2020-2021

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ : ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΦΟΙΤΗΤΕΣ-ΕΠΙΜΕΛΕΙΑ: ΒΟΪΚΟΣ
ΣΤΕΦΑΝΟΣ ΚΑΙ ΑΛΚΙΒΙΑΔΗΣ ΜΙΧΑΛΙΤΣΗΣ

[Συστήματα Μικροϋπολογιστών]

Απαντήσεις των ασκήσεων στην 3η ομάδα ασκήσεων
2 ασκήσεις που ακολουθούν είναι όλες ασκήσεις προσομοίωσης και εν συνεχεία, ακολουθούν θεωρητικές ασκήσεις

Στοιχεία φοιτητή

Ονοματεπώνυμο: Στέφανος Βόικος και Αλκιβιάδης Μιχαλίτσης

Α.Μ: el18162 και el18868 Ακαδημαϊκό εξάμηνο: 6ο

Σχολή: Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών (Η.Μ.Μ.Υ)

ΑΠΑΝΤΗΣΕΙΣ ΤΩΝ ΑΣΚΗΣΕΩΝ ΤΗΣ ΟΜΑΔΑΣ 1 ΣΤΟ ΜΑΘΗΜΑ «ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ»

1Η ΑΣΚΗΣΗ

```
        IN 10H
        LXI B,0064H         ;SMALL DELAY 100ms
        LXI D,0B00H         ;STDM

        MVI A,10H           ;SET SCREEN
        STA 0B00H           ;
        STA 0B01H           ;
        STA 0B02H           ;
        STA 0B03H           ;
        STA 0B04H           ;
        STA 0B05H           ;

        MVI A,0DH           ;RST 6.5 SWITCH
        SIM                 ;
        EI                  ;
START:
        JMP START

INTR_ROUTINE:
        EI

        MVI A,00H           ;LEDS ON
        STA 3000H

        MVI A,05H           ;[59-50] secs
        STA 0B05H
        MVI A,09H
SEG1:
        STA 0B04H
        CALL SCREEN
        DCR A                       ;TIME
        CPI 00H             ;TIME EXPIRATION
        JNZ SEG1            ;CHECK TIME EXPIRATION
```

```
        CALL ZERO

        MVI A,04H       ;[49-40] secs
        STA 0B05H
        MVI A,09H
SEG2:
        STA 0B04H
        CALL SCREEN
        DCR A                   ;TIME
        CPI 00H         ;TIME EXPIRATION
        JNZ SEG2        ;CHECK TIME EXPIRATION
        CALL ZERO

        MVI A,03H       ;[39-30] secs
        STA 0B05H
        MVI A,09H
SEG3:
        STA 0B04H
        CALL SCREEN
        DCR A                   ;TIME
        CPI 00H         ;TIME EXPIRATION
        JNZ SEG3        ;CHECK TIME EXPIRATION
        CALL ZERO

        MVI A,02H       ;[29-20] secs
        STA 0B05H
        MVI A,09H
SEG4:
        STA 0B04H
        CALL SCREEN
        DCR A                   ;TIME
        CPI 00H         ;TIME EXPIRATION
        JNZ SEG4        ;CHECK TIME EXPIRATION
        CALL ZERO

        MVI A,01H       ;[19-10] secs
        STA 0B05H
        MVI A,09H
SEG5:
        STA 0B04H
        CALL SCREEN
        DCR A                   ;TIME
        CPI 00H         ;TIME EXPIRATION
        JNZ SEG5        ;CHECK TIME EXPIRATION
        CALL ZERO

        MVI A,00H       ;[9-0] secs
        STA 0B05H
        MVI A,09H
SEG6:
```

```
        STA 0B04H
        CALL SCREEN
        DCR A                   ;TIME
        CPI 00H             ;TIME EXPIRATION
        JNZ SEG6            ;CHECK TIME EXPIRATION
        CALL ZERO

        MVI A,FFH           ;LEDS OFF
        STA 3000H           ;

        RET

SCREEN:                 ;SCREEN REFRESH
        PUSH PSW
        PUSH H
        PUSH D
        PUSH B
        CALL STDM
        MVI A,0AH           ;10*DELAY=1sec
CONT:
        CALL DCD
        CALL DELB
        DCR A
        CPI 00H
        JNZ CONT
        POP B
        POP D
        POP H
        POP PSW
        RET

ZERO:                   ;ZERO AT THE MIDDLE SEGMENTS OF THE
SCREEN
        MVI A,00H
        STA 0B02H
        CALL SCREEN
        CALL DELB
        RET

        END
```

```
;|----------------------|
;|LD6|LD5|LD4|LD3|LD2|LD1|
;|B05|B04|B03|B02|B01|B00|
;|FLOW <----------------|
;|[0] [0] [1] [1] [0] [0]|
;|ARA THEL0UME B02 & B03 |
;|______________________|
```

2H ΑΣΚΗΣΗ

```
        IN 10H
        MVI D,0FH    ;K1
        MVI C,EFH    ;K2
        MVI L,FFH    ;AUXILIARY VALUE
        LXI B,0B00H ;STDM VALUE

        MVI A,10H    ;RESET SCREEN
        STA 0B00H    ;
        STA 0B01H    ;
        STA 0B02H    ;
        STA 0B03H    ;
        STA 0B04H    ;
        STA 0B05H    ;

        MVI A,0DH    ;ENABLE RST 6.5 SWITCH
        SIM          ;
        EI           ;
START:
        STA 3000H    ;LEDS OUTPUT
        MOV A,L
        CALL OTHONI        ;SCREEN OUTPUT

        CMP D              ;COMPARE WITH K1
        JC ONE             ;A<K1
        JZ ONE             ;A=K1
        CMP E        ;COMPARE WITH K2
        JC TWO             ;A<K2
        JZ TWO             ;A=K2
        MVI A,FBH    ;[0,K1]-->FIRST LED
        JMP START    ;
ONE:
        MVI A,FEH    ;(K1,K2]-->SECOND LED
        JMP START    ;
TWO:
        MVI A,FDH    ;(K2,FFH]-->THIRD LED
        JMP START

OTHONI:              ;REFRESH SCREEN
        PUSH PSW
        PUSH H
        PUSH B
        PUSH D
        CALL STDM
        CALL DCD
        POP D
        POP B
        POP H
        POP PSW
```

```
        RET

INTR_ROUTINE:
        PUSH PSW

        CALL KIND   ;KEYBOARD INPUT
        STA 0B00H   ;LSB
        MOV L,A     ;RECENT SAVE TILL MSB
        CALL KIND   ;
        STA 0B01H   ;MSB

        RLC         ;SLIDING
        RLC         ;SLIDING
        RLC         ;SLIDING
        RLC         ;SLIDING
        ORA L       ;LSB,MSB
        MOV L,A

        POP PSW
        EI
        RET


        END
```

3^{H} ΑΣΚΗΣΗ

;Ερώτημα a)

```
SWAP MACRO Nible Q

        PUSH B

        PUSH D

        PUSH H

        MOV A,Nible Q

        CMP B

        JNZ CONT1 ;

        MOV A,B
```

```
CONT1: CMP C
        JNZ CONT2 ;
        MOV A,C
CONT2: CMP D
        JNZ CONT3 ;
        MOV A,D
CONT3: CMP E
        JNZ CONT4 ;
        MOV A,E
CONT4: CMP H
        JNZ CONT5 ;
        MOV A,H
CONT5: CMP L
        JNZ CONT6 ;
        MOV A,L
CONT6:  ;To A kataligei me to mikrotero noumero edw
        CMP C
        JNZ CONT7 ;
        MOV Nible Q,C
CONT7: CMP D
        JNZ CONT8 ;
        MOV Nible Q,D
CONT8: CMP E
        JNZ CONT9 ;
```

```
        MOV Nible Q,E
CONT9: CMP H
        JNZ CONT10 ;
        MOV Nible Q,H
CONT10: CMP L
        JNZ CONT11 ;
        MOV Nible Q,L
CONT11: MOV A,Nible Q
        MOV Nible Q,R
        MOV R,A
        POP PSW
        POP D
        POP H
        POP B
ENDM

;Ερώτημα b)

FILL MACRO RP,X,K
        PUSH PSW
        PUSH B
        PUSH H
        LXI H,RP
        MVI A,X ;the length of the area that we are going to fill with
```

```
        MVI B,K
CONT:   MOV M,B ;store number in memory
        INX H ;increase HL for next loop
        DCR A ;next number
        CPI 00H
        JNZ CONT ;continue if A is not 0
        POP H
        POP B
        POP PSW
ENDM

;Ερώτημα c)

RHLR MACRO n
        MOV A,n
        MOV B,H
NPERIH: RAR ;CY contains H's MSB,whatever it conteins
        DCR A ;next number,decrease
        CPI 00H
        JNZ NPERIH
        MOV H,B ;H's LSB is now old CY value (H is ready)
        MOV B,L
        MOV A,n
NPERIL: RAR ; CY contains Q's old MSB (CY is ready in the last loop of NPERIL.When A = 0)
```

```
        DCR A ;next number,decrease
        CPI 00H
        JNZ NPERIL
        MOV L,B ;And now,L is ready
ENDM
```

(Δεν χρησιμοποιήθηκε PUSH PSW, γιατί ο CY είναι flag και αν κάναμε POP PSW θα αλλοιωνόταν το αποτέλεσμα.)

$4^{Η}$ ΑΣΚΗΣΗ

Στην $4^{η}$ άσκηση, κάνω την επίλυση χειρόγραφα, τις απαντήσεις τις παραθέτω και είναι οι εξής:

Η RST 7.5 είναι hardware διακοπή και θα πάρει προτεραιότητα αντί της εκτέλεσης του υπόλοιπου προγράμματος (με την προϋπόθεση σαφώς ότι επιτρέπονται οι διακοπές και υπάρχει η κατάλληλη μάσκα).

Επομένως, θα συμβούν τα παρακάτω με αυτή τη σειρά:

1. Θα ολοκληρωθεί η εκτέλεση της τρέχουσας εντολής CALL 0880H και θα έχουμε (PC) = 0880H.

2. Θα σωθεί στη στοίβα ο PC, δηλαδή η τιμή 0880H, στις θέσεις μνήμης ((SP)-1) και ((SP)-2).

Συγκεκριμένα θα έχουμε (2FFFH) = 08H και (2FFEH) = 00H.

3. Θα αυξηθεί το μέγεθος της στοίβας κατά 2, δηλαδή (SP) = 3000H.

4. Θα αποθηκευτεί στον PC η διεύθυνση της διακοπής που αναγνωρίστηκε, δηλαδή (PC) = 0117H ;0034H.

5. Αφού τελειώσει η εκτέλεση της ρουτίνας εξυπηρέτησης της διακοπής, θα επαναφερθεί η παλιά τιμή του PC από τη στοίβα και θα μειωθεί το μέγεθος της στοίβας κατά δύο (επομένως ο SP θα αυξηθεί κατά 2). Και άρα θα έχουμε (PC) =

0880H και (SP) = 3000H. Σε αυτό το σημείο συνεχίζεται η εκτέλεση των εντολών στη διεύθυνση του JMP.

$5^{Η}$ ΑΣΚΗΣΗ

```
;Απάντηση στο α ερώτημα:

RST6.5: JMP INTRPT ;(via Data ready tube,make the interrupt)

START:  MVI D,40H ;64 (D will count how many inputs are remaining) (these are the total number of steps to do)
        MVI H,00H ;initialization, used for MSBs of sum
        MVI L,00H ;initialization, used for LSBs of sum
        MVI B,00H ;used for DAD later
        MVI A,1DH ;mask to allow RST 6.5
        SIM
RETURN: MOV A,D
        CPI 00H
        JZ FINISH
        EI
WAIT:   JMP WAIT
INTRPT: INX SP
        INX SP ;reduce stack by 2
        DCR D
        MOV A,D
```

```
        ANI 01H
        CPI 00H
        JNZ LSBS ;if D is odd then do LSBS (1st,3rd,5th move etc), else read 2nd part
of input
        IN 20H ;read MSBs
        ANI 0FH ;only X0-X3 matter
        RRC
        RRC
        RRC
        RRC ;move the MSBs in correct position
        MOV C,A ;temporarily store MSBs
        JMP RETURN
LSBS:   IN 20H ;read LSBs
        ANI 0FH ;only X0-X3 matter and it's true
        ADD C
        MOV C,A ;C now contains an 8-bit number
        DAD B ;add BC to HL (B is 0, C contains an 8-bit number)
        JMP RETURN
FINISH: DI
        DAD H ;adding HL to itself 4 times is the same as
        DAD H ;multiplying it by 16 (4 left rotates)
        DAD H ;H will contain the final 8-bit result
        DAD H ;L contains remaining bits (< 1)
        END
```

```
;Απάντηση στο β ερώτημα:
;Ίδια η λογική, απλώς τώρα έλεγχο θα κάνουμε με βάση όχι την διακοπή όπως
;είχαμε προηγουμένως, αλλά θα κάνουμε τώρα με βάση το X7. Έτσι, το πρόβλημα
;διαμορφώνεται ως εξής:

X7CHANGE:IN 20H ;Αναμονή για την επαναφορά του x7 σε 0
        ANI 80H
        JMP DATAREADY
START:  MVI D,40H ;64 (D will count how many inputs are remaining) (these are the total number of steps to do)
        MVI H,00H ;initialization, used for MSBs of sum
        MVI L,00H ;initialization, used for LSBs of sum
        MVI B,00H ;used for DAD later
        ;MVI A,1DH ;mask to allow RST 6.5 einai sxolio
        SIM
RETURN: MOV A,D
        CPI 00H
        JZ FINISH
        EI
WAIT:   ANI 80H ;Έλεγχος αν το x7 είναι 1 Here to ever see that instead of RST 6.5, now i check in wait
        JMP X7CHANGE ;I will jump to X7CHANGE in case that X7 changes from 0 to 1
        JMP WAIT ;loop if the X7 changes to 1 so as to jump to X7CHANGE
DATAREADY: INX SP
        INX SP ;reduce stack by 2
```

```
        DCR D
        MOV A,D
        ANI 01H
        CPI 00H
        JNZ LSBS ;if D is odd then do LSBS (1st,3rd,5th move etc), else read 2nd part
of input
        IN 20H ;read MSBs
        ANI 0FH ;only X0-X3 matter
        RRC
        RRC
        RRC
        RRC ;move the MSBs in correct position
        MOV C,A ;temporarily store MSBs
        JMP RETURN
LSBS:   IN 20H ;read LSBs
        ANI 0FH ;only X0-X3 matter and it's true
        ADD C
        MOV C,A ;C now contains an 8-bit number
        DAD B ;add BC to HL (B is 0, C contains an 8-bit number)
        JMP RETURN
FINISH: DI
        DAD H ;adding HL to itself 4 times is the same as
        DAD H ;multiplying it by 16 (4 left rotates)
        DAD H ;H will contain the final 8-bit result
        DAD H ;
```

END