Ε.Μ.Π. - ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧ. & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ & ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΑΘΗΝΑ 21. 9. 2021

ΓΡΑΠΤΗ ΕΞΕΤΑΣΗ ΣΤΟ ΜΑΘΗΜΑ"***Συστήματα Μικροϋπολογιστών***"

(ΘΕΜΑ 2ο – ΣΥΝΟΛΟ 4.5 Μονάδες)

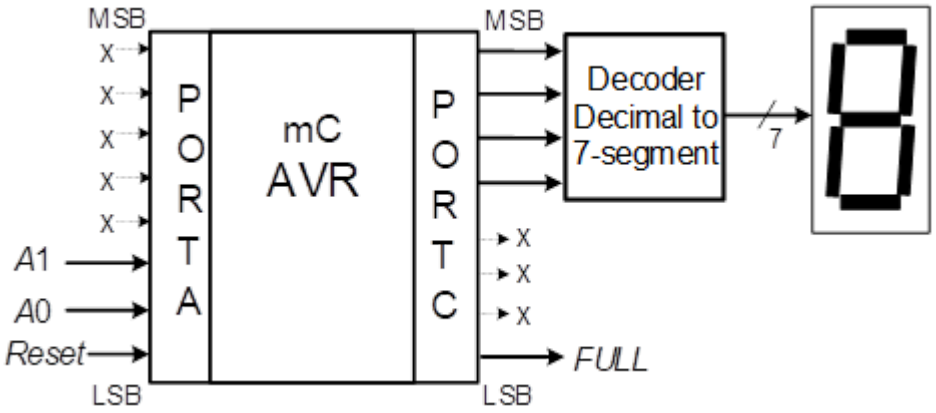
Έναρξη 12:30 - ΔΙΑΡΚΕΙΑ 60’ + 10’ Παράδοση: 13:40’

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: Θοδωρής Παπαρρηγόπουλος

(Ξεχασα να βάλω όνομα πριν)

ΘΕΜΑ 2ο: (4.5 ΜΟΝΑΔΕΣ): Σε ένα μικροελεγκτή AVR Mega16 που φαίνεται στο σχήμα, να υλοποιηθεί ένα σύστημα απεικόνισης των ελεύθερων θέσεων ενός parking. Υποθέτουμε ότι υπάρχει χωριστή είσοδος και έξοδος, εφοδιασμένες η κάθε μία με δέσμη φωτός και φωτοκύτταρα *A*0 και *A*1 αντίστοιχα. Κάθε όχημα που περνάει, διακόπτει τη δέσμη φωτός (το φωτοκύτταρο τότε δίνει 0) και στη συνέχεια αυτή επανέρχεται. Για την απλούστευση της λύσης υποθέτουμε ότι αποκλείεται η περίπτωση να έχουμε ταυτόχρονα είσοδο και έξοδο οχήματος. Το σύστημα διαθέτει είσοδο *Reset* που όταν τεθεί στο ‘1’, υποθέτοντας ότι τότε το parking είναι άδειο, να αρχικοποιεί τις ελεύθερες θέσεις στις 30. Επίσης στην εκκίνηση του συστήματος το parking να θεωρηθεί άδειο. Όταν ο αριθμός των ελεύθερων θέσεων είναι <10, αυτός να απεικονίζεται στο 7-segment display, αλλιώς να απεικονίζεται ο αριθμός 9 και να ανάβει το led (θετικής λογικής) με την ένδειξη FULL. Υποθέτουμε ότι αν το parking γεμίσει αποκλείεται η είσοδος άλλων οχημάτων. Για τη δική σας διευκόλυνση φτιάξτε ένα πρόχειρο διάγραμμα ροής. Δώστε το πρόγραμμα υλοποίησης του παραπάνω συστήματος σε assembly και σε C.

(Assembly: 2.5 ΜΟΝΑΔΕΣ και C: 2 ΜΟΝΑΔΕΣ)



C:

```
#include <avr/io.h>
#include <mega16.h>

char A1, A0;
char reset;
char input;
char output;
int free_parking_spots;
int main() {
	DDRA = 0x00; // set A as input
	DDRC = 0xFF; // set C as output
	free_parking_spots = 30;

	while (1) {
		input = PINA;
		A1 = input & 0x04;
		A0 = input & 0x02;
		reset = input & 0x01;
		if (reset == 1) {
			free_parking_spots = 30;
			continue;
		}
		if (A0 == 0) {
			free_parking_spots--;

		} else if (A1 == 0) {
			free_parking_spots++;
		}

		if(free_parking_spots < 10) {
			output = free_parking_spots << 4;
		} else {
			output = 0x91; // 10010001 -> 145
```

```
                }
                PORTC = output
        }
}
```

Assembly:

```
ser r24 ; set register r24 = 0xFF
out DDRC, r24 ;B is output port
clr r24 ; clear register to r24 = 0x00
out DDRA, r24 ; A is input port

ldi r25, 1EH ; r25 = 30 ;; r25 is our counter for free_parking_spots

START:
        in r24, PINA ;read input
        andi r24, 0x07 ; 0x07 = 0000 0111 binary

        ;; check reset
        mov r23, r24
        andi r23, 0x01
        cpi r23, 0x01
        breq RESET ; if branch is 0x01 jump to reset

        ;; check A0 = 0
        mov r23, r24
        andi r23, 0x02
        cpi r23, 0x02
        brne A0 ; if branch is not 0000 0010 it means A0 = 0 then jump A0

        ;; check A1 = 0
        mov r23, r24
        andi r23, 0x04
        cpi r23, 0x04
        brne A1 ;  if branch is not 0000 0100 it means A1 = 0 then jump A1
```

```
        rjmp CONTINUE

RESET:
        ldi r25, 1EH ;
        rjmp START

A0:
        dec r25 ; free_parking_spots--
        rjmp CONTINUE
A1:
        inc r25 ; free_parking_spots++
        rjmp CONTINUE

CONTINUE:
        cpi r25, 0x0A
        brlt LESS; branch less than 10
        rjmp MORE

LESS:
        mov r26, r25
        lsl r26 ;; rotate 4 times left
        lsl r26
        lsl r26
        lsl r26
        out PORTC, r26 ;
        rjmp START

MORE:
        ldi r26, 0x91 ; 10010001 -> 145
        out PORTC r26
        rjmp START
```