Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Ακαδημαϊκό Έτος 2020-2021

Εξάμηνο 7ο

Εργαστήριο Μικροϋπολογιστών

2η Εργαστηριακή Αναφορά

Κωστάκη Χριστίνα (el18136)
Τσάφος Αλέξανδρος (el18211)

Ζήτημα 2.1

assembly:

```
.include"m16def.inc"

.DEF A = r16        ;definition of registers
.DEF B = r17
.DEF C = r18
.DEF D = r19
.DEF input = r20
.DEF F0 = r21
.DEF E = r22        ;ancillary register

start:
ser r24
out DDRB, r24       ;set PORTB as output
clr r24
out DDRC, r24       ;set PORTC as input


in input, PINC
mov A, input        ;A = 1st LSB
lsr input
mov B, input        ;B = 2nd LSB
lsr input
mov C, input        ;C = 3rd LSB
lsr input
mov D, input        ;D = 4th LSB

FO:
mov F0, A           ;F0=A
com F0              ;F0=A'
and F0, B           ;F0=A'B

mov E, B            ;E=B
com E               ;E=B'
and E, C            ;E=B'C
and E, D            ;E=B'CD

or F0, E            ;F0=(A'B + B'CD)
com F0              ;F0=(A'B + B'CD)'
andi F0, 1          ;mask for 1st LSB
```

```
F1:
and A, C          ;A=AC
or B, D           ;B=B+D
and A, B          ;A=(AC)(B+D)=F1
lsl A
andi A, 2         ;mask for 2nd LSB
or F0, A          ;F0+F1
out PORTB, F0     ;output

rjmp start
```

C:

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

char A, B, C, D, F0, F1;

int main(void)
{
      DDRB = 0xFF;
      DDRC = 0x00;


      while (1){

            //Isolating the bits
            A = PINC & 0x01;
            B = PINC & 0x02;
            C = PINC & 0x04;
            D = PINC & 0x08;

            //Bringing the bit to the LSB
            B = B >> 1;
            C = C >> 2;
            D = D >> 3;

            F0 = !((!A)&B)|((!B)&C&D);
            F1 = (A&C)&(B+D);
            F1 = F1 << 1; //2nd LSB

            PORTB = F1|F0;
      }
}
```

Ζήτημα 2.2

```
.org 0x0                 ;code always starts at 0x0
rjmp reset
.org 0x4                 ;INT1 address is 0x4
rjmp ISR1

reset:
ldi r24 , low(RAMEND)    ;initialize stack pointer
out SPL , r24
ldi r24 , high(RAMEND)
out SPH , r24
ldi r24 ,( 1 << ISC10) | ( 1 << ISC11)
;interrupt starts at positive edge
out MCUCR,r24
ldi r24,( 1 << INT1)     ;enable INT1 interrupt
out GICR,r24

ser r26
out DDRC , r26           ;port C for output
out DDRB , r26           ;port B for output
clr r26
out DDRA , r26           ;port A for input

sei                      ;enable interrupts

loop:                    ;counter copied from ex2.1 (changed the port)
out PORTC , r26          ;counter's loop
ldi r24 , low(100)
ldi r25 , high(100)
rcall wait_msec           ;delay 100ms
inc r26
rjmp loop

ISR1:
cli                       ;Disbale interrupts
in r28, PINA              ;check PA7, PA6
andi r28, 0xC0
```

```
cpi r28,0xC0                  ;compare with 1100 0000
brne skip                     ;if PA7 and PA6 are set, increase
in r27, PORTB                  ;the counter
inc r27
out PORTB, r27
skip:
reti
```

Ζήτημα 2.3

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

char ret;
int counter;

void INT0_Enable(void){
      MCUCR = (1<<ISC01)|(1<<ISC00);//Positive edge enabling
      GICR = (1<<INT0);              // INT0 enabling
      asm("sei");                    // Enable Interrupts
}


ISR(INT0_vect){
      asm("cli");
      counter = 0x00;
      ret = PINB;
      //Counting the number of 1's in PINB
      for(int i = 0; i < 8; i++){
            if ((ret & 0x01) == 0x01) counter = counter + 0x01;
            ret = ret>>1;
      }
      //If PA2 is ON, we display the counter in binary form
      if((PINA & 0x04) == 0x04){
            PORTC = counter;
      }
      //Else, we display the same number of consecutive 1's
      //starting from the LSB
      else{
            ret = 0x00;
            for (int i = 0; i < counter; i++){
                  ret = ret + 0x01;
                  ret = ret << 1;
            }
            //Bringing the first 1 back to the LSB
            PORTC = ret >> 1;
      }
      asm("sei");
}


int main(void){
      DDRA = 0x00;      //Input
      DDRB = 0x00;      //Input
      DDRC = 0xFF;      //Output
      INT0_Enable();

      while (1){
            ret = 0x00; //Do nothing
            continue;
      }
}
```