22/5/21

Εργαστηριακή Άσκηση AVR

Συστήματα Μικροϋπολογιστών

Βαρδάκης Χριστόφορος 03118883
Διαμαντή Χριστίνα 03118203
Λυμπεράκης Γεώργιος 03118881

ΖΗΤΗΜΑ 4.1

Στην άσκησης αυτή καλούμαστε να γράψουμε σε Assembly ένα πρόγραμμα που επιτρέπει και σταματά τη κίνηση ενός κινούμενου LED χρησιμοποιώντας το push button PB0. Το LED εκτελεί περιοδική αριστερή και εν συνεχεία δεξιά ανάμεσα στα lsb και msb της PORTA, η οποία σταματά όσο είναι πατημένο το push button. Ο κώδικας φαίνετε παρακάτω:

```
.include "m16def.inc"

.DEF temp = r24
.DEF input = r21
.DEF output = r26
.DEF direction = r20 ; direction = 1  -> go right

clr temp ; initialize PORTB
out DDRB , temp ; as input
ser temp ; initialize PORTA
out DDRA , temp ;as output
out PORTB,temp ; pull-up B

ldi output, 0x01 ; initialize first LED
ldi direction, 0x00 ; initialize LED direction to the left

CHECK_B0:
in input, PINB ; read PORTB
andi input, 0x01 ; isolate input LSB  (the bit we are interested in)
cpi input, 0x01 ; check if B0 is pressed
breq CHECK_B0  ;if pressed keep checking until pressed again
cpi direction, 0x00 ; check LED direction
brne GO_RIGHT

GO_LEFT:
cpi output, 0x80 ; check if LED is on MSB
breq CHANGE_TO_RIGHT ; if true go right
out PORTA, output ; else turn on the next LED to the left
lsl output
jmp CHECK_B0 ; check if B0 remains pressed

CHANGE_TO_RIGHT:
ldi direction, 0x01

GO_RIGHT:
cpi output, 0x01 ; check if LED is on LSB to change direction
breq CHANGE_TO_LEFT
out PORTA, output
lsr output
jmp CHECK_B0

CHANGE_TO_LEFT:
ldi direction, 0x00 ; change direction to the left
jmp GO_LEFT
```

ΖΗΤΗΜΑ 4.2

Στην άσκησης αυτή καλούμαστε να υλοποιήσουμε σε ένα σύστημα AVR δύο λογικές συναρτήσεις *F0 = (ABC' + CD)'* και *F1= (A+B)·(C+D).* Παρακάτω φαίνετε η υλοποίηση σε C και Assembly αντίστοιχα:

```
#include <avr/io.h>
//F0 = (ABC' + CD)'
//F1= (A+B)*(C+D)

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

      while(1){

            temp= PINA ;       //read PINA
            A = temp & 0x01;//read 1st lsb and assign to variable A (2^0 = 1)
            B= temp & 0x02;    //read 2nd lsb (2^1 = 2)
            B = B >> 1;        //assign 2nd lsb to lsb of variable B
            C= temp & 0x04;    //read  3rd lsb (2^2 = 4)
            C = C>>2;          //assign 3rd lsb to lsb of variable C
            D= temp & 0x08;    //read  4th lsb (2^3 = 8)
            D = D>>3;          //assign 4th lsb to lsb of variable D

            F0= (~((A & B & (~C)) | (C & D) ))&0x01;  //evaluate F0 and keep only
                                                      //lsb due to bitwise operations

            F1= (((A | B ) & (C | D) ))&0x01; //evaluate F1 and keep only lsb due
                                                      //to bitwise operations

            F1 = F1 << 1;      //left shift of F1 to move value on 2nd lsb.
            PORTB =  F0 | F1;//show result(using bitwise or on F1 and F2 and ouput
                               //on PORTB)

      }
      return 0;
```

```
.include "m16def.inc"

;F0 = (ABC' + CD)'
;F1= (A+B)*(C+D)

.DEF A=r16 ; declare registers
.DEF B=r17
.DEF C=r18
.DEF CN=r19
.DEF D=r20
.DEF temp=r21
.DEF A2=r22
.DEF temp2=r23

start:
clr temp ; port A as input
out DDRA,temp
ser temp
out PORTA,temp ; pull-up of gate D
out DDRB,temp ; port as output

;F0 = (ABC' + CD)'
F0:
in temp, PINA ; read PORTA
mov temp2,temp ; create copy of input
mov A, temp ; store A on LSB of register A
lsr temp
mov B, temp ; store B on LSB of register B
lsr temp
mov CN, temp ; store C on LSB of register CN
com CN ; complement of C
mov C, temp ; store C on LSB of register C
lsr temp
mov D, temp ; store D on LSB of register D
and A, B ; A=A*B
and A,CN ; A=A*B*CN
and C,D ; C=C·D
or A,C ; compute A=(A*B*CN + C·D)
com A
andi A, 1 ; isolate LSB

;F1= (A+B)*(C+D)
F1:
mov A2, temp2 ; store A on LSB of register A
lsr temp2
lsr temp2
mov C, temp2  ; store C on register C
or A2, B ; A=A+B´
or C,D ; C=C+D
and A2,C ; compute A=(A+B)*(C+D)
andi A2, 1 ; isolate LSB
lsl A2  ; put F1 on right position
or A,A2 ;F1F0 on PORTB
andi A,0x03
out PORTB,A ; print result
rjmp F0 ; jump on F0 to repeat
```

ΖΗΤΗΜΑ 4.3

Στην άσκησης αυτή καλούμαστε να υλοποιήσουμε σε ένα σύστημα AVR το οποίο μετακινεί το LED μέσω τεσσάρων push button. Ποιο συγκεκριμένα με το πάτημα του πρώτου και του δεύτερου lsb γίνετε αριστερή και δεξιά ολίσθηση του LED, ενώ με το τρίτο και τέταρτο lsb το LED μετακινείται άμεσα στην έβδομη (128) και πρώτη θέση. Ως είσοδος χρησιμοποιείται η PORTA εμώ ως έξοδος η PORTB.

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

char x;
void wait(char,char);

int main(void)
{
	DDRA = 0xFF; //Port A as output
	DDRC = 0X00; //Port C as input

	x = 1;
	PORTA = x; //Turn on the LED0

	while (1)
	{

		if((PINC & 0x01) == 1) {//Left Rotation

			while((PINC & 0x01) == 1) { ; } //wait by doing nothing
			//wait(0x01,1);

			if (x == 128) x = 1; //check for overflow
			else x = x << 1;
		}

		else if ((PINC & 0x02) == 2){//Right Rotation

			while ((PINC & 0x02) == 2) { ; } //wait by doing nothing
			//wait(0x02,2);

			if (x == 1) x = 128;
			else x = x >> 1;
		}

		else if ((PINC & 0x04) == 4){//Move to MSB

			while ((PINC & 0x04) == 4) { ; } //wait by doing nothing
			//wait(0x04,4);

			x = 128;

		}
```

```
            else if ((PINC & 0x08) == 8){//Move to LSB

                  while((PINC & 0x08) == 8) { ; } //wait by doing nothing
                  //wait(0x08,8);

                  x = 1;
            }

            else { continue; }//if there is no new input then
                              //check again the switches

            PORTA = x;//Print the right output
      }

      return 0;
}

void wait(char bin,char num){
      while((PINC & bin) == num) { ; }
}
```