4η Ομάδα Ασκήσεων στα Συστήματα Μικροϋπολογιστών
Εργαστήριο-Ομάδα ΜΥ-105

6ο Εξάμηνο, Ακαδημαϊκή Περίοδος 2020 – 2021

Ονοματεπώνυμο	Αριθμός Μητρώου
Μπούφαλης Οδυσσεύς – Δημήτριος	el18118
Στεφανάκης Γεώργιος	el18436

Η παρούσα αναφορά περιέχει την ανάπτυξη κώδικα σε assembly και σε C για τον προγραμματισμό του μικροελεγκτή AVR ATmega16. Χρησιμοποιήθηκε ο simulator του Atmel Studio 7.0 προκειμένου να γίνει ο έλεγχος της ορθής λειτουργίας των προγραμμάτων.

Άσκηση 1η

```
.include "m16def.inc"

reset:      ldi r24,low(RAMEND) ; initializing stack pointer
            out SPL,r24
            ldi r24,high(RAMEND)
            out SPH,r24

            clr r24           ; initializing PORTB as input
            out DDRB,r24
            ser r24
            out DDRA,r24    ; initializing PORTA as output

            clr r25     ; using r25 as flag for up/down

            ldi r17 , 0x01  ; using r17 for output
            out PORTA , r17

main :
wait : in r26,PINB  ; we get the input from PortB
       andi r26,0x01; we mask PB0
       cpi r26,0x01 ; and check if it is pressed or not

       breq wait   ; if it is pressed we wait

       cpi r25,0x00  ; we check the flag
       breq upcounting
       rjmp downcounting



upcounting : lsl r17 ; shift left
             out PORTA,r17 ; and output
             cpi r17,0x80 ; if we have reached the last bit we set the flag to 1
             brne main  ; if not we go to main
             ldi r25,0x01 ; change the flag
             rjmp main ; return to main to check input

downcounting :  lsr r17 ; shift right
                out PORTA,r17 ; and output
                cpi r17,0x01 ; if we have reached the LSB set the flag to 0
```

```
brne main
ldi r25,0x00 ;
rjmp main ; return to main to check input
```

Το παραπάνω πρόγραμμα σε Assembly απεικονίζει ένα αναμμένο led στα bit της θύρας PA0-PA7 το οποίο κινείται από τα LSB προς τα MSB και αντίστροφα όταν φτάσει στο άλλο άκρο. Επίσης, αναλόγως με το αν είναι πατημένο ή όχι το push button PB0 η κίνηση σταματάει ή συνεχίζεται αντιστοίχως. Αρχικά, αρχικοποιούμε την στοίβα, θέτουμε την PORTA έξοδο και την PORTB είσοδο, θέτουμε όλα τα bit του καταχωρητή r25 ίσα με 0 για να τον χρησιμοποιήσουμε σαν flag και ανάβουμε το πρώτο led. Στη συνέχεια, κάθε φορά διαβάζουμε την είσοδο από το PINB και απομονώνουμε το LSB. Αν είναι πατημένο το PB0 η κίνηση σταματάει μέχρι να αφεθεί. Αν δεν είναι πατημένο κοιτάμε την σημαία μας. Αν αυτή είναι 0 πηγαίνουμε στο label upcounting όπου κάνουμε την αριστερή κίνηση του led με το output στο PORTA και ξαναπηγαίνουμε στην main για να διαβάσουμε το PINB αφού όμως πρώτα ελέγξουμε αν έχουμε φτάσει στο MSB οπότε θα πρέπει να θέσουμε την σημαία ίση με 1. Αν η σημαία είναι 1 τότε πηγαίνουμε στο label downcounting όπου κάνουμε την δεξιά κίνηση και επιστρέφουμε στην main αφού ελέγξουμε αν έχουμε φτάσει στο LSB οπότε θα πρέπει να θέσουμε τη σημαία ίση με 0. Τέλος, η επιλογή για το insert των breakpoints έγινε ως εξής:

- Στο label wait προκειμένου να περιμένουμε για αλλαγή στο input.
- Μετά το out PORTA,r17 στα labels upcounting και downcounting προκειμένου να φανεί η αλλαγή στην έξοδο.

Άσκηση 2η

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

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

 while (1) {
 A = PINA & 0x01 ; // masking the LSB
 B = PINA & 0x02 ; // masking the 2nd bit
 C = PINA & 0x04 ; // masking the 3rd bit
 D = PINA & 0x08 ; // masking the 4th bit
 B = B>>1; //shifting right
 C = C>>2; // shifting right
 D = D>>3; // shifting right
 F0 = ~((A & B & (~C)) | (C & D)); // the wanted expression
 F0 = F0 & 0x01;

 F1 = ( (A|B) & (C|D) ) ; // the 2nd wanted expression
 F1 = F1<<1; // shift left
 F1 = F1 & 0x02 ;
 AUX = F0 | F1 ;
 AUX = AUX & 0x03 ;
 PORTB = AUX;
 }
 return 0;
}
```

Ο παραπάνω κώδικας σε C διαβάζει τα 4 LSB της θύρας εισόδου PORTA που αντιστοιχούν στα A,B,C,D και αφού υπολογίσει τις λογικές συναρτήσεις F0=(ABC'+CD)' και F1=(A+B)(C+D) εμφανίζει τις τιμές τους στα 2 LSB της θύρας εξόδου PORTB αντιστοίχως. Αρχικά, θέτουμε την θύρα A ως είσοδο και την θύρα B ως έξοδο. Στη συνέχεια

διαβάζουμε την είσοδο από το PINA κάνοντας mask το LSB για το A, το 2ο bit για το B, το 3ο bit για το C και το 4ο bit για το D. Στη συνέχεια, μεταφέρουμε το χρήσιμο περιεχόμενο αυτών των 8bitων μεταβλητών στη ίδια θέση και συγκεκριμένα στο LSB έτσι ώστε να γίνουν σωστά οι bitwise operators &,|,~. Αφού υπολογίσουμε τις 2 λογικές συναρτήσεις και μεταφέρουμε το χρήσιμο περιεχόμενο του F1 στη θέση 1 από την θέση 0 για να το κάνουμε output στο PORTB1 χρησιμοποιούμε την βοηθητική μεταβλητή AUX όπου κρατάει το bitwise or των F0 και F1 και στη συνέχεια κάνουμε mask τα 2 LSB με το &0x03 (hex). Τέλος, στέλνουμε το αποτέλεσμα στην PORTB όπου ανάβουν ή όχι τα 2 τελευταία Leds ανάλογα με την αποτίμηση των λογικών εκφράσεων. Ο παρακάτω πίνακας αληθείας επιβεβαιώνει την προσομοίωσή μας:

A	B	C	D	F0	F1
0	0	0	0	1	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	0
0	1	0	0	1	1
0	1	0	1	1	1
0	1	1	0	1	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	1
1	0	1	0	1	1
1	0	1	1	0	1
1	1	0	0	0	0
1	1	0	1	0	1
1	1	1	0	1	1
1	1	1	1	0	1

Τέλος, προσθέσαμε ένα breakpoint στο τέλος του σώματος του while μετά το PORTB = AUX; έτσι ώστε να βλέπουμε την έξοδο κάθε φορά που γίνεται η αποτίμηση.

Άσκηση 3η

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

 char A;
 int main (void)
 {
 DDRA = 0xFF ; // initializing PORTA as output
 DDRC = 0x00 ; // initializing PORTC as input
 A = 0x01 ; // start from LSB
 PORTA = A;

 while (1) {
   if ((PINC & 0x01 ) == 1) {       //Check for push-button SW0
     while ((PINC & 0x01) == 1) ; // check for letting free
     if (A == 0x80) // if MSB is on
        A = 0x01 ; // light LSB
```

```
     else
        A = A << 1; // left shift
   }
   if ((PINC & 0x02) == 2) {        //Check for push-button SW1
    while ((PINC & 0x02) == 2) ; // check for letting free
    if (A == 0x01) // if LSB is on
      A = 0x80 ; // light MSB
    else
      A = A>>1; // right shift
   }

 if ((PINC & 0x04) == 4) { //Check for push-button SW2
  while ((PINC & 0x04) == 4) ; // check for letting free
  A = 0x80 ; // light MSB
 }



 if ((PINC & 0x08) == 8) { //Check for push-button SW3
   while ((PINC & 0x08) == 8) ; // check for letting free
   A = 0x01 ; // light LSB
   }

   PORTA = A; // pass A as output
   }
}
```

Στο παραπάνω πρόγραμμα σε C αρχικά ανάβουμε το led0 (μέσω της 8bitης μεταβλητής A) που είναι συνδεδεμένο στο 1ο bit της θύρας εξόδου PORTA. Στη συνέχεια διαβάζουμε την είσοδο από το PINC στα 4 LSB του οποίου είναι συνδεδεμένα τα buttons SW0-SW3. Έτσι έχουμε 4 if statements που το κάθε ένα κάνει mask το LSB,το 2ο LSB,το 3ο LSB και το 4ο LSB έτσι ώστε να καταλαβαίνει το πρόγραμμα πότε έχει πατηθεί το κάθε κουμπί και να εκτελεί την αντίστοιχη λειτουργία. Επίσης, στο σώμα του κώδικα του κάθε if υπάρχει ένα while loop το οποίο είναι true όσο το button παραμένει πατημένο. Συνεπώς, για να προχωρήσει η εκτέλεση του προγράμματος και να εκτελεστεί η λειτουργία η οποία προκαλείται από το εκάστοτε button πρέπει αφού πατήσουμε το button στη συνέχεια να το αφήσουμε. Τέλος, αφού εκτελεστεί η σωστή λειτουργία (shift left,shift right,light MSB,light LSB) στέλνουμε την μεταβλητή A στην output θύρα PORTA για να ανάψει το κατάλληλο led. Η εισαγωγή των breakpoints έγινε στα εξής σημεία:

- Από ένα breakpoint σε κάθε while των 4ων if statements έτσι ώστε να διακόπτει το πρόγραμμα για να αφήσει ο χρήστης το αντίστοιχο button.
- Αμέσως μετά την εντολή PORTA = A; έτσι ώστε να φανεί η νέα έξοδος.