ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΗΜΜΥ

ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ

Ακαδημαϊκό έτος 2019-2020

Εργαστηριακή Άσκηση

Ομάδα 75
Κυριάκου Δημήτρης – 03117601
Πεγειώτη Νάταλυ – 03117707

Ζήτημα 2.1

Για τη ρύθμιση της κατεύθυνσης του led χρησιμοποιείται η μεταβλητή direction. Όταν έχει τιμή 0 τότε το led κινείται προς τα αριστερά (εκτελείται το τμήμα leftshift) ενώ όταν έχει τιμή 1, το led κινείται προς τα δεξιά (εκτελείται το τμήμα righthift). Τα τμήματα leftshift και rightshift είναι επίσης υπεύθυνα να αλλάξουν την τιμή της μεταβλητής direction όταν εντοπίσουν ότι το led έχει φτάσει στο εκάστοτε άκρο. Σε περίπτωση που το push button PC2 (bit 2 του input) είναι πατημένο τότε το πρόγραμμα εκτελεί ατέρμονα βρόχο (η κίνηση του led σταματά), μέχρι να αποσυμπιεστεί ξανά το PC2. Το πρόγραμμα είναι συνεχούς λειτουργίας.

```
.include "m16def.inc"
.DEF input=r24
.DEF output=r25
.DEF direction=r22

reset:      ldi r24 , low(RAMEND)    ;initialize stack pointer

            out SPL , r24
            ldi r24 , high(RAMEND)
            out SPH , r24

            clr input                ;initialize PORTC for input
            out DDRC, input
            ser output               ;initialize PORTB for output
            out DDRB, output

            ldi output,1
            ldi direction,0          ;controls the direction of the led
                                     (0->left, 1->right)

main:       in input,PINC
            sbrc input,2             ;if input's bit 2 is on then jump to main
            jmp main
            out PORTB,output         ;turn on the led
            sbrs direction,0         ;if lbs of direction is 0  then call
                                     leftshift else call rightshift
            jmp leftshift
            jmp rightshift

leftshift:
            lsl output               ;left rotation
            sbrc output,7            ;if msb of output is 1 then change
                                     direction
            ldi direction,1
            jmp main

rightshift:
            lsr output               ;right rotation
            sbrc output,0            ;if lsb of output is 1 then change
                                     direction
            ldi direction,0
            jmp main
```

Ζήτημα 2.2

Αρχικά το πρόγραμμα διαβάζει την είσοδο από τα 4 lsb της θύρας εισόδου PORTB και την αποθηκεύει στις μεταβλητές A,B,C,D. Για κάθε μεταβλητή η τιμή του bit ολισθαίνει δεξιά μέχρι να φτάσει στη θέση lsb. Έτσι το περιεχόμενο των μεταβλητών είναι 0 ή 1 (true η false) και το πρόγραμμα δύναται να τις χρησιμοποιήσει για τις λογικές πράξεις υπολογισμού των F0 και F1. Όταν υπολογιστούν, το F1 ολισθαίνει κατά μία θέση αριστερά και εκτελείται αριθμητική bitwise πράξη με το F0 ώστε να τοποθετηθούν και τα δύο παράλληλα στις δύο lsb θέσεις της εξόδου PORTA.

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


int main(void)
{
      char x,A,B,C,D,F0,F1;
      DDRA=0xFF; // use PORTA as output
      DDRB=0x00; // use PORTB as input

    /* Replace with your application code */
    while (1)
    {
            x = PINB & 0x0F;  //isolation of the 4 LSBs
            A = x & 0x01;           //A is PB0
            B = x & 0x02;           //B is PB1
            B = B>>1;
            C = x & 0x04;           //C is PB2
            C = C>>2;
            D = x & 0x08;           //D is PB3
            D = D>>3;

            F0 = !( (A && !B ) || (B && !C && D));    //calculation of F0
            F1 = ((A || C) && (B || D));              //calculation of F1
            F1 = F1<<1;

            PORTA = (F0 | F1);      //output
    }
}
```

Ζήτημα 2.3

Το πρόγραμμα αποτελείται από έναν ατέρμονα βρόχο (φροντίζει για τη συνεχή λειτουργία) που ελέγχει διαδοχικά αν έχει πατηθεί κάποιο από τα push buttons. Στην περίπτωση που έχει πατηθεί κάποιο τότε μπαίνει σε ένα υποβρόχο που αναστέλλει τη λειτουργία του προγράμματος μέχρι να ξεπατηθεί. Μόλις ξεπατηθεί εκτελεί την ανάλογη διεργασία.

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


int main(void)
{
      DDRA=0x00; // use PORTA as input
      DDRB=0xFF; // use PORTB as output
      char current_state=0x01;
      PORTB = current_state;                    //initial state is PB0 on

    while (1)
    {
            if((PINA & 0x01) == 1){             //check if SW0 is pressed
                  while((PINA & 0x01) == 1);    //wait until it is released
                  if(current_state==0x01)       //if PB0 is on the turn on
PB7
                        current_state=0x80;
                  else
                        current_state=current_state >> 1; //else shift right
                  PORTB = current_state;
            }
            if((PINA & 0x02) == 2){             //check if SW1 is pressed
                  while((PINA & 0x02) == 2);    //wait until it is released
                  if(current_state==0x80)       //if PB7 is on the turn on
PB0
                        current_state=0x01;
                  else
                        current_state=current_state << 1; //else shift left
                  PORTB = current_state;
            }
            if((PINA & 0x04) == 4){             //check if SW2 is pressed
                  while((PINA & 0x04) == 4);    //wait until it is released
                  current_state=0x01;           //turn on PB0
                  PORTB = current_state;
            }
            if((PINA & 0x08) == 8){             //check if SW3 is pressed
                  while((PINA & 0x08) == 8);    //wait until it is released
                  current_state=0x80;           //turn on PB7
                  PORTB = current_state;
            }
    }
}
```