ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Συστήματα Μικροϋπολογιστών

4η Σειρά Ασκήσεων:

Διδάσκοντες:

Δ. Σούντρης

Ομάδα Β4:

Ειρήνη Δόντη

Α.Μ.: 03119839

Αθήνα 2022

Περιεχόμενα

Ζήτημα 4.1

Προγραμματίζουμε σε assembly και επιδεικνύουμε στο Atmel Studio πρόγραμμα που να απεικονίζει ένα αναμμένο led που αντιστοιχεί στα bit της θύρας PB0-PB7. Το αναμμένο led κινείται συνεχώς, ξεκινώντας από τα LSB προς τα MSB και αντίστροφα όταν φτάνει στο άλλο άκρο. Κάθε φορά το led μένει αναμμένο για ~0.5 sec και το επόμενο led ανάβει αμέσως. Η κίνηση του αναμμένου led ελέγχεται από το push button PA0. Όταν αυτό είναι πατημένο η κίνηση σταματάει, ενώ διαφορετικά συνεχίζεται.
Παρακάτω, παραθέτουμε τον ζητούμενο κώδικα assembly:

```
.include "m16def.inc"

reset:
    ldi r24 , low(RAMEND) ; initialize stack pointer
    out SPL , r24
    ldi r24 , high(RAMEND)
    out SPH , r24
    clr r22

main:
    clr r21         ;CLC TEMP
    out PORTA , r21 ;PORTA FOR INPUT
    ser r21         ;SET TEMP
    out PORTB,r21   ;PORTB FOR OUTPUT
    ldi r22, 0x01   ;LIGHT BIT
    out PORTB, r22  ;FLASH LSB
    ldi r20,7       ;r20 COUNTER VALUE 7

left:
    in r23, PINA        ;READ INPUT A
    subi r23,1
    breq left           ;NOT EXIT CONTINUE
    ldi r24 , low(500)  ;load r25:r24 with 500
    ldi r25 , high(500) ; delay 0.5 second
    rcall wait_msec
    clc
    rol r22
    out PORTB, r22   ;FLASH NEXT LED TO THE LEFT
    dec r20          ; counter = counter-1
    cpi r20, 0       ; IF counter == 0 => MSB => RIGHT ROTATION
    breq right
    rjmp left
```

```
right:
    in r23,PINA      ;READ INPUT A
    subi r23,1
    breq right       ;NOT EXIT CONTINUE
    ldi r24,low(500)    ;load r25:r24 with 500
    ldi r25,high(500)   ; delay 0.5 second
    rcall wait_msec
    clc
    ror r22
    out PORTB,r22  ;FLASH NEXT LED TO THE RIGHT
    inc r20        ; counter = counter + 1
    cpi r20,7      ; if counter = 7  => LSB => LEFT ROTATION
    breq left
    rjmp right

wait_msec: ;1 msec
    push r24          ;2 cycles (0.250usec)
    push r25          ;2 cycles (0.250usec)
    ldi r24,low(998)  ;1 cycle  (0.125 usec) - load r25:r24 with 998
    ldi r25,high(998) ;1 cycle  (0.125 usec)
    rcall wait_usec   ;3 cycles (0.375 usec) - delay 998.375 usec
    pop r25           ;2 cycles (0.250usec)
    pop r24           ;2 cycles (0.250usec)
    sbiw r24,1        ;2 cycles (0.250usec)
    brne wait_msec    ;1 or 2 cycles (0.125 or 0.250 usec)
    ret               ;4 cycles (0.500usec)

wait_usec:
    sbiw r24,1        ;2 cycles (0.250usec)
    nop               ;1 cycle  (0.125 usec)
    nop               ;1 cycle  (0.125 usec)
    nop               ;1 cycle  (0.125 usec)
    nop               ;1 cycle  (0.125 usec)
    brne wait_usec    ;1 or 2 cycles (0.125 or 0.250) usec
    ret               ;4 cycles (0.500 usec)
end:
```

Ζήτημα 4.2

Υλοποιούμε σε ένα σύστημα μικροελεγκτή AVR τις λογικές συναρτήσεις F0=ABC+B'C'D' και F1=(A+B+C)D.

Οι τιμές των F0-F1 εμφανίζονται, αντίστοιχα, στα δύο LSB της θύρας εισόδου PORTA (0-1).

Οι μεταβλητές εισόδου (A,B,C,D) υποθέτουμε ότι αντιστοιχούν στα 4 bit της θύρας εισόδου PORTB (0-3), με το A στο LSB.

Παρακάτω, παραθέτουμε το ζητούμενο κώδικα assembly:

```
.include "m16def.inc"
.DEF temp = r16
.DEF A = r17
.DEF B = r18
.DEF C = r19
.DEF D = r20
.DEF F0= r21
.DEF F1= r22
.DEF E = r15

reset:
ldi r24,low(RAMEND) ;initialize stack pointer
out SPL,r24
ldi r24,high(RAMEND) ;load 8-bit value in r24
out SPH,r24

start:
clr temp
out DDRB,temp ;set port B as input
ser temp ;put ones in every bit of the register
out DDRA,temp ;set port A as output


main:
clr F0 ;clear F0 - initialize with zeros
in temp,PINB ;put data from the input post B in temp
mov A,temp
andi A,0x01 ;isolate LSB using mask
lsr temp ;shift the value right
mov B,temp
andi B,0x01 ;isolate LSB using mask
lsr temp ;shift the value right
mov C,temp
andi C,0x01 ;isolate LSB using mask
lsr temp ;shift the value right
mov D,temp
```

```
andi D,0x01 ;isolate LSB

mov F0,A ;store register A in F0
and F0,B ;F0 = AB
and F0,C ;F0 = ABC
mov E,B
or E,C ;B or C
or E,D ;B or C or D
com E ;(B or C or D)' - De Morgan
or F0,E ;F0 = (ABC) + (B+C+D)'

mov F1,A ;store register A in F1
or F1,B ;A or B
or F1,C ;A or B or C
and F1,D ;F1 = (A+B+C)D
lsl F1 ;shift the value left in order to take the 2nd LSB for F1

andi F0, 0x01 ;isolate LSB
andi F1, 0x02 ;isolate 2nd LSB
or F0,F1
out PORTA,F0 ;show the exit F0 | F1
rjmp main
```

Παρακάτω, παραθέτουμε το ζητούμενο κώδικα σε C:

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

int main(void)
{
    char x, f0, f1, a, b, c, d;
    DDRA=0xFF; //initialize PORTA as output
    DDRB=0x00; //initialize PORTB as input
    while(1)
    {
        x = PINB; //contains all bits of port B in the input (0000 xxxx)
        a = x & 1; // isolate LSB
        b = (x >> 1) & 1; // shift the value right by one bit and isolate the new LSB
        c = (x >> 2) & 1; // shift the value right and isolate
        d = (x >> 3) & 1; // shift the valye right and isolate
        f1 = ((a || b || c) && d) << 1;// take F1 to 2nd most LSB
        f0 = ((a && b && c) || ((!b) && (!c) && (!d))); // F0

        f0 = f0 && 1; //ignore all bits except bit 1
        f1 = f1 & 2; //ignore all bits except bit 2
        PORTA = f0 | f1; // put the result of f0 and f1 in the output
    }
}
```

Ζήτημα 4.3

Γράφουμε πρόγραμμα σε C για τον προγραμματισμό AVR Atmega16 και εκτελούμε την προσομοίωση στο Atmel Studio 7 στο οποίο αρχικά ανάβει το led0 που είναι που είναι συνδεδεμένο στο bit0 της θύρας εξόδου PORTB. Επίσης, με το πάτημα των διακοπτών (Push-buttons) SW0-3 που υποθέτουμε ότι είναι συνδεδεμένα στα αντίστοιχα bit της θύρας εισόδου PORTA συμβαίνουν τα εξής:

- SW0 ολίσθηση-περιστροφή του led κατά μία θέση αριστερά (κυκλικά από LSB προς το MSB).
- SW1 ολίσθηση-περιστροφή του led κατά μία θέση δεξιά (κυκλικά από MSB προς το LSB).
- SW2 μετακίνηση του αναμμένου led κατά δύο θέσεις αριστερά (κυκλικά από LSB προς το MSB).
- SW3 μετακίνηση του αναμμένου led στη θέση MSB (led7).

Υποθέτουμε ότι η αρχική θέση του αναμμένου led είναι LSB (led0) και τα push-buttons δεν πατιούνται ταυτόχρονα (όλες οι αλλαγές γίνονται αφήνοντας τα push-buttons SWx (bitx PORTA)).

Παρακάτω, παραθέτουμε το ζητούμενο κώδικα σε C:

```c
#include <avr/io.h>
char x=1; //variable for storing output
int main(void)
{
    DDRB=0xFF; // Initializing PORTB as output
    DDRA=0x00; //Initializing PORTA as INPUT

    PORTB=x; //first output(led0=on)

    while(1){ //non-terminating program

        //if SW0 was pushed--left shift by 1
        if ((PINA & 0x01)==1){   //check if  push-button was pushed
            while((PINA & 0x01)==1);  //check if push-button was unpushed
            if (x!=128)
            x=x<<1;       //left shift
        else x=1;} //if x is 128(10000000),to prevent overflow we return to 1(00000001)


        //if SW1 was pushed--right shift by 1
        else if((PINA & 0x02)==2){     //check if  push-button was pushed
            while ((PINA & 0x02)==2); //check if  push-button was unpushed
            if (x!=1)
            x=x>>1; //right shift
        else x=128;}  //if x is 1(00000001),we return to 128(10000000)


        //if SW2 was pushed--left shift by 2
        else if ((PINA & 0x04)==4){ //check if  push-button was pushed
            while((PINA & 0x04)==4); //check if  push-button was unpushed
            if (x==64)  //to prevent overflow if x=64(01000000) then x is set equal to 1(00000001)
            x=1;

           else if (x==128) //otherwise if x=128(10000000) then x is set equal to 2(00000010)
           x=2;
        else x=x<<2;} //Finally,if none of the above conditions is true,theres no overflow hazard,we shift normally

        //if SW3 was pushed--set msb led on
        else if ((PINA & 0x08)==8){  //check if  push-button was pushed
            while( (PINA & 0x08)==8); //check if  push-button was unpushed
        x=128;}  //x=(10000000)

        PORTB=x; //set the output at leds

    }
}
```