Συμεών Ποργιώτης el18053

Στέφανος Τσώλος el18050

Βικέντιος Βιτάλης el18803

Ε.Μ.Π. - ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧ. ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΑΚΑΔ. ΕΤΟΣ 2020-2021

Άσκηση 1

```
reset:
ldi r24 , low(RAMEND) ; Αρχικοποίηση stack pointer
out SPL , r24 ; Πληροφορία περνιέται απο τον καταχωρητή στο PORT
ldi r24 , high(RAMEND)
out SPH , r24 ; Πληροφορία περνιέται απο τον καταχωρητή στο PORT
ser r26 ; Αρχικοποίηση της PORTA
out DDRA , r26 ; Για έξοδο
clr r26 ; Θέτουμε τον καταχωρητή r26 στην τιμή 0
out DDRB , r26 ; Για είσοδο
ldi r26,0x01 ; Φόρτωση άμεσωω δεδομένων = δεκαεξαδικό 01
out PORTA,r26
; Υπορουτίνα για να αναβουν τα LEDs απο LSB προς MSB
left:
pause_left:
        IN r25,PINB ;'Ελεγχος εισόδου
        cpi r25,0x01 ; Σύκγριση καταχωρητή r25 με δεκαεξαδικό 01
        breq pause_left ; Αν το αποτέλεσμα είναι 0, άλμα στην αρχή του βρόχου
        lsl r26 ; Logical Shift Left/Αριστερή ολίσθηση καταχωρητή r26
        out PORTA , r26 ; Πληροφορία περνιέται απο τον καταχωρητή στο PORT
```

```
        cpi r26,0x80 ; Σύκγριση καταχωρητή r26 με δεκαεξαδικό 80
        breq right ; Αν το αποτέλεσμα είναι 0, άλμα στον βρόχο right
        jmp left ; Άλμα χωρίς συνθήκη στον βρόχο left

; Υπορουτίνα για να αναβουν τα LEDs απο MSB προς LSB
right:
pause_right:
        IN r25,PINB ;'Ελεγχος εισόδου
        cpi r25,0x01 ; Σύκγριση καταχωρητή r25 με δεκαεξαδικό 01
        breq pause_right ; Αν το αποτέλεσμα είναι 0, άλμα στην αρχή του βρόχου
        lsr r26 ; Logical Shift Right/Δεξιά ολίσθηση καταχωρητή r26
        out PORTA , r26 ; Πληροφορία περνιέται απο τον καταχωρητή στο PORT
        cpi r26,0x01 ; Σύκγριση καταχωρητή r26 με δεκαεξαδικό 01
        breq left  ; Αν το αποτέλεσμα είναι 0, άλμα στον βρόχο left
        jmp right ;Άλμα χωρίς συνθήκη
```

Υλοποιούμε ένα πρόγραμμα σε Assembly το οποίο απεικονίζει ένα αναμμένο LED το οποίο αντιστοιχεί στα bit θύρας PA0-PA7. Το LED κινείται συνεχώς και ελέγχουμε την κίνησή του από το push button PB0.

Αρχικοποιούμε την στοίβα. Εκεί αποθηκεύεται η διεύθυνση η οποία βρισκόμαστε όταν καλούμε μια συνάρτηση, ο Program Counter. Έτσι όταν η συνάρτηση μας κάνει return, το πρόγραμμα μας ανακαλεί από τη στοίβα την θέση που βρισκόταν και γυρίζει εκεί. Αν δεν έχουμε αρχικοποιήσει τη στοίβα στο πρόγραμμα μας, όταν μπαίνουμε σε μια συνάρτηση δε θα επιστρέφει ποτέ ή θα επιστρέφει στην αρχή του προγράμματος (0) γιατί δε θα έχει ορισθεί στοίβα.

Χρησιμοποιήσαμε τις εντολές clr, ser και out για να θέσουμε την θύρα B ως είσοδο και την θύρα A ως έξοδο. Στη συνέχεια χρησιμοποιήσαμε έναν μετρητή (r26) αρχικοποιημένο στο 0, ο οποίος σε κάθε κίνηση του LED προς τα αριστερά αυξάνεται κατά ένα. Όταν αυτός ο μετρητής φτάσει στην τιμή 7, αρχίζουμε κίνηση προς τα δεξιά, μειώνοντας τον μετρητή κατά ένα κάθε φορά. Όταν μηδενιστεί ξανά, αρχίζουμε πάλι κίνηση προς τα αριστερά. Επιπλέον, με την εντολή in λαμβάνουμε την είσοδο από το PINB. 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.

Άσκηση 2:

```
#include <avr/io.h>
unsigned char a, b, c, d, f0, f1, notc; // Απρόσημη δήλωση χαρακτήρων

int main(void) {
	DDRB = 0xff;			// portB έξοδος
	DDRA = 0x00;			// portA είσοδος

	while(1)
		a = PINA & 0x01;	// a -> A bit0/Απομόνωση των κατάλληλων bits στην PORTA

		b = PINA & 0x02;	// b -> A bit1/Απομόνωση των κατάλληλων bits στην PORTA
		b = b >> 1;		// Μεταφορά ψηφίου στην ορθή του αξία

		c = PINA & 0x04;	// c -> A bit2/Απομόνωση των κατάλληλων bits στην PORTA
		c = c >> 2;		// Μεταφορά ψηφίου στην ορθή του αξία

		d = PINA & 0x08;	// d -> A bit3/Απομόνωση των κατάλληλων bits στην PORTA
		d = d >> 3;		// Μεταφορά ψηφίου στην ορθή του αξία

		f1 = (a | b) & (c | d);  // Υλοποιήση της λογικής συνάρτησης f1
		f1 = f1 << 1;		// Ολίσθηση αριστερά

		notc = c^0x01;		// Συμπλήρωμα του c με XOR Gate εισόδου c και δεκαεξαδικού 01
		f0 = ((a & b & notc) | (c & d));  // Υλοποιήση της λογικής συνάρτησης f0
		f0 = f0^0x1;		// Συμπλήρωμα f0 με χρήση XOR
		f0 = f0 + f1;		// Υπολογισμός αθροίσματος και
		PORTB = f0;		// Έξοδος στην PORTB
```

```
        }
        return 0;
}
```

Η άσκηση γράφτηκε σε C. Όλες οι μεταβλητές μας είναι τύπου char για να είναι 8-bit. Αρχικά θέτουμε 0 όλα τα bits του DDRA (PINA είναι είσοδος) και 1 όλα τα bits του DDRB (PORTB είναι έξοδος). Στη συνέχεια χρησιμοποιούμε εντολές ολίσθησης και bitwise and για να απομονώσουμε καθένα εκ των A, B, C, D σε ξεχωριστές μεταβλητές. Στη συνέχεια οι ζητούμενες λογικές πράξεις πραγματοποιούνται μεταξύ των νέων μεταβλητών που ορίσαμε και προκύπτουν τα ζητούμενα αποτελέσματα (F1) και (F0). Στο (F1) χρησιμοποιείται και μία αριστερή ολίσθηση για να βρεθεί στην επιθυμητή θέση (bit 1), ενώ το (F0) βρίσκεται εξ' αρχής στη σωστή θέση (bit 0). Στη συνέχεια απομονώνουμε μόνο τα συγκεκριμένα bits με bitwise and και τέλος τα βάζουμε στην έξοδο μαζί χρησιμοποιώντας bitwise or. Όλο το πρόγραμμα (εκτός από τις αρχικοποιήσεις) βρίσκεται μέσα σε ένα ατέρμονο loop, ώστε να έχουμε συνεχή λειτουργία.

Άσκηση 3:

```
#include <avr/io.h>
unsigned char x;                                    //Δήλωση απρόσημου χαρακτήρα

int main(void) {
        DDRA = 0xff;                                // portA Έξοδος
        DDRC = 0x00;                                // portC Είσοδος

        x = 1;                                              // Αρχικοποίηση
μεταβλητής για αρχικά αναμμένο led

        while(1) {
                if ((PINC & 0x01) == 1) {
                                                            // Χρήση πρώτου push-
button
                        while ((PINC & 0x01) == 1) {}       // Έλεγχος επαναφοράς
push-button
                        if(x == 0x80) {                     // Έλεγχος υπεχείλισης
                                x = 0x01;                   // Hex 01 = Decimal 1
```

```
                }
                else {
                        x = x << 1;                     // Ολίσθηση αριστερά
                }
        }

        if ((PINC & 0x02) == 2){
                                                        // Χρήση δευτέρου push-
button
                while ((PINC & 0x02) == 2) {}           // Έλεγχος επαναφοράς
push-button
                if(x == 0x01) {                 // Έλεγχος υπεχείλισης
                        x = 0x80;                       // Hex 80 = Decimal 128
                }
                else {
                        x = x >> 1;                     // Ολίσθηση δεξιά
                }
        }

        if ((PINC & 0x04) == 4){
                                                        // Χρήση τρίτου push-
button
                while ((PINC & 0x04) == 4) {}           // Έλεγχος επαναφοράς
push-button
                x = 0x80;                               // Hex 80 = Decimal 128
        }

        if ((PINC & 0x08) == 8){
                                                        // Χρήση τέταρτου push-
button
                while ((PINC & 0x08) == 8) {}           // Έλεγχος επαναφοράς
push-button
                x = 0x01;
```

```
                }


                PORTA = x;                                  // Έξοδος σε PORTA
        }
        return 0;
}
```

Ομοίως με την προηγούμενη άσκηση καθορίζουμε την είσοδο και την έξοδο. Το πρόγραμμα και πάλι περιέχει ένα ατέρμονο loop, μέσα στο οποίο σε κάθε επανάληψη ελέγχεται αν είναι αναμμένο κάποιο από τα τέσσερα LSB LEDs της θύρας Α. Αυτό πραγματοποιείται με σύγκριση του PORTA με τα 1, 2, 4 και 8. Καθεμία από αυτές τις τέσσερις περιπτώσεις αναμένει το σβήσιμο του LED με μια εντολή while και στη συνέχεια εκτελείται η αντίστοιχη λειτουργία. Στις περιπτώσεις όπου έχουμε κυκλική περιστροφή πραγματοποιούμε έλεγχο για το αν το LED έχει φτάσει στο άκρο, δηλαδή π.χ. στην περίπτωση της αριστερής περιστροφής, αν το LED είναι στη θέση 0 (PORTC == 1) τότε το μεταφέρουμε στη θέση 7 (PORTC = 0x80). Αλλιώς πραγματοποιείται εντολή ολίσθησης προς την επιθυμητή κατεύθυνση. Προκειμένου να ληφθεί οποιαδήποτε εντολή στο πρόγραμμα, αναμένεται η επαναφορά των push buttons. Τέλος, στις περιπτώσεις όπου θέλουμε μεταφορά του LED στο LSB ή στο MSB απλά θέτουμε στο PORTC τις τιμές 1 και 0x80 αντίστοιχα.