



Εργαστήριο Μικροϋπολογιστών

5^η Εργαστηριακή Άσκηση

Άσκηση 1

```
.def counter=r16          ; Περιέχει το τρέχον περιεχόμενο του μετρητή
.def intrcounter=r17      ; Μετρητής του πλήθους των διακοπών INT1
.def temp=r18

.org 0x0
    rjmp reset
.org 0x4          ; Διάνυσμα διακοπής INT1
    rjmp ISR1

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

    ; Επίτρεψη INT1.
    ldi temp, (1 << ISC10) | (1 << ISC11)
    out MCUCR, temp          ; Ορίζεται η διακοπή με σήμα θετικής ακμής.
    ldi temp, (1 << INT1)
    out GICR, temp           ; Επιτρέπεται η διακοπή INT1.
    sei                      ; Επίτρεψη διακοπών.

    ; Οι A,B τίθενται έξοδοι.
    ser temp
    out DDRA, temp
    out DDRB, temp
    ; Η D τίθεται είσοδος.
    clr temp
    out DDRD,temp
    ; Αρχικοποίηση counter.
    clr counter
```

```
;; Κυρίως πρόγραμμα, απεικονίζει έναν 8-bit μετρητή στη θύρα B.
```

```
loop:
    out PORTB, counter          ; Δείχνει το τρέχον περιεχόμενο του μετρητή,
    ldi r24, low(200)
    ldi r25, high(200)
    rcall wait_msec            ; περιμένει 0.2 sec,
    inc counter                ; τον αυξάνει
    rjmp loop                  ; και επαναλαμβάνει.
```

```
;; Όταν καλείται απεικονίζει στην θύρα A το πλήθος των διακοπών INT1,
;; αν το PD7 είναι ON.
```

```
ISR1:
    ; Σώσιμο των καταχωρητών.
    push r24
    push r25
    push temp
    in temp, SREG
    push temp
```

```
; Έλεγχος για αναπηδήσεις ώστε να μετρηθεί μία φορά η διακοπή.
```

```
check:
    ldi temp, (1 << INTF1)
    out GIFR, temp             ; Μηδενισμός του INTF1.
    ldi r24, low(5)
    ldi r25, high(5)
    rcall wait_msec            ; Αναμονή για 5 msec.
    in temp, GIFR
    sbrc temp, 7                ; Αν το INTF1 είναι 0 πάει στις κυρίως εντολές,
    rjmp check                 ; αλλιώς επαναλαμβάνει.

    inc intrcounter             ; Αυξάνει το πλήθος των διακοπών κατά 1.

    in temp, PIND               ; Διάβασμα της θύρας D.
    sbrc temp, 7                ; Αν το PD7 είναι 0, δεν εκτελείται η επόμενη
                                ; εντολή.
    out PORTA, intrcounter      ; Αλλιώς απεικονίζεται στη θύρα A, το πλήθος των
                                ; διακοπών.
```

```
; Επαναφορά των καταχωρητών και επιστροφή.
```

```
is1ret:
    pop temp
    out SREG, temp
    pop temp
    pop r25
    pop r24
    reti
```

```
;; Προκαλεί καθυστέρηση r25:r24 msec.
```

```
wait_msec:
    push r24                    ; 2 κύκλοι (0.250 msec)
    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 2

```

.def counter=r16                  ; περιέχει το τρέχον περιεχόμενο του μετρητή
.def answer=r17                  ; περιέχει το πλήθος των διακοπών της θύρας A που
                                   ; είναι ON

.def temp=r18

.org 0x0
        rjmp reset
.org 0x2
        rjmp ISR0                ; διάνυσμα διακοπής INTR0

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

        ; Επίτρεψη INT0.
        ldi temp,( 1 << ISC01) | ( 1 << ISC00)
        out MCUCR, temp           ; Ορίζεται η διακοπή με σήμα θετικής ακμής.
        ldi temp, ( 1 << INT0)
        out GICR, temp            ; Επιτρέπεται η διακοπή.
        sei                       ; Επίτρεψη διακοπών.

        ; Οι B,C τίθενται έξοδοι.
        ser temp
        out DDRB, temp

```

```

        out DDRC, temp
        ; Η Α τίθεται είσοδος.
        clr temp
        out DDRA, temp
        ; Αρχικοποίηση counter.
        clr counter

;; Κυρίως πρόγραμμα, απεικονίζει έναν 8-bit μετρητή στη θύρα B.
loop:
        out PORTB, counter          ; Δείχνει το τρέχον περιεχόμενο του μετρητή,
        ldi r24, low(200)
        ldi r25, high(200)
        rcall wait_msec             ; περιμένει 0.2 sec,
        inc counter                 ; τον αυξάνει
        rjmp loop                   ; και επαναλαμβάνει.

;; Όταν καλείται ανάβει τόσα led της θύρας C όσα switch της A
;; είναι ON (αρχίζοντας από το LSB).
ISR0:
        ; Σώσιμο καταχωρητών.
        push r24
        push r25
        push counter
        push temp
        in temp, SREG
        push temp

; Έλεγχος για αναπηδήσεις ώστε να μετρηθεί μία φορά η διακοπή.
check:
        ldi temp, (1 << INT0)
        out GIFR, temp              ; Μηδενισμός του INTF0.
        ldi r24, low(5)
        ldi r25, high(5)
        rcall wait_msec             ; Αναμονή για 5 msec.
        in temp, GIFR
        sbrc temp, 6                 ; Αν το INTF0 είναι 0 πάει στις κυρίως εντολές,
        rjmp check                  ; αλλιώς επαναλαμβάνει.

        ldi answer, 0                ; Στον answer θα σχηματιστεί η απάντηση.
        ldi counter, 8               ; Ο counter χρησιμοποιείται ως μετρητής των bits.
        in temp, PINA

stillcnt:
        rol temp                     ; Ολίσθηση μέσω κρατουμένου.
        brcc nextdig                 ; Αν το Carry είναι 0 πάει στο nextdig.
        lsl answer                    ; Αλλιώς ολισθαίνει αριστερά τον answer
        inc answer                    ; και προσθέται ακόμη μία μονάδα.
nextdig:
        dec counter                  ; Μειώνει το μετρητή και
        brne stillcnt                ; επαναλαμβάνει συνολικά 8 φορές (μία για κάθε bit).

        out PORTC, answer            ; Απεικόνιση της εξόδου στη θύρα C.

```

```
; Επαναφορά των καταχωρητών και επιστροφή.  
pop temp  
out SREG, temp  
pop temp  
pop counter  
pop r25  
pop r24  
reti
```

;; Προκαλεί καθυστέρηση r25:r24 msec.

wait_msec:

```
push r24 ; 2 κύκλοι (0.250 msec)  
push r25 ; 2 κύκλοι  
ldi r24 , low(998) ; φόρτωσε τον καταχ. r25:r24 με 998 (1 κύκλος -  
; 0.125 msec)  
ldi r25 , high(998) ; 1 κύκλος (0.125 msec)  
rcall wait_usec ; 3 κύκλοι (0.375 msec), προκαλεί συνολικά  
; καθυστέρηση 998.375 msec  
pop r25 ; 2 κύκλοι (0.250 msec)  
pop r24 ; 2 κύκλοι  
sbw r24 , 1 ; 2 κύκλοι  
brne wait_msec ; 1 ή 2 κύκλοι (0.125 ή 0.250 msec)  
ret ; 4 κύκλοι (0.500 msec)
```

;; Προκαλεί καθυστέρηση r25:r24 msec.

wait_usec:

```
sbw r24 , 1 ; 2 κύκλοι (0.250 msec)  
nop ; 1 κύκλος (0.125 msec)  
nop ; 1 κύκλος (0.125 msec)  
nop ; 1 κύκλος (0.125 msec)  
nop ; 1 κύκλος (0.125 msec)  
brne wait_usec ; 1 ή 2 κύκλοι (0.125 ή 0.250 msec)  
ret ; 4 κύκλοι (0.500 msec)
```

Άσκηση 3

```

;; Κάθε φορά που επανατίθεται ο μετρητής ακολουθούν δύο φάσεις. Στην πρώτη φάση που
;; διαρκεί 0.5 sec, αν πρόκειται για ανανέωση είναι αναμμένα όλα τα leds, αλλιώς μόνο
;; το PB0. Στη δεύτερη φάση που διαρκεί 3.5 sec, είναι αναμμένο μόνο το PB0.

.def alreadyon=r16          ; Περιέχει στα bit 0 και 4 την φάση που είμαστε: 00
                             ; όλα σβηστά, 01 δεύτερη φάση, 11 πρώτη φάση.
.def output=r17             ; Περιέχει τη τρέχουσα μορφή των led της θύρας B.
.def temp=r18

; 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

; 4 sec: 0x85ee
; 3.5 sec: 0x9530
; 0.5 sec: 0xf0be

.org 0x0
    rjmp reset
.org 0x4          ; διάνυσμα διακοπής INTR1
    rjmp ISR1
.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

    ; Η B τίθεται έξοδος.
    ser temp
    out DDRB, temp
    ; Οι A,D τίθενται είσοδοι.
    clr temp
    out DDRA, temp
    out DDRD, 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

    ; Επίτρεψη INT1.
    ldi temp, (1 << ISC11) | (1 << ISC10)

```

```

    out MCUCR, temp          ; Ορίζεται η διακοπή με σήμα θετικής ακμής.
    ldi temp, ( 1 << INT1)
    out GICR, temp           ; Επίτρεψη της διακοπής.

    ldi alreadyon, 0x00      ; Αρχικά δεν είναι αναμμένο τίποτα.

    sei                      ; Επίτρεψη διακοπών.

;; Κυρίως πρόγραμμα.
;; Ελέγχεται αν το PA7 είναι πατημένο και καλείται η flashtime αναλόγως.
start:
    in temp, PINA
    sbrc temp, 7             ; Αν δεν πατήθηκε το PA7,
    rjmp start              ; επαναλαμβάνει,
    rcall flashtime         ; αλλιώς καλεί τη flashtime

    ; Έλεγχος για αναπηδήσεις ώστε να μετρηθεί μία φορά το πάτημα του PA7.
    push r24
    push r25
check1:
    in temp, PINA
    ldi r24, low(5)
    ldi r25, high(5)
    rcall wait_msec         ; Αναμονή για 5 msec.
    in temp, PINA
    sbrc temp, 7            ; Αν το PA7 είναι 0 πάει στις κυρίως εντολές,
    rjmp check1            ; αλλιώς επαναλαμβάνει.

    pop r25
    pop r24
    rjmp start              ; αλλιώς επανέρχεται στο start.

;; Καλείται όταν πατιέται κάποιος από τους αισθητήρες κίνησης ή γίνεται διακοπή INTR1
;; Επαναθέτει το μετρητή και ανάβει όλα τα led αν πρόκειται για ανανέωση.
flashtime:
    push r24
    push r25
    ser output
    sbrc alreadyon, 0       ; Αν δεν είναι ήδη αναμμένο το PB0 σημαίνει ότι δεν
    andi output, 0x01      ; είναι ανανέωση, άρα ανάβει μόνο το PB0.
    ldi alreadyon, 0x11    ; Σε κάθε περίπτωση η διαδικασία του μετρητή εκκινεί
    ; από την αρχή.

    ; Τίθεται ο μετρητής για την πρώτη φάση των 0.5 sec, όπου αν ήταν ανανέωση, θα είναι
    ; όλα αναμμένα (αλλιώς μόνο το PB0 όπως πρέπει).
    ldi r25, 0xf0          ; 0 μετρητής τίθεται αρχικά στα 0.5 sec.
    ldi r24, 0xbe
    ; ldi r25, 0xff
    ; ldi r24, 0xf8
    out TCNT1H, r25
    out TCNT1L, r24

```

```
    out PORTB, output          ; Απεικονίζεται η έξοδος της πρώτης φάσης.
    pop r25
    pop r24
    ret

;; Όταν καλείται, απλώς καλείται η flashlight
ISR1:
    push r24
    push r25
    push temp
    in temp, SREG
    push temp

; Έλεγχος για αναπηδήσεις ώστε να μετρηθεί μία φορά η διακοπή.
check2:
    ldi temp, (1 << INTF1)
    out GIFR, temp             ; Μηδενισμός του INTF1.
    ldi r24, low(5)
    ldi r25, high(5)
    rcall wait_msec            ; Αναμονή για 5 msec.
    in temp, GIFR
    sbrc temp, 7                ; Αν το INTF1 είναι 0 πάει στις κυρίως εντολές,
    rjmp check2                ; αλλιώς επαναλαμβάνει.

    rcall flashtime

    pop temp
    out SREG, temp
    pop temp
    pop r25
    pop r24
    reti

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

    sbrc alreadyon, 4           ; Αν το 4ο bit του alreadyon είναι OFF σημαίνει ότι
    rjmp closeall              ; τέλειωσε η δεύτερη φάση, οπότε πάει στο closeall.

; Αλλιώς τέλειωσε η πρώτη φάση και πρέπει να αφήσει ανοιχτό μόνο το PB0
; και να περιμένει άλλα 3.5 sec.
    andi alreadyon, 0x0f        ; Το 4ο bit γίνεται 0 διότι τέλειωσε η πρώτη φάση.
    ldi r25, 0x95                ; Θέτει τον χρονιστή στα 3.5 sec.
    ldi r24, 0x30
    ; ldi r25, 0xff
    ; ldi r24, 0xf0
    out TCNT1H, r25
    out TCNT1L, r24
```



```

        andi output, 0x01          ; Στη δεύτερη φάση απομένει μόνο το PB0 αναμμένο.
        out PORTB, output
        rjmp timoffret

closeall:                                ; Έρχεται εδώ όταν τελειώνει η δεύτερη φάση.
        ldi alreadyon, 0x00        ; Τίποτα δεν είναι αναμμένο πλέον.
        ldi output, 0
        out PORTB, output          ; Σβήνει όλα τα led της PORTB.

; Επαναφορά των καταχωρητών και επιστροφή.
timoffret:
        pop temp
        out SREG, temp
        pop temp
        reti

;; Προκαλεί καθυστέρηση r25:r24 msec.
wait_msec:
        push r24                    ; 2 κύκλοι (0.250 msec)
        push r25                    ; 2 κύκλοι
        ldi r24 , low(998)          ; φόρτωσε τον καταχ. r25:r24 με 998 (1 κύκλος -
                                   ; 0.125 msec)
        ldi r25 , high(998)         ; 1 κύκλος (0.125 msec)
        rcall wait_usec             ; 3 κύκλοι (0.375 msec), προκαλεί συνολικά
                                   ; καθυστέρηση 998.375 msec
        pop r25                     ; 2 κύκλοι (0.250 msec)
        pop r24                     ; 2 κύκλοι
        sbiw r24 , 1                ; 2 κύκλοι
        brne wait_msec              ; 1 ή 2 κύκλοι (0.125 ή 0.250 msec)
        ret                         ; 4 κύκλοι (0.500 msec)

;; Προκαλεί καθυστέρηση r25:r24 msec.
wait_usec:
        sbiw r24 , 1                ; 2 κύκλοι (0.250 msec)
        nop                         ; 1 κύκλος (0.125 msec)
        nop                         ; 1 κύκλος (0.125 msec)
        nop                         ; 1 κύκλος (0.125 msec)
        nop                         ; 1 κύκλος (0.125 msec)
        nop                         ; 1 κύκλος (0.125 msec)
        brne wait_usec              ; 1 ή 2 κύκλοι (0.125 ή 0.250 msec)
        ret                         ; 4 κύκλοι (0.500 msec)

```