<u>ΕΘΝΙΚΌ ΜΕΤΣΌΒΙΟ ΠΟΛΥΤΕΧΝΕΊΟ</u>

<u>ΣΧΟΛΉ ΗΛΕΚΤΡΟΛΌΓΩΝ ΜΗΧΑΝΙΚΏΝ ΚΑΙ ΜΗΧΑΝΙΚΏΝ ΥΠΟΛΟΓΙΣΤΏΝ</u>

<u>ΕΡΓΑΣΤΉΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΏΝ</u>

(2020-2021)

2η ΟΜΆΔΑ ΑΣΚΉΣΕΩΝ

<u>Ονοματεπώνυμο</u>:

- Χρήστος Τσούφης

<u>Αριθμός Μητρώου</u>:

- 03117176

<u>Ομάδα Εργαστηρίου</u>:

- Β15

<u>Εξέταση – Επίδειξη</u>:

- 4/11/2020

1^{η} Άσκηση

Ο πηγαίος κώδικας, μαζί με τα απαραίτητα σχόλια:

Σε **C**:

```c
#include <avr/io.h>
int main(void)
{
        char x, a, b, c, d, f0, f1;
        DDRC = 0x00;                            //input port
        DDRB = 0xFF;                            //output port
        while (1)
        {
                asm("break");
                x = PINC;                       //input from switches
                a = x & 1;
                b = (x >> 1) & 1;               //rotations
                c = (x >> 2) & 1;
                d = (x >> 3) & 1;
                f0 = ~(((~a) & b) | ((~b) & c & d)); //f0
                f1 = (a & c) & (b | d) << 1;    //f1
                f0 = f0 & 1;                    //ignore all bits except bit 1
                f1 = f1 & 2;                    //ignore all bits except bit 2
                PORTB = f0 | f1;
        }
}
```

Σε **assembly**:

```
.include "m16def.inc"
.DEF A = r16
.DEF B = r17
.DEF C = r18
.DEF D = r19
.DEF temp = r20
.DEF temp2 = r21
.DEF res = r22

start:
       clr res
       clr temp
       out DDRC,temp          ; Port C input
       ser temp
       out PORTC,temp         ; Port C pull-up
       out DDRB,temp          ; Port B output

calc:
       break
       in temp,PINC
       mov A,temp             ; A in LSB of reg A
       lsr temp
       mov B,temp             ; similar for B
       lsr temp
       mov C,temp             ; similar for C
       lsr temp
       mov D,temp             ; similar for D

       mov temp,A
       com temp
       and temp,B
       mov temp2,B
       com temp2
       and temp2,C
       and temp2,D
       or temp,temp2
       com temp
       andi temp,1
       mov res,temp
       break

       and A,C
       or B,D
       and A,B
       andi A,1
       break
       lsl A                  ; move F1 to 1st bit

       or res,A
       out PORTB,res
       rjmp calc
```

2η Άσκηση

Ο πηγαίος κώδικας, μαζί με τα απαραίτητα σχόλια:

```
.include "m16def.inc"
.DEF tmp = r16
.DEF count = r17
.DEF INTcount = r18
.cseg

.org 0x0
rjmp reset

.org 0x4 ;necessary for the jmp to ISR1
rjmp ISR1

reset: ;initializations
       ldi tmp,(1 << ISC11) | (1 << ISC10)       ; INT1 at rising edge
       out MCUCR, tmp

       ldi tmp,(1 << INT1) ; INT1 enable (PD3)
       out GICR, tmp
       sei

       ser tmp
       out DDRC, tmp ; PORTC Output

main:
       out PORTC, count

       inc count ;increase count
       break
       rjmp main

ISR1:
       in tmp, SREG ; push status reg to stack
       push tmp

       clr tmp
       out DDRA, tmp

       sbic PINA,7  ;if equal to 0, don’t count the interrupt & return to main
       sbis PINA,6  ;if equal to PINA6=1 & PINA7=1, continue
       rjmp return

       ser tmp ;increase temp
       out DDRB, tmp ; Output PORTB for counting interrupts

       inc INTcount
       out PORTB, INTcount

return:
       pop tmp
       out SREG, tmp ;recover status register
       reti
```

3η Άσκηση

Ο πηγαίος κώδικας, μαζί με τα απαραίτητα σχόλια:

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

char x, y;

ISR (INT0_vect)
{
        x=PINA; //LED A
        y=PINB; //LED B
        x  &= 0x04; // keep PA2, mask x

        int i, count, leds;
        count=0; //count is used for the quantity of switches of PORTB
        leds=0x00;
        for(i=0;i<8;i++) //calculates how many switches are ON
        {
                if ( (y & 0x01)==1 ) //check LSB of y
                {
                        count+=1 ;
                        leds = leds <<1; //every time I find a LED ON, rotate leds
                        leds+=1;         //then add one
                }
                y = y>>1; //rotate
        }

        PORTC = (x==4) ? count : leds; //if x=4 (A2 is ON), output count in
binary, else output leds
}
int main(void)
{
        DDRA=0; //initialize led in 0
        DDRB=0; //initialize led in 0
        PORTC=0; //initialize led in 0
        DDRC=0xFF;

        GICR|=(1<<INT0); // enable INT0
        MCUCR=0x03; // rising edge
        sei(); //enable interrupts
        while(1) //active loop
        {
                asm("break");
        }

}
```