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

ΑΘΗΝΑ 21. 9. 2021

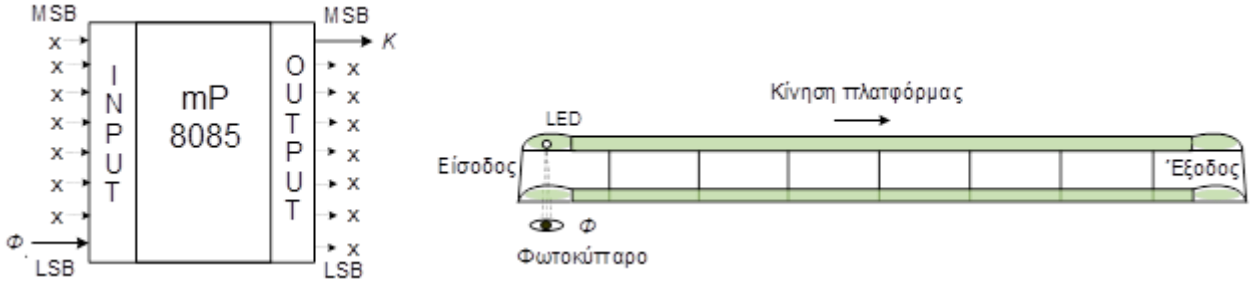
ΓΡΑΠΤΗ ΕΞΕΤΑΣΗ ΣΤΟ ΜΑΘΗΜΑ "***Συστήματα Μικροϋπολογιστών***"

(ΘΕΜΑ 1ο – ΣΥΝΟΛΟ 3.5 Μονάδες)

Έναρξη 11:30 - ΔΙΑΡΚΕΙΑ 50' + 10' Παράδοση: 12:30'

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: Κυριακόπουλος Γεώργιος – el 18153 . . . . . .

ΘΕΜΑ 1α: (2 ΜΟΝΑΔΕΣ): Δίνεται μΥ-Σ που διαθέτει δύο 8-bit θύρες: μία εισόδου (διεύθ. 20^{HEX}) και μία εξόδου (διεύθ. 30^{HEX}). Να γραφεί πρόγραμμα assembly σε 8085 που να υλοποιηθεί ένα σύστημα οδήγησης κυλιόμενης πλατφόρμα μονής κατεύθυνσης η οποία να ενεργοποιείται από το φωτοκύτταρο *Φ*. Συγκεκριμένα, αν ένας επιβάτης εισέρχεται στην πλατφόρμα, όταν είναι ακίνητη, διακόπτει δέσμη φωτός (γίνεται *Φ*=0) και τότε τίθεται σε κίνηση η πλατφόρμα με το σήμα εξόδου *K* (για *K*=1 έχουμε κίνηση). Η κίνηση να σταματά ~10 sec μετά την τελευταία διακοπή του φωτοκυττάρου *Φ* (χρόνος ικανός για να αδειάσει η πλατφόρμα από επιβάτες). Μπορείτε να κάνετε χρήση της ρουτίνας χρονοκαθυστέρησης DSEC των 50 msec.



ΘΕΜΑ 1β: (0.8 ΜΟΝΑΔΕΣ): Απαντήστε στα παρακάτω ερωτήματα (σύντομα και αιτιολογημένα):

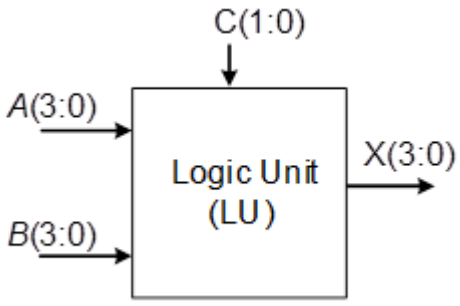
(i) Δώστε τη μακροεντολή *MIN* που μετακινεί τον ελάχιστον των καταχωρητών *B, D, E* στον καταχωρητή *A*, χωρίς να επηρεάζεται η τιμή των καταχωρητών (πλην φυσικά του A). (0.4 ΜΟΝΑΔΕΣ)

(ii) Εξηγήστε τη λειτουργική διαφορά των καθυστερήσεων που προκαλούνται μέσω ρουτινών χρονοκαθυστέρησης και μέσω μετρητών-χρονιστών (πλεονεκτήματα, μειονεκτήματα). (0.2 ΜΟΝΑΔΕΣ)

(iii) Πώς λειτουργούν οι εντολές σχετικού άλματος και κλήσης ρουτίνας; Εξηγήστε την χρησιμότητά τους. (0.2 ΜΟΝΑΔΕΣ)

ΘΕΜΑ 1γ: (0.7 ΜΟΝΑΔΕΣ): Να δοθεί το κύκλωμα (σχηματικό διάγραμμα) και η δομική περιγραφή σε Verilog της μονάδας LU που η λειτουργία της φαίνεται στο διπλανό πίνακα και σχήμα. Μπορείτε να κάνετε χρήση των βασικών πυλών: XOR(*x,a,b*), OR(*x,a,b*), AND(*x,a,b*) και INV(*x,a*) θεωρώντας τις μεταβλητές *a* και *b* ως εισόδους. Το σύμβολο '∨' δηλώνει την πράξη OR ενώ το '·' την πράξη AND. Υποθέτουμε ότι οι μεταβλητές *A, B* και *X* είναι των 4-bit. Η λέξη ελέγχου *C*(1:0) είναι των 2-bit.
Επίσης να δοθεί η περιγραφή Verilog του ίδιου κυκλώματος σε μορφή ροής δεδομένων ή σε μοντελοποίηση συμπεριφοράς.

C_1C_0	Έξοδος X
00	$A \vee B$
01	'1111'
10	$A \cdot B$
11	'0000'



Θέμα 1α:

```
START:
    IN 20H
    ANI 01H
    CPI 01H
    JZ START

    MVI A,80H
    OUT 30H

INITIALIZE:
    MVI B,C8H

LOOP:
    CALL DSEC

    IN 20H
    ANI 01H
    CPI 00H
    JZ INITIALIZE

    DCR B
    JNZ LOOP

    MVI A,00H
    OUT 30H
    JMP START
```

Θέμα 1β1:

```
MIN MACRO
    PUSH PSW
    PUSH C
    MOV A,B
    CMP D
    JC CMPBE
    MOV A,E
    CMP D
    JC FINALE
    MOV C,D
    JMP END
CMPBE:
    CMP E
    JC FINALB
    MOV C,E
    JMP END
FINALB:
    MOV C,B
    JMP END
FINALE:
    MOV C,E
END:
    POP PSW
    MOV A,C
    POP C
```

```
ENDM
```

Θέμα 1γ:

```
module mux2x1 (
    output [3:0] out,
    input [3:0] C, D,
    input select
);

tri out;
bufif0(out, C, select);
bufif1(out, D, select);

endmodule

module LU (
    output [3:0] X,
    input [3:0] A, B,
    input [1:0] C
);
wire w1, w2, w3, w4;

or G1(w1, A, B);
and G2(w2, A, B);
mux2x1 C1(w3, 4'b1111, 4'b0000, C[1]);
mux2x1 C2(w4, w1, w2, C[1]);
mux2x1 C3(X, w3, w4, C[0]);
endmodule
```

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

ΑΘΗΝΑ 21. 9. 2021

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

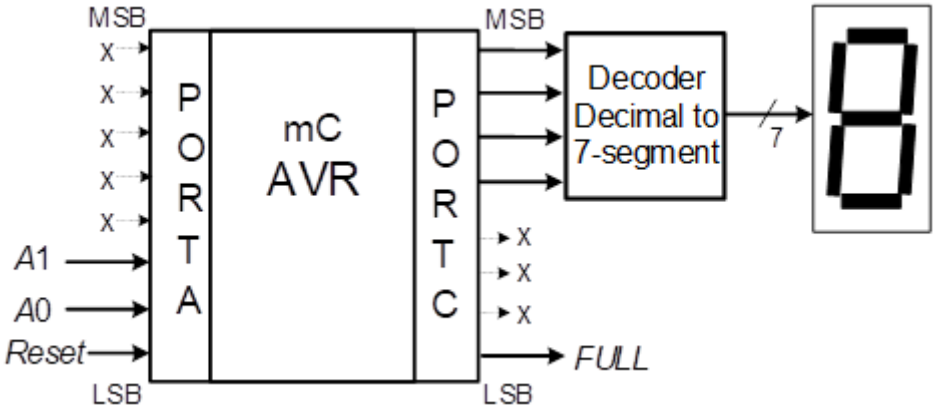
(ΘΕΜΑ 2ο – ΣΥΝΟΛΟ 4.5 Μονάδες)

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
```

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

ΑΘΗΝΑ 21. 9. 2021

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

(ΘΕΜΑ 3ο – ΣΥΝΟΛΟ 2 Μονάδες)

Έναρξη 13:40' - ΔΙΑΡΚΕΙΑ 30' + 10' Παράδοση: 14:20'

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: Κυριακόπουλος Γεώργιος – el18153 . . . . . .

ΘΕΜΑ 3ο: (2 ΜΟΝΑΔΕΣ): Σε ένα προσωπικό υπολογιστή, να γραφεί πρόγραμμα σε Assembly μΕ 80x86 που να δέχεται από το πληκτρολόγιο τέσσερις (4) δεκαεξαδικούς αριθμούς (H_3, H_2, H_1, H_0 με τη σειρά αυτή) για να αποτελέσουν δυο μονοψήφιους και ένα διψήφιο δεκαεξαδικό αριθμό και να κάνει τον εξής υπολογισμό:
$P = H_3 + H_2 \times (H_1 \times 16 + H_0)$. Το πρόγραμμα τυπώνει στην οθόνη τα μηνύματα εισόδου και τους εισαγόμενους αριθμούς. Όταν συμπληρωθούν 4 έγκυροι δεκαεξαδικοί αριθμοί να αναμένει τον χαρακτήρα 'Η' και μετά να τυπώνει το αποτέλεσμα σε δεκαδική μορφή 3 ψηφίων αν δεν υπάρχει υπερχείλιση (δηλ. <1000). Αν όμως υπάρχει υπερχείλιση (>999) τότε να τυπώνει το μήνυμα OVERF, αυστηρά όπως φαίνεται παρακάτω:

DOSE 1ο ARITHMO = *D*
DOSE 2ο ARITHMO = *7*
DOSE 3ο ARITHMO = *5C*
APOTELESMA = *657* ή APOTELESMA = YPERX

Να θεωρήσετε δεδομένες τις μακροεντολές (σελ. 361-2, 373) του βιβλίου και μπορείτε να κάνετε χρήση των ρουτινών DEC_KEYB και PRINT_HEX χωρίς να συμπεριλάβετε τον κώδικά τους. Για την διευκόλυνσή σας, δίνονται οι πρώτες εντολές που αποτελούν τον 'σκελετό' του ζητούμενου προγράμματος.

ΑΠΑΝΤΗΣΗ

```
INCLUDE     MACROS

DATA_SEG    SEGMENT
  MSG1      DB 0AH,0DH, 'DOSE 1ο ARITHMO =  $'
  MSG2      DB 0AH,0DH, 'DOSE 2ο ARITHMO = $'
  MSG3      DB 0AH,0DH, 'DOSE 3ο ARITHMO = $'
  MSG4      DB 0AH,0DH, 'APOTELESMA = $'
DATA_SEG    ENDS

CODE_SEG    SEGMENT
  ASSUME   CS:CODE_SEG, DS:DATA_SEG

MAIN  PROC  FAR
      MOV   AX, DATA_SEG
      MOV   DS, AX
. . .
```

```
INCLUDE MACROS

DATA_SEG SEGMENT
    MSG1 DB 0AH,0DH, 'DOSE 1o ARITHMO = $'
    MSG2 DB 0AH,0DH, 'DOSE 2o ARITHMO = $'
    MSG3 DB 0AH,0DH, 'DOSE 3o ARITHMO = $'
    MSG4 DB 0AH,0DH, 'APOTELESMA = $'
    MSG5 DB 0AH,0DH, 'APOTELESMA = OVERF$'
DATA_SEG ENDS

CODE_SEG SEGMENT
    ASSUME CS:CODE_SEG, DS:DATA_SEG

MAIN PROC FAR
    MOV AX, DATA_SEG
    MOV DS, AX

START:
    PRINT_STR MSG1
    CALL HEX_KEYB
    CMP AL,'Q'
    JE QUIT

    MOV BL, AL          ; BL = H3

    PRINT_STR MSG2
    CALL HEX_KEYB
    CMP AL,'Q'
    JE QUIT

    MOV CL, AL          ; CL = H2

    PRINT_STR MSG3
    CALL HEX_KEYB
    CMP AL,'Q'
    JE QUIT

    AND AL,0FH
    ROL AL, 1
    ROL AL, 1
    ROL AL, 1
    ROL AL, 1

    MOV DL, AL          ; DL = H1*16

    CALL HEX_KEYB
    CMP AL,'Q'
    JE QUIT

    ADD AL, DL          ; AL = H1*16 + H0
    MOV AH, 0
    MUL CL              ; AX = H2*(H1*16 + H0)

    MOV BH, 0
    ADD AX, BX          ; AX = H3 + H2*(H1*16 + H0)
```

```
    MOV DX, AX

    WAIT_H:
        READ
        CMP AL, 'H'
        JNE WAIT_H

    CMP DX, 999
    JG OVERF

    MOV AX, DX
    MOV DL, AH

    CALL PRINT_HEX      ; MSB OF 3 DIGITS

    MOV DL, AL
    ROR DL, 1
    ROR DL, 1
    ROR DL, 1
    ROR DL, 1
    AND DL, 0FH
    CALL PRINT_HEX      ; 2ND DIGIT

    MOV DL, AL
    AND DL, 0FH
    CALL PRINT_HEX      ; 3RD DIGIT
    JMP START

    OVERF:
        PRINT_STR MSG5
        JMP START

    QUIT:
        EXIT
MAIN ENDP
```