ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ

ΤΡΙΤΗ ΟΜΑΔΑ ΑΣΚΗΣΕΩΝ

Ναταλία Μπούρδη

ΑΜ: el19031

Ανδρέας Παπαθανασίου

ΑΜ: el19212

Άσκηση 1

Το Πρόγραμμα της Άσκησης – Κώδικας Assembly:

```
IN 10H
LXI B,000AH       ; 10ms delay
MVI E,00H
MVI A,1DH         ; Αρχικοποίηση διακοπών
SIM ;
EI

LOOP1:
MVI A,FFH         ; LEDs OFF
STA 3000H
JMP LOOP1

INTR_ROUTINE:
MVI D,2EH         ; Μετρητης αλλαγων καταστασης των LEDs (2D=45)


REPEAT:
CALL BLINK
CALL PRINT
CALL DELB_
CALL BLINK
CALL DELB_
DCR D
JZ END_
JMP REPEAT

END_:
MVI A,1DH         ; Αρχικοποίηση διακοπών
SIM
EI
RET



BLINK:            ; Αλλαγη καταστασης των LEDs
MOV A,E
STA 3000H
CMA
MOV E,A
RET



DELB_:            ; 500ms delay με ανανεωση των 7-segment displays καθε
10msec
MVI A,32H         ; 50x10msec delays
DELB_LOOP:
CALL DELB
```

```
CALL DCD
DCR A
JNZ DELB_LOOP
RET



PRINT:              ;εμφανιζουμε το περιεχομενο του καταχωρητη D στα δυο 7-
segment displays
CALL SPACES
LXI H,0905H         ;δειυθυνση που θα γραφούν τα συμβολα προς απεικονιση

PUSH B
PUSH D
MVI B,00H           ;Κραταει δεκαδες
MVI C,00H           ;Κραταει μοναδες

CONVERT:
DCR D               ;Μειωνουμε τον D
JZ CONT
CALL DEC_           ;Αυξανουμε το ζευγος B,C
JMP CONVERT

DEC_:               ;Σε καθε επαναληψη αφαιρουμε 1 απο τον D
INR C               ;και προσθετουμε 1 στο ζευγος B,C σε δεκαδικη αναπαρασταση
MOV A,C
CPI 0AH
JNZ RET_
INR B
MVI C,00H
RET_:
RET

CONT:
MOV M,B             ;store B to memory
DCX H               ;move pointer to the other address
MOV M,C             ;store C
LXI D,0900H         ;Διευθυνση μνημης από όπου θα διαβασουν οι STDM και DCD
CALL STDM
CALL DCD            ;απεικονιση με χρηση των STDM, DCD
POP D
POP B
RET

SPACES:             ;Αρχικοποιηση με κενα στα ψηφια που θα τυπωνονται
LXI H,0900H         ;χρησιμοποιουμε τη διευθυνση 0900H
MVI M,10H
INX H
MVI M,10H
INX H
MVI M,10H
INX H
MVI M,10H
INX H
MVI M,10H
```

```
INX H
MVI M,10H
RET

END
```

Άσκηση 2

Το Πρόγραμμα της Άσκησης – Κώδικας Assembly:

```
IN 10H

MVI A,10H          ; empty character code
STA 0B00H
STA 0B01H
STA 0B02H          ; will place low digit here
STA 0B03H          ; high digit
STA 0B04H
STA 0B05H


MVI A,1DH
SIM
EI

MVI C,40H          ; 64 DEC
MVI D,80H          ; 128 DEC
MVI E,C0H          ; 192 DEC

LOOP1:
    ;CALL DCD
    JMP LOOP1

INTR_ROUTINE:
    PUSH B
    PUSH D         ; saving register values...
    STA 0B02H      ; will place low digit here
    STA 0B03H      ; will place high digit here
    LXI D,0B00H
    CALL STDM
    CALL KIND      ; A <- digit pressed (high)
    ANI 0FH
    STA 0B03H      ; placing high digit for displaying later
    RLC
    RLC
    RLC
    RLC
    MOV H,A
    CALL KIND      ; A <- digit pressed (low)
    ANI 0FH
    STA 0B02H      ; placing low digit for displaying later
    ORA H          ; A now has both digits
```

```
LEDS:
    POP D
    POP B          ; restoring C,D,E values to check for the LEDs
    CMP E
    JZ LED2
    CMP E
    JNC LED3
    CMP D
    JZ LED1
    CMP D
    JNC LED2
    CMP C
    JZ LED0
    CMP C
    JNC LED1
    JMP LED0

LED0:
    MVI A,FEH
    STA 3000H
    JMP EXIT
LED3:
    MVI A,F7H
    STA 3000H
    JMP EXIT
LED1:
    MVI A,FDH
    STA 3000H
    JMP EXIT
LED2:
    MVI A,FBH
    STA 3000H
    JMP EXIT

DELB_:   ;~500ms delay με ανανεωση των 7-segment displays καθε 10msec
    MVI A,32H           ;50x10msec delays
    DELB_LOOP:
    CALL DELB
    CALL DCD
    DCR A
    JNZ DELB_LOOP
    RET


EXIT:
    PUSH D
    PUSH B         ; saving registers again...
    LXI D,0B00H    ; for STDM
    CALL STDM
    CALL DCD

    LXI B,000AH    ; 10msec delay for DELB (see DELB_ for the final delay
duration)
    CALL DELB_     ; to view leds
```

```
    MVI A,FFH     ; reset leds
    STA 3000H
    POP B
    POP D         ; restore registers...

    MVI A,1DH     ; initialize interrupts
    SIM
    EI
    RET
END
```

ΆΣΚΗΣΗ 3

Α.

```
INR16 MACRO ADDR
      PUSH H      ; Αποθηκευουμε τους H,L στη στοιβα
      LXI H,ADDR  ; H,L<-ADDR
      MOV M,A     ; A<-(ADDR)
      ADI 01H     ; Αυξανουμε το Xlow.
                  ; Το CY γινεται 1 σε περιπτωση υπερχειλισης
      MOV M,A     ; (ADDR)<-A
      INX H       ; Αυξανουμε την τιμη του ζευγους H,L
      MOV A,M     ; A<-(ADDR+1)
      ACI 00H     ; Προσθετουμε στο Xhigh το κρατουμενο σε
                  ; περιπτωση υπερχειλισης
      MOV M,A     ; (ADDR+1)<-A
      POP H       ; Επαναφέρουμε τους καταχωρητές H,L
      ENDM
```

Β.

```
FILL MACRO ADDR,K
      LXI H,ADDR  ; H,L<-ADDR
      MVI A,K     ; A<-K
      REPEAT:
      MOV M,A     ; ((H),(L))<-A
      INX H       ; Αυξανουμε την τιμη του ζευγους H,L
      DCR A       ; Μειωνουμε την τιμη του A
      JNZ REPEAT  ; Αν A=0 τελος
      ENDM
```

Γ.

```
RHLL MACRO Q,R
```

```
MOV A,R
RAL         ; Περιστροφη του R αριστερα μεσω κρατουμενου
MOV R,A     ; R0<-CY και CY<-R7
MOV A,Q
RAL         ; Περιστροφη του Q αριστερα μεσω κρατουμενου
MOV Q,A     ; Q0<-R7 και CY<-Q7
ENDM
```

Άσκηση 4

Όταν αναγνωριστεί διακοπή από τον μΕ αρχικά ολοκληρώνεται η εκτέλεση της τρέχουσας εντολής, έπειτα σώζεται στη στοίβα ο PC και τέλος ο PC λαμβάνει τη διεύθυνση της διακοπής που αναγνωρίστηκε. Η ρουτίνα εξυπηρέτησης που θα κληθεί από τη διεύθυνση αυτή θα αναλάβει να σώσει την κατάσταση του μΕ και τους καταχωρητές που θα χρησιμοποιήσει.

	Λειτουργίες του μΕ	Περιγραφή
JMP 2200H	**PC<-2200H**	Η εντολή JMP εκτελείται μέχρι το τέλος, άρα PC<-2200H
RST6.5:	**(SP-1)<-PCH** **(SP-2)<-PCL**	(2FFFH)<-22H (2FFEH)<-00H
	SP<-SP-2	SP<-2FFEH
	PC<-0034H	PC<-0034H (Η διεύθυνση της διακοπής RST6.5)
Εκτέλεση ρουτίνας εξυπηρέτησης…	...	
Επιστροφή από τη ρουτίνα εξυπηρέτησης	**PCL<-(SP)** **PCH<-(SP+1)** **SP<-SP+2**	PCL<-00H PCH<-22H SP<-3000H

Αμέσως πριν την κλήση της ρουτίνας εξυπηρέτησης της διακοπής έχουμε:

PC	SP	Διεύθυνση	Περιεχόμενο
2200H	3000H	2FFFH	22H
		2FFEH	00H

Στις διευθύνσεις 2FFFH, 2FFEH εξακολουθούν μετά την εκτέλεση της ρουτίνας εξυπηρέτησης της διακοπής και της επαναφοράς του stack pointer να υπάρχουν οι τιμές 22H και 00H αντίστοιχα. Όμως, επειδή η στοίβα μεγαλώνει προς τα κάτω οι παραπάνω διευθύνσεις δεν ανήκουν στην στοίβα. Επομένως, τα περιεχόμενά της είναι τα ίδια με αυτά που ήταν πριν την διακοπή.

ΆΣΚΗΣΗ 5

A.

```
; Initialize registers
MVI B,00H        ; Overflow counter
MVI C,00H        ; Sum of the inputs
MVI E,20H        ; Counter for the 32 reads
LXI H,9000H


LOOP:
JMP LOOP         ; Wait for interrupts


RST5.5:          ; Interrupt routine
     IN 20H      ; Read 4 digits from the device
     MOV M,A     ; Store them in memory
     INX H       ; Increase the address
     DCR E       ; Decrement counter
     JZ EXIT     ; Exit if all data is read
     RET         ; Else return

EXIT:

LXI H,9000H      ; Restore to the initial address
MVI E,10H        ; Counter for the 16 numbers to be summed


SUM:
     MOV A,M     ; Read 4 LSBs from memory
     RRC
     RRC
     RRC
```

```
      RRC         ; put LSBs in the correct position
      MOV D,A     ; Store them temporarily
      INX H       ; Increase memory address
      MOV A,M     ; Read 4 MSBs from memory
      INX H       ; Increase memory address
      ORA D       ; Now A has the whole number
      ADD C       ; Add to the sum
      JNC CONT    ; If there is overflow increment B
      INR B
CONT:
      MOV C,A     ; Store the sum in C
      DCR E
      JNZ SUM     ; If all 16 numbers are summed, continue


; The sum is now stored in the B,C register pair
; We need to divide it by 16
STC
CMC         ; Clear carry
MVI E,04H  ; 4 right rotations needed to divide the sum by 16
DIVIDE:
      MOV A,B
      RAR         ; Rotate B through carry
      MOV B,A
      MOV A,C
      RAR         ; Rotate C through carry
      MOV C,A     ; Now the value stored in B,C is divided by 2
      DCR E
      JNZ DIVIDE

MOV A,C           ; Now A has the result!
RET               ; To return from the interrupt routine
```

B.

```
; Initialize registers
MVI B,00H         ; Overflow counter
MVI C,00H         ; Sum of the inputs
MVI E,20H         ; Counter for the 32 reads
LXI H,9000H


LOOP:
IN 20H
```

```
MOV B,A
ANI 01H
JNZ LOOP         ; Wait for 'DATA READY' to become 0

DATA_READY:
      MOV M,B    ; Store them in memory
      INX H      ; Increase the address
      DCR E      ; Decrement counter
      JZ EXIT    ; Exit if all data is read
WAIT_FOR_1:
      IN 20H
      ANI 01H
      JZ WAIT_FOR_1    ; Wait for 'DATA READY' to return to 1
      JMP LOOP         ; When 'DATA READY' = 1 return

EXIT:
LXI H,9000H            ; Restore to the initial address
MVI E,10H        ; Counter for the 16 numbers to be summed


SUM:
      MOV A,M    ; Read 4 LSBs from memory
      RRC
      RRC
      RRC
      RRC        ; put LSBs in the correct position
      MOV D,A    ; Store them temporarily
      INX H      ; Increase memory address
      MOV A,M    ; Read 4 MSBs from memory
      ANI F0H    ; Clear 'data ready' bit
      INX H      ; Increase memory address
      ORA D      ; Now A has the whole number
      ADD C      ; Add to the sum
      JNC CONT   ; If there is overflow increment B
      INR B
CONT:
      MOV C,A    ; Store the sum in C
      DCR E
      JNZ SUM    ; If all 16 numbers are summed, continue


; The sum is now stored in the B,C register pair
; We need to divide it by 16
STC
CMC        ; Clear carry
MVI E,04H  ; 4 right rotations needed to divide the sum by 16
DIVIDE:
      MOV A,B
      RAR        ; Rotate B through carry
      MOV B,A
      MOV A,C
```

```
      RAR         ; Rotate C through carry
      MOV C,A     ; Now the value stored in B,C is divided by 2
      DCR E
      JNZ DIVIDE

MOV A,C           ; Now A has the result!
```