Εργαστήριο Μικροϋπολογιστών
2η Εργαστηριακή Άσκηση
Ονοματεπώνυμα-Α.Μ.:Μοίρας Αλέξανδρος-03118081
Παντελαίος Δημήτριος-03118049

Ζήτημα 2.1

Assembly:

```
.DEF Input=r16
.DEF A=r17
.DEF B=r18
.DEF C=r19
.DEF D=r20
.DEF Acomp = r21
.DEF Bcomp = r22
.DEF f0 = r23
.DEF f1 = r24

start:
	ldi r24, low(RAMEND) ; Αρχικοποίηση στοίβας στο τέλος της RAM
	out SPL, r24
	ldi r24, high(RAMEND)
	out SPH, r24
	clr r24
	out DDRC , r24		;αρχικοποίηση PORTC ως input
	ser r24
	out PORTC , r24
	out DDRB , r24		;αρχικοποίηση PORTB ως output

readinput:
	in Input, PINC		;διαβασμα εισοδου
	mov A, Input		;μεταφορα bit 0 εισόδου στο lsb του A

	mov B, Input
	lsr B			;μεταφορα bit 1 εισόδου στο lsb του B

	mov C, Input
	lsr C
	lsr C			;μεταφορα bit 2 εισόδου στο lsb του C

	mov D, Input
	lsr D
	lsr D
	lsr D			;μεταφορα bit 3 εισόδου στο lsb του D

	mov Acomp, A		;A'
	com Acomp

	mov Bcomp, B		;B'
	com Bcomp

	and Acomp, B		;A'B στον Acomp
	and Bcomp, C		;B'C στον Bcomp
	and Bcomp, D		;B'CD στον Bcomp
	or Acomp, Bcomp		;A'B + B'CD στον Acomp

	mov f0, Acomp		;μεταφορα αποτελεσματος στο LSB του f0
	com f0			;(A'B + B'CD)'
	andi f0, 0x01		;κραταμε μονο LSB του f0
```

```
        and A, C            ;AC στον A
        or B, D             ;B+D στον B
        and A, B            ;(AC)(B+D) στον A

        mov f1, A           ;μεταφορα αποτελεσματος στο LSB του f1
        andi f1, 0x01       ;κραταμε μονο LSB του f1
        lsl f1              ;μεταφερουμε αποτελεσμα στο bit 1 του f1

        or f1, f0           ;bit 0 το f0, bit 1 το f1
        out PORTB, f1       ;δινουμε στην εξοδο B τα αποτελεσματα
        jmp readinput
```

C:

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


char x, a, b, c, d, f0, f1;

int main(void)
{
        DDRB = 0xFF;  //αρχικοποίηση PORTB ως output
        DDRC = 0x00;  //αρχικοποίηση PORTC ως input
        while (1)
        {
                x = PINC & 0x0F;//απομόνωση 4 LSB της εισόδου
                a = x & 0x01;   //απομόνωση του bit0 της εισόδου
                b = x & 0x02;   //απομόνωση του bit1 της εισόδου
                b = b >> 1;     //φέρνουμε το bit στο LSB του B
                c = x & 0x04;   //απομόνωση του bit2 της εισόδου
                c = c >> 2;     //φέρνουμε το bit στο LSB του C
                d = x & 0x08;   //απομόνωση του bit3 της εισόδου
                d = d >> 3;     //φέρνουμε το bit στο LSB του D

                f0 = ~(((~a) & b) | ((~b) & c & d));
                f1 = (a & c) & (b | d);
                f1 = f1 << 1;   //φέρνουμε το f1 στο bit1

                PORTB = (f0 & 0x01) | (f1 & 0x02);  //εκτύπωση της εξόδου κρατώντας μονο 2
                                                     //LSB που μας ενδιαφέρουν
        }
        return 0;
}
```

Ζήτημα 2.2

```
.include "m16def.inc"

.DEF temp=r21
.DEF counter = r17


jmp start
.org 4
jmp interrupt1
reti

start: ldi r24, low(RAMEND)        ; Αρχικοποίηση στοίβας στο τέλος της RAM
out SPL, r24                       ; Είναι σημαντικό καθώς η στοίβα θα χρησιμοποιηθεί
ldi r24, high(RAMEND)              ; για το Return Address μετά από κάθε interrupt
out SPH, r24
clr counter
clr temp
out DDRA,temp                      ; PINA ως είσοδος
ser temp                           ; PORTC, PORTB ως έξοδοι
out DDRC,temp
out DDRB, temp

LDI temp,0x0C                      ; Η διακοπή θα γίνεται στη θετική ακμή του σήματος
OUT MCUCR,temp                     ; στην ακίδα INT1
ldi temp ,( 1 << INT1)             ; Ενεργοποίησε τη διακοπή INT1
out GICR , temp
SEI                                ; Ενεργοποιήση των διακοπών


clr r26
loop: out PORTC , r26
ldi r24 , low(100)
ldi r25 , high(100)
;rcall wait_msec
inc r26                            ; Αύξησε τον μετρητή
rjmp loop                          ; Επανέλαβε

interrupt1:
sbis PINA, 7                       ; Αν το MSB του PINA είναι on παρέλειψε την εντολή που
                                   ; παραλείπει την αύξηση του μετρητή
jmp go_back
sbis PINA, 6                       ; Αν το δεύτερο MSB του PINA είναι on παρέλειψε την
                                   ; εντολή που παραλείπει την αύξηση του μετρητή
jmp go_back
inc counter                        ; Αύξησε τον μετρητή των διακοπών και απεικόνισε τον
out PORTB, counter                 ; Εκτυπωσε εξοδο
go_back: sei                       ; Ενεργοποιήσε πάλι τις διακοπές
reti                               ; και επέστρεψε απο την διακοπή στον μετρητή
```

Ζήτημα 2.3

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

char input, counter, x;

ISR(INT0_vect){
	counter = 0x00; //Αρχικοποίηση μετρητή των switches που είναι on
	input = PINB;
	for(int i=0; i<8 ;i++){ // Μέτρηση των switches που είναι On
		x = input & 0x01; //Κάθε φορά έλεγχος του τελευταίου bit της εισόδου
		if (x == 0x01){
			counter++;
		}
		input = input >> 1; //και ολίσθηση της εισόδου δεξιά για έλεγχο επόμενου
bit
	}
	input = PINA & 0x04; //Απομόνωση του PA2
	if (input == 0x04){ //Αν είναι ON απεικόνισε τον αριθμό των switches που είναι On
στην έξοδο
		PORTC = counter;
	}
	else { //Αν είναι off άναψε τόσα led στην έξοδο όσα switches είναι on.
		x = 0x00;
		for(int i=0; i<counter; i++){ //φτιάχνεται βάζοντας έναν έναν τους άσσους
από δεξιά
			x = x << 1;
			x++;
		}
		PORTC = x;
	}
}

int main(void)
{
	DDRA = 0x00; //PINA ως είσοδος
	DDRB = 0x00; //PINB ως είσοδος
	DDRC = 0xFF; //PORTC ως έξοδος
	GICR = 0x40; //Ενεργοποιήση διακοπής INT0
	MCUCR = 0x03; //Η διακοπή θα γίνεται στη θετική ακμή του σήματος στην ακίδα INT0
	asm("sei"); //Ενεργοποίηση διακοπών
	while(1){
		asm("nop"); //Τη χρησιμοποιούμε για να μπορούμε να κάνουμε βηματική
εκτέλεση έχοντας infinite loop
	};
}
```