2η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ
ΓΙΑ ΤΟ ΜΑΘΗΜΑ "Εργαστήριο Μικροϋπολογιστών"
(Στο Σύστημα AVR16mega))
Παραλίκας Ηλίας 03116605
Ιωάννης Φαρδέλας 03113190

1. Η άσκηση ζητάει να κάνουμε τις εξής λογικές πράξεις

F0 = (A'B + B'CD)'
F1= (A*C)*(B+D)

με είσοδο
A,B,C,D = PORTC[0],PORTC[1] ,PORTC[2] ,PORTC[3]
και έξοδο
F0 ,F1 =PORTB[0],PORTB[1]

Assembly
ο κώδικας έχει 2 τμήματα.
Είσοδος
Λογικές Πράξεις

```
.INCLUDE "m16def.inc"

.def input = r17
.def temp = r18
.def temp2=r19
.def A = r20
.def B = r21
.def C = r22
.def D = r23
.def F0 = r24
.def F1 = r25


main:
        clr temp
        out DDRC ,temp
        ser temp
        out DDRB,temp
;*******************************Είσοδος*********************************
        in input , PINC             ; διαβάζω το PINC
        mov A,input
        andi A,0x01                 ; απομονώνω το LSB στον A με μάσκα

        mov B,input
        andi B,0x02                 ; είναι στην δεύτερη θέση
        ror B                       ; αλλά εμείς θέλουμε να κάνουμε λογικές
                                    ; πράξεις bit-wise, άρα πρέπει να είναι όλα
                                    ; στην ίδια θέση, άρα rotate

        mov C,input                 ; παρόμοια για το C & D
        andi C,0x04
        ror C
        ror C
```

```
        mov D,input
        andi D,0x08
        ror D
        ror D
        ror D
 ;*****************************Είσοδος*********************************

;***************************Λογικές Πράξεις*****************************

        mov temp2,B
        com temp2                   ; φτιάχνουμε το B’
        andi temp2 ,0x01            ; όταν κάναμε comp όλα τα μηδενικά έγιναν 1

        mov temp,A
        com temp                    ; A’
        andi temp ,0x01


        and temp,B                  ; temp=  A'B
        and temp2,C                 ; temp2= B'C
        and temp2,D                 ; temp2= B'CD
        or temp,temp2               ; temp=  A'B+B'CD
        mov F0,temp

        mov temp,A                  ; temp = A
        and temp,C                  ; temp = AC

        mov temp2,B                 ; temp2= B
        or temp2,D                  ; temp2= B+D
        and temp,temp2              ; temp =(AC)(B+D)
        mov F1,temp

        clc                         ; θα κάνουμε rotate,για να φέρουμε
                                    ; το F1 στην δεύτερη θέση
                                    ; το οποίο θα βάλει το Carry
                                    ; στην αρχή, οπότε το κάνουμε 0
        rol F1                      ;
        com F0                      ; F0 = (A'B+B'CD)’
        andi F0,0x01                ;
        add F0,F1
        out PORTB,F0                ; βγάζουμε την έξοδο στοPORTB
        rjmp main
end:
```

```
;***************************Λογικές Πράξεις*****************************
```

C

ο κώδικας έχει ακριβώς την ίδια λογική, μόνο που οι εντολές είναι πιο συμπυκνωμένες

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


unsigned char A,B,C,D,NOTA,NOTB,F0,F1;
```

```
int main(void){


        DDRC=0x00;                  // C= input
        DDRB=0xff;                  // B= output
        while(1){
;*******************************Είσοδος*********************************
                A = PINC &0x01;     // διαβάζω και μάσκα στη ίδια γραμμή
                B = PINC &0x02;
                B = B >> 1;         // rotate για τους ίδιους λόγους
                C = PINC & 0x04;
                C = C >> 2;
                D = PINC &0x08;
                D = D >> 3;
;*******************************Είσοδος*********************************

;***************************Λογικές Πράξεις**************************
                NOTA = A ^ 0x01;    // xor συμπληρώνει το τελευταίο μπιτ, ενώ
                                    // κρατάει τα άλλα 0
                NOTB = B ^ 0x01;
                F0 = (NOTA & B)|(NOTB & C & D);
                F0 = F0 ^ 0x01; // F0'
                F1 = (A & C) & (B | D);
                F1 = F1 << 1;
                F0 = F0 + F1;
;***************************Λογικές Πράξεις**************************
                PORTB = F0; // output to the port B


        }
}
```

2 Το ζητούμενο είναι ένας μετρητής διακοπών, υπό συνθήκες, ο οποίος θα τρέχει παράλληλα με έναν μετρητή χρόνου.

Ο κώδικας έχει 3 τμήματα
Ενεργοποίηση διακοπών
Μετρητής χρόνου
Έλεγχος διακοπών

```
.include "m16def.inc"

                .org 0x0
                rjmp RESET
                .org 0x4
                rjmp ISR1


RESET:
        .def temp = r20
        .def input =r21
        .def interupt_counter=r22
        .def timer_counter=r26

        clr interupt_counter
        clr timer_counter

        ldi temp,LOW(RAMEND)        ; βάζουμε τον stack pointer στο τέλος της
        out SPL,temp                ; RAM
        ldi temp,HIGH(RAMEND)
        out SPH,temp

;**********Ενεργοποίηση διακοπών**********
        ldi r24,(1<<ISC11)|(1<<ISC10)
                                    ; διακοπή στην ανερχόμενη ακμή του INT1
        out MCUCR,r24
        ldi r24 ,(1<<INT1)
        out GICR,r24                ; Ενεργοποίησε τη διακοπή INT1
        sei                         ; enable interupts
;**********Ενεργοποίηση διακοπών**********
        ser temp                    ; θύρα D έξοδος
        out DDRC,temp

;**********Μετρητής χρόνου**********
loop:
        out PORTC,timer_counter     ; δείξε τον μετρητή
        ;ldi r24,low(100)           ; wait
        ;ldi r25,high(100)
        ;rcall wait_msec
        inc timer_counter           ; αύξηση
        rjmp loop                   ; ξανά
;**********Μετρητής χρόνου**********


;**********Έλεγχος διακοπών**********
ISR1:
        cli                         ; δεν δεχόμαστε άλλες διακοπές όσο
                                    ; εξυπηρετούμε άλλη
```

```
        in temp, SREG               ; αποθήκευση του System Register
        push temp
        clr temp
        out DDRA,temp               ; A= είσοδος
        ser temp
        out DDRB,temp               ; B = έξοδος

        inc interupt_counter        ; μετράω μία ακόμα διακοπή

        in input,PINA               ; κοιτάω ,από την A
        cpi input,0xC0              ; τα 2 MSB
        brne dont_display_the_interupt_counter
                                    ; εάν δεν είναι και τα 2 ON
                                    ; δεν δείχνω τίποτα
        out PORTB,interupt_counter  ; αλλιώς στο PORTB

dont_display_the_interupt_counter:
        ;ldi r24,low(100)
        ;ldi r25,high(100)
        ;rcall wait_msec
        pop temp
        out SREG,temp               ; επαναφέρω την τιμή του SREG
        reti                        ; επιστρέφω
```

;********Έλεγχος διακοπών**********

3 Η άσκηση μας ζητάει να φτιάξουμε ένα πρόγραμμα το οποίο , ενώ στο σώμα του δεν κάνει τίποτα, busy waiting, στην περίπτωση διακοπής μετράει τα αναμέβα PINB's και βγάζει έξοδο, ανάλογα με το PA2

Ο κώδικας έχει 2 τμήματα
Ενεργοποίηση διακοπών
Εξυπηρέτηση διακοπών

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


unsigned char A,B,temp;
char x,c;
volatile int flag=1;



;**********************Εξυπηρέτηση διακοπών**********************
 ISR(INT0_vect){
      cli();                        // κλείνω τις περαιτέρω διακοπές

      x=0x00;
      if((PINB &0x01) == 0x01) x++;
      if((PINB &0x02) == 0x02) x++;
      if((PINB &0x04) == 0x04) x++;
      if((PINB &0x08) == 0x08) x++;
      if((PINB &0x10) == 0x10) x++;
      if((PINB &0x20) == 0x20) x++;
      if((PINB &0x40) == 0x40) x++;
      if((PINB &0x80) == 0x80) x++; // μετράω τον αριθμό των PINB


        if((PINA & 0x04) == 0x04){  // εάν  PA2 ON θέλω την τιμή των αναμένων B
                c=0x00;
                while(x>0){
                        c=c<<1;
                        c=c+1;
                        x=x-1;
                }
                PORTC=c;            // εμφανίζω


        }
        else{
                PORTC=x;            // αλλιώς με ενδιαφέρει απλά η θέση και
                                    // τα εμφανίζω όπως είναι
        }
        sei();                      // επιτρέπω διακοπές
}
;**********************Εξυπηρέτηση διακοπών**********************


int main(){

;**********************Ενεργοποίηση διακοπών**********************
        GICR=(1<<INT0);             // ενεργοποίησε INT0
        MCUCR = (1<<ISC01|1<<ISC00);
                                    // στην θετική ακμή
```

```
        sei();
;**********Ενεργοποίηση διακοπών**********
        DDRA = 0x00;                // A είσοδος
        DDRB = 0x00;                // B είσοδος
        DDRC = 0xff;                // C έξοδος
        while(1){
                PORTC=0x00;         // busy wait
                }
        }
```