

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών
Υπολογιστών

1η Σειρά Ασκήσεων

Μάθημα: Εργαστήριο Μικροϋπολογιστών

Εξάμηνο: 7^ο

Ονοματεπώνυμο: Αλεξοπούλου Γεωργία, Γκενάκου Ζωή

Ζήτημα 1.1:

A) Να υλοποιηθεί κώδικας assembly, για τον μικροελεγκτή ATmega328, στη μορφή που εμφανίζεται παρακάτω, ο οποίος να παράγει ρυθμιζόμενες χρονικές καθυστερήσεις με απόλυτη ακρίβεια.

```
...  
rcall wait_x_msec  
...  
wait_x_msec:  
....  
ret
```

```
.include "m328PBdef.inc"
```

```
.equ FOSC_MHZ=16    ;MHz  
.equ DEL_mS=100     ;mS  
.equ F1=FOSC_MHZ*DEL_mS
```

```
reset:  
    ldi r24,low(RAMEND)  
    out SPL,r24
```

```

    ldi r24,high(RAMEND)
    out SPH,r24

    ser r24
    out DDRD,r24
    out PORTD,r24

    clr r26

main:
    ldi r24, low(F1-2)
    ldi r25, high(F1-2)
    rcall wait_x_msec
    inc r26
    out PORTD,r26
    rjmp main

wait_x_msec:
    ldi    r23, 249        ; (1 cycle)
    nop

delay_one:
    dec r23                ; 1 cycle
    nop
    brne delay_one        ; 1 or 2 cycles
    ;total group delay 996 cycles

    ldi    r23, 249        ; (1 cycle)

delay_two:
    dec r23                ; 1 cycle
    nop
    brne delay_two        ; 1 or 2 cycles
    ;total group delay 996 cycles

    ldi    r23, 249        ; (1 cycle)

delay_three:
    ldi    r23, 249        ; (1 cycle)
loop:
    dec r23                ; 1 cycle

```

```

nop                ; 1 cycle
brne loop          ; 1 or 2 cycles

sbiw r24 ,1        ; 2 cycles
brne delay_three   ; 1 or 2 cycles

ret                ;4 cycles

```

Για την υλοποίηση του ζητήματος θα χρησιμοποιήσουμε ως βάση την υπορουτίνα delay_mS που έχει δοθεί ως παράδειγμα στο εργαστήριο.

Για να επιτευχθεί απόλυτη ακρίβεια στην υπορουτίνα καθυστέρησης, θα πρέπει οι κύκλοι που θα μεσολαβούν για μια καθυστέρηση (πχ 100ms) να προκύπτουν ως εξής:

- Η συχνότητα του μικροεπεξεργαστή είναι 16MHz, δηλαδή 16000000 κύκλοι ανα second.
Επομένως 1 κύκλος κάνει $\frac{1}{16 \cdot 10^6} s = 0.0625 \mu s$ και $1ms = 16000$ κύκλοι
Επομένως μια καθυστέρηση 100ms προκύπτει από την εκτέλεση 1600000 κύκλων

Δηλαδή στόχος μας για να επιτύχουμε απόλυτη ακρίβεια στην χρονική καθυστέρηση, μεταξύ της εντολής rcall wait_x_msec και της επόμενης εντολής πρέπει να διαμεσολαβούν 1600000 κύκλοι.

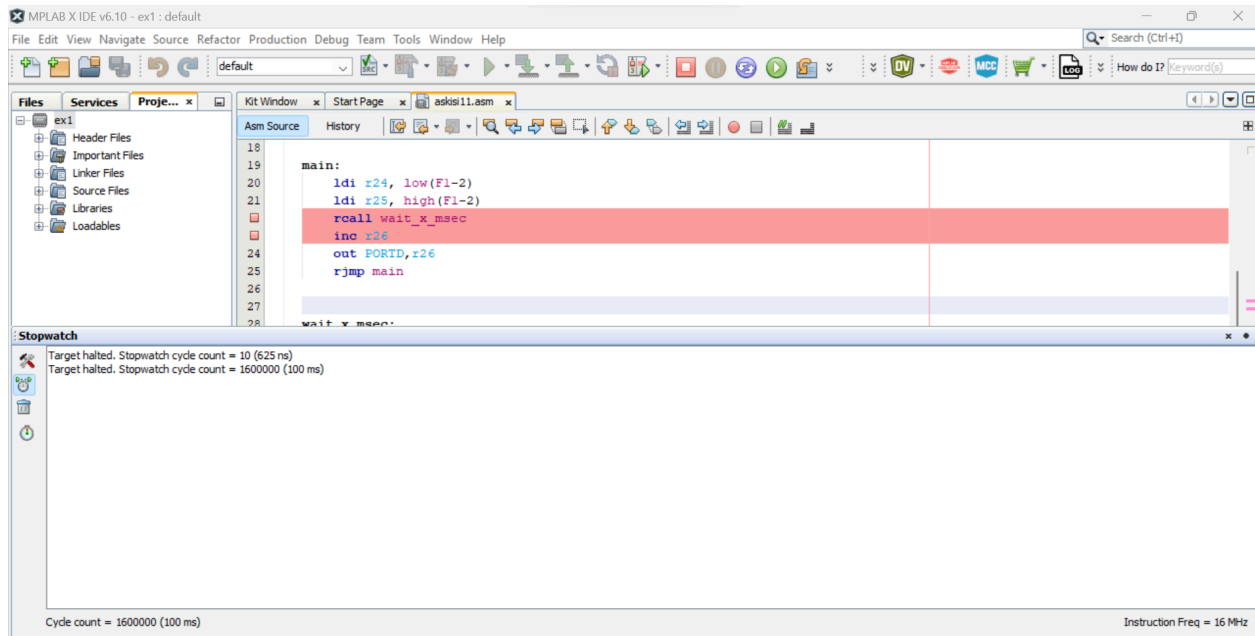
Στο παράδειγμα του εργαστηρίου διαμεσολαβούσαν μεταξύ των εντολών rcall delay_mS και της επόμενης εντολής 1600006 κύκλοι. Για να γλιτώσουμε μερικούς κύκλους, σπάμε την ήδη υπάρχουσα λούπα σε 3 μικρότερες προκειμένου μετά το ret της υπορουτίνας να έχουν μεσολαβήσει 1600000 ακριβώς κύκλοι.

Άρα με αυτή την υλοποίηση προκύπτουν:

3 cycles (rcall) + 2 cycles (ldi + nop) + 996 (delay_one) + 996 (delay_two) + 1000*1997 + 999 + 4 = 1600000 κύκλοι ακριβώς.

Τρέχοντας το παραπάνω πρόγραμμα στον Simulator του MPLABX, και τοποθετώντας δύο breakpoints στις εντολές rcall wait_x_msec και inc r26, μπορούμε να υπολογίσουμε τον χρόνο και τους κύκλους που μεσολαβούν μεταξύ των δύο εντολών με το εργαλείο Stopwatch.

Όπως βλέπουμε παρακάτω καταφέραμε με απόλυτη ακρίβεια να εισάγουμε χρονοκαθυστέρηση 100ms.



Ζήτημα 1.2:

Να υλοποιηθεί κώδικας assembly, για τον μικροελεγκτή ATmega328, για τον υπολογισμό των λογικών συναρτήσεων:

$$F0 = (A' \cdot B' \cdot C' + D)'$$

$$F1 = (A' + C) \cdot (B' + D)'$$

Ο υπολογισμός των συναρτήσεων να εισαχθεί σε ένα loop, το οποίο θα εκτελεστεί 5 φορές. Σε κάθε κύκλο η μεταβλητή A θα αυξάνεται κατά 0x01, η μεταβλητή B θα αυξάνεται κατά 0x02, η μεταβλητή C θα αυξάνεται κατά 0x04 και η μεταβλητή D θα αυξάνεται κατά 0x05.

Οι μεταβλητές A, B, C, D είναι μεγέθους ενός byte και έχουν αρχικές τιμές A=0x45, B=0x23, C=0x21 και D=0x01.

Υπολογισμοί για ζήτημα 1.2:

A	B	C	D	F0	F1
0x45	0x23	0x21	0x01	0x66	0xBA
0x46	0x25	0x25	0x06	0x61	0xB9
0x47	0x27	0x29	0x0B	0x64	0xB8
0x48	0x29	0x2D	0x10	0x6D	0xBF
0x49	0x2B	0x31	0x15	0x6A	0xB6

```

.include "m328PBdef.inc"
.equ FOSC_MHZ=16 ;MHz
.equ DEL_mS=100 ;mS
.equ F2=FOSC_MHZ*DEL_mS

.def A = r16 ; Variable A
.def B = r17 ; Variable B
.def C = r18 ; Variable C
.def D = r19 ; Variable D
.def F0 = r20 ; Result for F0
.def F1 = r21 ; Result for F1
.def i = r22 ; Loop counter

reset:
    ldi r24,LOW(RAMEND) ; Initialize stack pointer
    out SPL,r24
    ldi r24,HIGH(RAMEND)
    out SPH,r24

    ldi A, 0x45 ; Initialize variables
    nop
    ldi B, 0x23
    nop
    ldi C, 0x21
    nop
    ldi D, 0x01
    nop
    ldi i, 0x05

main:
    clr F0
    clr F1

    rcall F_0
    rcall F_1

    ldi r24, low(F2)
    ldi r25, high(F2)
    rcall delay_mS

    ; Increment variables

```

```

ldi r23, 0x01    ; Increment value for A
add A, r23
ldi r23, 0x02    ; Increment value for B
add B, r23
ldi r23, 0x04    ; Increment value for C
add C, r23
ldi r23, 0x05    ; Increment value for D
add D, r23

; Decrement counter and check if the loop should continue
dec i
brne main
breq end

```

F_0:

```

mov r23, A
com r23      ;A'
mov r26, B
com r26      ;B'
and r23, r26 ;A'*B' in r23
mov r27, C
com r27      ;C'
and r23, r27 ;A'*B'*C' in r23
or r23, D     ;A'*B'*C'+D in r23
com r23      ;(A'*B'*C'+D)'
mov F0, r23   ;to F0
ret

```

F_1:

```

clr r23
clr r26
mov r23, A
com r23      ;A'
or r23, C    ;A'+C sto r23
mov r26, B
com r26      ;B'
mov r28, D
com r28      ;D'
or r26, r28   ;B'+D' sto r26
mov F1, r23   ;A'+C sto F1
and F1, r26   ;(A'+C)*(B'+D')
ret

```

```

delay_mS:

    ;total group delay 996 cycles
delay_inner:
    ldi    r23, 249        ; (1 cycle)
loop_inn:
    dec r23                ; 1 cycle
    nop                    ; 1 cycle
    brne loop_inn         ; 1 or 2 cycles

    sbiw r24 ,1           ; 2 cycles
    brne delay_inner      ; 1 or 2 cycles

    ret                    ;4 cycles

end:

```

Η λειτουργία του κώδικα είναι σχετικά απλή και επεξηγείται στα σχόλια. Για να επιβεβαιώσουμε την ορθή λειτουργία του προγράμματος τοποθετούμε breakpoint μετά την κλήση των υπορουτινών F_0 και F_1 και ελέγχουμε τα αποτελέσματα σε κάθε επανάληψη. Τα αποτελέσματα παρατίθενται στις επόμενες εικόνες:

Variables x				
Name	Type	Address	Value	
<input checked="" type="checkbox"/> r22	SFR	0x16	0x05	
<input checked="" type="checkbox"/> r21	SFR	0x15	0xBA	
<input checked="" type="checkbox"/> r20	SFR	0x14	0x66	
<input checked="" type="checkbox"/> r19	SFR	0x13	0x01	
<input checked="" type="checkbox"/> r18	SFR	0x12	0x21	
<input checked="" type="checkbox"/> r17	SFR	0x11	0x23	
<input checked="" type="checkbox"/> r16	SFR	0x10	0x45	
<Enter new watch>				

Variables x				
Name	Type	Address	Value	
<input checked="" type="checkbox"/> r22	SFR	0x16	0x04	
<input checked="" type="checkbox"/> r21	SFR	0x15	0xB9	
<input checked="" type="checkbox"/> r20	SFR	0x14	0x61	
<input checked="" type="checkbox"/> r19	SFR	0x13	0x06	
<input checked="" type="checkbox"/> r18	SFR	0x12	0x25	
<input checked="" type="checkbox"/> r17	SFR	0x11	0x25	
<input checked="" type="checkbox"/> r16	SFR	0x10	0x46	
<Enter new watch>				

Variables				
Name	Type	Address	Value	
<input checked="" type="checkbox"/> r22	SFR	0x16	0x03	
<input checked="" type="checkbox"/> r21	SFR	0x15	0xB8	
<input checked="" type="checkbox"/> r20	SFR	0x14	0x64	
<input checked="" type="checkbox"/> r19	SFR	0x13	0x08	
<input checked="" type="checkbox"/> r18	SFR	0x12	0x29	
<input checked="" type="checkbox"/> r17	SFR	0x11	0x27	
<input checked="" type="checkbox"/> r16	SFR	0x10	0x47	
<Enter new watch>				

Variables				
Name	Type	Address	Value	
<input checked="" type="checkbox"/> r22	SFR	0x16	0x02	
<input checked="" type="checkbox"/> r21	SFR	0x15	0xBF	
<input checked="" type="checkbox"/> r20	SFR	0x14	0x6D	
<input checked="" type="checkbox"/> r19	SFR	0x13	0x10	
<input checked="" type="checkbox"/> r18	SFR	0x12	0x2D	
<input checked="" type="checkbox"/> r17	SFR	0x11	0x29	
<input checked="" type="checkbox"/> r16	SFR	0x10	0x48	
<Enter new watch>				

Variables				
Name	Type	Address	Value	
<input checked="" type="checkbox"/> r22	SFR	0x16	0x01	
<input checked="" type="checkbox"/> r21	SFR	0x15	0xB6	
<input checked="" type="checkbox"/> r20	SFR	0x14	0x6A	
<input checked="" type="checkbox"/> r19	SFR	0x13	0x15	
<input checked="" type="checkbox"/> r18	SFR	0x12	0x31	
<input checked="" type="checkbox"/> r17	SFR	0x11	0x28	
<input checked="" type="checkbox"/> r16	SFR	0x10	0x49	
<Enter new watch>				

Ζήτηση 1.3:

Να υλοποιηθεί κώδικας assembly, για τον μικροελεγκτή ATmega328, ο οποίος να ελέγχει ένα αυτοματισμό βαγονέτου που κινείται συνεχώς, αρχικά από δεξιά προς τα αριστερά και στη συνέχεια αντίστροφα.

Το βαγονέτο να προσομοιώνεται με ένα bit της θύρας εξόδου PORTD που κινείται συνεχώς από το LSb προς το MSb και αντίστροφα.

- Η κίνησή του βαγονέτου κατά μία θέση, θα γίνεται κάθε 1,5 sec περίπου
- Η κατεύθυνση της κίνησης να αποθηκεύεται στο T flag του SREG.
- Το βαγονέτο, κάθε φορά που αλλάζει κατεύθυνση, θα κάνει μία πρόσθετη στάση 2 sec περίπου δηλ. θα παραμένει στα άκρα 3,5 sec περίπου.
- Για τη δημιουργία των χρονικών καθυστερήσεων να χρησιμοποιηθεί ο κώδικας που υλοποιήθηκε στο ζήτημα 1.1


```

ldi r24, low(F1-2)          ; Load F1 delay
ldi r25, high(F1-2)
rcall wait_x_msec           ; Call delay subroutine
lsr leds                    ; Right shift LEDs state
rcall test_direction2       ; Call direction test subroutine
brtc left                   ; If T flag is cleared, go left
rjmp right                  ; Otherwise, continue right

test_direction1:
    cpi leds, 0b10000000    ; Test if LEDs are at the leftmost
position
    brne left               ; If not, continue left
    breq change_to_right    ; If yes, change direction to right

test_direction2:
    cpi leds, 0b00000001    ; Test if LEDs are at the rightmost
position
    brne right              ; If not, continue right
    breq change_to_left     ; If yes, change direction to left

change_to_right:
    set                      ; Set the T flag for right direction
    ldi r24, low(F2-2)       ; Load F2 delay
    ldi r25, high(F2-2)
    rcall wait_x_msec        ; Call delay subroutine
    rjmp right

change_to_left:
    clt                      ; Clear the T flag for left direction
    ldi r24, low(F2-2)       ; Load F2 delay
    ldi r25, high(F2-2)
    rcall wait_x_msec        ; Call delay subroutine
    rjmp left

wait_x_msec:
    ldi r23, 249
    nop

delay_one:

```

```

    dec r23
    nop
    brne delay_one

    ldi r23, 249

delay_two:
    dec r23
    nop
    brne delay_two

    ldi r23, 249

delay_three:
    ldi r23, 249
loop:
    dec r23
    nop
    brne loop

    sbiw r24, 1
    brne delay_three

    ret

```

Η λειτουργία του κώδικα εξηγείται στα σχόλια. Η κατεύθυνση του βαγονέτου καθορίζεται από την κατάσταση του T flag του SREG. Εκτελώντας το πρόγραμμα στον Simulator του MPLABX μπορούμε να επιβεβαιώσουμε την λειτουργία του προγράμματος. Επιπλέον μπορούμε να εξετάσουμε και τις χρονικές καθυστερήσεις που εισάγουμε.

Χρησιμοποιούμε πάλι το εργαλείο Stopwatch και βάζουμε breakpoints πριν τις εντολές lsl και lsr όπου τα LED μετακινούνται.

Μπορούμε να δούμε ότι οι πρώτες 7 χρονοκαθυστερήσεις που αντιστοιχούν στις 7 πρώτες μετακινήσεις του βαγονέτου είναι 1.5s, ενώ στην πρώτη αλλαγή κατεύθυνσης έχουμε καθυστέρηση 3.5s, ακριβώς όπως μας ζητείται.

