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

ΑΘΗΝΑ 21. 9. 2021

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

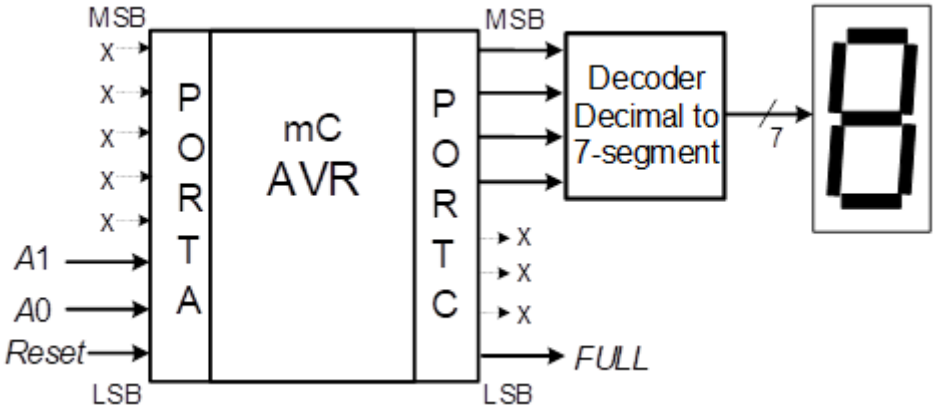
(ΘΕΜΑ 2ο – ΣΥΝΟΛΟ 4.5 Μονάδες)

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

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: Κυριακόπουλος Γεώργιος – el8153 . . . . . .

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:

```c
#include <mega16.h>

int main(void) {
    DDRA = 0x00;
    DDRC = 0xFF;
    char a1, a0, reset, in, out;
    int counter = 30;

    PORTC = 0x91;

    while(1) {
        in = PINA & 0x07;
        reset = in & 0x01;
        if(reset == 0x01) {
            counter = 30;
            PORTC = 0x91;
        }
```

```
        else {
            a0 = in & 0x02;
            a1 = in & 0x04;
            if (a0 != 0x02) {
                counter--;
                if(counter < 10) {
                    out = counter << 4;
                    out = out & 0xF0;
                }
                else {
                    out = 0x91;
                }
                PORTC = out;
            }
            else if(a1 != 0x04) {
                counter++;
                if(counter < 10) {
                    out = counter << 4;
                    out = out & 0xF0;
                }
                else {
                    out = 0x91;
                }
                PORTC = out;
            }
        }
    }
}
```

AVR:

```
.include "m16def.inc"
.DEF temp = r16
.DEF counter = r17
.DEF input = r18
.DEF output = r19
.DEF a1 = r20
.DEF a0 = r21
.DEF reset = r22

start:
    clr temp
    out DDRA,temp
    ser temp
    out DDRC,temp

    ldi counter,0x1E
    ldi output,0x91
    out PORTC,output

loop:
    in input,PINA
    andi input,0x07
    mov reset,input
    andi reset,0x01
    cpi reset,0x01
```

```
    breq resetcounter

    mov a0,input
    mov a1,input
    andi a0,0x02
    andi a1,0x04

    cpi a0,0x02
    brneq carinput

    cpi a1,0x04
    brneq caroutput
    rjmp loop

carinput:
    dec counter
    cpi counter,0x0A
    brlt lessthanten
    ldi output,0x91
    out PORTC,output
    rjmp loop

caroutput:
    inc counter
    cpi counter,0x0A
    brlt lessthanten
    ldi output,0x91
    out PORTC,output
    rjmp loop

lessthanten:
    mov output,counter
    lsl
    lsl
    lsl
    lsl
    andi output,0xF0
    out PORTC,output
    rjmp loop

resetcounter:
    ldi counter,0x1E
    ldi output,0x91
    out PORTC,output
    rjmp loop
end:
    .exit
```