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

ΑΘΗΝΑ 25. 6. 2021

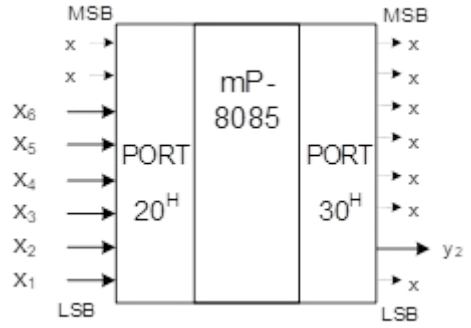
ΓΡΑΠΤΗ ΕΞΕΤΑΣΗ ΣΤΟ ΜΑΘΗΜΑ "***Συστήματα Μικροϋπολογιστών***"

(ΘΕΜΑ 1ο – ΣΥΝΟΛΟ 3.5 Μονάδες)

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

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: ΓΙΩΡΓΟΣ ΚΥΡΙΑΚΟΠΟΥΛΟΣ - el18153

ΘΕΜΑ 1α: (1.5 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 υπολογίζει τη λογική συνάρτηση $y_2 = x_1 \cdot x_2 \cdot x_3 \cdot x_4 + x_5 \cdot x_6$.



ΘΕΜΑ 1β: (1.3 ΜΟΝΑΔΑ): Απαντήστε στα παρακάτω ερωτήματα (σύντομα και αιτιολογημένα):

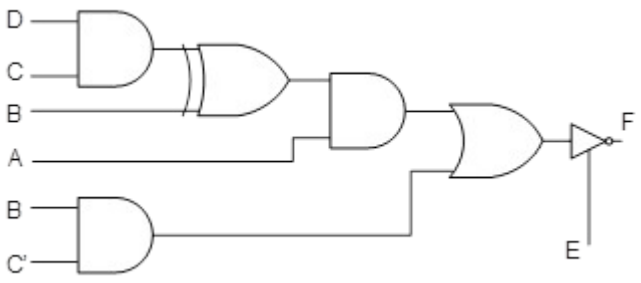
(i) Δώστε τη μακροεντολή *MOVING n* που μετακινεί το περιεχόμενο ενός εκ των καταχωρητών *B, C, D, E* στον καταχωρητή *A*, για *n* = 0, 1, 2, 3 αντίστοιχα. Για άλλη τιμή του *n* να μην κάνει καμία λειτουργία. (0.5 ΜΟΝΑΔΕΣ)

(ii) Να αναφέρετε τα πλεονεκτήματα που παρέχουν οι διακοπές στα μΥ-Σ. Τί πρόβλημα μπορεί να προκύψει αν μια διακοπή προκαλείται από παλμό μεγάλης ή και μικρής διάρκειας και γιατί; Να προτείνετε λύσεις για την αποφυγή των ενδεχόμενων προβλημάτων. (0.4 ΜΟΝΑΔΕΣ)

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

(iv) Πότε είναι χρήσιμη και πλεονεκτική η χρήση των Μακροεντολών σε σχέση με τις Ρουτίνες; (0.2 ΜΟΝΑΔΕΣ)

ΘΕΜΑ 1γ: (0.7 ΜΟΝΑΔΕΣ):
Δώστε την περιγραφή Verilog του παρακάτω κυκλώματος σε **επίπεδο πυλών** και σε μορφή **ροής δεδομένων**.



1α)

```
START:
   LDA 2000H
   MOV B, A
   MOV C, A
   RAR
   ANA C
   MOV C, A

   MOV A, B
   RAR
   RAR
   ANA C
   MOV C, A

   MOV A, B
   RAR
   RAR
   RAR
   ANA C
   MOV C, A       ; C(LSB) = x1x2x3x4

   MOV A, B
   RAR
   RAR
   RAR
   RAR
   MOV D, A
   RAR
   ANA D          ; A(LSB) = x5x6

   ORA C          ; A(LSB) = x1x2x3x4 + x5x6
   ANI 01H
   RAL

   STA 3000H

   JMP START

   END
```

1β1)

```
MOVING MACRO N
   MVI A, N       ; We only use A and return it so nothing to push/pop
   CPI 00H
   JZ MOVBA
   CPI 01H
   JZ MOVCA
   CPI 02H
   JZ MOVDA
   CPI 03H
   JZ MOVEA
   JMP EXIT
MOVBA:
   MOV A, B
   JMP EXIT
MOVCA:
   MOV A,
   JMP EXIT
MOVDA:
```

```
   MOV A,D
   JMP EXIT
MOVEA:
   MOV A,E
   JMP EXIT
EXIT:  ENDM
```

1γ)

```
module verilog (F, A, B, C, D, E);
   output F;
   input A, B, C, D, E;
   wire w1, w2, w3, w4, w5, w6;

   not G1 (w1, C);
   and
      G2 (w2, D, C),
      G3 (w3, B, w1);

   xor G4 (w4, w2, B);
   and G5 (w5, A, w4);
   or G6 (w6, w5, w3);
   notif(F, w6, E);
endmodule
```

Μοντελοποίησης

```
module verilog (F, A, B, C, D, E);
   output F;
   input A, B, C, D, E;

   assign F = (E)?(~((((D&C)^B)&A)|(B&(~C)))):1'bz;
endmodule
```

1β4)

Με τη χρήση ρουτινών έχουμε οικονομία στη μνήμη. Επίσης με χρήση μακροεντολών οι εντολές τους εισέρχονται στο κυρίως πρόγραμμα πριν το χρόνο μετάφρασης ενώ με τις ρουτίνες αυτό γίνεται κατά την εκτέλεση του προγράμματος. Από άποψη ταχύτητας οι μακροεντολές δίνουν καλύτερα αποτελέσματα αφού δεν επιβαρύνουν το πρόγραμμα με εντολές κλήσης και επιστροφής ρουτινών.

1β2)
Διαχείριση I/O με καλύτερο τρόπο.
Καλύτερη εκμετάλλευση του χρόνου του με δηλαδή υπολογιστικής ισχύος.
Ασύγχρονη ανταπόκριση.

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

ΑΘΗΝΑ 25. 6. 2021

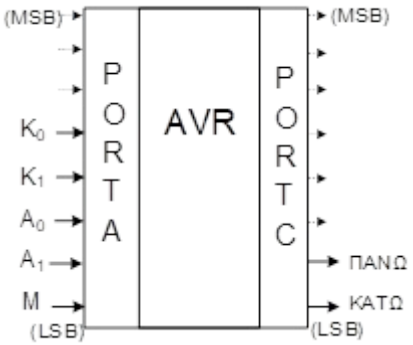
ΓΡΑΠΤΗ ΕΞΕΤΑΣΗ ΣΤΟ ΜΑΘΗΜΑ "***Συστήματα Μικροϋπολογιστών***"

(ΘΕΜΑ 2ο – ΣΥΝΟΛΟ 4.5 Μονάδες)

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

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: ΓΕΩΡΓΙΟΣ ΚΥΡΙΑΚΟΠΟΥΛΟΣ - el18153

ΘΕΜΑ 2ο: (4.5 ΜΟΝΑΔΕΣ): Σε ένα μικροελεγκτή AVR Mega16 που αξιοποιεί μία θύρα εισόδου και μία εξόδου, όπως φαίνεται στο διπλανό σχήμα, να υλοποιηθεί ένα σύστημα οδήγησης ενός ανελκυστήρα δυο θέσεων (ισογείου και 1ου ορόφου). Η κίνηση προς το ισόγειο ή τον 1ο όροφο ελέγχεται από τους εξωτερικούς διακόπτες (Push-Buttons) *K*0 και *K*1 αντίστοιχα καθώς και από έναν εσωτερικό διακόπτη (Push-Button) *M*. Για να δοθεί εντολή από τους διακόπτες αυτούς, προϋπόθεση είναι το βαγόνι να είναι σταματημένο στο ισόγειο ή στον 1ο όροφο. Όταν κινείται πρέπει να σταματάει από το πρόγραμμα με βάση τους αισθητήρες *A*0 και *A*1 που είναι τερματικοί διακόπτες και οι οποίοι δίνουν λογικό 1 αυτόματα όταν ο θάλαμος φτάνει στο ισόγειο ή στον 1ο όροφο αντίστοιχα. Υποθέτουμε ότι κατά την εκκίνηση του συστήματος, ο θάλαμος πρέπει να βρίσκεται στο ισόγειο, αλλιώς πριν δεχτεί οποιαδήποτε εντολή να μεταφέρεται σε αυτή τη θέση αυτόματα.



Αναλυτικά, αν ο θάλαμος φτάσει στο ισόγειο, τότε πρέπει να σταματάει η κίνησή του και να ελέγχονται οι διακόπτες *K*1 και *M*. Αν ένας από αυτούς είναι ενεργοποιημένος (=1) τότε έχουμε κίνηση προς τα πάνω. Αντίστοιχα αν ο θάλαμος φτάσει στον 1ο όροφο, τότε πρέπει να σταματάει η κίνησή του και να ελέγχονται οι διακόπτες *K*0 και *M*. Αν ένας από αυτούς είναι ενεργοποιημένος (=1) τότε έχουμε κίνηση προς τα κάτω. Δώστε το αντίστοιχο πρόγραμμα σε assembly και σε C.

(Assembly: 2.5 ΜΟΝΑΔΕΣ και C: 2 ΜΟΝΑΔΕΣ)

C:

```
int main(void){
   DDRC=0xFF;      // output on PORTC
   DDRA=0x00;      // input on PORTA

   while((PINA & 0x04) == 0){      // if not on floor go there (output down = 1)
      PORTC = 1;
   }
   while(1) {
      if ((PINA & 0x01) == 1){    // if M is pressed
         if ((PINA & 0x10) == 16){    // must move to floor
            while ((PINA & 0x04) != 4){
               PORTC = 1;    // if not on floor move till you are there
            }
         }
         if ((PINA & 0x08) == 8){    // must move to 1st floor
            while ((PINA & 0x02) != 2){
               PORTC = 2;    // if not on 1st floor move till you are there
            }
         }
      }
   }
}
```

Assembly:

```
include “m16def.inc”
reset:
   ldi r24, low(RAMEND)
   out SPL, r24
   ldi r24, high(RAMEND)
   out SPH, r24
   ser r24
   out DDRC, r24
   clr r24
   out DDRA, r24
   ldi r24, 0b11111111
   out PORTA , r24

   in r24, PINA
   and r24, 0b00000100   ; check a0
   cpi r24, 0            ; if not on ground
   jnz mov_ground        ; go to ground

main:
   ldi r22,0b00000000
   out PORTC, r22        ; stop moving
   in r24, PINA
   mov r23,r24
   and r24, 0b00000001   ; if moving
   cpi r24, 0
   jnz moving            ; check where to
   jmp main              ; else loop

moving:
   mov r24,r23
   and r24, 0b00001000    ; move to 1st floor
   cpi r24, 0
   jnz mov_floor
   mov r24,r23
   and r24, 0b00010000    ; move to ground
   cpi r24, 0
   jnz mov_ground
   jmp main

mov_ground:
   in r24,PINA
   and r24, 0b00000100    ; if on ground loop main
   jnz main
   ldi r22,0b00000001
   out PORTC, r22         ; else move to ground and check again
   jmp mov_ground

mov_floor:
   in r24,PINA
   and r24, 0b00000010    ; if on floor loop main
   jnz main
   ldi r22,0b00000010     ; else move to floor and check again
   out PORTC, r22
   jmp mov_floor
```

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

ΑΘΗΝΑ 25. 6. 2021

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

(ΘΕΜΑ 3ο – ΣΥΝΟΛΟ 2 Μονάδες)

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

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: ΓΙΩΡΓΟΣ ΚΥΡΙΑΚΟΠΟΥΛΟΣ - el18153

ΘΕΜΑ 3ο: (2 ΜΟΝΑΔΕΣ): Σε ένα προσωπικό υπολογιστή, να γραφεί πρόγραμμα σε Assembly μΕ 80x86 που να δέχεται από το πληκτρολόγιο τέσσερις (4) δεκαδικούς αριθμούς (D_3, D_2, D_1, D_0 με τη σειρά αυτή) για να αποτελέσουν ένα διψήφιο και δυο μονοψήφιους δεκαδικούς αριθμούς και να κάνει τον εξής υπολογισμό: $P=(D_3\times 10 + D_2)\times(D_1 + D_0)$. Το πρόγραμμα τυπώνει στην οθόνη τα μηνύματα εισόδου και τους εισαγόμενους αριθμούς. Όταν συμπληρωθούν 4 έγκυροι δεκαδικοί αριθμοί να αναμένει τον χαρακτήρα 'h' και μετά να τυπώνει το αποτέλεσμα σε δεκαεξαδική μορφή 3 ψηφίων αν είναι <400Hex, αλλιώς το μήνυμα yperx, αυστηρά όπως φαίνεται παρακάτω:

DOSE 1ο ARITHMO = *58*
DOSE 2ο ARITHMO = *7*
DOSE 3ο ARITHMO = *9*
APOTELESMA = *3A0* 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 1ο ARITHMO = $'
       MSG2 DB 0AH,0DH, 'DOSE 2ο ARITHMO = $'
  MSG3      DB 0AH,0DH, 'DOSE 3ο ARITHMO = $'
  MSG4      DB 0AH,0DH, 'APOTELESMA = $'
  MSG5   DB 0AH,0DH, 'APOTELESMA = yperx'
DATA_SEG  ENDS
```

```
CODE_SEG  SEGMENT
        ASSUME   CS:CODE_SEG, DS:DATA_SEG

MAIN PROC FAR
   MOV        AX, DATA_SEG
   MOV        DS, AX

ADDR1:
   PRINT_STR MSG1
   CALL DEC_KEYB
   CMP AL, 'Q'
   JE QUITMAIN
   MOV BL, 10
   MUL BL          ; AL = D3*10
   MOV BL, AL      ; BL = D3*10

   CALL DEC_KEYB
   CMP AL, 'Q'
   JE QUITMAIN
   ADD AL, BL      ; AL = D3*10 + D2
   MOV BL, AL      ; BL = D3*10 + D2

   PRINT_STR MSG2
   CALL DEC_KEYB
   CMP AL, 'Q'
   JE QUITMAIN
   MOV CL, AL      ; CL = D1

   CALL DEC_KEYB
   CMP AL, 'Q'
   JE QUITMAIN
   ADD AL, CL      ; AL = D1 + D0

   MUL BL          ; AL = (D3*10 + D2) * (D1 + D0)

   CPM 0400H, AX
   JGE YPERX

CHECK:
   CALL WAIT_H   ; Wait for H
   CMP AL, 'H'
   JE ADDR2
   JMP CHECK

ADDR2:
   ROL AX, 1       ; 4 left rotate for msbs to become lsbs
   ROL AX, 1
   ROL AX, 1
   ROL AX, 1
   MOV DL, AL        ; keep 4 lsbs of dl that are the msbs of exit
   AND DL, 0FH
   PUSH AX
   PRINT_STR MSG4
   CALL PRINT_HEX
   POP AX
   LOOP ADDR2
   JMP ADDR1

YPERX:
   PRINT_STR MSG5
   JMP ADDR1
```

```
QUITMAIN:
    EXIT

    MAIN ENDP
    CODE_SEG ENDS
    END MAIN


WAIT_H PROC NEAR
    IGNORE:
    READ
    CMP AL, 'Q'
    JE QUIT
    CMP AL, 'H'
    JNE IGNORE
    QUIT:
    RET
WAIT_H ENDP
CODE_SEG ENDS
```