Ιωάννης Γεωργακόπουλος - 03111512
Νικόλαος Μαρούσης - 03111025

Εργαστήριο Μικροϋπολογιστών

3η Εργαστηριακή Άσκηση

Άσκηση 1

```
; Αρχικοποίηση της στοίβας.
      ldi r16, HIGH(RAMEND)     ; Το άνω byte του τέλους της μνήμης
      out SPH,r16               ; τίθεται στον stack pointer (high)
      ldi r16, LOW(RAMEND)      ; κι όμοια το κάτω byte.
      out SPL,r16

      ser r23                   ; r23=0xff
      out DDRB,r23              ; Θέτει τον B θύρα εξόδου.
      clr r23                   ; r23=0x00
      out DDRA,r23              ; Θέτει τον A θύρα εισόδου.
      ldi r23,1                 ; Αρχικοποίηση του μετρητή r23.
                                ; Θα χρησιμοποιηθεί για να αποθηκεύει την τρέχουσα
                                ; κατάσταση.

; Απεικονίζει ένα αναμμένο led που ολισθαίνει δεξιά κι αριστερά στην B.
; Σταματάει την κίνηση όταν δεν είναι πατημένο το PA0.
left:
      cpi r23,0x80              ; Έλεγχος αν έχει φτάσει στο αριστερό άκρο.
      breq right                ; Αν ναι, αρχίζει την κίνηση προς δεξιά.
      lsl r23                   ; Αλλιώς ολίσθηση αριστερά κατά μία θέση.
      call stop_check           ; Έλεγχος αν είναι πατημένο το PA0.
      out PORTB,r23             ; Έχει επιστρέψει εδώ, όταν έχει πατηθεί το PA0 και
                                ; δείχνει τη νέα τιμή.
      ldi r24,low(500)          ; Τοποθετείται στον r25:r24 η τιμή 500
      ldi r25,high(500)         ; για πρόκληση καθυστέρησης 500msec=0.5sec
      call wait_msec            ; από την wait_msec.
      rjmp left                 ; Επανάληψη.
right:
      cpi r23,0x01              ; Όμοια με προηγουμένως, έλεγχος αν έφτασε στο δεξί
άκρο.
      breq left                 ; Αν ναι κίνηση προς τα αριστερά.
      lsr r23                   ; Αλλιώς ολίσθηση μία θέση δεξιά.
      call stop_check
      out PORTB,r23
```

```
	ldi r24,low(500)
	ldi r25,high(500)
	call wait_msec
	rjmp right

; Ελέγχει αν είναι πατημένο το PA0 (και σταματάει την κίνηση αν όχι).
stop_check:
	in r22,PINA		; Διάβασμα της θύρας A.
	andi r22,0x01		; Απομόνωση του LSB (PA0).
	cpi r22,0x01		; Έλεγχος αν είναι 1 (πατημένο).
	brne stop_check		; Αν όχι επανάληψη,
	ret			; αλλιώς επιστροφή.

; 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 κ=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)

; 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)
```

Άσκηση 2

```
; Αρχικοποίηση της στοίβας.
	LDI r16, HIGH(RAMEND)	; Το άνω byte του τέλους της μνήμης
	OUT SPH,r16		; τίθεται στον stack pointer (high)
	LDI r16, LOW(RAMEND)	; κι όμοια το κάτω byte.
	OUT SPL,r16

	ser r26
	out DDRB, r26		; Η PORTB τίθεται έξοδος.
	clr r26
	out DDRA,r26		; Η PORTA τίθεται είσοδος.
flash:
```

```
; Άναμμα της PORTA.
       rcall on
       in r26, pina              ; Διάβασμα της PORTA.
       rcall delay               ; Διάρκεια ανάμματος βάσει της τιμής των PA0-PA3.
; Σβήσιμο της PORTA.
       rcall off
       in r26, pina              ; Τα PA4-PA7 καθορίζουν τη διάρκεια του σβησίματος
       swap r26                  ; Μεταφορά των PA4-PA7 στα PA0-PA3.
       rcall delay
       rjmp flash

; Προκαλεί καθυστέρηση (x+1)*200 msec όπου x τα 4 LSB του r26.
delay:
       andi r26,0x0f             ; Απομόνωση των 4 LSB του r26.
       inc r26                   ; r26 <- r26 + 1.
       ldi r27,200               ; r27 <- 200
       mul r26,r27               ; r1:r0 <- r27*r26 = 200 * (x+1).
       movw r24,r0               ; r25:r24 <- r1:r0.
       rcall wait_msec           ; Καθυστέρηση 200(x+1) msec.
       ret                       ; Επιστροφή.

; Ανάβει την πόρτα B.
on: ser r26
       out PORTB , r26
       ret

; Σβήνει την πόρτα B.
off: clr r26
       out PORTB , r26
       ret

; 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)
```

Άσκηση 3

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

int main(void)
{
       DDRA = 0xff;                     // Η θύρα Α τίθεται έξοδος.
       DDRC = 0x00;                     // Η θύρα C τίθεται είσοδος.

       volatile unsigned char t, new;   //  Στον t αποθηκεύεται η είσοδος και στον
new
                                        //  αποθηκεύεται προσωρινά κάθε νέα είσοδος.
       unsigned char u, out;            //  Στον u αποθηκεύεται η θέση του MSB της
                                        // εισόδου και στο out η τρέχουσα έξοδος.

       out = 0x80;                      // Αρχικά απεικονίζεται το bit
       PORTA = out;                     // στο MSB.

    /* Κάθε φορά που πατιέται και αφήνεται ένα από τα SW0-4
     * εκτελείται μία αντίστοιχη λειτουργία στο απεικονιζόμενο bit
     * της θύρας B με προτεραιότητα από SW4 σε SW0 αν πατηθούν πολλά μαζί.
     */
    while (1)                           // Διαρκής επανάληψη.
    {
             /* Αναμονή μέχρι να πατηθεί κάποιο από τα SW0-4*/
             while (!(t = PINC & 0x1f));      // Επαναλαμβάνει όσο 5 LSB του PINC =
0.

             /* Αναμονή μέχρι κάποιο κουμπί να μην είναι πλέον πατημένο.
              * Αν όσο είναι πατημένο κάποιο κουμπί πατηθεί κάποιο μεγαλύτερο,
              * ενημερώνεται ο t, αλλιώς παραμένει ως έχει, μέχρι να ελευθερωθεί
              * κάποιο από όλα.
              */
             while((new = PINC & 0x1f) >= t) t = new;      // Αν πατηθεί κάποιο
                                                           // ενημερώνεται το t.
             t ^= new;                  // Τώρα ο t έχει άσσους σε όσα κουμπιά από 1
                                        // έγιναν 0 (δηλαδή όσα αφέθηκαν).

             /* Εύρεση του μέγιστου ψηφίου της εισόδου που είναι άσσος. */
             u = 0x10;                  // Αρχικοποιεί τον u στο 0x10,
             while (!(t & 0x10)){       // ώστε στη συνέχεια να ελέγχει αν είναι 0 ο
t
                   t <<= 1;             // κι αν ναι, να ολισθαίνει κι άλλο τον t
                   u >>= 1;             // και να χαμηλώνει το bit του u
             }                          // μέχρις ότου το u να αποκτήσει τη θέση του
                                        // μεγαλύτερου ψηφίου του t που είναι 1
```

```
            /* Εύρεση της λειτουργίας που καθορίζεται από το u και εκτέλεση της. */
            switch (u){
                  /* SW4 - Το bit έρχεται στο MSB. */
                  case 0x10:
                        out = 0x80;
                        break;
                  /* SW3 - Ολίσθηση κατά δύο θέσεις αριστερά. */
                  case 0x08:
                        if (out & 0xC0) out >>= 6;// Αν βρίσκεται στα δύο MSB
                                                  // πρέπει να πάει στην αρχή πάλι
                        else out <<= 2;           // οπότε το ολισθαίνουμε 6 θ
                                                  // θέσεις δεξιά.
                        break;
                  /* SW2 - Ολίσθηση κατά δύο θέσεις δεξιά. */
                  case 0x04:
                        if (out & 0x03) out <<= 6;       // Όμοια για τα δύο LSB.
                        else out >>= 2;
                        break;
                  /* SW1 - Ολίσθηση κατά μία θέση αριστερά. */
                  case 0x02:
                        if (out & 0x80) out >>= 7;       // Όμοια για το ένα MSB.
                        else out <<= 1;
                        break;
                  /* SW0 - Ολίσθηση κατά μία θέση δεξιά. */
                  case 0x01:
                        if (out & 0x01) out <<= 7;       // Όμοια για το ένα LSB.
                        else out >>= 1;
                        break;
                  }
            PORTA = out;                     // Απεικόνιση της εξόδου στη θύρα C.
      }
}
```