Συστήματα Μικρουπολογιστών – Εργαστηριακή Αναφορά

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Ακαδημαϊκό έτος : 2018 – 2019 Εξάμηνο : 6ό
Βόσινας Κωνσταντίνος ΑΜ: 03116435
Μέλη ομάδας : Ανδριόπουλος Κωνσταντίνος
Πεολίδης Αχιλλέας

Άσκηση 1η

```
.include "m16def.inc"
reset: ldi r24 , low(RAMEND) ; initialize stack pointer
out SPL , r24
ldi r24 , high(RAMEND)
out SPH , r24
ser r24 ; initialize PORTA for output
out DDRB , r24
clr r27
out DDRA, r27
ser r26
andi r26,1  ;r26 = 00000001

left:
input1: in r27, PINA    ;Check if A0 is pressed
    ror r27
    brcs input1         ;If not, repeat until it's pressed
out PORTB , r26         ;Turn on current led
ldi r24 , low(500) ; load r25:r24 with 500
ldi r25 , high(500) ; delay 1 second
rcall wait_msec
lsl r26         ; shift left once
cpi r26 , 128   ;compare with 128, B7 to be turned on
brlo left   ;if lower continue with left
rjmp right  ;else go to right

right:
input2: in r27, PINA    ;Check if A0 is pressed
    ror r27
    brcs input2
out PORTB , r26
ldi r24 , low(500) ; load r25:r24 with 500
ldi r25 , high(500) ; delay 1 second
rcall wait_msec
lsr r26     ; shift right once
cpi r26 , 1 ; compare with 1 similarily as before
brne right ; if not equal continue with right
rjmp left   ;else go the other direction


wait_msec:
push r24 ; 2 κύκλοι (0.250 μsec)
push r25 ; 2 κύκλοι
ldi r24 , low(998) ; φόρτωσε τον καταχ. r25:r24 με 998 (1 κύκλος - 0.125 μsec)
ldi r25 , high(998) ; 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)
```

```
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)
```

Άσκηση 2η – AVR

```
.include "m16def.inc"
reset: ldi r24 , low(RAMEND)    ; initialize stack pointer
out SPL , r24
ldi r24 , high(RAMEND)
out SPH , r24
ser r24             ; initialize PORTB for output
out DDRB , r24
clr r27
out DDRA, r27

main:
clr r27
in r26,PINA
mov r20, r26
andi r20,1          ;isolating bit A in r20

lsr r26             ; logigal shift right is used for the next bit
mov r21, r26
andi r21,1          ; isolating bit B in r21

lsr r26
mov r22, r26
andi r22,1          ; isolating bit C in r22

lsr r26
mov r23, r26
andi r23,1          ; isolating bit D in r23

calculate_F1:

mov r24,r20
or r24,r21          ; (A + B)

mov r25,r22
or r25,r23          ; (C + D)

and r24,r25             ; (A+B)(C+D)

mov r27,r24
lsl r27             ; F1 corresponds to PORTB(1)

calculate_F0:

mov r24,r20
and r24,r21
and r24,r22             ; 'ABC'

mov r25, r22
```

```
com r25              ; computing C'
andi r25,1           ; isolating the lsb
and r25,r23              ; (C'D)

or r24,r25
com r24              ; (ABC + C'D)'
andi r24,1           ; isolating the lsb
add r27,r24

out PORTB,r27

rjmp main
```

Άσκηση 2η – C

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

char input,A,B,C,D,fout,f0,f1;

int main(void){

    DDRB = 0xFF; //output PORTB (0-1)
    DDRA = 0x00; //input PORTA (0-3)

    while(1){

        input = PINA & 0x0F;
        A = input & 0x01; //isolating each one of
        B = input & 0x02; //the A,B,C,D bits
        B = B>>1;
        C = input & 0x04;
        C = C>>2;
        D = input & 0x08;
        D = B>>3;

        f1 = (A | B) & (C | D); //implementing f1

        f1 = f1<<1; //f1 corresponds to PORTB(1)
        f1 = f1 & 0x02; //isolating the bit needed

        f0 = ~(((A & B) & C) | ((~C)&D)); //implementing f2
        f0 = f0 & 0x01; //isolating the bit needed
        fout = f0+f1;

        PORTB = fout; //f1 in PORTB(1) and f0 in PORTB(0)

    }
    return 0;

}
```

Άσκηση 3η

```
#include <avr/io.h>
char x;

int main(void)
{
    DDRA=0xFF; //output
    DDRC=0x00; //input


    x = 1; // initialization for first LED


    while(1){

        if ((PINC & 0x01) == 1){ // push-button SW0 check
            while ((PINC & 0x01) == 1);
            if (x==128) // overflow check
                x = 1;
            else
                x = x<<1; // left shift logical
        }

        if ((PINC & 0x02) == 2){ // push-button SW1 check
            while ((PINC & 0x02) == 2);
            if (x==1) // overflow check
                x = 128;
            else
                x = x>>1; // right shift logical
        }

        if ((PINC & 0x04) == 4){ // push-button SW3 check
            while ((PINC & 0x04) == 4);
            x=128;

        }

        if ((PINC & 0x08) == 8){ // push-button SW4 check
            while ((PINC & 0x08) == 8);
            x=1;

        }


        PORTA = x; // Έξοδος σε PORTA
    }
    return 0;
}
```