Εργαστήριο Μικροϋπολογιστών – 2η Σειρά Ασκήσεων

Κυριακόπουλος Γιώργος – el18153
Τζελέπης Σεραφείμ – el18849

1η Άσκηση:

C version

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

unsigned char A, B, C, D, notA, notB, F0, F1, answer, temp;

int main(void) {
    DDRC = 0x00;                    // input from C
    DDRB = 0xFF;                    // output at B

    while(1) {
        A = PINC & 0x01;            // keep LSB
        B = PINC & 0x02;            // keep 2nd LSB
        B = B >> 1;                 // shift it to LSB so LSB(B) = B
        C = PINC & 0x04;            // keep 3rd LSB
        C = C >> 2;                 // shift it to LSB so LSB(C) = C
        D = PINC & 0x08;            // keep 4th LSB
        D = D >> 3;                 // shift it to LSB so LSB(D) = D

        notA = A ^ 0x01;            // complement A
        notB = B ^ 0x01;            // complement B

        temp = ((notA & B) |
                (notB & C & D));    // temp = (A'B + B'CD)
        F0 = temp ^ 0x01;           // complement temp to get F0

        F1 = ((A & C) &
              (B | D));             // F1 = (AC)(B+D)
        F1 = F1 << 1;               // shift F1 to 2nd LSB
        answer = F0 + F1;           // add F1 and F0 to get output
        PORTB = answer;             // output answer at B
    }
}
```

Assembly version

```
.include "m16def.inc"

.DEF A = r16
.DEF B = r17
.DEF C = r18
.DEF D = r19
.DEF F0 = r20
.DEF F1 = r21
.DEF TEMP = r22
.DEF ANSWER = r23

stack:
        ldi r24, low(RAMEND)    ; initialize stack pointer
        out SPL, r24
        ldi r24, high(RAMEND)
        out SPH, r24
IO_set:
        clr r24
        out DDRC, r24           ; input from C
        ser r24
        out DDRB, r24           ; output at B
main:
        in A, PINC              ; load input on A
        mov TEMP, A             ; backup input on TEMP
        andi A, 0x01            ; LSB(A) = A

        mov B, TEMP             ; load input on B
        andi B, 0x02
        lsr B                   ; LSB(B) = B

        mov C, TEMP             ; load input on C
        andi C, 0x04
        lsr C
        lsr C                   ; LSB(C) = C

        mov D, TEMP             ; load input on D
        andi D, 0x08
        lsr D
        lsr D
        lsr D                   ; LSB(D) = D
```

```
mov TEMP, A             ; TEMP = A
com TEMP                ; TEMP = A'
and TEMP, B             ; TEMP = A'B
mov F0, TEMP            ; F0 = A'B

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

or F0, TEMP             ; F0 = A'B + B'CD
com F0                  ; F0 = (A'B + B'CD)'

mov TEMP, A             ; TEMP = A
and TEMP, C             ; TEMP = AC
mov F1, TEMP            ; F1 = AC

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

andi F0, 0x01           ; keep only LSB
andi F1, 0x01           ; keep only LSB
lsl F1                  ; move F1 to 2nd LSB
mov ANSWER, F1          ; ANSWER = 000000(F1)0
or ANSWER, F0           ; ANSWER = 000000(F1)(F0)
out PORTB, ANSWER       ; output ANSWER at B

rjmp main               ; restart the program
```

2η Άσκηση:

```
.org 0x0
    rjmp reset
.org 0x4
    rjmp ISR1

.DEF COUNTER = r20
.DEF TEMP = r21

reset:
    ldi r24 , low(RAMEND)                ; initialize stack pointer
    out SPL , r24
    ldi r24 , high(RAMEND)
    out SPH , r24

    ldi r23, (1 << ISC11)|(1 << ISC10)   ; rising edge of INT1
    out MCUCR, r23
    ldi r23, (1 << INT1)                 ; enable INT1
    out GICR, r23
    sei                                  ; enable interrupts
IO_set:
    clr r26
    out DDRA, r26                        ; input from A
    ser r26
    out DDRC, r26                        ; output at C
    out DDRB, r26                        ; output at B (INT COUNTER)
    clr r26                              ; initialize counter
    clr COUNTER                          ; initialize INT COUNTER
loop:
    out PORTC, r26                       ; sent counter to output
    inc r26                              ; increase counter
    rjmp loop                            ; repeat until interrupt

ISR1:
    push r26                             ; save r26
    in r26, SREG
    push r26                             ; save SREG

    in TEMP, PINA                        ; load input to TEMP
    andi TEMP, 0xC0                      ; keep only A7 and A6
    cpi TEMP, 0xC0                       ; check if A7 and A6 are on
    brne EXITINT                         ; if they are not exit
```

```
    inc COUNTER                         ; increase INT COUNTER
    out PORTB, COUNTER                  ; display it on PORTB LEDs
EXITINT:
    pop r26                             ; restore SREG
    out SREG, r26
    pop r26                             ; restore r26
    reti                                ; return to loop
```

3η Άσκηση:

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

volatile unsigned char A, B, counter, flag = 1;

ISR(INT0_vect) {
    A = PINA & 0x04;                    // keep only PA2
    B = PINB;                           // load input from B

    if(A) {                             // if PA2 is ON
        counter = 0x00;                 // initialize counter
        for (int i = 0; i < 8; i++) {   // for loop 8 times
            if(B & 0x01) {              // if LSB is 1
                counter++;              // increase counter
            }
            B = B >> 1;                 // rotate input right
        }
        PORTC = counter;                // output counter to C
    }
    else {                              // if PA2 is OFF
        counter = 0x00;                 // initialize counter
        for (int i = 0; i < 8; i++) {   // for loop 8 times
            if(B & 0x01) {              // if LSB is 1
                counter = counter << 1; // shift counter left
                counter++;              // increase counter
            }
            B = B >> 1;                 // rotate input right
        }
        PORTC = counter;                // output counter to C
    }
}

int main(void) {
    MCUCR = 0x03;                       // rising edge of INT0
    GICR = 0x40;                        // enable INT0

    DDRA = 0x00;                        // input from A
    DDRB = 0x00;                        // input from B
    DDRC = 0xFF;                        // output at C

    while(flag) {                       // infinite loop
```

```
        sei();                          // enable interrupts
    }
}
```