Ιωάννης Γεωργακόπουλος - 03111512
Νικόλαος Μαρούσης - 03111025

Εργαστήριο Μικροϋπολογιστών

6η Εργαστηριακή Άσκηση

Άσκηση 1

```
; Δημιουργία της Στοίβας.
      ldi r24,LOW(RAMEND)
      out SPL,r24
      ldi r25,HIGH(RAMEND)
      out SPH,r25

; Θέση των PORTA, PORTC ως εισόδους.
      clr r24
      out DDRA,r24
      out DDRC,r24
; Θέση της PORTB ως έξοδο.
      ser r24
      out DDRB,r24

; Αρχικοποίηση καταχωρητή r26 που θα χρησιμοποιηθεί στην έξοδο.
start:
      clr r26
      in r24,PINA         ; Διάβασμα της θύρας A.

; GATE 04 (XNOR)
      mov r25,r24
      lsl r25             ; Προετοιμασία για σύγκριση PB7 - PB6.
      push r24
      push r25
      andi r24,0x80       ; Απομόνωση των PA7 και PA6.
      andi r25,0x80
      add r24,r25         ; Αν τα bit 7 των καταχωρητών είναι ίδια (NXOR)
                          ; (1+1=0 ή 0+0=0)
      sbrs r24,7
      ori r26,0x08        ; τότε θέτουμε στο Led εξόδου PB3 λογικό 1.

      pop r25
      pop r24             ; Επαναφορά καταχωρητών για περαιτέρω επεξεργασία.
```

```
; GATE 03 (NOR)
      lsl r24
      lsl r24           ; Shift left μέχρι να έρθει στο MSB του r24 το PA5.
      lsl r25
      lsl r25           ; Shift left μέχρι να έρθει στο MSB του r25 το PA4.
      push r24
      push r25
      andi r24,0x80     ; Απομόνωση των PA5 και PA4.
      andi r25,0x80
      or r24,r25
      com r24           ; Υλοποίηση της NOR me OR και αντιστροφή bit (COM).
      sbrc r24,7        ; Αν βγει 1 από την NOR,
      ori r26,0x04      ; τότε θέτουμε στο led εξόδου PB2 λογικό 1.

      pop r25
      pop r24           ; Επαναφορά καταχωρητών για περαιτέρω επεξεργασία.

; GATE 02 (OR)
      lsl r24
      lsl r24           ; Shift left μέχρι να έρθει στο MSB του r24 το PA3.
      lsl r25
      lsl r25           ; Shift left μέχρι να έρθει στο MSB του r25 το PA2.
      push r24
      push r25
      andi r24,0x80     ; Απομόνωση των PA3 και PA2.
      andi r25,0x80
      or r24,r25
      sbrc r24,7        ; Αν βγει 1 από την OR,
      ori r26,0x02      ; τότε θέτουμε στο led εξόδου PB1 λογικό 1.
      mov r27,r26
      andi r27,0x02     ; Αποθηκεύουμε στο 2ο bit του r27 για την GATE 05.

      pop r25
      pop r24           ; Επαναφορά καταχωρητών για περαιτέρω επεξεργασία.

; GATE 01 (XOR)
      lsl r24
      lsl r24           ; Shift left μέχρι να έρθει στο MSB του r24 το PA1.
      lsl r25
      lsl r25           ; Shift left μέχρι να έρθει στο MSB του r25 το PA0.
      push r24
      push r25
      andi r24,0x80     ; Απομόνωση των PA1 και PA0.
      andi r25,0x80
      add r24,r25       ; Αν τα bit 7 των κάταχωρητών είναι διαφορετικά (XOR)
                        ; (1+0=1 ή 0+1=1),
      sbrc r24,7        ; τότε αποθηκεύουμε το αποτέλεσμα στο
      ori r27,0x01      ; 1ο bit του r27 για την GATE 05.

      pop r25
      pop r24
```

```
; GATE 05 (AND)
      mov r24,r27         ; Χρησιμοποιώ τον r24 ως temp για να υλοποιήσω την AND.
      lsr r24             ; Ολισθαίνουμε το bit 1 στη θέση του bit 0, ώστε να τα
                          ; συγκρίνουμε.
      and r24,r27
      sbrc r24,0          ; Αν το bit 0 είναι 1,
      ori r26,0x01        ; τότε θέτουμε το bit 0 εξόδου λογικό 1.


; Έλεγχος διακοπτών PC0-7
      in r27,PINC
      eor r26,r27         ; Αντιστρέφονται τόσα bits εξόδου του B
      out PORTB,r26       ; όσα έχουν 1 στον C.

      rjmp start          ; Διαρκής Επανάληψη.
```

Άσκηση 2

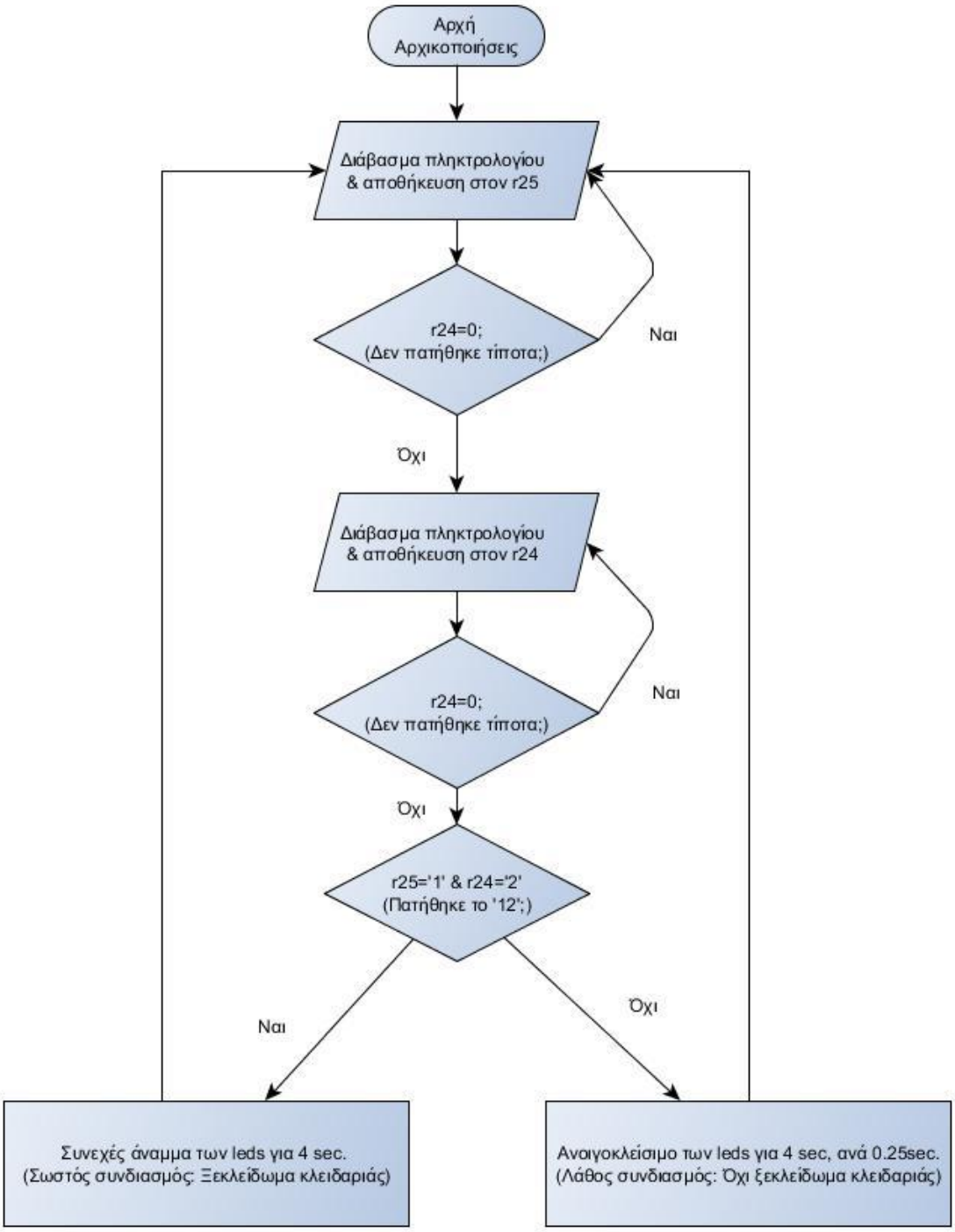
```
#include <avr/io.h>

int main(void)
{
      volatile unsigned char input;
      unsigned char answer;

      DDRA = 0x00;        //είσοδος A
      DDRC = 0xff;        //έξοδος C

      while (1)
      {
            /* Διάβασμα εισόδου. */
            input = PINA;
            /* FO=answer(5)=(ABC+CD+DE)' */
            if (((input & 0b00000111) == 0b00000111) ||
                ((input & 0b00001100) == 0b00001100) ||
                ((input & 0b00011000) == 0b00011000))
                  answer = 0b00000000;
            else answer = 0b00100000;
            /* F1=answer(6)=ABC+D'E' */
            if (((input & 0b00000111) == 0b00000111) ||
                ((input & 0b00011000) == 0b00000000))
                  gmanswer |= 0b01000000;
            /* F2=answer(7)=F0+F1 */
            if (answer & 0b01100000) answer |= 0b10000000;
            /* Εμφάνιση εξόδου. */
            PORTC = answer;
      }
}
```

Άσκηση 3


Αρχή
Αρχικοποιήσεις
Διάβασμα πληκτρολογίου
& αποθήκευση στον r25
r24=0;
(Δεν πατήθηκε τίποτα;)
Ναι
Όχι
Διάβασμα πληκτρολογίου
& αποθήκευση στον r24
r24=0;
(Δεν πατήθηκε τίποτα;)
Ναι
Όχι
r25='1' & r24='2'
(Πατήθηκε το '12';)
Ναι
Όχι
Συνεχές άναμμα των leds για 4 sec.
(Σωστός συνδιασμός: Ξεκλείδωμα κλειδαριάς)
Αναιγοκλείσιμο των leds για 4 sec, ανά 0.25sec.
(Λάθος συνδιασμός: Όχι ξεκλείδωμα κλειδαριάς)

```
.INCLUDE "m16def.inc"

; ---- Αρχή τμήματος δεδομένων.
.DSEG
_tmp_: .byte 2
; ---- Τέλος τμήματος δεδομένων.

.CSEG

; Δημιουργία Στοίβας.
       ldi r24,LOW(RAMEND)
       out SPL,r24
       ldi r25,HIGH(RAMEND)
       out SPH,r25

; Θέση της PORTB ως έξοδος.
       ser r24
       out DDRB,r24


; Αρχικοποίηση του DDRC για την ανάγνωση του πληκτρολογίου.
       ldi r24,(1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
       out DDRC,r24
; Αρχική κλήση για την αρχικοποίηση της _tmp_ .
       call scan_keypad_rising_edge

start:
       ldi r24,10         ; Θέτουμε 10ms καθυστέρηση για αποφυγή σπινθιρισμών.
       rcall scan_keypad_rising_edge          ; Διάβασμα του πληκτρολογίου.
       rcall keypad_to_ascii                  ; Μετατροπή σε ASCII.
       cpi r24,0          ; Αν δεν πατήθηκε τίποτα
       breq start         ; ξαναγυρνάει στο start
       mov r25,r24        ; Αποθήκευση 1ου ψηφίου στον r25

       push r25
SndB:
       ldi r24,10
       rcall scan_keypad_rising_edge          ; Διάβασμα του πληκτρολογίου.
       rcall keypad_to_ascii                  ; Μετατροπή σε ASCII.
       cpi r24,0          ; Αν δεν πατήθηκε τίποτα
       breq SndB          ; ξαναγυρνάει στο SndB

       pop r25
       cpi r25,'1'        ; Έλεγχος αν το πρώτο πλήκτρο που πατήθηκε είναι το '1'
       brne  WRONG        ; και  αν όχι ...
       cpi r24,'2'        ; Έλεγχος αν το δεύτερο πλήκτρο που πατήθηκε είναι το '2'
       brne WRONG         ; και  αν όχι ...
       rcall on_4s        ; Εφόσον πατήθηκε το ζεύγος '12', καλείται η on_4s
       rjmp start         ; και επιστροφή (διαρκής επανάληψη)

WRONG:
       rcall blink_4s
```

```
	rjmp start
; Καλούμενες υπορουτίνες

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

```
	ret
	clr r24
	ret

scan_keypad_rising_edge:
	mov r22,r24	; 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
	ldi r25,0	; καθορίζεται από τον 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


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	; μετάφερε το αποτέλεσμα στους καταχωρητές r25:r24
	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.


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), 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)

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)

; Ανάβει τα leds της PORTB για 4 sec.
on_4s:
	ser r26
	out PORTB,r26		; Άναμμα των LEDs της PORTB.
	ldi r24,low(4000)
	ldi r25,high(4000)
	rcall wait_msec	; Καθυστέρηση 4sec.
	clr r26
	out PORTB,r26		; Σβήσιμο των LEDs της PORTB
	ret			; κι επιστροφή.

; Αναβοσβήνει τα leds της PORTB για 4 sec, με συχνότητα 0.25 sec.
blink_4s:
	ser r26		; r26: έξοδος  (αρχικά άναμμα)
	ldi r27, 16		; r27: counter (4/0.25=16)
repat:
	out PORTB, r26	; Άναμμα ή σβήσιμο των LEDs της PORTB.
```

```
        ldi r24,low(250)
        ldi r25,high(250)
        rcall wait_msec     ; Καθυστέρηση 0.25 sec.
        com r26             ; Εναλλαγή bits εξόδου (Άναμμα/Σβήσιμο)
        dec r27             ; Μείωση counter
        cpi r27, 0          ; Όσο ο counter δεν ειναι 0
        brne repat          ; επανάλαβε
        clr r26
        out PORTB, r26      ; σβήσιμο για σιγουριά
        ret
```