Εργαστήριο Μικροϋπολογιστών
Θοδωρής Παπαρρηγόπουλος
el18040
Ομάδα 21

4ο Εργαστήριο

Άσκηση 1

```
.DSEG
_tmp_: .byte 2
; ---- ÔÝëïò ôìÞìáôïò äåäïìÝíùí

.CSEG
.include "m16def.inc"

.def GPreg1=r18 ;general purpose register1
.def GPreg2=r19 ;general purpose register2
.def counter=r20 ;general purpose counter
.def Cx=r17
.def level=r16
.def on_off=r21
.def lcd_flag=r28

.macro CHECK_IF_ZERO
;macro prokeimenou na mhn grafoume sunexeia
;ean einai mhden oi r25,r24
	cpi r24,0x00

	brne END_NOT_EQUAL
	cpi r25,0x00

	brne END_NOT_EQUAL

	ldi GPreg1,0x01 ;true
	rjmp END

	END_NOT_EQUAL:
	ldi GPreg1,0x00 ;false

	END:
.endm

.macro CHECK_FOR_VALID_r24

	cpi r24,@0
	brne NOT_EQUAL

	ldi GPreg2,0x01 ;true
	rjmp END1

	NOT_EQUAL:
	ldi GPreg2,0x00 ;false

	END1:
.endm

.macro SET_LEDS_ON
```

```
push r18
ser r18
out PORTB,r18
pop r18
.endm

.macro SET_LEDS_OFF
push r20
clr r20
out PORTB,r20
pop r20
.endm



.macro WELCOME_USER

	rcall lcd_init_sim

	push r25
	push r24

	ldi lcd_flag, 0x02

	clr r24
	out TIMSK, r24

	cpi level, 0x04
	brlt MHN_SBHSA_OLA
	ldi r25, 0x80
	out PORTB, r25
	rjmp lets_write

	MHN_SBHSA_OLA:
	in r25, PINB
	ori r25, 0x80
	out PORTB, r25

	lets_write:

	ldi r24,'W'
	rcall lcd_data_sim ; áðïóôïëÞ åíüò byte äåäïìÝíùí óôïí åëåãêôÞ ôçò ïèüíçò lcd
	ldi r24,'E'
	rcall lcd_data_sim
	ldi r24,'L'
	rcall lcd_data_sim
	ldi r24,'C'
	rcall lcd_data_sim
	ldi r24,'O'
	rcall lcd_data_sim
	ldi r24,'M'
	rcall lcd_data_sim
	ldi r24,'E'
	rcall lcd_data_sim
	ldi r24,' '
	rcall lcd_data_sim
	ldi r24,'2'
	rcall lcd_data_sim
	ldi r24,'1'
	rcall lcd_data_sim
```

```
        ldi r24,low(4000)
        ldi r25,high(4000)
        rcall wait_msec                 ; DELAY 4 SECONDS (MACRO)

        in r25, PINB
        andi r25, 0x7F
        out PORTB, r25

        pop r24

        ldi r25, (1 << TOIE1)
        out TIMSK, r25

        pop r25

        ldi lcd_flag, 0x02
.endm

.macro WRONG_PASSWORD
        push r25
        push r24

        in r24, PINB
        ori r24, 0x80
        out PORTB, r24

        ldi r24,low(500)
        ldi r25,high(500)
        rcall wait_msec

        in r24, PINB
        andi r24, 0x7F
        out PORTB, r24

        ldi r24,low(500)
        ldi r25,high(500)
        rcall wait_msec


        in r24, PINB
        ori r24, 0x80
        out PORTB, r24

        ldi r24,low(500)
        ldi r25,high(500)
        rcall wait_msec


        in r24, PINB
        andi r24, 0x7F
        out PORTB, r24

        ldi r24,low(500)
        ldi r25,high(500)
        rcall wait_msec

        in r24, PINB
        ori r24, 0x80
        out PORTB, r24
```

```
        ldi r24,low(500)
        ldi r25,high(500)
        rcall wait_msec

        in r24, PINB
        andi r24, 0x7F
        out PORTB, r24

        ldi r24,low(500)
        ldi r25,high(500)
        rcall wait_msec


        in r24, PINB
        ori r24, 0x80
        out PORTB, r24

        ldi r24,low(500)
        ldi r25,high(500)
        rcall wait_msec


        in r24, PINB
        andi r24, 0x7F
        out PORTB, r24

        ldi r24,low(500)
        ldi r25,high(500)
        rcall wait_msec

        ldi lcd_flag, 0x02

        pop r24
        pop r25
.endm


.macro GAS_DETECTED

        push r24

        cpi lcd_flag, 0x00
        breq END_GAS_DETECTED
        rcall lcd_init_sim
        ldi lcd_flag, 0x00

        ldi r24,'G'
        rcall lcd_data_sim ; áðïóôïëÞ åíüò byte äåäïìÝíùí óôïí åëåãêôÞ ôçò ïèüíçò lcd
        ldi r24,'A'
        rcall lcd_data_sim
        ldi r24,'S'
        rcall lcd_data_sim
        ldi r24,' '
        rcall lcd_data_sim
        ldi r24,'D'
        rcall lcd_data_sim
        ldi r24,'E'
        rcall lcd_data_sim
        ldi r24,'T'
```

```
        rcall lcd_data_sim
        ldi r24,'E'
        rcall lcd_data_sim
        ldi r24,'C'
        rcall lcd_data_sim
        ldi r24,'T'
        rcall lcd_data_sim
        ldi r24,'E'
        rcall lcd_data_sim
        ldi r24,'D'
        rcall lcd_data_sim
        ldi r24,'!'
        rcall lcd_data_sim

        END_GAS_DETECTED:

        pop r24
.endm

.macro CLEAR
        push r24

        cpi lcd_flag, 0x01
        breq END_CLEAR
        rcall lcd_init_sim
        ldi lcd_flag, 0x01

        ldi r24,'C'
        rcall lcd_data_sim ; áðïóôïëÞ åíüò byte äåäïìÝíùí óôïí åëåãêôÞ ôçò ïèüíçò lcd
        ldi r24,'L'
        rcall lcd_data_sim
        ldi r24,'E'
        rcall lcd_data_sim
        ldi r24,'A'
        rcall lcd_data_sim
        ldi r24,'R'
        rcall lcd_data_sim

        END_CLEAR:

        pop r24
.endm

.macro TIMER1_INIT
        push r24

        ldi r24 ,(1<<TOIE1) ; åíåñãïðïßçóç äéáêïðÞò ôõðùí÷åßéóçò ôïõ ìåôñçôÞ TCNT1
        out TIMSK ,r24 ; ãéá ôïí timer1
        ldi r24 ,(1<<CS12) | (0<<CS11) | (1<<CS10) ; CK/1024
        out TCCR1B ,r24
        ldi r24,0xFC ; áñ÷éêïðïßçóç ôïõ TCNT1
        out TCNT1H ,r24 ; ãéá õðåñ÷åßéóç ìåôÜ áðü 5 sec
        ldi r24 ,0xF3
        out TCNT1L ,r24
        clr r24

        pop r24
.endm

.macro ADC_INIT
```

```
	push r24

	ldi r24, 0x40
	out ADMUX, r24
	ldi r24, (1<<ADEN)|(1<<ADIE)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)
	out ADCSRA, r24
	clr r24

	pop r24
.endm



.org 0x00
rjmp start
.org 0x10
rjmp TIMER1_INTERRUPT
.org 0x1C
rjmp ADC_INTERRUPT


; Replace with your application code
start:
	ldi r24, low(RAMEND)
		out SPL, r24

	ldi r24, high(RAMEND)
		out SPH, r24


	ser r24 ; r24 = FF
	out DDRB, r24 ; initialize  port b
	out DDRD, r24 ; and d for output


	clr r24
	ldi r24, (1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4) ; èÝôåé ùò åîüäïõò ôá 4 MSB
	out DDRC, r24 ; ôçò èýñáò PORTC

	ldi lcd_flag, 0x02

	ADC_INIT
	TIMER1_INIT



	sei

	clr on_off

	rcall lcd_init_sim

PROGRAM:

	clr r24
	clr r25
```

```
LOOP:
rcall scan_keypad_rising_edge_sim

CHECK_IF_ZERO ;check if r25 == 0 & r24 == 0
cpi GPreg1,0x01
breq LOOP


rcall keypad_to_ascii_sim


CHECK_FOR_VALID_r24 '2'
sbrs GPreg2, 0
rjmp ABORT



LOOP1:
rcall scan_keypad_rising_edge_sim

CHECK_IF_ZERO ;check if r25 == 0 & r24 == 0
cpi GPreg1,0x01
breq LOOP1

movw r26,r24 ;temporary regs for result
rcall keypad_to_ascii_sim

CHECK_FOR_VALID_r24 '1'
cpi GPreg2,0x00
breq ABORT

WELCOME_USER

rjmp PROGRAM

ABORT:

WRONG_PASSWORD

rjmp PROGRAM


ADC_INTERRUPT:
push r26
push r25
push r24
push r23
push r22
in r25, ADCL
in r26, ADCH
andi r26, 0x03
cpi r26, 0x03
breq seven_leds_on_off
cpi r26, 0x02
breq six_leds_on_off
cpi r26, 0x01
breq five_leds_on_off
cpi r25, 0xCE
brsh four_leds_on_off
cpi r25, 0x7E
```

```
brsh three_leds_on
cpi r25, 0x49
brsh two_leds_on

one_led_on:
CLEAR
ldi level, 0x01
in r22, PINB
andi r22, 0x80
ori r22, 0x01
out PORTB, r22
rjmp END_ADC

six_leds_on_off:
rjmp six_leds_on_off1

five_leds_on_off:
rjmp five_leds_on_off1

four_leds_on_off:
rjmp four_leds_on_off1

three_leds_on:
rjmp three_leds_on1

two_leds_on:
rjmp two_leds_on1

seven_leds_on_off:
GAS_DETECTED
ldi level, 0x07
in r22, PINB
andi r22, 0x80
cpi on_off, 0x00
breq write_on_seven
out PORTB, r22
ldi on_off, 0x00
rjmp END_ADC

write_on_seven:
ori r22, 0x7F
out PORTB, r22
ldi on_off, 0x01
rjmp END_ADC


six_leds_on_off1:
GAS_DETECTED
ldi level, 0x06
in r22, PINB
andi r22, 0x80
cpi on_off, 0x00
breq write_on_six
out PORTB, r22
ldi on_off, 0x00
rjmp END_ADC

write_on_six:
ori r22, 0x3F
out PORTB, r22
```

```
ldi on_off, 0x01
rjmp END_ADC


five_leds_on_off1:
GAS_DETECTED
ldi level, 0x05
in r22, PINB
andi r22, 0x80
cpi on_off, 0x00
breq write_on_five
out PORTB, r22
ldi on_off, 0x00
rjmp END_ADC

write_on_five:
ori r22, 0x1F
out PORTB, r22
ldi on_off, 0x01
rjmp END_ADC

four_leds_on_off1:
GAS_DETECTED
ldi level, 0x04
in r22, PINB
andi r22, 0x80
cpi on_off, 0x00
breq write_on_four
out PORTB, r22
ldi on_off, 0x00
rjmp END_ADC

write_on_four:
ori r22, 0x0F
out PORTB, r22
ldi on_off, 0x01
rjmp END_ADC


three_leds_on1:
CLEAR
ldi level, 0x03
in r22, PINB
andi r22, 0x80
ori r22, 0x07
out PORTB, r22
rjmp END_ADC


two_leds_on1:
CLEAR
ldi level, 0x02
in r22, PINB
andi r22, 0x80
ori r22, 0x03
out PORTB, r22

END_ADC:
pop r22
pop r23
```

```
pop r24
pop r25
pop r26

reti


TIMER1_INTERRUPT:
push r26
push r25
push r24

in r26, ADCSRA
ldi r25, (1 << ADSC)
or r26, r25
out ADCSRA, r26
ldi r24,0xFC ; áñ÷éêïðïßçóç ôïõ TCNT1
out TCNT1H ,r24 ; ãéá õðåñ÷åßëéóç ìåôÜ áðü 5 sec
ldi r24 ,0xF3
out TCNT1L ,r24

pop r24
pop r25
pop r26

reti


scan_row_sim:
out PORTC, r25 ; ç áíôßóôïé÷ç ãñáììÞ ôßèåôáé óôï ëïãéêü ‘1’
push r24 ; ôìÞìá êþäéêá ðïõ ðñïóôßèåôáé ãéá ôç óùóÞ
push r25 ; ëåéôïõñãßá ôïõ ðñïãñáììáôïò áðïìáêñõóìÝíçò
ldi r24,low(500) ; ðñüôáóçò
ldi r25,high(500)
rcall wait_usec
pop r25
pop r24 ; ôÝëïò ôìÞìá êþäéêá
nop
nop ; êáèõóôÝñçóç ãéá íá ðñïëÜâåé íá ãßíåé ç áëëáãÞ êáÜóôáóçò
in r24, PINC ; åðéóôñÝöïõí ïé èÝóåéò (óôÞëåò) ôùí äéáêïðþí ðïõ åßíáé ðéåóìÝíïé
andi r24 ,0x0f ; áðïìïíþíïóáé ôá 4 LSB üðïõ ôá ‘1’ äåß÷íïõí ðïõ åßíáé ðáôçìÝíïé
ret ; ïé äéáêüðôåò

scan_keypad_sim:
push r26 ; áðïèÞêåõóå ôïõò êáôá÷ùñçôÝò r27:r26 ãéáôé ôïõò
push r27 ; áëëÜæïõìå ìÝóá óôçí ñïõôßíá
ldi r25 , 0x10 ; Ýëåãîå ôçí ðñþôç ãñáììÞ ôïõ ðëçêôñïëïãßïõ (PC4: 1 2 3 A)
rcall scan_row_sim
swap r24 ; áðïèÞêåõóå ôï áðïôÝëåóìá
mov r27, r24 ; óôá 4 msb ôïõ r27
ldi r25 ,0x20 ; Ýëåãîå ôç äåýôåñç ãñáììÞ ôïõ ðëçêôñïëïãßïõ (PC5: 4 5 6 B)
rcall scan_row_sim
add r27, r24 ; áðïèÞêåõóå ôï áðïôÝëåóìá óôá 4 lsb ôïõ r27
ldi r25 , 0x40 ; Ýëåãîå ôçí ôñßôç ãñáììÞ ôïõ ðëçêôñïëïãßïõ (PC6: 7 8 9 C)
rcall scan_row_sim
swap r24 ; áðïèÞêåõóå ôï áðïôÝëåóìá
mov r26, r24 ; óôá 4 msb ôïõ r26
ldi r25 ,0x80 ; Ýëåãîå ôçí ôÝôáñôç ãñáììÞ ôïõ ðëçêôñïëïãßïõ (PC7: * 0 # D)
rcall scan_row_sim
add r26, r24 ; áðïèÞêåõóå ôï áðïôÝëåóìá óôá 4 lsb ôïõ r26
```

```
movw r24, r26 ; ìåôÝöåñå ôï áðïôÝëáóìá óôïõò êáôá÷ùñçôÝò r25:r24
clr r26 ; ðñïóôÝèçêå ãéá ôçí áðïìáêñõóìÝíç ðñüôáóç
out PORTC,r26 ; ðñïóôÝèçêå ãéá ôçí áðïìáêñõóìÝíç ðñüôáóç
pop r27 ; åðáíÜöåñå ôïõò êáôá÷ùñçôÝò r27:r26
pop r26
ret

scan_keypad_rising_edge_sim:
push r22 ; áðïèÞêåõóå ôïõò êáôá÷ùñçôÝò r23:r22 êáé ôïõò
push r23 ; r26:r27 ãéáôé ôïõò áëëÜæïõìå ìÝóá óôçí ñïõôßíá
push r26
push r27
rcall scan_keypad_sim ; Ýëåãîå ôï ðëçêôñïëüãéï ãéá ðéåóìÝíïõò äéáêüðôåò
push r24 ; êáé áðïèÞêåõóå ôï áðïôÝëåóìá
push r25
ldi r24 ,15 ; êáèõóôÝñçóå 15 ms (ôõðéêÝò ôéìÝò 10-20 msec ðïõ êáèïñßæåôáé áðü ôïí
ldi r25 ,0 ; êáôáóêåõáóôÞ ôïõ ðëçêôñïëïãßïõ – ÷ñïíïäéÜñêåéá óðéíèçñéóìþí)
rcall wait_msec
rcall scan_keypad_sim ; Ýëåãîå ôï ðëçêôñïëüãéï îáíÜ êáé áðüññéøå
pop r23 ; üóá ðëÞêôñá åìöáíßæïôí óðéíèçñéóìü
pop r22
and r24 ,r22
and r25 ,r23
ldi r26 ,low(_tmp_) ; öüñôùóå ôçí êáôÜóôáóç ôùí äéáêïðþí óôçí
ldi r27 ,high(_tmp_) ; ðñïçãïýìåíç êëÞóç ôçò ñïõôßíáò óôïõò r27:r26
ld r23 ,X+
ld r22 ,X
st X ,r24 ; áðïèÞêåõóå óôç RAM ôç íÝá êáôÜóôáóç
st -X ,r25 ; ôùí äéáêïðþí
com r23
com r22 ; âñåò ôïõò äéáêüðôåò ðïõ Ý÷ïõí «ìüëéò» ðáôçèåß
and r24 ,r22
and r25 ,r23
pop r27 ; åðáíÜöåñå ôïõò êáôá÷ùñçôÝò r27:r26
pop r26 ; êáé r23:r22
pop r23
pop r22
ret

keypad_to_ascii_sim:
push r26 ; áðïèÞêåõóå ôïõò êáôá÷ùñçôÝò r27:r26 ãéáôé ôïõò
push r27 ; áëëÜæïõìå ìÝóá óôç ñïõôßíá
movw r26 ,r24 ; ëïãéêü ‘1’ óôéò èÝóåéò ôïõ êáôá÷ùñçôÞ r26 äçëþíïõí
; ôá ðáñáêÜôù óýìâïëá êáé áñéèìïýò
ldi r24 ,'*'
; r26
;C 9 8 7 D # 0 *
sbrc r26 ,0
rjmp return_ascii
ldi r24 ,'0'
sbrc r26 ,1
rjmp return_ascii
ldi r24 ,'#'
sbrc r26 ,2
rjmp return_ascii
ldi r24 ,'D'
sbrc r26 ,3 ; ái äåí åßíáé ‘1’ðáñáêÜìðåé ôçí ret, áëëéþò (ái åßíáé ‘1’)
rjmp return_ascii ; åðéóôñÝöåé ìå ôïí êáôá÷ùñçôÞ r24 ôçí ASCII ôéìÞ ôïõ D.
ldi r24 ,'7'
sbrc r26 ,4
```

```
rjmp return_ascii
ldi r24 ,'8'
sbrc r26 ,5
rjmp return_ascii
ldi r24 ,'9'
sbrc r26 ,6
rjmp return_ascii ;
ldi r24 ,'C'
sbrc r26 ,7
rjmp return_ascii
ldi r24 ,'4' ; ëïãéêü ‘1’ óôéò èÝóåéò ôïõ êáôá÷ùñçôÞ r27 äçëþíïõí
sbrc r27 ,0 ; ôá ðáñáêÜôù óýìâïëá êáé áñéèìïýò
rjmp return_ascii
ldi r24 ,'5'
;r27
;Á 3 2 1 B 6 5 4
sbrc r27 ,1
rjmp return_ascii
ldi r24 ,'6'
sbrc r27 ,2
rjmp return_ascii
ldi r24 ,'B'
sbrc r27 ,3
rjmp return_ascii
ldi r24 ,'1'
sbrc r27 ,4
rjmp return_ascii ;
ldi r24 ,'2'
sbrc r27 ,5
rjmp return_ascii
ldi r24 ,'3'
sbrc r27 ,6
rjmp return_ascii
ldi r24 ,'A'
sbrc r27 ,7
rjmp return_ascii
clr r24
rjmp return_ascii
return_ascii:
pop r27 ; åðáíÜöåñå ôïõò êáôá÷ùñçôÝò r27:r26
pop r26
ret

write_2_nibbles_sim:
push r24 ; ôìÞìá êþäéêá ðïõ ðñïóôßèåôáé ãéá ôç óùóôÞ
push r25 ; ëåéôïõñãßá ôïõ ðñïãñáììáôïò áðïìáêñõóìÝíçò
ldi r24 ,low(6000) ; ðñüôáóçò
ldi r25 ,high(6000)
rcall wait_usec
pop r25
pop r24 ; ôÝëïò ôìÞìá êþäéêá
push r24 ; óôÝëíåé ôá 4 MSB
in r25, PIND ; äéáâÜæïíôáé ôá 4 LSB êáé ôá îáíáóôÝëíïõìå
andi r25, 0x0f ; ãéá íá ìçí ÷áëÜóïõìå ôçí üðïéá ðñïçãïýìåíç êáôÜóôáóç
andi r24, 0xf0 ; áðïìïíþíïôáé ôá 4 MSB êáé
add r24, r25 ; óõíáõÜæïíôáé ìå ôá ðñïûðÜñ÷ïíôá 4 LSB
out PORTD, r24 ; êáé äßíïíôáé óôçí Ýîïäï
sbi PORTD, PD3 ; äçìéïõñãåßôáé ðáëüò Enable óôïí áêñïäÝôç PD3
cbi PORTD, PD3 ; PD3=1 êáé ìåôÜ PD3=0
push r24 ; ôìÞìá êþäéêá ðïõ ðñïóôßèåôáé ãéá ôç óùóôÞ
```

```
push r25 ; ëåéôïõñãßá ôïõ ðñïãñáììáôïò áðïìáêñõóìÝíçò
ldi r24 ,low(6000) ; ðñüóâáóçò
ldi r25 ,high(6000)
rcall wait_usec
pop r25
pop r24 ; ôÝëïò ôìÞìá êþäéêá
pop r24 ; óôÝëíåé ôá 4 LSB. ÁíáêÔôáé ôï byte.
swap r24 ; åíáëëÜóïíôáé ôá 4 MSB ìå ôá 4 LSB
andi r24 ,0xf0 ; ðïõ ìå ôçí óåéñÜ ôïõò áðïóÝëëïíôáé
add r24, r25
out PORTD, r24
sbi PORTD, PD3 ; ÍÝïò ðáëìüò Enable
cbi PORTD, PD3
ret

lcd_data_sim:
push r24
push r25
sbi PORTD,PD2
rcall write_2_nibbles_sim
ldi r24,43
ldi r25,0
rcall wait_usec
pop r25
pop r24
ret

lcd_command_sim:
push r24 ; áðïèÞêåõóå ôïõò êáôÁ÷ùñçôÝò r25:r24 ãéáôß ôïõò
push r25 ; áëëÜæïõìå ìÝóá óôç ñïõôßíá
cbi PORTD, PD2 ; åðéëïãÞ ôïõ êáôÁ÷ùñçôÞ åíôïëþí (PD2=0)
rcall write_2_nibbles_sim ; áðïóôïëÞ ôçò åíôïëÞò êáé áíáìïíÞ 39ìsec
ldi r24, 39 ; ãéá ôçí ïëïêëÞñùóç ôçò åêôÝëåóçò ôçò áðü ôïí åëåãêôÞ ôçò lcd.
ldi r25, 0 ; ÓÇÌ.: õðÜñ÷ïõí äýï åíôïëÝò, ïé clear display êáé return home,
rcall wait_usec ; ðïõ áðáéôïýí óçìáíôéêÜ ìåãáëýôåñï ÷ñïíéêü äéÜóôçìá.
pop r25 ; åðáíÜöåñå ôïõò êáôÁ÷ùñçôÝò r25:r24
pop r24
ret

lcd_init_sim:
push r24 ; áðïèÞêåõóå ôïõò êáôÁ÷ùñçôÝò r25:r24 ãéáôß ôïõò
push r25 ; áëëÜæïõìå ìÝóá óôç ñïõôßíá

ldi r24, 40 ; ¼ôáí ï åëåãêôÞò ôçò lcd ôñïöïäïôåßôáé ìå
ldi r25, 0 ; ñåýìá åêôåëåß ôçí äéêÞ ôïõ áñ÷éêïðïßçóç.
rcall wait_msec ; ÁíáìïíÞ 40 msec ìÝ÷ñé áõôÞ íá ïëïêëçñùèåß.
ldi r24, 0x30 ; åíôïëÞ ìåôÜâáóçò óå 8 bit mode
out PORTD, r24 ; åðåéäÞ äåí ìðïñïýìå íá åßìáóôå âÝâáéïé
sbi PORTD, PD3 ; ãéá ôç äéáìüñùóç åéóüäïõ ôïõ åëåãêôÞ
cbi PORTD, PD3 ; ôçò ïèüíçò, ç åíôïëÞ áðïóôÝëåôáé äýï öïñÝò
ldi r24, 39
ldi r25, 0 ; åÜí ï åëåãêôÞò ôçò ïèüíçò âñßóêåôáé óå 8-bit mode
rcall wait_usec ; äåí èá óõìåßåß ôßðïôá, áëëÜ áí ï åëåãêôÞò Ý÷åé äéáìüñùóç
; åéóüäïõ 4 bit èá ìåôáâåß óå äéáìüñùóç 8 bit
push r24 ; ôìÞìá êþäéêá ðïõ ðñïóôßèåôáé ãéá ôç óùóôÞ
push r25 ; ëåéôïõñãßá ôïõ ðñïãñáììáôïò áðïìáêñõóìÝíçò
ldi r24,low(1000) ; ðñüóâáóçò
ldi r25,high(1000)
rcall wait_usec
pop r25
```

```
       pop r24 ; ôÝëïò ôìÞìá êþäéêá
       ldi r24, 0x30
       out PORTD, r24
       sbi PORTD, PD3
       cbi PORTD, PD3
       ldi r24,39
       ldi r25,0
       rcall wait_usec
       push r24 ; ôìÞìá êþäéêá ðïõ ðñïóôßèåôáé ãéá ôç óùóôÞ
       push r25 ; ëåéôïõñãßá ôïõ ðñïãñáììáôïò áðìíáêñóíÝíçò
       ldi r24 ,low(1000) ; ðñüóâáóçò
       ldi r25 ,high(1000)
       rcall wait_usec
       pop r25
       pop r24 ; ôÝëïò ôìÞìá êþäéêá
       ldi r24,0x20 ; áëëáãÞ óå 4-bit mode
       out PORTD, r24
       sbi PORTD, PD3
       cbi PORTD, PD3
       ldi r24,39
       ldi r25,0
       rcall wait_usec
       push r24 ; ôìÞìá êþäéêá ðïõ ðñïóôßèåôáé ãéá ôç óùóôÞ
       push r25 ; ëåéôïõñãßá ôïõ ðñïãñáììáôïò áðìíáêñóíÝíçò
       ldi r24 ,low(1000) ; ðñüóâáóçò
       ldi r25 ,high(1000)
       rcall wait_usec
       pop r25
       pop r24 ; ôÝëïò ôìÞìá êþäéêá
       ldi r24,0x28 ; åðéëïãÞ ÷áñáêÞñùí ìåãÝèïõò 5x8 êïõêßäùí
       rcall lcd_command_sim ; êáé åìÖíéóç äýï ãñáììþí óôçí ïèüíç
       ldi r24,0x0c ; åíåñãïðïßçóç ôçò ïèüíçò, áðüêñøç ôïõ êÝñóïñá
       rcall lcd_command_sim
       ldi r24,0x01 ; êáèáñéóìüò ôçò ïèüíçò
       rcall lcd_command_sim
       ldi r24, low(1530)
       ldi r25, high(1530)
       rcall wait_usec
       ldi r24 ,0x06 ; åíåñãïðïßçóç áõôùìáôçò áýîçóçò êáôÜ 1 ôçò äéåýèõíóçò
       rcall lcd_command_sim ; ðïõ åßíáé áðìçêåõìÝíç óôïí ìåôñçôÞ äéåõèýíóåùí êáé
       ; áðåíåñãïðïßçóç ôçò ïëßóèçóçò ïëüêëçñçò ôçò ïèüíçò
       pop r25 ; åðáíÜöåñå ôïõò êáôá÷ùñçôÝò r25:r24
       pop r24
       ret
       wait_msec:
       push r24                        ; 2 êýêëïé (0.250 ìsec)
       push r25                        ; 2 êýêëïé
       ldi r24 , low(998)          ; öüñôùóå ôïí êáôá÷. r25:r24 ìå 998 (1 êýêëïò - 0.125 ìsec)
       ldi r25 , high(998)         ; 1 êýêëïò (0.125 ìsec)
       rcall wait_usec                 ; 3 êýêëïé (0.375 ìsec), ðñïêáëåß óõíïëéêÜ êáèõóÝñçóç
998.375 ìsec
       pop r25                                 ; 2 êýêëïé (0.250 ìsec)
       pop r24                                 ; 2 êýêëïé
       sbiw r24 , 1                ; 2 êýêëïé
       brne wait_msec                  ; 1 Þ 2 êýêëïé (0.125 Þ 0.250 ìsec)
       ret                                     ; 4 êýêëïé (0.500 ìsec)

       wait_usec:
       sbiw r24 ,1                 ; 2 êýêëïé (0.250 ìsec)
       nop                             ; 1 êýêëïò (0.125 ìsec)
```

```
nop                             ; 1 êýêëïò (0.125 ìsec)
nop                             ; 1 êýêëïò (0.125 ìsec)
nop                             ; 1 êýêëïò (0.125 ìsec)
brne wait_usec                ; 1 Þ 2 êýêëïé (0.125 Þ 0.250 ìsec)
ret                             ; 4 êýêëïé (0.500 ìsec)
```

Άσκηση 2

```
/*
 * Avr3Ask2.c
 *
 * Created: 12/12/2021 10:02:18 PM
 * Author : thodpap
 */

#define F_CPU 8000000                    // FREQUENCY OF ATMEGA16
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

/* Functions Declarations */
static void SUCCESS();
static void BLINK_FAIL();

static unsigned char scan_row(int i);
static unsigned char swap(unsigned char x);
static void scan_keypad();
static int scan_keypad_rising_edge();
static unsigned char keypad_to_ascii();
static unsigned char calculate_LED(void);
static int calculate_cx ();
void ADC_init(void);

/* Global Variables */
unsigned char mem[2], key_reg[2];
unsigned char first, second; // first: First key and second: Second Key;
static int Cx = 0; // CO concentration in ppm
static unsigned char leds = 0x00, flag_MSB = 0x00, blink_flag = 0, success_flag = 0;
int count = 0;


ISR(ADC_vect) { // ADC Interrupt routine
	Cx = calculate_cx();
	leds = calculate_LED();

	PORTB = flag_MSB << 7; // LEDS Off
	if (Cx > 70 && blink_flag)
		PORTB |= leds;
	else if (Cx <= 70){
		PORTB |= leds;
	}
}

ISR(TIMER1_OVF_vect) {
	ADCSRA |= (1<<ADSC);          // Start the next conversion
	TCNT1 = 64755;                                          //Timer set to overflow in 100
msec
	TCCR1B = (1<<CS12) | (0<<CS11) | (1<<CS10); // Start again.

	if (count == 2) // Here we change the flag for the alarm (Blink_flag) every 2 Timer
Interrupts
	{
		blink_flag = !blink_flag;
		if (success_flag == 1)
```

```
            blink_flag = 1; // HUGE_flag Turns ON only when we are in SUCCESS Mode (Correct
password Typed) So we by pass Blink_flag to carry leds ON.
            count = 0;
       }
       ++count;
}


int main(void)
{
    DDRB = 0xFF;        // PORTB => OUTPUT
    DDRC = 0xF0;        // KEYPAD: PORTC[7:4] => OUTPUT, PORTC[3:0] => INPUT

       ADC_init();

       TIMSK = (1 << TOIE1); //Timer1 ,interrupt enable
       TCCR1B = (1<<CS12) | (0<<CS11) | (1<<CS10);     //frequency of Timer1 8MHz/1024
       TCNT1 = 64755;              //Timer set to overflow in 100 msec

       asm("sei");                        // enable interupts

    while (1) {
           mem[0] = 0;    // INITIALIZE RAM
           mem[1] = 0;
           PORTB = 0;

           while (!scan_keypad_rising_edge()) {}
              first = keypad_to_ascii();

           // GET SECOND DIGIT
              while(!scan_keypad_rising_edge()){}
              second = keypad_to_ascii();

              if (first == '2' && second == '1') {
                     SUCCESS();
              } else {
                     BLINK_FAIL();
              }
    }
}

void SUCCESS() {
       success_flag = 1;
       flag_MSB = 1;
       PORTB = 0x80 | leds;
       _delay_ms(4000);
       PORTB = 0x00 | leds;
       flag_MSB = 0;
       success_flag = 0;
}

void BLINK_FAIL() {
       for (int i = 0; i < 4; ++i) {
              flag_MSB = 1;
              PORTB = 0x80 | leds; // 0xff;
              _delay_ms(500);

              flag_MSB = 0;
              PORTB = 0x00 | leds;
              _delay_ms(500);
```

```
	}
}

unsigned char scan_row(int i) {   // i = 1,2,3,4
	unsigned char a = ( 1 << 3 ); // SKIP 3 LSB
	a = (a << i);                 // SELECT ROW ACCORDING TO FUNCTION INPUT i
	PORTC = a;                          // WE SELECT ROW BY SETTING CORRESPONDING BIT TO 1
	_delay_us(500);                     // DELAY FOR REMOTE USAGE
	return PINC & 0x0F;           // WE READ THE 4 LSB, '1' INDICATES SWITCH PUSHED
}

/* FUNCTION TO SWAP LO WITH HO BITS */
unsigned char swap(unsigned char x) {
	return ((x & 0x0F) << 4 | (x & 0xF0) >> 4);
}

/* SCAN ROWS(1..4) *DIFFERENT ORDER FROM EXERSISE DOCUMENT*
* FIRST ROW: PC4->PC0: 1,  PC4->PC1: 2,  PC4->PC2: 3, PC4->PC3: A
* SECOND ROW: PC5->PC0: 4,  PC5->PC1: 5,  PC5->PC2: 6, PC5->PC3: B
* THIRD ROW: PC6->PC0: 7,  PC6->PC1: 8,  PC6->PC2: 9, PC6->PC3: C
* FOURTH ROW: PC7->PC0: *,  PC7->PC1: 0,  PC7->PC2: #, PC7->PC3: D
*/
void scan_keypad() {
	unsigned char i;

	// check row 1, 0b0001-ROW CORRESPONDING TO PC4
	i = scan_row(1);
	key_reg[1] = swap(i);      //key_reg[1] = first_row(4 MSB)-0000

	// check row 2, 0b0010-ROW CORRESPONDING TO PC5
	i = scan_row(2);
	key_reg[1] += i;           //key_reg[1] = first_row(4 MSB)-second_row(4 LSB)

	// check row 3, 0b0100-ROW CORRESPONDING TO PC6
	i = scan_row(3);
	key_reg[0] = swap(i);      //key_reg[0] = third_row(4 MSB) -0000

	// check row 4, 0b1000-ROW CORRESPONDING TO PC7
	i = scan_row(4);
	key_reg[0] += i;           //key_reg[0] = third_row(4 MSB)-fourth_row(4 LSB)
	PORTC = 0x00;              // added for remote usage
}

int scan_keypad_rising_edge() {
	// CHECK KEYPAD
	scan_keypad();                                    // RETURNS RESULTS IN key_reg
	// ADD TEMPORARY VARIABLES
	unsigned char tmp_keypad[2];
	tmp_keypad[0] = key_reg[0];             //tmp_keypad HOLD ACQUIRED DATA FROM
SCAN_KEYPAD()
	tmp_keypad[1] = key_reg[1];

	_delay_ms(0x15);                              // APOFYGH SPINTHIRISMOU


	scan_keypad();
	key_reg[0] &= tmp_keypad[0];            // APPORIPSE TIS TIMES POU EMFANISAN
SPINTHIRISMO
	key_reg[1] &= tmp_keypad[1];
```

```
	tmp_keypad[0] = mem[0];                         // BRING LAST STATE OF SWITCHES FROM RAM
TO tmp_keypad
	tmp_keypad[1] = mem[1];

	mem[0] = key_reg[0];                    // STORE NEW KEYPAD STATE IN RAM FOR FUTURE
CALL
	mem[1] = key_reg[1];


	key_reg[0] &= ~tmp_keypad[0];           // FIND KEYPAD SWITCHES THAT HAVE JUST BEEN
PRESSED
	key_reg[1] &= ~tmp_keypad[1];

	return (key_reg[0] || key_reg[1]); // 16 BIT VALUE INDICATING FRESHLY PRESSED SWITCHES -
RETURNS 0 IF NO SWITCH PRESSED
}

/* CONVERT VALUE TO ASCII CODE *CHECK COMMENT ABOVE SCAN_KEYPAD FOR CORRESPONDENCE
* key_reg[0] = third_row(4 MSB)-fourth_row(4 LSB)
* key_reg[1] = first_row(4 MSB)-second_row(4 LSB)
* LSB -> MSB == LEFT -> RIGHT IN KEYPAD */
unsigned char keypad_to_ascii() {
	if (key_reg[0] & 0x01)
		return '*';

	if (key_reg[0] & 0x02)
		return '0';

	if (key_reg[0] & 0x04)
		return '#';

	if (key_reg[0] & 0x08)
		return 'D';

	if (key_reg[0] & 0x10)
		return '7';

	if (key_reg[0] & 0x20)
		return '8';

	if (key_reg[0] & 0x40)
		return '9';

	if (key_reg[0] & 0x80)
		return 'C';

	if (key_reg[1] & 0x01)
		return '4';

	if (key_reg[1] & 0x02)
		return '5';

	if (key_reg[1] & 0x04)
		return '6';

	if (key_reg[1] & 0x08)
		return 'B';

	if (key_reg[1] & 0x10)
		return '1';
```

```
	if (key_reg[1] & 0x20)
		return '2';

	if (key_reg[1] & 0x40)
		return '3';

	if (key_reg[1] & 0x80)
		return 'A';

	// Nothing Found
	return 0;
}


unsigned char calculate_LED(void){
	if (Cx < 30)
		return 0x01; // if   0 <= CO < 30 ppm LEDS_PORTB ->   X0000001
	else if (Cx < 50)
		return 0x03; // if  30 <=  CO < 50 ppm LEDS_PORTB ->  X0000011
	else if (Cx < 70)
		return 0x07; // if  50 <=  CO < 70 ppm LEDS_PORTB ->  X0000111
	else if (Cx < 80)
		return 0x0F; // if  70 <=  CO < 80 ppm LEDS_PORTB ->  X0001111
	else if (Cx < 105)
		return 0x1F;// if  80 <=  CO < 105 ppm LEDS_PORTB ->  X0011111
	else if (Cx < 140)
		return 0x3F;// if 105 <=  CO < 140 ppm LEDS_PORTB ->  X0111111
	else
		return 0x7F;                        // if CO >= 140 ppm LEDS_PORTB ->
	X1111111
}

void ADC_init(void) // Initialize ADC
{
	ADMUX = 0x40;
	ADCSRA = ( 1 << ADEN | 1 << ADIE | 1 << ADPS2 | 1 << ADPS1 | 1 << ADPS0 );
}

int calculate_cx () {
	volatile float sensitivity = 129.0, Vgas0 = 0.1;
	volatile float Vin = ADC * 5.0/1024.0; // Vin  = (ADC/5)/1024
	volatile float M = sensitivity * 0.0001; // Cx = (1/M) * (Vin - Vgas0)
	return (1/M) * (Vin - Vgas0);
}
```