



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ  
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ  
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ  
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ  
ΑΚΑΔ. ΕΤΟΣ 2022-2023

ΑΘΗΝΑ 13 Οκτωβρίου 2022

## **1η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ ΓΙΑ ΤΟ ΜΑΘΗΜΑ "Εργαστήριο Μικροϋπολογιστών"**

Ανάπτυξη κώδικα για το μικροελεγκτή ATmega328 και προσομοίωση της εκτέλεσής του στο αναπτυξιακό περιβάλλον MPLAB X

**Αναφορά 1<sup>ης</sup> Εργαστηριακής Άσκησης**

**Ραπτόπουλος Πέτρος (el19145)  
Σαφός Κωνσταντίνος (el19172)**

## Ζήτημα 1.1:

Παρακάτω φαίνεται ο κώδικας σε Assembly που υλοποιεί τα ζητούμενα της άσκησης.

Η ορθή λειτουργία του κώδικα έχει ελεγχθεί στο περιβάλλον προσομοίωσης MPLAB X .

**Σημείωση:** Ο ακριβής τρόπος λειτουργίας του προγράμματος υποδεικνύεται μέσω σχολίων σε εντολές του κώδικα.

```
.include "m328Pbdef.inc"

reset:
    ldi r24 , low(RAMEND)        ; initialize stack pointer
    out SPL , r24
    ldi r24 , high(RAMEND)
    out SPH , r24

main:
    ldi r24 , low(273)           ; load r25:r24 with how many milliseconds
    ldi r25 , high(273)          ; the function wait_x_msec must wait
    rcall wait_x_msec            ; call function wait_x_msec (3 cycles)
    rjmp main                    ; loop for ever

wait4:
    ret                          ; just return to the caller (4 cycles)

wait1m:
    ldi r26, 98 ; (1 cycle)

loop1:
    rcall wait4 ; 7 cycles total
    dec r26      ; 1 cycle total
    brne loop1   ; 2 cycles if taken 1 if not
    rcall wait4 ; 7 cycles total
    nop          ; do nothing (1 cycle)
    nop          ; do nothing (1 cycle)
    ret          ; return to the caller (4 cycles)

; Functionality of the wait_x_msec explained:
; x is passed as parameter to "double" register r25,r24
; wait_x_msec waits (x-1)msec - 1 cycle with the help of loop2
; loop waits the rest of the cycles needed
; before loop2 a check if x was 1 is essential

wait_x_msec:
    ldi r26, 98
loop:
    ; loop repeated 98 times
    rcall wait4 ; call (3 cycles) and wait4 (4 cycles)
    dec r26      ; r26-- (1 cycle)
    brne loop    ; if r26 == 0 then exit loop
    ; brne takes 2 cycles for each jump to loop
    ; 1 cycle for last check
    ; So, loop takes 98*5-1 cycles in total
    rcall wait4 ; 7 cycles total
    nop          ; do nothing (1 cycle)
    nop          ; do nothing (1 cycle)
    sbiw r24, 1 ; subtract 1 from word (16 bit) register
                  ; formed by r25,r24 (2 cycles)
    breq retu    ; if x was 1 then return (2 cycles)
    nop          ; do nothing (1 cycle)
    nop          ; do nothing (1 cycle)
loop2:
    ; if loop2 is executed y times then it takes 1000*y-1 cycles
    rcall wait1m ; 996 cycles total (y)
    sbiw r24, 1 ; 2 cycles
    brne loop2   ; 2 or 1 cycles
retu: ret        ; return to the caller (4 cycles)
```

## Ζήτημα 1.2:

Παρακάτω φαίνεται ο κώδικας σε Assembly που υλοποιεί τα ζητούμενα της άσκησης.

Η ορθή λειτουργία του κώδικα έχει ελεγχθεί στο περιβάλλον προσομοίωσης MPLAB X .

**Σημείωση:** Ο ακριβής τρόπος λειτουργίας του προγράμματος υποδεικνύεται μέσω σχολίων σε εντολές του κώδικα.

```
.include "m328Pbdef.inc"
; define which register is used with each name
.DEF A=r16
.DEF Ainc=r25
.DEF B=r17
.DEF Binc=r26
.DEF C=r18
.DEF Cinc=r27
.DEF D=r19
.DEF Dinc=r28
.DEF A=r16
.DEF DN=r20
.DEF temp=r21
.DEF F0=r22
.DEF F1=r23
.DEF counter=r24

reset:
    ldi r24 , low(RAMEND)      ; initialize stack pointer
    out SPL , r24
    ldi r24 , high(RAMEND)
    out SPH , r24

main:
    ldi counter, 6             ; repeat six times
    ldi A, 0x55                ; initial values
    ldi B, 0x43
    ldi C, 0x22
    ldi D, 0x02

    ldi Ainc, 0x02             ; how much each variable
    ldi Binc, 0x03             ; increases
    ldi Cinc, 0x04
    ldi Dinc, 0x05

loop:
    mov DN, D
    com DN                    ; DN = D'

    mov F0, A
    or F0, B                  ; F0 = A + B (temporarily)

    mov temp, B
    or temp, DN               ; temp = B + D'

    and F0, temp              ; F0 = (A+B)(B+D') (final)
                                ; (A'B'+B'D)'=(A+B)(B+D')

    mov F1, A
    or F1, C                  ; F1 = A + C (temporarily)

    and F1, temp              ; F1 = (A+C)(B+D') (final)

    add A, Ainc               ; increase the variables
    add B, Binc
    add C, Cinc
    add D, Dinc
    dec counter               ; decrease counter
    brne loop                 ; if counter > 0 repeat
                                ; exit otherwise
```

Με τη βοήθεια του debugger παρακολουθούμε την τιμή των μεταβλητών για κάθε επανάληψη. Έχουμε:

A	B	C	D	F0	F1
0x55	0x43	0x22	0x02	0x57	0x77
0x57	0x46	0x26	0x07	0x56	0x76
0x59	0x49	0x2A	0x0C	0x59	0x7B
0x5B	0x4C	0x2E	0x11	0x4E	0x6E
0x5D	0x4F	0x32	0x16	0x4F	0x6F
0x5F	0x52	0x36	0x1B	0x56	0x76

### Ζήτημα 1.3:

Παρακάτω φαίνεται ο κώδικας σε Assembly που υλοποιεί τα ζητούμενα της άσκησης.

Η ορθή λειτουργία του κώδικα έχει ελεγχθεί στο περιβάλλον προσομοίωσης MPLAB X .

**Σημείωση:** Ο ακριβής τρόπος λειτουργίας του προγράμματος υποδεικνύεται μέσω σχολίων σε εντολές του κώδικα.

```
.include "m328PBdef.inc"
; define which register is used with each name
.DEF temp = r16
.DEF leds = r17

reset:
    ldi r24, low(RAMEND)    ;Initialize stack pointer
    out SPL, r24
    ldi r24, high(RAMEND)
    out SPH, r24

main:
    ldi leds,0x01          ; initialize led state
    ser temp
    out DDRD,temp          ; PORTD output
    bset 6                  ; set 6th bit (T flag) of state
                           ; register to 1
; 1 -> going left, 0 -> going right
left:
    out PORTD, leds        ; output led state
    ldi r25, HIGH(500)
    ldi r24, LOW(500)
    rcall wait_x_msec      ; wait 0.5sec
    lsl leds               ; move 1 to the left
    sbrs leds, 7           ; if last-left led skip jump
    rjmp left              ; otherwise repeat left movement

last_left:
    out PORTD, leds        ; output led state
    ldi r25, HIGH(1000)
    ldi r24, LOW(1000)
    rcall wait_x_msec      ; wait 1sec and go right
                           ; attention! the program will wait
                           ; another 0.5sec after right: label
                           ; for last_left state
    bclr 6                  ; set 6th bit (T flag) of state
                           ; register to 0
; 1 -> going left, 0 -> going right
right:
    out PORTD, leds        ; output led state
    ldi r25, HIGH(500)
    ldi r24, LOW(500)
    rcall wait_x_msec      ; wait 0.5sec
    lsr leds               ; move 1 right
    sbrs leds, 0           ; if last-right led skip jump
    rjmp right             ; otherwise repeat right movement

last_right:
    out PORTD, leds        ; output led state
    ldi r25, HIGH(1000)
    ldi r24, LOW(1000)
    rcall wait_x_msec      ; wait 1sec and go left
                           ; attention! the program will wait
                           ; another 0.5sec after left: label
                           ; for last_right state
    bset 6                  ; set 6th bit (T flag) of state
                           ; register to 1
; 1 -> going left, 0 -> going right
    rjmp left              ; repeat
```

**Σημείωση:** Για να είναι ο παραπάνω κώδικας πιο ευανάγνωστος παραλείπεται το σώμα της wait\_x\_msec.

Για να κάνει compile ο παραπάνω κώδικας αρκεί να προστεθούν στο τέλος οι γραμμές assembly του Ζητήματος 1.1 από την wait4: και κάτω.