ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΣΥΣΤΗΜΑΤΑ ΜΙΚΟΫΠΟΛΟΓΙΣΤΩΝ - 6ο ΕΞΑΜΗΝΟ
ΑΝΑΦΟΡΑ 3ης ΟΜΑΔΑΣ ΑΣΚΗΣΕΩΝ

ΓΕΩΡΓΙΑΔΗ ΔΑΦΝΗ 03120189
ΓΕΩΡΓΑΚΟΠΟΥΛΟΣ ΓΕΩΡΓΙΟΣ 03120827

ΑΣΚΗΣΕΙΣ ΠΡΟΣΟΜΟΙΩΣΗΣ

Disclaimer: ο προσομοιωτής 8085 τρέχει σε Windows 7 μέσα σε virtual box.

ΑΣΚΗΣΗ 1

Το πρόγραμμα που υλοποιήσαμε είναι το ακόλουθο:

```
        IN 10H
        MVI A,10H
        ;Βαζω σε όλες τις θέσεις μνήμης το περιεχόμενο του A
        STA 0B00H
        STA 0B01H
        STA 0B02H
        STA 0B03H
        STA 0B04H
        STA 0B05H
        MVI A,0DH
        SIM                     ;Ενεργοποίηση της Μασκας Διακοπών
;Τα περιεχόμενα του συσσωρευτή χρησιμοποιούνται για προγραμματισμό της μασκας διακοπών
        EI                      ;Ενεργοποίηση των Interrupts
INF_LOOP:
        JMP INF_LOOP
INTR_ROUTINE:
        POP H
        EI
        MVI A,00H
        STA 3000H
        MVI D,06H               ; Θα κάνουμε 6 επαναλήψεις για τα 60 δευτερόλεπτα
        MOV A,D
        DCR A
        STA 0B01H               ; Τοποθετώ αρχικά τις δεκάδες στον 7-segment
RELOAD_SECONDS:
        MVI A,09H               ; Seconds = 9
BIG_LOOP:
        STA 0B00H               ; Τοποθετώ τα δευτερόλεπτα στο 7-segment
        CALL SCREEN             ; Τυπωνω την τιμή
        DCR A                   ; Μειώνω κατα 1 τα δευτερόλεπτα
        CPI 00H
        JNZ BIG_LOOP
        MVI A,00H
        STA 0B00H
        CALL SCREEN
        DCR D                   ;Ελεγχος για τις Δεκάδες
        JZ EXIT
        MOV A,D
        DCR A
        STA 0B01H
        JMP RELOAD_SECONDS
SCREEN:                         ;Ανανέωση της οθόνης
        LXI B,0064H
        PUSH PSW                ; Store το A και το Flag
        PUSH H
        PUSH D
        PUSH B
        LXI D,0B00H
        CALL STDM
        MVI A,0AH
```

```
SMALL_LOOP:
        CALL DCD
        CALL DELB
        DCR A
        CPI 00H
        JNZ SMALL_LOOP
        POP B
        POP D                   ; Επαναφορά του D
        POP H
        POP PSW
        RET
EXIT:
        MVI A,FFH               ; Σβηνω τα LEDs
        STA 3000H
        JMP INF_LOOP

        END
```

ΠΡΟΣΟΧΗ:
Για να τρέξει το πρόγραμμα αφού το κάνουμε assemble, πατάμε RUN. Το πρόγραμμα θα σταματήσει μόνο του. Πρέπει να πατήσουμε το πλήκτρο FETCH UP και έπειτα ξανά το RUN.
Παρατηρήσεις:

1. !! Στο παραδοτέο zip εχουμε συμπεριλάβει ένα βίντεο στο οποίο ελέγχουμε την ορθότητα του προγράμματος !!
   Επεξήγηση βίντεο:
   Στο βίντεο τρέχουμε το πρόγραμμα και στην συνέχεια πατάμε το πλήκτρο INTR. Παρατηρούμε ότι ανοίγουν τα LEDs και ξεκινάει ένα ‘χρονόμετρο’ που φαίνεται στο 7-segment display. Μολις τελειώσει το χρονόμετρο σβηνουν τα LEDs και ξαναπατάμε το INTR. Ξαναξεκινάει το χρονόμετρο όμως αυτή την φορά ξαναπατάμε το INTR πριν τελειώσει ο χρόνος και βλέπουμε το χρονόμετρο να ξαναξεκινάει από την αρχή, όπως διευκρινήζεται στην εκφώνηση
   Η ταχύτητα της αντίστροφης μέτρησης ελεγχεται από την μπάρα πανω δεξία. Σε κανονική λειτουργία είναι ακριβως 1 λεπτό. Στο βιντεο η ταχύτητα είναι πιο γρήγορη.
2. Για την υλοποίηση του χρονομέτρου χρησιμοποιούμε 2 loop, ένα για τις δεκάδες και ένα για τις μονάδες των δευτερολέπτων. Το 7-segment display ενημερώνται κάθε 1 δευτερόλεπτο χάρη στην υπορουτίνα DELB
3. Στην ασκηση αυτή, καθώς καλούμε αρκετές υπορουτίνες, χρησιμοποιούμε την στοίβα του 8085 για να αποθηκεύουμε προσωρινά τα δεδομένα των καταχωρητών, ώστε να μην υπάρχει περίπτωση να τα επηρεάσουμε.
4. Πατώντας το κουμπί INTR δημιουργούμε διπλή διακοπή, ωστόσο δεν επηρεάζει κάπως την λειτουργία του προγράμματος σε αυτή την άσκηση.

ΑΣΚΗΣΗ 2

Το πρόγραμμα που υλοποιήσαμε είναι το ακόλουθο:

```
        IN 10H
        MVI A,10H               ;Αρχικοποίηση του 7-segment
        STA 0B00H
        STA 0B01H
        STA 0B02H
        STA 0B03H
        STA 0B04H
        STA 0B05H

        MVI A,0DH               ;Ενεργοποίηση διακοπής RST 6.5
        SIM
        EI
MAIN_PROGRAM:
        JMP MAIN_PROGRAM
INTR_ROUTINE:
        CALL KIND       ;Εισοδος απο το πληκτρολόγιο -> Μονάδες
        STA 0B04H
        CALL DISPLAY
        MOV B,A
        CALL KIND       ;Εισοδος απο το πληκτρολόγιο -> Δεκάδες
        STA 0B05H
        CALL DISPLAY
        MVI D,32H               ;K1 = 50
        MVI E,C8H               ;K2 = 200
        RLC
        RLC
        RLC
        RLC
        ORA B
        MOV B,A         ;Ο Τελικός Αριθμός ειναι το L
        EI
        MOV A,B
        CMP D           ;Συγκριση με το K1
        JC RANGE1
        JZ RANGE1
        CMP E           ;Συγκριση με το K2
        JC RANGE2
        JZ RANGE2
        MVI A,FBH       ;Σε όποια αλλη περίπτωση ανοίγω το 3ο LSB LED
        STA 3000H
        RET

DISPLAY:
         PUSH PSW
         PUSH H
         PUSH D
         PUSH B
         LXI D,0B00H
         CALL STDM
         CALL DCD
         POP B
         POP D
         POP H
         POP PSW
         RET
RANGE1:
         MVI A,FEH       ; 1ο LSB LED
         STA 3000H
         RET
RANGE2:
         MVI A,FDH       ; 2ο LSB LED
         STA 3000H
         RET

         END
```

ΠΡΟΣΟΧΗ:

Για να τρέξει το πρόγραμμα αφού το κάνουμε assemble, πατάμε RUN. Το πρόγραμμα θα σταματήσει μόνο του. Πρέπει να πατήσουμε το πλήκτρο FETCH UP και έπειτα ξανά το RUN.

Παρατηρήσεις:

1. !! Στο παραδοτέο zip εχουμε συμπεριλάβει ένα βίντεο στο οποίο ελέγχουμε την ορθότητα του προγράμματος !!
   Επεξήγηση βίντεο:
   Στο βίντεο τρέχουμε ηδη το πρόγραμμα και πατάμε το πλήκτρο INTR. Εμφανίζεται η τιμή df που ήταν η προηγούμε που έιχαμε καταχωρήσει. Εισάγουμε μια νέα τιμή, την τιμή 30, ωστόσο λόγω της ιδιοτροπίας του προσωμοιωτή πρεπει να πατήσουμε πρωτα 2 τυχαια πληκτρα και έπειτα την τιμή που θέλουμε. Μολις πατάμε το τελευταίο πλήκτρο (3) βλέπουμε να ανάβει το 1ο LSB LED, όπως ζητούσε η εκφώνηση γιατι αυτός ο αριθμός ανήκει στο πρώτο πεδίο τιμών, είναι δηλαδή μικρότερος του K1 που έχουμε θέσει (K1=32H). Επειτα ξαναπατάμε το πλήκτρο INTR και εισάγουμε την τιμή 88, οπότε και ανάβει το δευτερο LED καθώς K1<88<K2 (K2= C8H). Τέλος κανουμε την ίδια διαδικασία για την τιμη DF, και ανάβει το 3ο LSB LED , όπως ακριβώς θέλαμε.
   Κάθε φορά που εισάγουμε μια τιμή πρέπει αν λαμβάνουμε υπόψιν μας την ιδιοτροπία του simulator που εξηγείται στην συνέχει.
2. Πατώντας το πλήκτρο INTR δημιουργούμε διπλή διακοπή, μια όταν πατήσαμε το κουμπί και μια όταν το αφήσαμε. Επομένως η ρουτίνα εξυπηρέτησης τρέχει 2 φορές. Αρα στο τέλος βλέπουμε αποτέλεσμα μόνο για την δευτερη φορά που μας επιτρέπει να εισάγουμε τιμές.
3. Οι έλεγχοι για να βρούμε σε πιο πεδίο τιμών βρίσκεται ο αριθμός που εισάγαμε από το πληκτρολόγιο γίνονται στο 16δικό σύστημα.
4. Στην ασκηση αυτή, καθώς καλούμε αρκετές υπορουτίνες, χρησιμοποιούμε την στοίβα του 8085 για να αποθηκεύουμε προσωρινά τα δεδομένα των καταχωρητών, ώστε να μην υπάρχει περίπτωση να τα επηρεάσουμε.

ΘΕΩΡΗΤΙΚΈΣ ΑΣΚΗΣΕΙΣ

ΑΣΚΗΣΗ 3

α) Η μακροεντολή SWAP Nible Q

```
SWAP_NIBBLE_Q MACRO Q
   PUSH PSW          Save the status register
   MOV A, Q          Move the value of Q to A
   RLC               Rotate left A four times to swap the nibbles
   RLC
   RLC
   RLC
   MOV Q, A          Move the result back to Q
   MOV A, M          Move the value in the memory location pointed to by HL to A
   RRC               Rotate right A four times to swap the nibbles
   RRC
   RRC
   RRC
   MOV M, A          Move the result to the memory location pointed to by HL
   POP PSW           Restore the status register
ENDM
```

β) Η μακροεντολή FILL RP, X, K

```
FILL MACRO ADDR, L, K
   PUSH PSW
   PUSH H

   LXI H, L          Load the length into the H-L register pair
   MOV A, H          Move the higher byte of the length to A

   CPI 0             Compare A with 0
   JNZ NOT_ZERO      Jump to NOT_ZERO if A is not zero

   LXI H, 256        Set H to 256 if A is zero
   JMP CONTINUE

NOT_ZERO:
   LXI H, L          Load the length into the H-L register pair

CONTINUE:
   LXI H, ADDR       Load the starting address into the H-L register pair
   START:
      MVI M, K       Store the value K at the memory location pointed to by HL
      INX H          Increment HL to point to the next memory location
      DCX B          Decrement BC to track the remaining length
      JNZ START      Jump back to START if BC is not zero

   POP H
   POP PSW
ENDM
```

γ) Η μακροεντολή RHLR

```
RHLR MACRO
   MOV A, L          Move the contents of register L to A
   RRC               Rotate right through carry
   MOV L, A          Move the rotated value back to register L

   MOV A, H          Move the contents of register H to A
   RRC               Rotate right through carry
   MOV H, A          Move the rotated value back to register H
ENDM
```

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

ΑΣΚΗΣΗ 4

Στον μικροεπεξεργαστή 8085, όταν εκτελείται η εντολή «CALL», ο program counter ωθείται στην στοίβα και ο PC φορτώνεται με τη διεύθυνση που καθορίζεται στην οδηγία. Ο stack pointer μειώνεται κατά δύο για να προσαρμόσει την ωθούμενη τιμή του PC.

Η διακοπή RST 5.5 συμβαίνει στο μέσω της CALL 0900H, άρα θα ολοκληρωθεί η εκτέλεση της τρέχουσας εντολής.
- η τρέχουσα τιμή του program counter 0840H, προωθείται στην στοίβα,
- ο program counter φορτώνει την διεύθυνση 0900H,
- ο δείκτης στοίβας μειώνεται κατά 2 θέσεις,

Άρα:
PC = 0900H
SP = 2FFEH (3000H - 2)
Έπειτα σώζεται η τιμή του μετρητή προγράμματος και η κατάσταση του 8085 και εκτελείται η ρουτίνα εξυπηρέτησης της διακοπής RST 5.5. Το RST 5.5 αντιστοιχεί στη διεύθυνση 0028H.

Μέσω της RESET¨
- η τρέχουσα τιμή του program counter προωθείται στην στοίβα
- ο program counter φορτώνεται με την σταθερή διεύθυνση που αντιστοιχεί η εντολή RST,

Μετά το RESET λοιπόν έχουμε:
PC= 0028H
SP = 2FFCH (2FFEH -2)

ΑΣΚΗΣΗ 5

Ο κώδικας για την άσκηση 5 ειναι ο ακόλουθος:

```
        MVI A,0Eh               ; Μάσκα διακοπών
        SIM
        LXI H,0                 ; Συσσωρευτής δεδομένων
        MVI C,64d
        EI
ADDR:                           ; Αναμονή δεδομένων
        MVI A,C
        CPI 0
        JNZ ADDR                ; Έλεγχος εισόδου όλων των δεδομένων
        DI                      ; Απενεργοποίηση διακοπών
        DAD H                   ; 3 φορές πρόσθεση του H-L στον εαυτό του για ολίσθηση 3 φορές αριστερά
        DAD H
        DAD H
        MOV A,L
        ANI 80H                 ; Κρατάω το 1ο ψηφίο του A (δλδ του αριθμού μου)
        MVI L, 00H
        CPI 00H
        JNZ PLUS_ONE            ; Ανάλογα αυτό το ψηφίο βλέπω αν χρειάζεται στρογγυλοποίηση προς τα πανω
        HLT
        END
PLUS_ONE:
        INR H
        HLT                     ; Τελος Προγραμματος
        END
```

```
0034:
        JMP RST6.5
```

```
RST6.5:
        PUSH PSW
        MOV A,C
        ANI 01H                 ; Για το LSB
        JPO GET4MSB             ; Έλεγχος αν λάβαμε τα LSB ή τα MSB του δεδομένου
        IN PORT_IN
        ANI 0FH                 ; Για τα 4 LSB της πόρτας
        MOV B,A                 ; Προσωρινή αποθήκευση
        JMP HAVE_4LSB           ; Επιστροφή στο πρόγραμμα ADDR
GET_4MSB:
        IN PORT_IN
        ANI 0FH
        RLC                     ; 4 φορές ολίσθηση και ενωση με τα LSB του δεδομένου
        RLC
        RLC
        RLC
        ORA B

        MVI D,0
        MVI E,A
        DAD D                   ; Πρόσθεση δεδομένων
HAVE_4LSB:
        POP PSW
        EI
```

Παρατηρήσεις:

1. 1. Το βασικό προγραμμα ειναι το loop ADDR με βάση τον μετρητή C. Σε κάθε διακοπή RST6.5 ο μετρητής αυτός μειώνεται κατα 1, έχωντας αρχική τιμή 64 (Εχουμε 32 δεδομένα με 2 μέρη).
2. Σε κάθε διακοποή χρησιμοποιούμε μεια στοίβα για να αποθηκευσουμε το PSW. Επίσης σε κάθε διακοπή ελέγχουμε εαν έχουμε πάρει MSB ή LSB με βαση τον μετρητή C.
3. Στο τέλος πριν την στρογγυλοποίηση, απενεργοποιούμε τις διακοπές με την εντολή DI ωστε να μην υπαρξει confilct.