Εργαστήριο Μικροϋπολογιστών

2η Εργαστηριακή Αναφορά(AVR)

Αμπατζή Ναυσικά (031 17 198)
Δήμος Δημήτρης (031 17 165)

7ο Εξάμηνο - Ροή Υ
Φθινόπωρο 2020

1 Λογικές Συναρτήσεις

Το ζητούμενο πρόγραμμα δέχεται ως είσοδο τις μεταβλητές A,B,C,D από τα bit 0-3 της θύρας εισόδου PORTC και υλοποιεί τις λογικές συναρτήσεις F0 = (A'B + B'CD)' και F1 = (AC)(B+D). Το αποτέλεσμα των F0 και F1 εμφανίζεται στη θύρα εξόδου PORTB(bits 0 και 1 αντίστοιχα). Το πρόγραμμα δίνεται σε assembly και σε C.

ASSEMBLY:

```
.include "m16def.inc"
.DEF A = r16
.DEF B = r17
.DEF C = r18
.DEF D = r19
.DEF F = r20
.DEF T = r21
.DEF b = r22

stack: ldi r24 , low(RAMEND)
       out SPL , r24
       ldi r24 , high(RAMEND)
       out SPH , r24
```

```
IO_set: ser r24              ;initialize PORTB
            out DDRB, r24    ;for output
            clr r24          ;initialize PORTC
                out DDRC, r24  ;for input

main: clr F           ;ready F
          in T, PINC  ;T <-- input
          mov A,T     ;LSB(A) = A
          lsr T
          mov B,T     ;LSB(B) = B
          lsr T
          mov C,T     ;LSB(C) = C
          lsr T
          mov D,T     ;LSB(D) = D
          mov F,A     ;save A in F
          com F       ;LSB(F) = A'
          and F,B     ;LSB(F) = A'B
          mov b,B     ;save B in b
          com B       ;LSB(B) = B'
          and B,C     ;LSB(B) = B'C
          and B,D     ;LSB(B) = B'CD
          or F,B      ;LSB(F) = (A'B + B'CD)
          com F       ;LSB(F) = (A'B + B'CD)' <---

          mov B,b     ;restore B
          and A,C     ;LSB(A) = AC
          or B,D      ;LSB(B) = B + D
          and A,B     ;LSB(A) = (AC)(B + D) <---
          lsl A       ;A1 = (AC)(B + D)
          andi A,2    ;A = F1
          andi F,1    ;F = F0A ; F = F + A = OUTPUT
          add F,A     ;F = F + A = OUTPUT
          out PORTB,F
          rjmp main
```

C:

```
#include <avr/io.h>
unsigned char x,F0,A,B,C,D,F1;

int main(void)
{
                DDRC = 0xff;     // PORTC->output
                DDRB = 0x00;     // PORTB ->input

    while (1)
    {
                x = PINB;        //save input in x
                A = x & 0x01;    //LSB(PINB) = A
                B = x & 0x02;
                B = B>>1;        //2nd LSB(PINB) = B
                C = x & 0x04;
                C = C>>2;        // 3rd LSB(PINB) = C
                D = x & 0x08;
                D = D>>3;        //4rth LSB(PINB) = D

                F0 = ((~A & B) | (~B & C & B));
                F0 = (~F0);      //com F0
                F0 = F0 & 0x01; //keep only LSB from F0

                F1 = (A&C)&(B|D);
                F1 = F1<<1;      //2nd LSB has now F1
                F0 = F0+F1;      //output at F0 ->1rst LSB F0
                                 //2nd LSB F1
                PORTC = F0;
        }
}
```

2 Μετρητής - Διακοπή INT1

Το ζητούμενο πρόγραμμα απεικονίζει στα dip switches της PORTC την συνεχή μέτρηση από 0 έως 255.Όταν προκληθεί η διακοπή INT1 η ρουτίνα εξυπηρέτησης διακοπής INT1 αυξάνει κατά 1 τον μετρητή διακοπών. Εάν τα dip switches PA6 και PA7 είναι στο λογικό '1' τότε στην θύρα εξόδου PORTC απεικονίζεται σε δυαδική μορφή ο αριθμός των μέρχι τότε διακοπών. Η υλοποίηση γίνεται σε assembly.

```
.include "m16def.inc"
.org 0x0
rjmp reset
.org 0x4      ;address of INT1 is at 0x4
rjmp ISR1

reset: ldi r24 ,( 1 << ISC11) | ( 1 << ISC10)
           out MCUCR , r24
           ldi r24 ,( 1 << INT1) ;INT1 is allowed
           out GICR , r24
           sei

IO_set: ser r24         ;initialize PORTC
        out PORTC, r24  ;for output
        clr r24         ;initialize PORTA
        out PORTA, r24  ;for input when INT1 ocures


main_program:out PORTC ,r26     ;show counter at PORTC
             inc r26            ;increase counter
             rjmp main_program  ;loop


ISR1: in r21,PINA         ;load PINA
      andi r21,0xc0       ;check bits 6 and 7
      cpi r21,0xc0
      breq cont           ;if bits 6 and 7 ->'1' continue
      inc r23             ;else increase the counter
                          ;for the interrupts INT1 till now
      reti                ;and return at the main program
```

```
cont:   inc r23          ;increase the counter for the interrupts
                         ;INT1 till now
        out PORTB,r23    ;show the counter at PORTB
        reti
```

3 Μέτρηση ON dip switches της PORTB - Διακοπή INT0

Το ζητούμενο πρόγραμμα αναμένει την διακοπή INT0. Όταν προκληθεί η ρουτίνα εξυπηρέτησης της INT0 μετράει πόσα dip switches της PORTB είναι ON. Εάν το PA2 είναι OFF ανάβει τόσα LEDs της θύρας PORTC όσα τα ON dip switches της PORTB, ξεκινώντας από το LSB. Διαφορετικά απεικονίζει σε δυαδική μορφή τον αριθμό των ON dip switches της PORTB στην PORTC. Η υλοποίηση γίνεται σε C.

```
#include <avr/io.h>
#include <avr/interrupt.h>
unsigned char Led,check,b,c,counter,cnt,a,i;

//interrupt [EXT_INT0]
ISR(INT0_vect)  // External Interrupt 0 service routine
        {
                b = PORTB;
                a = 0x01;                //a = 1
                Led = PINA;
                cnt = 0x00;
        for(i=0; i<8; i++){
                        c = b & a;       //mask LSB of PORTB
                        if(c==0x01){
                                cnt++;
                        }
                        b = b>>1;    //rotate so the 2nd LSB
                                     //is checked
        }

        check = Led & 0x04;        //check only LSB 2 from PORTA

        if(check == 0x00){         //if PA2 is OFF
                a = 0x01;
                Led=0x00;
```

```
                for(i=1; i<=cnt; i++){
                        Led = Led+a;
                        a = a<<1;
                }
                PORTC = Led;  //number of ON dip switches
                              //(PORTB)->PORTC
        }

        else{PORTC = cnt;}    //if PA2 is ON, binary number
                              //of ON dip switches(PORTB)->PORTC
}

int main(void){
        DDRC = 0xff;     //PORTC->output
        DDRB = 0x00;     //PORTB ->input
        DDRA = 0x00;     //PORTA ->input

        MCUCR = (1<<ISC01 | 1<< ISC00); //enable INT0 interrupt
        GICR = (1<<INT0);
        sei     ();       //enable all interrupts
        while(1){};       // wait for INT0

}
```