Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχ. Και Μηχ. Υπολογιστών
Εργαστήριο Μικροϋπολογιστών , 7ο εξάμηνο - Ροή Υ
Ακαδημαϊκή Περίοδος : 2011-2012

5η ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ

Ομάδα: C16
Γερακάρης Βασίλης Α.Μ.: 03108092
Διβόλης Αλέξανδρος Α.Μ.: 03107238
Ίεντσεκ Στέργιος Α.Μ.: 03108690

Άσκηση 1.1

Σε αυτή την άσκηση υλοποιήσαμε ένα πρόγραμμα, σε assembly του AVR, το οποίο ανάβει τα οκτώ LEDs της PORTA, ένα-ένα από πάνω προς τα κάτω και κάθε φορά που είναι πατημένο το push button PB0 αναστέλλεται η μετακίνηση του αναμμένου LED. Παρατίθεται το αρχείο **wait.asm** το οποίο υλοποιεί τις καθυστερήσεις και ο κώδικας του κυρίως προγράμματος:

wait.asm:

```
wait_usec:
       sbiw r24 ,1          ; 2 κύκλοι (0.250 μsec)
       nop                  ; 1 κύκλος (0.125 μsec)
       nop                  ; 1 κύκλος (0.125 μsec)
       nop                  ; 1 κύκλος (0.125 μsec)
       nop                  ; 1 κύκλος (0.125 μsec)
       brne wait_usec       ; 1 ή 2 κύκλοι (0.125 ή 0.250 μsec)
       ret                  ; 4 κύκλοι (0.500 μsec)

wait_msec:
       push r24             ; 2 κύκλοι (0.250 μsec)
       push r25             ; 2 κύκλοι
       ldi r24 , 0xe6       ; φόρτωσε τον καταχ. r25:r24 με 998 (1 κύκλος - 0.125 μsec)
       ldi r25 , 0x03       ; 1 κύκλος (0.125 μsec)
       rcall wait_usec      ; 3 κύκλοι (0.375 μsec), προκαλεί συνολικά καθυστέρηση 998.375 μsec
       pop r25              ; 2 κύκλοι (0.250 μsec)
       pop r24              ; 2 κύκλοι
       sbiw r24 , 1         ; 2 κύκλοι
       brne wait_msec       ; 1 ή 2 κύκλοι (0.125 ή 0.250 μsec)
       ret                  ; 4 κύκλοι (0.500 μsec)
```

main.asm:

```
.include "m16def.inc"
.def temp=r16
.cseg
.org 0

	ldi temp,HIGH(RAMEND)
	out SPH,temp
	ldi temp,LOW(RAMEND)
	out SPL,temp
reset:
	ser r26
	out DDRA,r26		;Set PORTA as output
	clr r26
	out DDRB,r26		;Set PORTB as input
	ser r26
	out PORTB,r26		;Activate pull-up resistors
	ldi r26,1		;r26 will control the LEDs

down:
	out PORTA,r26		;Turn on the LED
	ldi r24,0xf4		;Set 0x01f4 = 0d500 to r25:r24
	ldi r25,0x01
	rcall wait_msec		;wait for 500 sec
	jmp button_down		;Check if button pressed
bpd:
	rol r26			;If not set the next LED
	brcc down		;Loop until the 8th LED
	ror r26			;Set the 7th LED
	ror r26
up:
	out PORTA, r26		;Do the same job backwards
	ldi r24,0xf4		;Set 0x01f4 = 0d500 to r25:r24
	ldi r25,0x01
	rcall wait_msec
	jmp button_up
bpu:
	ror r26
	brcc up
	rol r26
	rol r26
	jmp down

button_up:			;Button check when you go down
	in  r23,PINB		;Read input from PORTB
	bst r23, 0		;If nothing pressed exit
	brtc exit_up
pressed_up:
	in  r23, PINB		;Else wait until button released
	bst r23,0
	brts pressed_up
exit_up:
	jmp bpu

button_down:			;Button check when you go up
	in  r23, PINB
	bst r23, 0
	brtc exit_down
pressed_down:
	in  r23, PINB
	bst r23,0
	brts pressed_down
exit_down:
	jmp bpd

.include "wait.asm"
```

Άσκηση 1.2

Σε αυτή την άσκηση τροποποιήσαμε το πρόγραμμα του Πίνακα 1.1, σε assembly του AVR, ώστε να ελέγχουμε μέσω των διακοπτών της PORTB το χρονικό διάστημα για το οποίο θα παραμένουν αναμμένα ή σβηστά τα LEDs. Πιο συγκεκριμένα τα 4 LSbits καθορίζουν το άναμμα και τα 4 MSbits το σβήσιμο, η καθυστέρηση δίνεται από τον τύπο $D = x + 1 * 100\ msec,\ x \in [0,15]$. Παρακάτω παρατίθεται ο ζητούμενος κώδικας:

```
.include "m16def.inc"
.def temp=r16
.cseg
.org 0

	ldi temp,HIGH(RAMEND)
	out SPH,temp
	ldi temp,LOW(RAMEND)
	out SPL,temp

	ser r26			; Αρχικοποίηση της PORTA
	out DDRA, r26		; για έξοδο
	clr r26
	out DDRB, r26		; Αρχικοποίηση της PORTB
	ser r26			; για είσοδο
	out PORTB, r26
flash:
	in r23, PINB		; Διάβασε την είσοδο
	mov r22, r23
	rcall on		; Άναψε τα LEDs
	andi r23, 0x0f		; Κράτα τα 4 LSB (leds on)
	inc r23			; Αύξησε τον αριθμό κατά 1
wait_on:
	ldi r24, 0x64		; r25:r24 = 100 ms delay
	ldi r25, 0x00
	rcall wait_msec		; Επανάλαβε x+1 καθυστερήσεις των 100 ms
	dec r23
	brne wait_on

	rcall off		; Σβήσε τα LEDs
	swap r22		; Κράτα τα 4 MSB (leds off)
	andi r22, 0x0f
	inc r22			; Αύξησε τον αριθμό κατά 1 (x+1)
wait_off:
	ldi r24, 0x64		; r25:r24 = 100 ms delay
	ldi r25, 0x00
	rcall wait_msec		; Επανάλαβε x+1 καθυστερήσεις των 100 ms
	dec r22
	brne wait_off
	rjmp flash		; Επανάλαβε

;Υπορουτίνα για να ανάβουν τα LEDs
on:
	ser r26			; Θέσε τη θύρα εξόδου των LED
	out PORTA , r26
	ret			; Γύρισε στο κύριο πρόγραμμα

;Υπορουτίνα για να σβήνουν τα LEDs
off:
	clr r26			; Μηδένισε τη θύρα εξόδου των LED
	out PORTA , r26
	ret			; Γύρισε στο κύριο πρόγραμμα

.include "wait.asm"
```

Άσκηση 1.3

Σε αυτή την άσκηση υλοποιήσαμε ένα πρόγραμμα σε C το οποίο ελέγχει τα LEDs της PORTB με βάση τους διακόπτες της PORTD. Αρχικά είναι αναμμένο το 1^{ο} LED, οι διακόπτες SW1-SW5 έχουν την εξής λειτουργία:

- SW1 μετακίνηση του led μια θέση αριστερά (κυκλικά).
- SW2 μετακίνηση του led μια θέση δεξιά (κυκλικά).
- SW3 μετακίνηση του led δυο (2) θέσεις αριστερά (κυκλικά).
- SW4 μετακίνηση του led δυο (2) θέσεις δεξιά (κυκλικά).
- SW5 μετακίνηση του αναμένου led στην αρχική του θέση (LSB - led0).

Παρακάτω παρατίθεται ο κώδικας μας:

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

int input_check(void);
unsigned char SW5(unsigned char input);
unsigned char SW4(unsigned char input);
unsigned char SW3(unsigned char input);
unsigned char SW2(unsigned char input);
unsigned char SW1(unsigned char input);

int main(void)
{
    unsigned char state = 0x01;                  //Set initial state to 1
    int input;

    DDRB = 0xff;                                 //Set PORTB as output
    DDRD = 0x00;                                 //Set PORTD as input
    PORTD = 0xff;
    PORTB = state;                               //Set LEDs to initial state

    for(;;) {
        while ((input = input_check()) == 0){}   //Wait for button press
        if (input == 5) {                        //According to the input
                state = SW5(state);              //do the corresponding job
                PORTB = state;
        } else if (input == 4) {
                state = SW4(state);
                PORTB = state;
        } else if (input == 3) {
                state = SW3(state);
                PORTB = state;
        } else if (input == 2) {
                state = SW2(state);
                PORTB = state;
        } else {
                state = SW1(state);
                PORTB = state;
        }
    }
    return -1;
```

```
int input_check(void)
{
    int end = 0;
    int ret = 0;
    unsigned char input;

    input = PIND;                                   //Read the input from PORTD
    input = input & 0x1f;                           //Keep the 5 first bits
    if (input == 0)                                 //Check every possible button
        return 0;
    for (;;) {
    /*
    * The ret value checking in the clauses
    * guarantees that only the pressed button of
    * maximum value will take effect
    */
        if (input >= 16)
            ret = 5;
        else if ((input >= 8) && (ret <= 4))
            ret = 4;
        else if ((input >= 4) && (ret <= 3))
            ret = 3;
        else if ((input >= 2) && (ret <= 2))
            ret = 2;
        else if ((input >= 1) && (ret <= 1))
            ret = 1;
        else if (input == 0)                        //The button was released
            end = 1;
        if (end == 1)
            return ret;
        input = PIND;
        input = input & 0x1f;
    }
    return -1;
}

unsigned char SW5(unsigned char input)          //SW5 sets the 1st LED
{
    return 0x01;
}

unsigned char SW4(unsigned char input)          //SW4 sets the LED two places right
{
    if (input > 2)
        return input / 4;
    else
        return input * 64;
}

unsigned char SW3(unsigned char input)          //SW3 sets the LED two places left
{
    return (input * 4) % 255;
}

unsigned char SW2(unsigned char input)          //SW2 sets the LED one place right
{
    if (input > 1)
        return input / 2;
    else
        return 128;
}

unsigned char SW1(unsigned char input)          //SW2 sets the LED one place left
{
    return (input * 2) % 255;
}
```