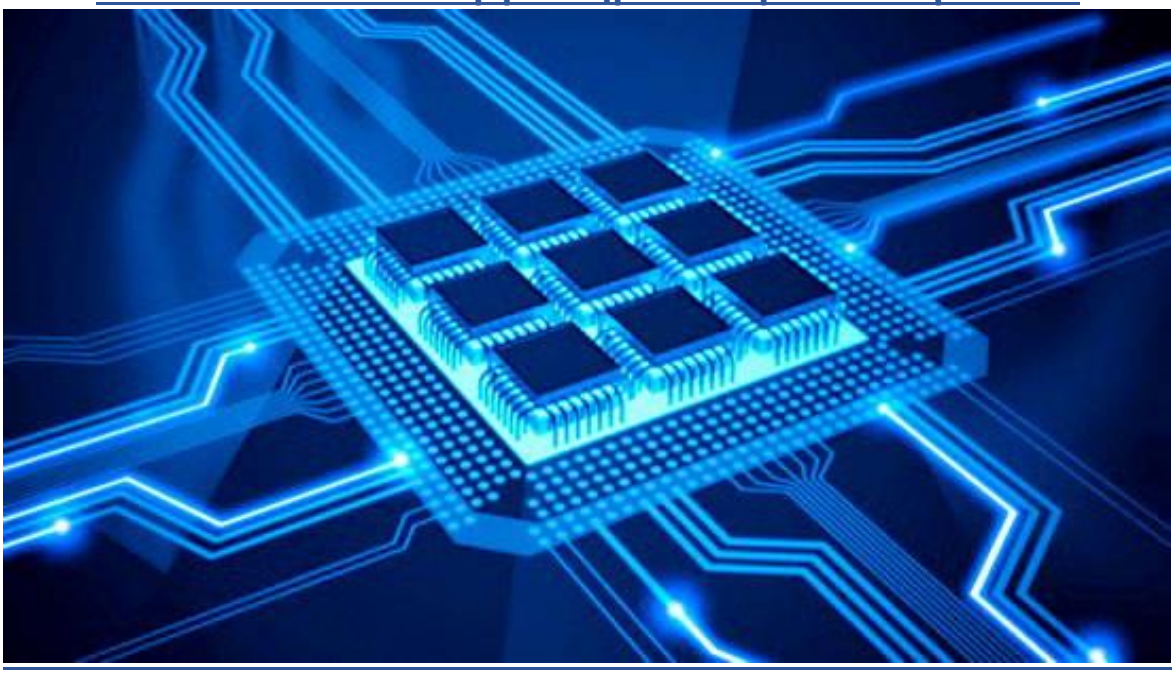
# 4η ΕΡΓΑΣΤΗΡΙΑΚΗ ΆΣΚΗΣΗ
# ΓΙΑ ΤΟ ΜΑΘΗΜΑ "Εργαστήριο Μικροϋπολογιστών"



**Ομάδα: 16**
**Ημερομηνία επίδειξης: 8/11/2017**
**Μέλη:**
**Κερασιώτης Ιωάννης, Α.Μ.:03114951**
**Πευκιανάκης Κωνσταντίνος, Α.Μ.: 03114897**
**Ραφτόπουλος Ευάγγελος, Α.Μ.:03114743**

## ΆΣΚΗΣΗ 1

Η άσκηση αυτή ζητάει να εμφανίζεται ένα αναμμένο led στις θύρες PB0-PB7 το οποίο να κινείται συνεχώς και όταν φτάνει στα άκρα να αλλάζει κατεύθυνση. Το κάθε led θα πρέπει να μένει αναμμένο για 0,5 sec. Η κίνηση ελέγχεται από το PA0.

Αρχικά αρχικοποιούμε την στοίβα αφού κάνουμε κλήση ρουτίνας Και ορίζουμε την είσοδο (A) και τη έξοδο (B). θεωρούμε ότι το led θα ξεκινήσει από την δεξιά θέση και θα κινηθεί προς τα αριστερά. Έχουμε σημαίες για την φορά της κίνησης των LEDs και για το αν είναι πατημένο ή όχι το κουμπί. Επίσης ελέγχουμε τις ακραίες περιπτώσεις γιατί αλλιώς θα εμφανιστεί σφάλμα. Δηλαδή όταν έχει φτάσει στο MSB τότε θα αλλάξει φορά προς τα δεξιά. Αντίστοιχα και για το LSB. Στην ρουτίνα χρονοκαθυστέρησης (η οποία δίνεται στο pdf της εκφώνησης) έχουμε 500 ως είσοδο αφού 1->1msec άρα 500->500ms.
Ο κώδικας αυτής της άσκησης δίνεται παρακάτω

```
.include "m16def.inc"      ;προσθήκη αρχείου κεφαλίδας που επιτρέπει την διαχείριση των PORTs μέσω των
                           ;συνολικών ετικετών τους
.def ledStatus=r16         ;καταχωρητής για το ενεργό bit
.def temp=r17              ;προσωρινός καταχωρητής
.def foraKinhshs=r18       ;σημαία που δηλώνει την φορά της κίνησης των LEDs (11..11->δεξιά ολίσθηση, 00…00->
                           ; αριστερή ολίσθηση
.def pushButtonStatus=r19  ;σημαία για το αν είναι πατημένο ή όχι το κουμπί ( 00..00->αφημένο, 11..11->πατημένο)
```

```
main:
        ldi temp,low(RAMEND)    ;αρχικοποίηση δείκτη στοίβας αφού γίνεται κάλεσμα ρουτίνας
        out SPL,temp
        ldi temp,high(RAMEND)
        out SPH,temp
        clr temp
        out DDRA,temp           ;η θύρα A ορίζεται ως είσοδος δηλαδή από εκεί θα διαβάζουμε τα δεδομένα
        out PORTA,temp
        ser temp
        out DDRB,temp           ;η θύρα B ορίζεται ως έξοδος δηλαδή από εκεί θα εμφανίζουμε τα δεδομένα εξόδου
        ldi ledStatus,0b00000001  ; τα LEDs είναι αρνητικής λογικής
        clr foraKinhshs           ;ορίζω αριστερή φορά κίνησης
        clr pushButtonStatus
        flash:
        rcall pushButtonState   ; κλήση υποπρογράμματος για τον έλεγχο του push button (LSB θύρας A)
        rcall flashLed          ; κλήση υποπρογράμματος για άναμμά των LEDs
        cpi pushButtonStatus,0x00
        breq flash
        rcall halfSecDelay      ; κλήση υποπρογράμματος για χρονοκαθυστέρηση 0,5 sec
        rcall enhmerwshLed      ; κλίση υποπρογράμματος για ενημέρωση των LEDs με την φορά περιστροφής και
                                ;τις ακραίες περιπτώσεις
        rjmp flash              ; το πρόγραμμα είναι συνεχούς λειτουργίας

pushButtonState:
        in temp,PINA            ; διάβασμα της θύρας A
        andi temp,0x01          ;θέλουμε να εξετάζεται μόνο το LSB
        cpi temp,0x01           ;έλεγχος για το αν πατήθηκε το PA0
        brne stop               ;αν όχι πήγαινε στο stop
        ldi pushButtonStatus,0xff  ;αλλιώς ενημέρωσε την σημαία για την κατάσταση του push button
        rjmp exitPush
stop:
        ldi pushButtonStatus,0x00  ;ενημέρωσε την σημαία για το push button ώστε να δηλώνει το άφημμα
exitPush:
        ret

flashLed:
        out PORTB,ledStatus
        ret

halfSecDelay:               ;ρουτίνα χρονοκαθυστέρησης
        ldi r24,low(500)
        ldi r25,high(500)
m_sec:
        push r24
        push r25
        ldi r24,low(998)
        ldi r25,high(998)
wait_usecBlock:
        sbiw r24,1
        nop
        nop
        nop
```

```
        nop
        brne wait_usecBlock
        pop r25
        pop r24
        sbiw r24,1
        brne m_sec
        ret

enhmerwshLed:
testMSB:
        cpi ledStatus,0b10000000    ;έλεγχος αν το αναμμένο led είναι το MSB
        brne testLSB
        ldi foraKinhshs,0xFF        ;αν ναι άλλαξε την φορά της κίνησης
        jmp shiftBlock
testLSB:
        cpi ledStatus,0b00000001    ;έλεγχος αν το αναμμένο led είναι το MSB
        brne shiftBlock
        ldi foraKinhshs,0x00        ;αν ναι άλλαξε την φορά της κίνησης
shiftBlock:
        cpi foraKinhshs,0           ;έλεγχος για το αν το led κινείται προς τα αριστερά
        brne prosDeksia             ;αν όχι τότε κάνε άλμα
prosAristera:
        lsl ledStatus               ;αριστερή ολίσθηση
        rjmp exitProc
prosDeksia:
        lsr ledStatus               ;δεξιά ολίσθηση
exitProc:
        ret
```

## ΆΣΚΗΣΗ 2

Η άσκηση αυτή ζητάει να εναλλάσσονται τα αναμμένα με τα σβηστά LEDs κατά χρόνο που υπολογίζεται από την σχέση (χ+1)*200msec, όπου το χ για το άναμμά δίνεται από τα dip switches PA0-PA3 και το χ για το σβήσιμο δίνεται από τα dip switches PA4-PA7. Γενικά ακολουθήσαμε τον κώδικα του πίνακα 4.1 και προσθέσαμε ένα κομμάτι κώδικα που υλοποιεί την πράξη (χ+1)*200 που το αποτέλεσμά της χρησιμοποιείται στην ρουτίνα χρονοκαθυστέρησης.

Ο κώδικας αυτής της άσκησης είναι ο παρακάτω:

```
.include "m16def.inc"     ;προσθήκη αρχείου κεφαλίδας που επιτρέπει την διαχείριση των PORTs μέσω των
                          ; συνολικών ετικετών τους
.def temp=r16
.def dipSwitchesValue=r17
ldi temp,low(RAMEND)   ;αρχικοποίηση δείκτη στοίβας αφού γίνεται κάλεσμα ρουτίνας
out SPL,temp
ldi temp,high(RAMEND)
out SPH,temp
clr temp
out DDRA,temp        ;η θύρα Α ορίζεται ως είσοδος δηλαδή από εκεί θα διαβάζουμε τα δεδομένα
out PORTA,temp
ser temp
```

```
out DDRB,temp        ;η θύρα Β ορίζεται ως έξοδος δηλαδή από εκεί θα εμφανίζουμε τα δεδομένα εξόδου
main:
        rcall on
        in dipSwitchesValue,PINA
        rcall dipSwitchesState
        rcall delay
        rcall off
        in dipSwitchesValue,PINA
        swap dipSwitchesValue
        rcall dipSwitchesState
        rcall delay
        rjmp main
on:
        clr temp
        out PORTB,temp
        ret
off:
        ser temp
        out PORTB,temp
        ret
dipSwitchesState:
        andi dipSwitchesValue,0x0F
        inc dipSwitchesValue                ;σχηματισμός του (χ+1)
        ldi temp,200
        mul dipSwitchesValue,temp           ;R1:R0<-(x+1)* 200
        mov r24,r0
        mov r25,r1
        ret
delay:          ;συνάρτηση που προκαλεί τόση χρονοκαθυστέρηση όσο είναι η τιμή του καταχωρητή r25:r24
        push r24
        push r25
        ldi r24,low(998)        ;998 γιατί τα δύο είναι τα δύο push
        ldi r25,high(998)
wait_usecBlock:
        sbiw r24,1
        nop
        nop
        nop
        nop
        brne wait_usecBlock
        pop r25
        pop r24
        sbiw r24,1
        brne delay
        ret
```

## ΆΣΚΗΣΗ 3

Για το άφημα και το πάτημα έχουμε ορίσει δύο σημαίες την state και την input που με τον κατάλληλο συνδυασμό τους προχωράει στις επιθυμητές λειτουργίες
Ο κώδικας της άσκησης 3 είναι ο παρακάτω:

```
#define DDRC (* (int *)0x34)
#define PORTC (* (int *)0x35)
#define PINC (* (int *)0x33)
#define DDRA (* (int *)0x3A)
#define PORTA (* (int *)0x3B)

int main(void)
{
	/*#ifdef NO_HEADER
	volatile int *DDRC = (int *)0x34;
	volatile int *PORTC = (int *)0x35;
	volatile int *PINC = (int *)0x33;
	volatile int *DDRA = (int *)0x3A;
	volatile int *PORTA = (int *)0x3B;
	#endif*/

	DDRC = 0x00;		// port c = input
	PORTC = 0x00;		// disable pull-up resistances

	DDRA = 0xFF;		// port a = output

	unsigned char input, output = 0x80, state = 0, state_tmp;

	while (1) {
		PORTA = output;

		input = (PINC & 0x1F) << 3;
		state = state | input;

		state_tmp = state;

		int operation;
		for (operation = 4; operation >= 0; operation--) {
			if ( (state_tmp & 0x80) && !(input & 0x80) ) break;
			state_tmp = state_tmp << 1;
			input = input << 1;
		}

		switch (operation) {
			case 0:
				output = (output >> 1) | (output << 7); //ROTATE μια θέση δεξιά
				state = state & 0xF7;
				break;
			case 1:
				output = (output << 1) | (output >> 7); // ROTATE μια θέση αριστερά
				state = state & 0xEF;
				break;
```

```
                        case 2:
                                output = (output >> 2) | (output << 6); //ROTATE δύο θέσεις δεξιά
                                state = state & 0xDF;
                                break;
                        case 3:
                                output = (output << 2) | (output >> 6); //ROTATE δύο θέσεις αριστερά
                                state = state & 0xBF;
                                break;
                        case 4:
                                output = 0x80;
                                state = state & 0x7F;
                                break;
                }

        }
        return 0;
}
```