out PORTB,r26
ldi r24,low(100)            ;r25:r24 = 100
```

```
ldi r25,high(100)
rcall wait_msec               ;delay 0.1 sec
clr r26
out PORTB,r26
ldi r24,low(100)              ;r25:r24 = 100
ldi r25,high(100)
rcall wait_msec               ;delay 0.1 sec
rjmp loop_alarm_on

exit_alarm:

ldi alarm,1                   ;update the flag and return
ret

alarm_off:

ldi r24,'A'
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
ldi r24 ,0x0C                 ; display on ,cursor on, blinking on
rcall lcd_command

alarm_off_loop:

rjmp alarm_off_loop
ret

;================================================================================

;==================READFROMTHEKEYPADROUTINES=====================================

scan_row:

;nop
ldi r25 ,0x08
back_:

lsl r25
dec r24
brne back_
out PORTC ,r25
nop
nop
in r24 ,PINC
andi r24 ,0x0f
ret
scan_keypad:

;nop
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:

;nop
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 ,low(_tmp_)
ldi r27 ,high(_tmp_)
ld r23 ,X+
ld r22 ,X
st X ,r24
st -X ,r25
com r23
com r22
and r24 ,r22
and r25 ,r23
ret


keypad_to_ascii:              ;routine to tranduce the result in r24 to an ascii code in
order to compare it afterwards
movw r26 ,r24
ldi r24 ,'*'
sbrc r26 ,0
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 ,'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
ldi r24 ,'0'
sbrc r26 ,1
ret
ldi r24 ,'4'
sbrc r27 ,0
ret
clr r24
ret
;================================================================================



;=====================TIMEROVERFLOWINGROUTINE=========================================
TIMER1:

mov r24,flag
cpi r24,7                    ;if we have entered the code right we show ALARM OFF
brne intruder_timer
welcome_timer:

rcall alarm_off
rjmp exit_timer
intruder_timer:

rcall alarm_on               ;else ring the alarm
rjmp exit_timer

exit_timer:

reti
;================================================================================

;=======================DELAYROUTINES=============================================
wait_usec:

sbiw r24 ,1                  ;2 cycle (0.250 micro sec)
nop                          ;1 cycle (0.125 micro sec)
nop
nop
nop
brne wait_usec               ;1 or 2 cycles (0.125 or 0.250 micro sec)
ret                          ;4 cycles (0.500 micro sec)

wait_msec:

push r24
push r25
ldi r24 , low(998)
ldi r25 , high(998)
rcall wait_usec
pop r25
pop r24
sbiw r24 , 1
brne wait_msec
ret
```

Άσκηση 3

Σε αυτή την άσκηση υλοποιήσαμε μία ρουτίνα η οποία κάθε φορά απεικονίζει στο LCD display την δυαδική τιμή (σε συμπλήρωμα ως προς 2) που διαβάζει από την θύρα PORTB και τη δεκαδική τιμή του σε μορφή τριών ψηφίων με το πρόσημο

```
.include "m16def.inc"
.def hundreds=r17
.def decades=r18
.def units=r19
.def counter=r20

ldi r26,low(RAMEND)           ;initialize stack pointer
out SPL,r26
ldi r26,high(RAMEND)
out SPH,r26
clr r26
out DDRB,r26                  ;set PINB as input
ldi r24,0xfc
out DDRD ,r24                 ;set PORTD as output
call lcd_init                 ;initialize the lcd screen
start:

in r26,PINB                   ;read the input
mov r23,r26
backagain:

ldi r24 ,0x01                 ; clear display
rcall lcd_command
ldi r24 ,low(1530)
ldi r25 ,high(1530)
rcall wait_usec
mov r28,r26                   ;r28 has the value of the input
ldi counter,8
loopa:

mov r30,r28                   ;loop to show the binary number we read to the lcd screen
andi r30,0x80
cpi r30,0x80
brne not_equal
ldi r24,'1'
jmp show
not_equal:

ldi r24,'0'
show: call lcd_data
lsl r28
dec counter
cpi counter,0
brne loopa

ldi r24,'='                   ;show '=' to the lcd screen
call lcd_data

mov r27,r26
andi r26,0x80
cpi r26,0x80                  ;check if the value we read is negative(check if the 7-bit
is equal to 1)
brne positive
ldi r24,'-'                   ;if so get the compliment of 2 of the value and show '-' to
the lcd screen
call lcd_data
neg r27
jmp next
positive:

ldi r24,'+'                   ;if not just show '+' to the lcd screen
call lcd_data
next:                         ;get the bcd value of the binary number we read
ldi hundreds,0
ldi decades,0
ldi units,0
```

```
cpi r27,100
brlo hundreds_ok
ldi hundreds,1
subi r27,100
hundreds_ok:

cpi r27,10
brlo decades_ok
inc decades
subi r27,10
jmp hundreds_ok
decades_ok:

mov units,r27

ldi r21,0x30
add hundreds,r21              ;show the number of hundreds to the lcd screen
mov r24,hundreds
call lcd_data

ldi r21,0x30
add decades,r21               ;show the number of decades to the lcd screen
mov r24,decades
call lcd_data

ldi r21,0x30
add units,r21                 ;show the number of units to the lcd screen
mov r24,units
call lcd_data
wait: in r26,PINB
cp r26,r23
breq wait
mov r23,r26
jmp backagain                 ;continuous functionality

;====================================LCDROUTINES=====================================

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

;===================================Delay routines================================
wait_usec:

sbiw r24 ,1                 ;2 cycle (0.250 micro sec)
nop                         ;1 cycle (0.125 micro sec)
nop
nop
nop
brne wait_usec              ;1 or 2 cycles (0.125 or 0.250 micro sec)
ret                         ;4 cycles (0.500 micro sec)

wait_msec:

push r24
push r25
ldi r24 , low(998)
ldi r25 , high(998)
rcall wait_usec
pop r25
pop r24
sbiw r24 , 1
brne wait_msec
ret
```