Συστήματα Μικροϋπολογιστών 2021 (6ο εξάμηνο)

Φιλιππόπουλος Ορφέας el18082
Παπαρρηγόπουλος Θοδωρής el18040
Γραμμένου Δέσποινα el18061

Άσκηση 1

Αρχικά με τη βοήθεια των εντολώς ser και clc γράφουμε στο καταχωρητή DDRA άσσους προκειμένου να είναι η θύρα Α θύρα εξόδου και στη συνέχεια γράφουμε μηδενικά στο καταχωρητή DDRB προκειμένου να είναι η θύρα Β θύρα εισόδου.

Στη συνέχεια χρησιμοποιήσαμε έναν μετρητή (r24) αρχικοποιημένο στο 0, ο οποίος σε κάθε κίνηση (η κίνηση του LED προς τα αριστερά επιτυγχάνεται με την εντολή lsl) του LED προς τα αριστερά αυξάνεται κατά ένα. Όταν αυτός ο μετρητής φτάσει στην τιμή 7 αρχίζουμε κίνηση (η κίνηση του LED προς τα δεξιά επιτυγχάνεται με χρήση της εντολής lsr) προς τα δεξιά, μειώνοντας τον μετρητή κατά ένα κάθε φορά. Όταν μηδενιστεί αρχίζουμε πάλι κίνηση προς τα αριστερά. Επιπλέον, με την εντολή in λαμβάνουμε την είσοδο από το PINB και με την εντολή andi κρατάμε μόνο το PB0. Αν σε οποιοδήποτε στάδιο της κίνησης το bit γίνει 1, η κίνηση σταματάει προσωρινά και συνεχίζει όταν το bit ξαναγίνει 0. Αυτό επιτυγχάνεται με τις εντολές cpi και breq.

Κώδικας:

```
;
; AssemblerApplication1.asm
;
; Created: 5/16/2021 3:13:49 PM
; Author : Admin
;


.include "m16def.inc"

reset:
	ser r24
	out DDRA, r24 ;A is output port
	clr r24
	out DDRB, r24 ;B is input port
	ldi r25, 0x01
	;r24 is our counter for led position

; Replace with your application code
left:
	in r26, PINB ;read B port (input port)
	andi r26, 0x01 ;isolate first bit (PB0)
	cpi r26, 0x01
	breq left ;if 1 do nothing
```

```
	out PORTA, r25 ;else continue process
	inc r24 ;increase counter
	lsl r25 ;logical shift left
	cpi r24, 0x07 ;if counter == 7 then we jump to right which will open PA7 (we
already did lsl) and then will do logical shift right
	breq right
	rjmp left
	;nop



right:
	in r26, PINB ;read B port (input port)
	andi r26, 0x01 ;isolate first bit (PB0)
	cpi r26, 0x01
	breq right ;if 1 do nothing
	out PORTA, r25 ;else continue process
	dec r24 ;decrease counter
	lsr r25 ;logical shift right
	cpi r24, 0x00 ;if counter == 1 then we jump to left which will open PA0 (we
already did lsr) and then will do logical shift left
	breq left
	rjmp right
	;nop
```

Άσκηση 2

Η άσκηση αυτή γράφτηκε και σε assembly και σε C και έχουν παραδοθεί και οι 2. Ωστόσο εξεταστήκαμε μόνο στη C και για το λόγο αυτό μόνο αυτή θα περιγραφεί.

Αρχικά όλες οι μεταβλητές είναι chars (δηλαδή 8-μπιτοι) ενώ αρχικοποιούμε και τα ports (δηλαδή γράφουμε άσσους στο DDRB προκειμένου το port B να αρχικοποιηθεί ως έξοδος και γράφουμε και μηδενικά στο DDRA προκειμένου να αρχικοποιηθεί το port A ως είσοδος. Στη συνέχεια μέσα στο ατέρμονο loop (while(1)) χρησιμοποιούμε εντολές ολίσθησης και bitwise and για να απομονώσουμε καθένα εκ των A, B, C, D σε ξεχωριστές μεταβλητές. Στη συνέχεια (μέσα πάλι στο while(1)) πραγματοποιούμε τις λογικές πράξεις και αποθηκεύουμε τα αποτελέσματα στις μεταβλητές F0 και F1 αντίστοιχα. Πραγματοποιούμε μία αριστερή ολίσθηση στο F1 (προκειμένου να πάει το αποτέλεσμα το 2ο bit) και κάνουμε OR μεταξύ του F0 και του F1 αποθηκεύοντας το αποτέλεσμα στο PORTB προκειμένου να γραφεί στην έξοδο. Κατά αυτό το τρόπο το PORTB θα έχει στο 1ο bit το F0 και στο 2ο bit το F1 ενώ τα υπόλοιπα bits θα είναι ίσα με 0.

Κώδικες:

Assembly:

```
;
; AssemblerApplication1.asm
;
; Created: 5/16/2021 3:13:49 PM
; Author : Admin
;


.include "m16def.inc"

ser r24
out DDRB, r24 ;B is output port
clr r24
out DDRA, r24 ;A is input port


reset:

	in r25, PINA ;xxxxDCBA, original
	mov r26, r25
	andi r26, 0x01 ;for A
	lsr r25
	mov r24, r25
	andi r24, 0x01 ;for B
	lsr r25
	mov r23, r25
	andi r23, 0x01 ;for C
	lsr r25
	mov r22, r25
	andi r22, 0x01 ;for D

F0:
```

```
        mov r0, r26 ;temp for A
        mov r1, r24 ;temp for B
        mov r2, r23 ;temp for C
        mov r3, r22 ;temp for D

        com r2
        and r0, r2 ; AC'

        and r0, r1 ;ABC'

        com r2
        and r2, r3 ;CD

        or r0, r2 ;ABC'+CD
        com r0 ;result of F0
        mov r16, r0
        andi r16, 0x01

nop

F1:
        or r26, r24 ;A+B
        or r23, r22 ;C+D
        and r26, r23 ;result of F1

        lsl r26
        or r16, r26 ; results ofr F1,F0 to the to LSB's of port B


out PORTB, r16 ;result to port B
jmp reset
```

C:

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

//initializations
char A;
char B;
char C;
char D;
char temp;
char F0;
char F1;

int main(void)
{ // add breakpoint here
        DDRB = 0xFF;
        DDRA = 0x00;

   while(1)
   {
      A = PINA & 0x01;
                temp = PINA >> 1;
                B = temp & 0x01;
                temp = temp >> 1;
                C = temp & 0x01;
```

```
            temp = temp >> 1;
            D = temp & 0x01;

            F0 = !(A & B & (!C & 0x01) | C & D);
            F1 = (A | B) & (C | D);

            F1 = F1 << 1;
            PORTB = F1 | F0;

  }
}
```

Άσκηση 3

Αρχικά γράφουμε άσσους στο DDRA προκειμένου η θύρα A να είναι θύρα εξόδου και γράφουμε μηδενικά στο DDRC προκειμένου η θύρα C να είναι θύρα εισόδου.

Στη συνέχεια, το πρόγραμμα περιέχει ένα ατέρμονο loop, μέσα στο οποίο σε κάθε επανάληψη ελέγχεται αν είναι πατημένο κάποιο από τα SWx (με x = 0,1,2,3). Αυτό πραγματοποιείται με σύγκριση του PORTC με τα 1, 2, 4 και 8. Στη συνέχεια εάν κάποιο από αυτά είναι πατημένο μπαίνουμε στην if που του αντιστοιχεί και εκτελούμε ένα while μέχρι το button αυτό να αφεθεί (προσομοιώνοντας έτσι την άσκηση στο πραγματικό κόσμο). Όταν αυτό αφεθεί τότε εκτελούμε και το ζητούμενο το οποίο του αντιστοιχεί.

Κώδικας:

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

int main(void){

	DDRA = 0xFF; //port A output
	DDRC = 0x00; // port C input (SWx's)
	char led;

	led = 0x01;
	PORTA = led;


	while(1){

		if((PINC & 0x01) == 1){
			while((PINC & 0x01) == 1){}
			if(led == 128){
				led = 0x01; // led = 1 if before led == 128
			}
			else led = led << 1;
		}

		if((PINC & 0x02) == 2){
			while((PINC & 0x02) == 2){}
			if(led == 1){
				led = 1 << 7; // led = 128 if before led == 0
			}
			else led = led >> 1;
		}

		if((PINC & 0x04) == 4){
			while((PINC & 0x04) == 4){}
			led = 1 << 7;
		}

		if((PINC & 0x08) == 8){
			while((PINC & 0x08) == 8){}
			led = 0x01;
		}
```

```
            PORTA = led;
        }

}
```