ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΙΣΤΩΝ

ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ – 7ο ΕΞΑΜΗΝΟ

Δημητρίου Αγγελική – ΑΜ: 03117106

Τζομάκα Αφροδίτη – ΑΜ: 03117107

Ομάδα Β8

2η Εργαστηριακή Άσκηση

ΖΗΤΗΜΑ 2.1

Αρχικά απομονώνουμε τα bit της εισόδου (A, B, C, D) στα lsb των καταχωρητών/μεταβλητών (assembly/C) και στη συνέχεια μέσω λογικών εντολών/τελεστών υπολογίζουμε διαδοχικά τις επιμέρους λογικές πράξεις και τις συνθέτουμε για να δημιουργήσουμε το σωστό αποτέλεσμα στα bit της εξόδου (F0, F1). Τα Bit αυτά τα απεικονίζουμε μετατοπίζοντάς τα στις σωστές θέσεις (1ο και 2ο LSB).

(Assembly)

```
.include "m16def.inc"
.DEF A=r16                ; registers declaration
.DEF AN=r17
.DEF B=r18
.DEF BN=r19
.DEF C=r20
.DEF D=r21
.DEF temp=r22

.cseg
.org 0                    ; starting address
reset: clr temp
       out DDRC,temp      ; port C for input
       ser temp
       out PORTC,temp ; pull-up for port C
       out DDRB,temp  ; port B for ouput
main:  in temp,PINC       ; reading from portC
       mov A,temp         ; A in As LSB
       mov AN,temp
       com AN          ; A' in ANs LSB
       lsr temp        ; shift and repeat
       mov B,temp
       mov BN,temp     ; Bs complement
       com BN          ; in BNs LSB and same for the other bits
       lsr temp
       mov C,temp
       lsr temp
       mov D,temp
       and AN,B        ; AN<-A'B
       and BN,C
```

```
        and BN,D        ; BN<-B'CD
        or AN,BN
        com AN
        andi AN,1       ; LSB(AN)<-F0
        and A,C         ; A<-AC
        or B,D          ; B<-B+D
        and A,B
        andi A,1        ; LSB(A)<-F1
        lsl A          ; shift to display F1 in pin1 of portA
        add A,AN       ; A = 0000 00F1F0
        out PORTB,A
        rjmp main
```

(C)

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

char a,b,c,d,not_a,not_b,f0,f1;

int main(void)
{
    DDRB=0xFF; //initializing PORTB as output
    DDRC=0x00; //initializing PORTA as input

       while (1)
    {
              a = PINC & 0x01;         //mask lsb  (A)
              b = (PINC & 0x02)>>1;    //mask 2nd lsb and shift one pos right (B)
              c = (PINC & 0x04)>>2;    //now (C) in lsb
              d = (PINC & 0x08)>>3;    //now (D) in lsb


              not_a = (~a) & 0x01;     //complements (masked in lsb)
              not_b = (~b) & 0x01;
//logic expressions using logic and, or (1st one needs the mask (~))
              f0 = (~((not_a && b) || (not_b && c && d))) & 0x01;
              f1 = ((a && c) && (b || d));

              PORTB = (f0 | (f1<<1)) ; //shift f1 1 pos left and do bitwise or for
output
    }
}
```

ΖΗΤΗΜΑ 2.2

Ξεκινάμε αρχικοποιώντας τον δείκτη της στοίβας του προγράμματος ώστε να μπορούμε μετά την εκτέλεση της ρουτίνας εξυπηρέτησης να επιστρέψουμε στη σωστή θέση και θέτοντας τις κατάλληλες τιμές στους καταχωρητές MCUCR, GICR για την ενεργοποίηση της διακοπής INT1. Το κυρίως πρόγραμμα αποτελεί έναν κλασσικό μετρητή με έλεγχο υπερχείλισης, όπου στην συγκεκριμένη περίπτωση έχουμε

μηδενισμό και μέτρημα ξανά από την αρχή. Στη συνέχεια, η ρουτίνα εξυπηρέτησης αποτελεί και αυτή έναν μετρητή, των διακοπών INT1 αυτήν την φορά, ο οποίος μόνο εάν είναι πατημένα τα PA6-PA7 ανανεώνει και εμφανίζει το αποτέλεσμα της μέτρησης στα led της PORTB. Για την ορθή λειτουργία του προγράμματος, σώζει στην στοίβα τον καταχωρητή r26 (που χρησιμοποιείται από το κυρίως πρόγραμμα) όπως και τον SREG.

```
.include "m16def.inc"
    .org 0x0
    rjmp reset
    .org 0x4                    ;INT1 service routine always at 0x04
    rjmp ISR1

reset:
    ldi r24,low(RAMEND)         ;Initialize stack pointer
    out SPL,r24
    ldi r24,high(RAMEND)
    out SPH,r24
    cli
    ldi r24 ,( 1 << ISC11) | ( 1 << ISC10) ;interrupt at rising edge for INT1
    out MCUCR , r24
    ldi r24 ,(1 << INT1)                   ;activate INT1
    out GICR , r24
    sei                                    ;interrupt enable

    clr r16                    ;counter of interrupts
    clr r26
    out DDRA, r26              ;PORTA as input
    ser r26
    out PORTA, r26             ;pull-up for PORTA
    out DDRC, r26              ;PORTC as output
    out DDRB, r26              ;PORTB as output
    clr r26                    ;counter of main program initialized at zero

loop:
    out PORTC, r26             ;show counter
    inc r26                    ;increase counter
    cpi r26,255
    brne overflowcheck
    out PORTC, r26
    clr r26
overflowcheck:
    rjmp loop

ISR1:
    push r26                   ;save counter of main program
    in r26, SREG
    push r26                   ;and SREG

    inc r16                    ;increase interrupt counter
    in r17, PINA               ;check for PA6-PA7
    andi r17, 0b11000000
    cpi r17, 0b11000000
    brne noshow                ;if both ON show interrupt counter
    out PORTB, r16

noshow:
    pop r26
    out SREG, r26
    pop r26
    reti                       ;return to main program
```

ΖΗΤΗΜΑ 2.3

Το πρόγραμμα ξεκινάει κλασσικά με την αρχικοποίηση των καταχωρητών των διακοπών για την ενεργοποίηση της διακοπής INT0 και των πυλών εισόδου και εξόδου και ύστερα περιμένει την εμφάνιση κάποιας διακοπής. Η ρουτίνα εξυπηρέτησης διακοπής INT0 λαμβάνει είσοδο από το PORTB και σειριακά μετράει τα bit που είναι 1 και ετοιμάζει τις 2 μορφές εξόδου που ζητά η άσκηση ταυτόχρονα. Ανάλογα με το PA2 εμφανίζει το κατάλληλο αποτέλεσμα στο PORTC.

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

unsigned char cnt, x, y;
unsigned char tmp;

ISR(INT0_vect)
{
	x = PINB;			//save PORTB dip switches at x
	y = PINA & 0x04;		//save PA2 at y
	cnt=0x00;			//initialize counters
	tmp=0x00;
	while (x != 0x00)
	{
		if(x & 0x01)	//checking if each bit of x is 1
		{
			cnt++;				//binary form
			tmp = (tmp << 1) + 1;		//1s starting from lsb
		}
		x >>= 1;
	}

	PORTC = y ? tmp : cnt;	//show output depending on y (PA2)
}

int main(void)
{
	cli();
	MCUCR = (1<<ISC01)|(1<<ISC00);	//rising edge
	GICR = (1<<INT0);		//set interrupt INT0
	sei();				//interrupt enable

	DDRC=0xff;	//initializing PORTC as output
	DDRA = 0x00;
	DDRB = 0x00;	//initializing PORTA and PORTB as inputs



	while(1){
		asm("nop");
	}
}
```