ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ

3η Εργαστηριακή Άσκηση

Ακαδημαϊκό έτος 2020-2021

Χριστίνα Προεστάκη | ΑΜ : 03118877

Νικόλαος Μπέλλος | ΑΜ : 03118183

1η Άσκηση

Κώδικας ASSEMBLY για 8085

```
; === INITIALIZE DISPLAY (BLANK) ===;
      IN 10H            ;disables memory protection
      LXI H,0AA0H       ;initial memory address of display digits
      MVI A,06H         ;sets counter (6)
LOOP_BLANK:
      MVI M,10H         ;blank value to display position
      INX H             ;increase display digit position
      DCR A             ;decrease counter
      CPI 00H           ;check if counter is zero
      JNZ LOOP_BLANK

; === ADD RST6.5 TO INTERRUPT MASK ===
      MVI A,0DH
      SIM
      EI

; === DISPLAY ZEROS UNTIL INTERRUPT ===
INF:
      LXI H,0AA4H       ;load digit 0 to first 2 digits from left
      MVI M,00H
      INX H
      MVI M,00H
LOAD:
      CALL DISP
      JMP LOAD

; === 6-SEGMENT-DISPLAY ROUTINE ===
DISP:                   ;displays digits stored in 0AA0-0AA5
      PUSH D            ;stores D
      LXI D,0AA0H       ;first address of 6 continuous addresses
      CALL STDM
      CALL DCD
      POP D             ;restores D
```

```
      RET

; === INTERRUPT ROUTINE ===
INTR_ROUTINE:
      MVI E,3CH         ;E = 60(s) (global counter)
      LXI B,0032H       ;B = 500(ms)
      EI                ;enable interrupts

INIT:
      MVI D,0AH         ;D = 10 (seconds counter)
      CALL DECR_SEC

      MVI A,00H         ;A = 00000000
      STA 3000H         ;switch on all LEDs
L1:
      CALL DISP         ;display seconds remaining
      CALL DELB         ;500ms delay
      DCR D             ;decrease seconds counter
      JNZ L1            ;loop 10 times for hight refresh rate

      DCR E             ;decrease counter
      JNZ INIT

      MVI A,FFH         ;A = 11111111
      STA 3000H         ;switch off all LEDs
      JMP INF

; === REFRESH TIMER ===
DECR_SEC:
      PUSH B            ;store B (decimal digits)
      PUSH H            ;store H
      MVI B,FFH         ;B = 1111 1111
      MOV A,E           ;A = global counter

L2:
      INR B             ;B = B + 1 (increase value of B by one) - decimal
      SUI 0AH           ;A = A - 10 (decreases A by 10s) - units
      JNC L2            ;loop until A = 0
      ADI 0AH           ;Correct A so it is not negative

      LXI H,0AA4H       ;display
      MOV M,A           ;display (A) - units
      INX H
      MOV M,B           ;display (B) - decimals

      POP H             ;restore H
      POP B             ;restore B
```

```
        RET
        END
```

2η Άσκηση

```
        IN    10H
        MVI   A,10H       ;A is zero

        STA   0A00H       ;7-segment displays are zero
        STA   0A01H
        STA   0A02H
        STA   0A03H
        STA   0A04H
        STA   0A05H

        MVI   A,0DH       ;RST6.5 interrupt mask
        SIM
        EI                ;enable interrupts

INTR:
        DI
        PUSH  PSW
        CALL  KIND
        STA   0A00H
        RLC
        RLC
        RLC
        RLC               ;move to MSBs
        MOV   B,A         ;save MSBs in B
        CALL  KIND
        STA   0A01H
        ADD   B           ;MSBs and LSBs
        MOV   B,A         ;save in B
        PUSH  D           ;save D and E
        LXI   D,0A00H
        CALL  STDM
        CALL  DCD
        POP   D           ;restore D and E

        MOV   A,B
        INR   D
        CMP   D           ;if number < D
        JC    ONE         ;first area
        INR   E
        CMP   E           ;if number < E
        JC    TWO         ;second area
```

```
        JNC    THREE        ;if number > E third area
        POP    PSW
        EI
        RET

ONE:
        MVI    A,FEH
        STA    3000H
        RET

TWO:    MVI    A,FDH
        STA    3000H
        RET

THREE:MVI    A,FBH
        STA    3000H
        RET

LOOP1:
        JMP    LOOP1    ;wait next interrupt
END
```

3η Άσκηση

α) Μακροεντολή "SWAP Nible MACRO Q"

Εναλλάσει το χαμηλότερης αξίας HEX ψηφίο με το υψηλότερης των καταχωρητών γενικού σκοπού B, C, D, E, H και L καθώς και της θέσης μνήμης που 'δείχνει' ο διπλός καταχωρητής H-L

Κώδικας ASSEMBLY για 8085

```
SWAP Nible MACRO Q
        PUSH PSW
        MOV A,Q
        RLC
        RLC
        RLC
        RLC
        MOV Q,A
        MOV A,M

        RRC
        RRC
        RRC
        RRC
        RRC
        RRC
        RRC
```

```
        RRC
        MOV M,A
        POP PSW
ENDM
```

β) Μακροεντολή "FILL RP, X, K" σε 8085

Γεμίζει ένα τμήμα της μνήμης με μία σταθερά K (0-255, 8 bits). Το μέγεθος του τμήματος X μπορεί να είναι από 1 ως 256. Η αρχική διεύθυνση ορίζεται από τον RP (BC, DE, HL) , το μήκος (X) και η σταθερά K καθορίζονται από τις παραμέτρους. Για (X)=0 το μέγεθος του τμήματος είναι ίσο με 256.

Κώδικας ASSEMBLY για 8085

```
FILL MACRO RP, X, K
        PUSH PSW
        PUSH H

        MVI A,X
        LXI H,RP
LOOP:
        MVI M,K
        INX
        DCR A
        CPI 00H
        JZ LOOP

        POP H
        POP PSW
ENDM
```

γ) Μακροεντολή "RHLR n" σε 8085

Περιστρέφει τα περιεχόμενα του κρατουμένου CY, των καταχωρητών H και L κατά n ψηφία δεξιά. Έτσι η μακροεντολή συμπεριφέρεται στα CY, H και L σαν να είναι ένας 17-bit καταχωρητής με το CY ως το πιο σημαντικό ψηφίο.

Κώδικας ASSEMBLY για 8085

```
RHLR MACRO n
        PUSH PSW
        PUSH B

        MVI B,n

LOOP:
        MOV A,L
        RAL
        MOV L,A
```

```
        MOV A,H
        RAL
        MOV H,A

        DCR B
        MOV A,B
        CPI 00H
        JZ LOOP

        POP B
        POP PSW
ENDM
```

4η Άσκηση

Αρχική κατάσταση :

(PC) = 0800H	
(SP)	(SP+1)
00H	30H

(1) Λόγω του ότι η διακοπή συμβαίνει κατά τη διάρκεια εκτέλεσης της εντολής **CALL 0880H** θα πρέπει πρώτα να τελειώσει η εκτέλεσή της. Επομένως η τιμή του PC θα αποθηκευτεί στο σωρό (μετατοπίζοντας τον SP κατά δύο θέσεις) και η καινούργια τιμή PC θα είναι η 0880H. Η κατάσταση των καταχωρητών θα είναι :

(PC) = 0880H			
(SP)	(SP+1)	(SP+2)	(SP+3)
00H	08H	00H	30H

(2) Στη συνέχεια εκτελείται η εντολή που έχει οριστεί στη θέση της διακοπής, η τιμή του μετρητή προγράμματος (PC) θα αποθηκευτεί πάλι στο σωρό και στη θέση του θα μπει η διακοπή RST 7.5 (η οποία θα πρέπει προηγουμένως να έχει προστεθεί στη μάσκα διακοπών και να ακολουθεί μία εντολή ενεργοποίησης της μάσκας - EI). Η κατάσταση των καταχωρητών θα είναι :

(PC) = (RST 7.5)					
(SP)	(SP+1)	(SP+2)	(SP+3)	(SP+4)	(SP+5)
80H	08H	00H	08H	00H	30H

(3) Αφού τελειώσει η εκτέλεση της ρουτινας διακοπής RST 7.5 η τελευταία διεύθυνση του σωρού επανέρχεται στο μετρητή προγράμματος και συνεχίζει η εκτέλεση του προγράμματος. Τέλος, αφού τελειώσει και εκτέλεση της ρουτίνας που αρχίζει στη διεύθυνση 0880H στο μετρητή προγράμματος (SP) επανέρχεται η τιμή 0800H που ήταν αποθηκευμένη στο σωρό και εκτελείται η επόμενη εντολή στη θέση 0801H. Τότε η κατάσταση των καταχωρητών θα είναι :

(PC) = 0801H	
(SP)	(SP+1)
00H	30H

5η Άσκηση

(α)

```
MVI   A,0DH       ;RST6.5 interrupt mask
      SIM                 ;Store Interrupt Mask
      LXI   H,0           ;H-L is zero
      MVI   C,64d         ;Counter C = 64 (64 steps)
      EI                  ;Enable Interrupts
ADDR:
      MVI   A,C
      CPI   0             ;Is C zero?
      JNZ   ADDR          ;If C != 0, loop
      DI                  ;else if, disable interrupts
      DAD   H             ;
      DAD   H
     DAD   H
      HLT

0034H:
      JMP   RST6.5

RST6.5:
      PUSH  PSW
      MOV   A,C
      ANI   01H           ;LSB of counter
      CPI   0             ;if even -> LSB
      JNZ   MSB           ;if odd -> MSB
      IN    20H           ;read 4 LSBs
```

```
        ANI   0FH         ;delete previous 4 MSBs
        MOV   B,A         ;save 4 LBSs
        DCR   C           ;decrease counter
        JMP   LOOP        ;wait for next interrupt

MSB:
        IN    20H         ;read 4 LSBs (MSBs of the whole number)
        ANI   0FH         ;delete previous MSBs
        RLC
        RLC
        RLC
        RLC               ;rotate to place 4 LSBs as MSBs
        ORA   B           ;A or B - join MSBs with LSBs
        MVI   D,00H       ;D is zero
        MOV   E           ;save number in D-E
        DAD   D           ;DE + HL (D = 0, C = 8bit number)
        DCR   C           ;decrease counter

LOOP:
        POP PSW
        EI
        RET
```

(β)

```
        LXI   H,0         ;H-L is zero
        MVI   C,64d       ;Counter C = 64 (64 steps)
ADDR:
        MVI   A,C
        CPI   0           ;Is C zero?
        JZ    MEAN
        IN    20H         ;check MSB
        ANI   10000000b
        CPI   0
        JNZ   READ
        JMP   ADDR        ;If C != 0, loop

MEAN:
        DI                ;else if, disable interrupts
```

```
        DAD   H          ;
        DAD   H
        HLT


READ:
        MOV   A,C
        ANI   01H        ;LSB of counter
        CPI   0          ;if even -> LSB
        JNZ   MSB        ;if odd -> MSB
        IN    20H        ;read 4 LSBs
        ANI   0FH        ;delete previous 4 MSBs
        MOV   B,A        ;save 4 LBSs
        DCR   C          ;decrease counter
        JMP   ADDR       ;wait for next interrupt

MSB:
        IN    20H        ;read 4 LSBs (MSBs of the whole number)
        ANI   0FH        ;delete previous MSBs
        RLC
        RLC
        RLC
        RLC              ;rotate to place 4 LSBs as MSBs
        ORA   B          ;A or B - join MSBs with LSBs
        MVI   D,00H      ;D is zero
        MOV   E          ;save number in D-E
        DAD   D          ;DE + HL (D = 0, C = 8bit number)
        DCR   C          ;decrease counter
```