Συστήματα Μικροϋπολογιστών

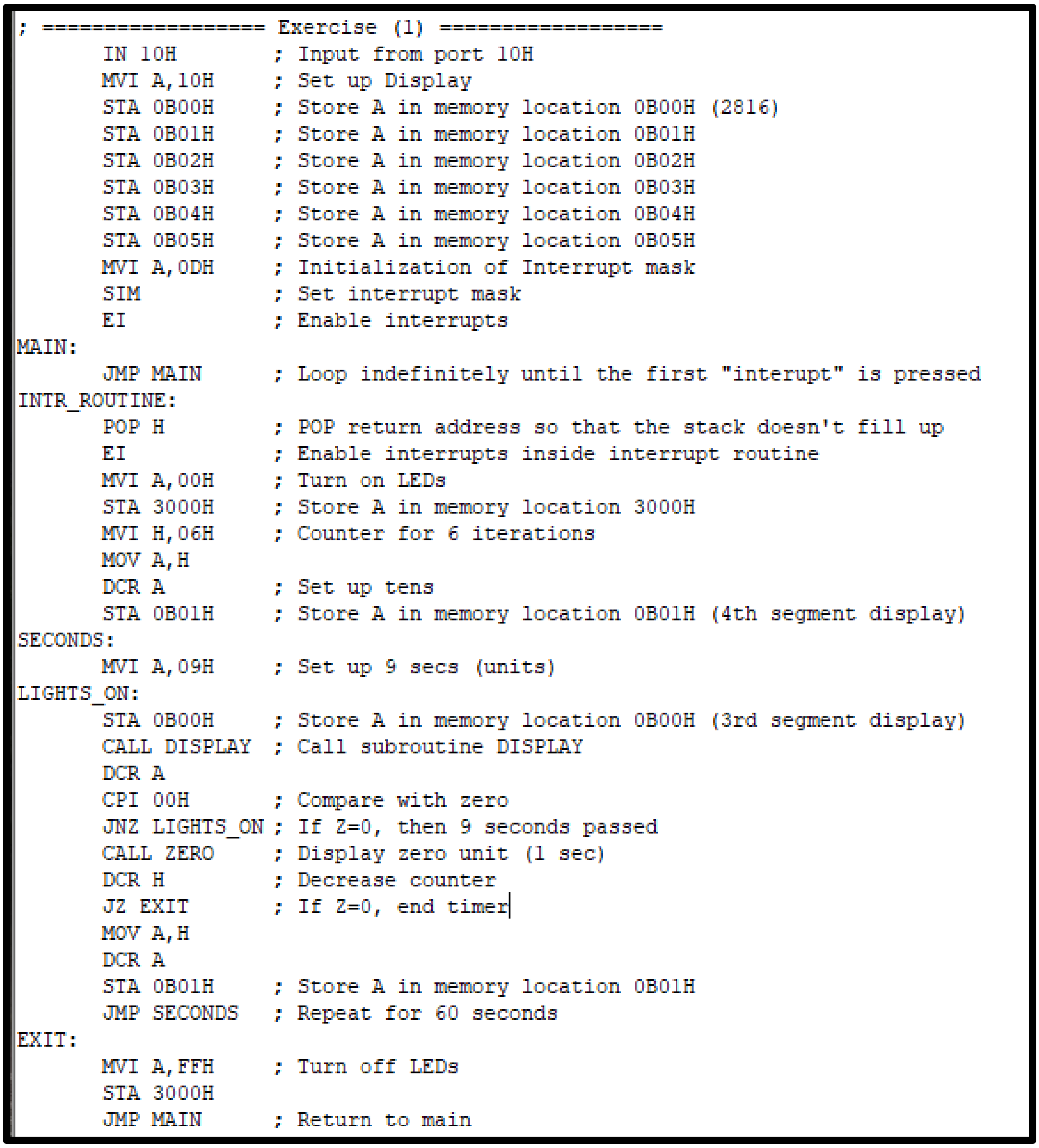
3η Ομάδα Ασκήσεων

Συμμετέχοντες:

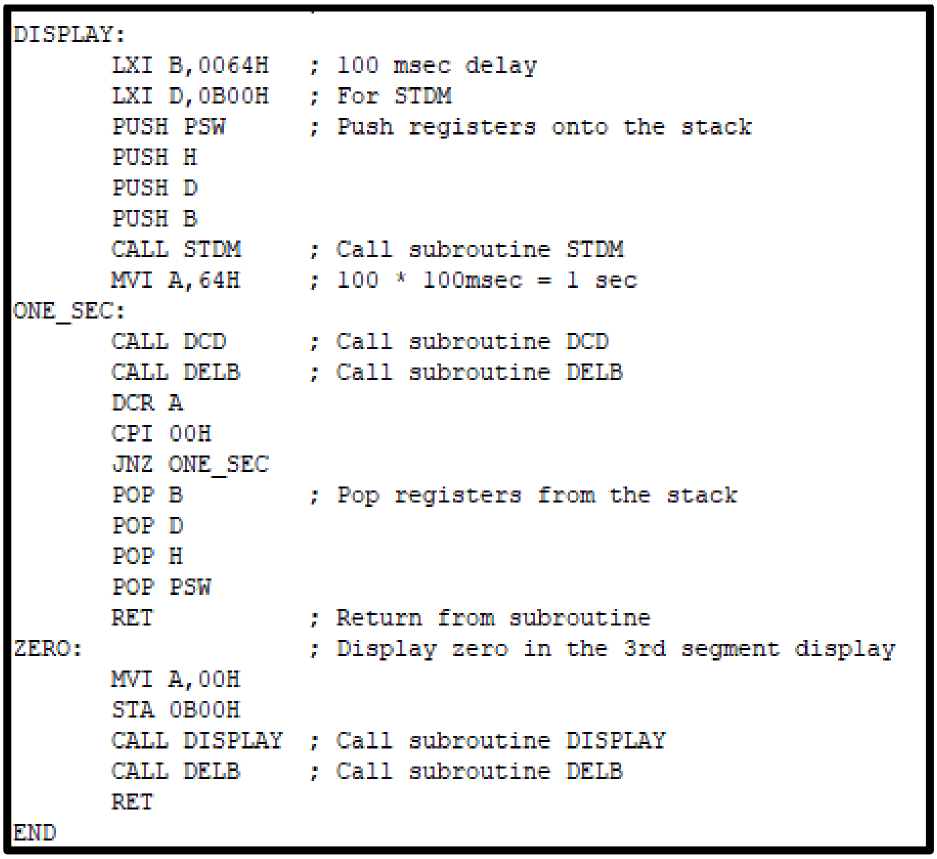
Αλέξανδρος Σκούρας, 03120105

Ιωάννης Τσαντήλας, 03120883

1η Άσκηση

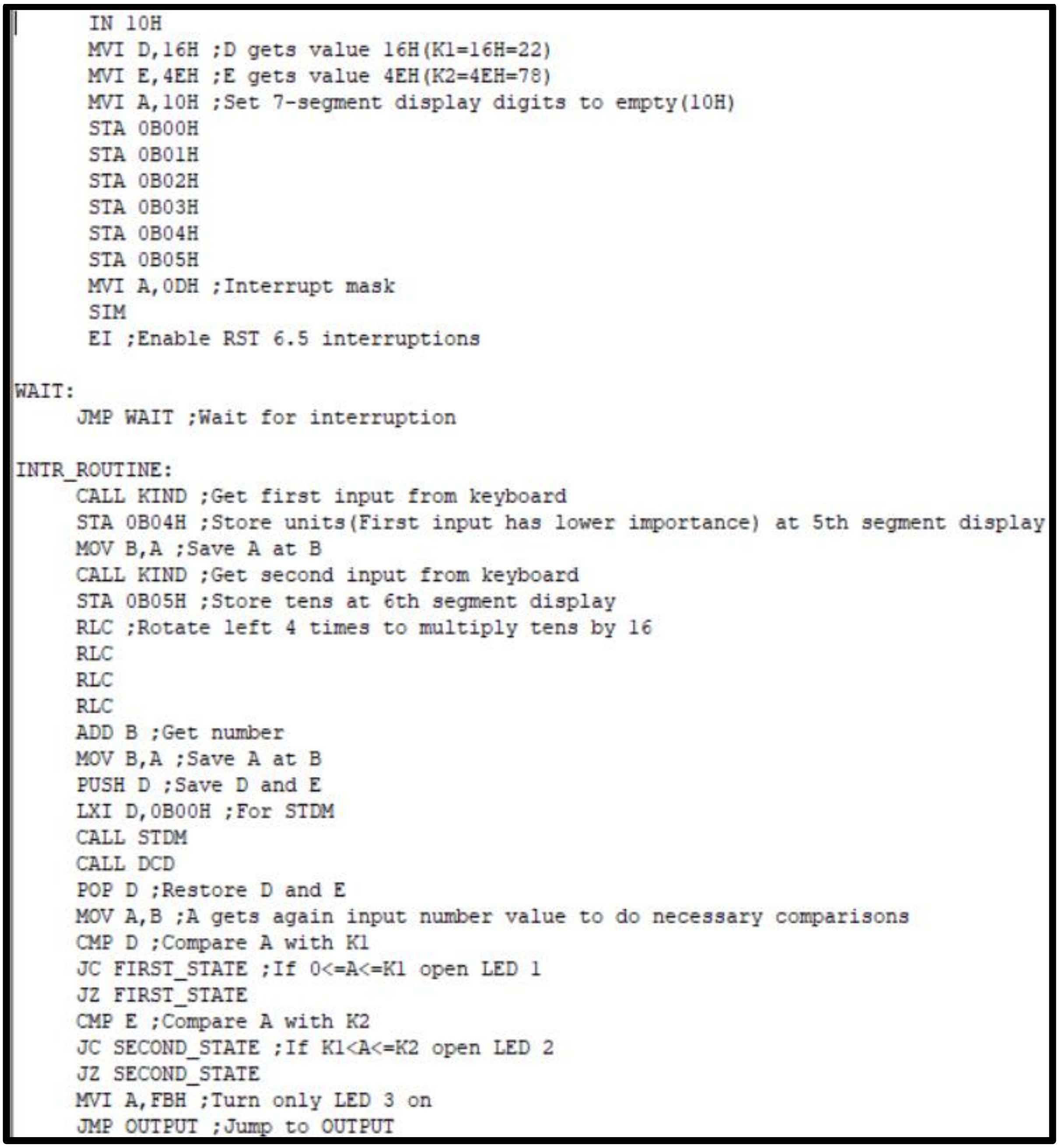


```
; ================== Exercise (1) ==================
       IN 10H        ; Input from port 10H
       MVI A,10H     ; Set up Display
       STA 0B00H     ; Store A in memory location 0B00H (2816)
       STA 0B01H     ; Store A in memory location 0B01H
       STA 0B02H     ; Store A in memory location 0B02H
       STA 0B03H     ; Store A in memory location 0B03H
       STA 0B04H     ; Store A in memory location 0B04H
       STA 0B05H     ; Store A in memory location 0B05H
       MVI A,0DH     ; Initialization of Interrupt mask
       SIM           ; Set interrupt mask
       EI            ; Enable interrupts
MAIN:
       JMP MAIN      ; Loop indefinitely until the first "interupt" is pressed
INTR_ROUTINE:
       POP H         ; POP return address so that the stack doesn't fill up
       EI            ; Enable interrupts inside interrupt routine
       MVI A,00H     ; Turn on LEDs
       STA 3000H     ; Store A in memory location 3000H
       MVI H,06H     ; Counter for 6 iterations
       MOV A,H
       DCR A         ; Set up tens
       STA 0B01H     ; Store A in memory location 0B01H (4th segment display)
SECONDS:
       MVI A,09H     ; Set up 9 secs (units)
LIGHTS_ON:
       STA 0B00H     ; Store A in memory location 0B00H (3rd segment display)
       CALL DISPLAY  ; Call subroutine DISPLAY
       DCR A
       CPI 00H       ; Compare with zero
       JNZ LIGHTS_ON ; If Z=0, then 9 seconds passed
       CALL ZERO     ; Display zero unit (1 sec)
       DCR H         ; Decrease counter
       JZ EXIT       ; If Z=0, end timer
       MOV A,H
       DCR A
       STA 0B01H     ; Store A in memory location 0B01H
       JMP SECONDS   ; Repeat for 60 seconds
EXIT:
       MVI A,FFH     ; Turn off LEDs
       STA 3000H
       JMP MAIN      ; Return to main
```

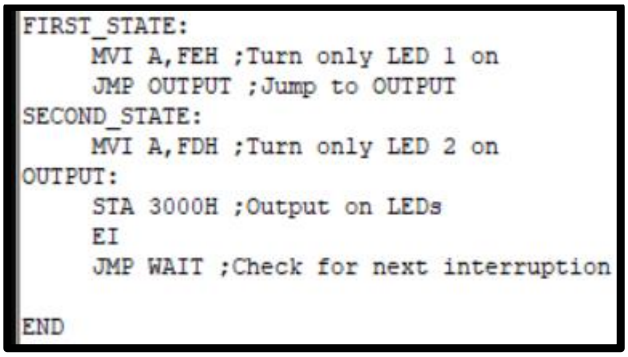


```
DISPLAY:
       LXI B,0064H   ; 100 msec delay
       LXI D,0B00H   ; For STDM
       PUSH PSW      ; Push registers onto the stack
       PUSH H
       PUSH D
       PUSH B
       CALL STDM     ; Call subroutine STDM
       MVI A,64H     ; 100 * 100msec = 1 sec
ONE_SEC:
       CALL DCD      ; Call subroutine DCD
       CALL DELB     ; Call subroutine DELB
       DCR A
       CPI 00H
       JNZ ONE_SEC
       POP B         ; Pop registers from the stack
       POP D
       POP H
       POP PSW
       RET           ; Return from subroutine
ZERO:                ; Display zero in the 3rd segment display
       MVI A,00H
       STA 0B00H
       CALL DISPLAY  ; Call subroutine DISPLAY
       CALL DELB     ; Call subroutine DELB
       RET
END
```

2^{η} Άσκηση

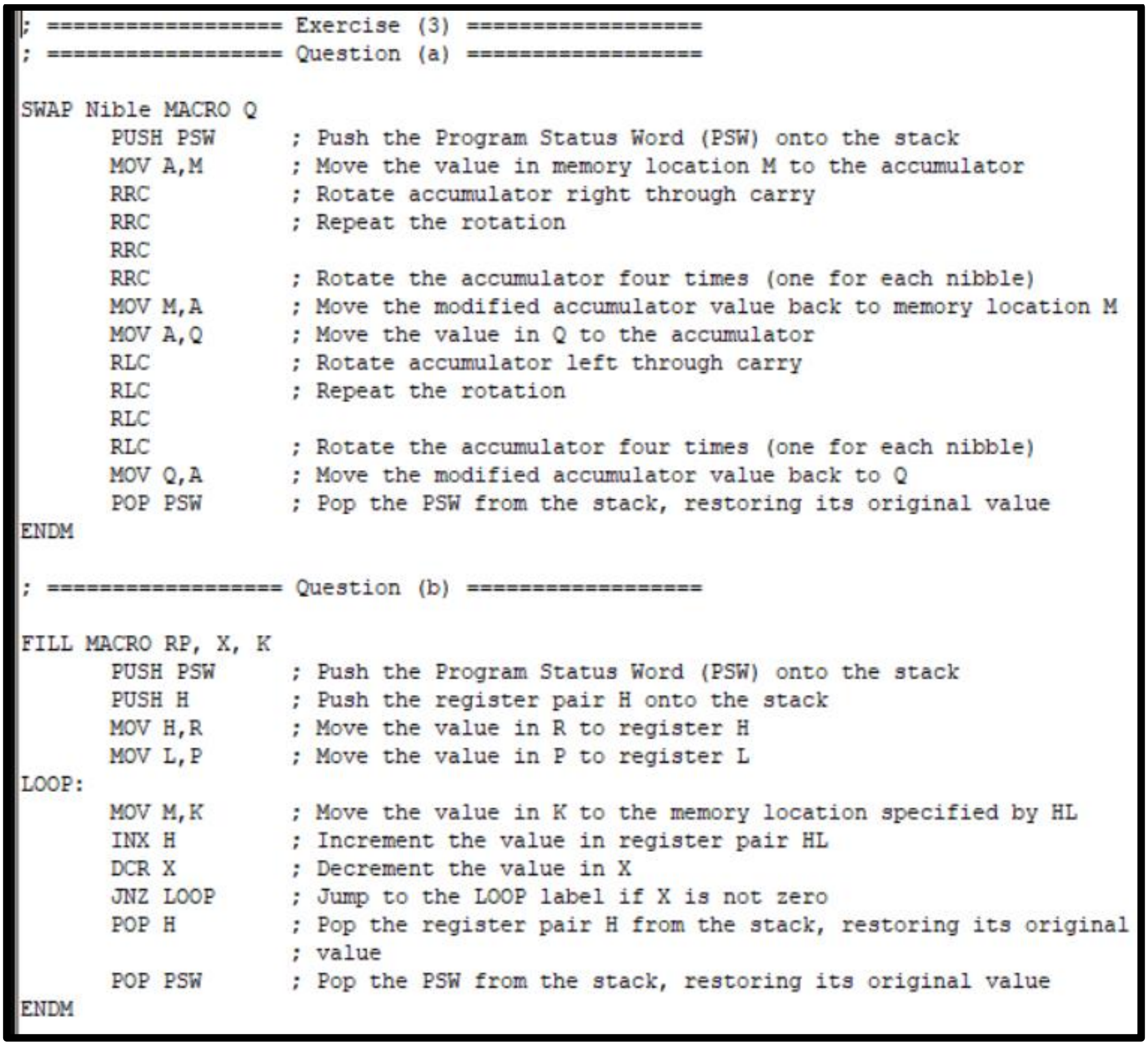


```
      IN 10H
      MVI D,16H ;D gets value 16H(K1=16H=22)
      MVI E,4EH ;E gets value 4EH(K2=4EH=78)
      MVI A,10H ;Set 7-segment display digits to empty(10H)
      STA 0B00H
      STA 0B01H
      STA 0B02H
      STA 0B03H
      STA 0B04H
      STA 0B05H
      MVI A,0DH ;Interrupt mask
      SIM
      EI ;Enable RST 6.5 interruptions

WAIT:
     JMP WAIT ;Wait for interruption

INTR_ROUTINE:
     CALL KIND ;Get first input from keyboard
     STA 0B04H ;Store units(First input has lower importance) at 5th segment display
     MOV B,A ;Save A at B
     CALL KIND ;Get second input from keyboard
     STA 0B05H ;Store tens at 6th segment display
     RLC ;Rotate left 4 times to multiply tens by 16
     RLC
     RLC
     RLC
     ADD B ;Get number
     MOV B,A ;Save A at B
     PUSH D ;Save D and E
     LXI D,0B00H ;For STDM
     CALL STDM
     CALL DCD
     POP D ;Restore D and E
     MOV A,B ;A gets again input number value to do necessary comparisons
     CMP D ;Compare A with K1
     JC FIRST_STATE ;If 0<=A<=K1 open LED 1
     JZ FIRST_STATE
     CMP E ;Compare A with K2
     JC SECOND_STATE ;If K1<A<=K2 open LED 2
     JZ SECOND_STATE
     MVI A,FBH ;Turn only LED 3 on
     JMP OUTPUT ;Jump to OUTPUT
```

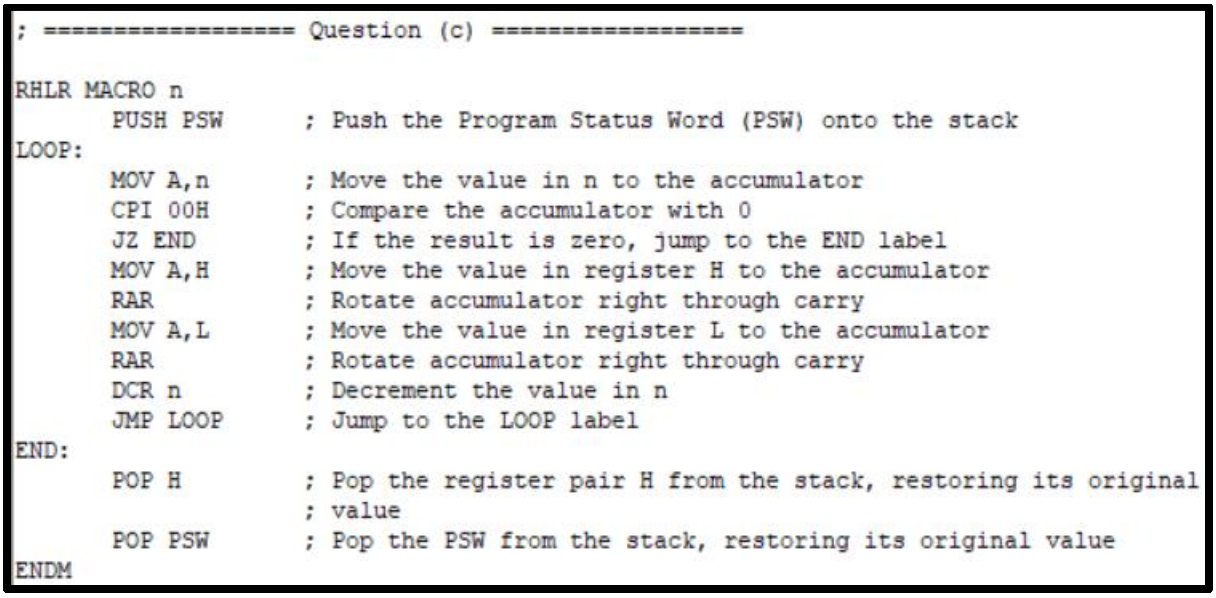


```
FIRST_STATE:
     MVI A,FEH ;Turn only LED 1 on
     JMP OUTPUT ;Jump to OUTPUT
SECOND_STATE:
     MVI A,FDH ;Turn only LED 2 on
OUTPUT:
     STA 3000H ;Output on LEDs
     EI
     JMP WAIT ;Check for next interruption

END
```

3η Άσκηση



```
; ================== Exercise (3) ==================
; ================== Question (a) ==================

SWAP Nible MACRO Q
       PUSH PSW      ; Push the Program Status Word (PSW) onto the stack
       MOV A,M       ; Move the value in memory location M to the accumulator
       RRC           ; Rotate accumulator right through carry
       RRC           ; Repeat the rotation
       RRC
       RRC           ; Rotate the accumulator four times (one for each nibble)
       MOV M,A       ; Move the modified accumulator value back to memory location M
       MOV A,Q       ; Move the value in Q to the accumulator
       RLC           ; Rotate accumulator left through carry
       RLC           ; Repeat the rotation
       RLC
       RLC           ; Rotate the accumulator four times (one for each nibble)
       MOV Q,A       ; Move the modified accumulator value back to Q
       POP PSW       ; Pop the PSW from the stack, restoring its original value
ENDM

; ================== Question (b) ==================

FILL MACRO RP, X, K
       PUSH PSW      ; Push the Program Status Word (PSW) onto the stack
       PUSH H        ; Push the register pair H onto the stack
       MOV H,R       ; Move the value in R to register H
       MOV L,P       ; Move the value in P to register L
LOOP:
       MOV M,K       ; Move the value in K to the memory location specified by HL
       INX H         ; Increment the value in register pair HL
       DCR X         ; Decrement the value in X
       JNZ LOOP      ; Jump to the LOOP label if X is not zero
       POP H         ; Pop the register pair H from the stack, restoring its original
                     ; value
       POP PSW       ; Pop the PSW from the stack, restoring its original value
ENDM
```



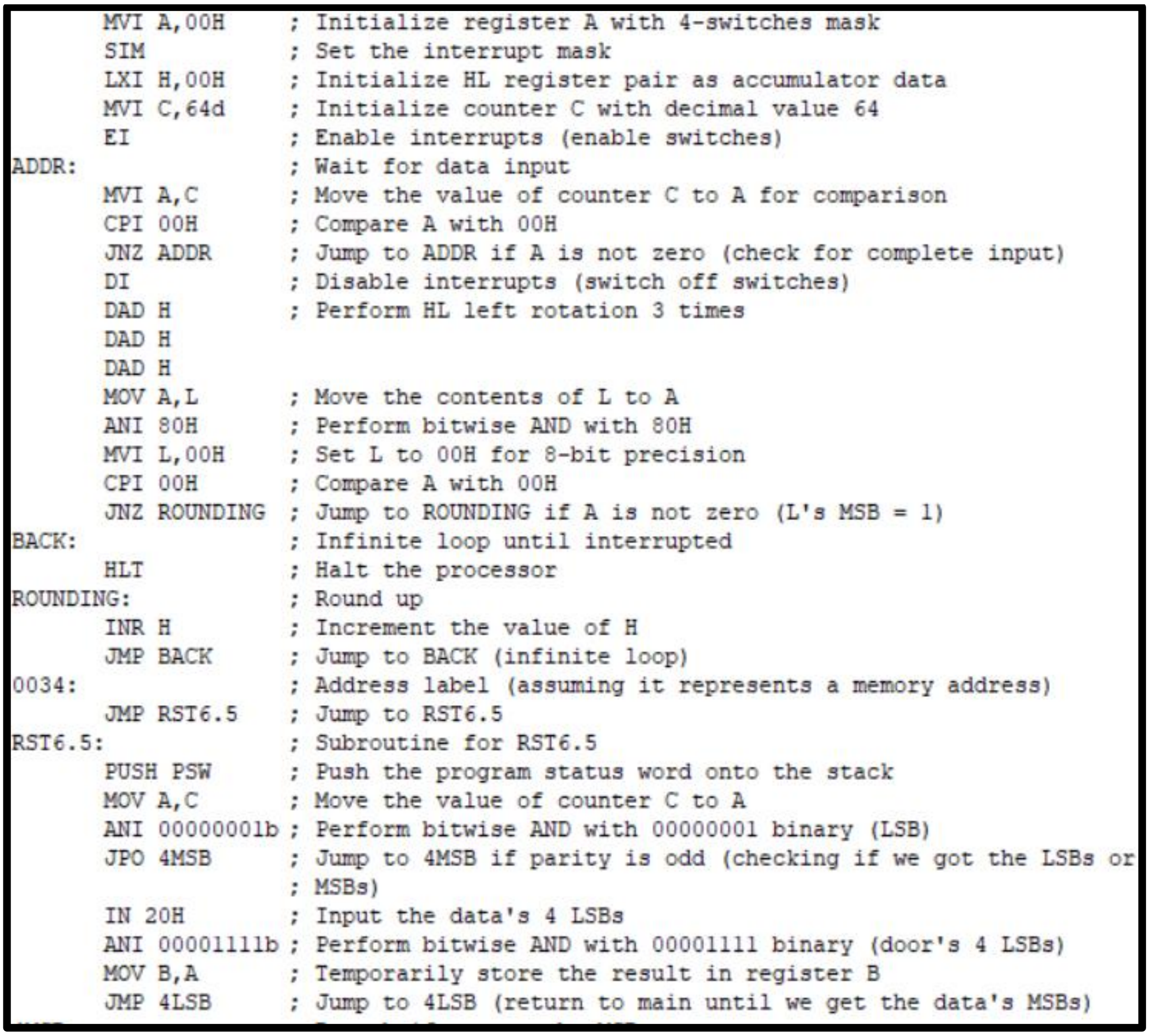
```
; ================== Question (c) ==================

RHLR MACRO n
       PUSH PSW      ; Push the Program Status Word (PSW) onto the stack
LOOP:
       MOV A,n       ; Move the value in n to the accumulator
       CPI 00H       ; Compare the accumulator with 0
       JZ END        ; If the result is zero, jump to the END label
       MOV A,H       ; Move the value in register H to the accumulator
       RAR           ; Rotate accumulator right through carry
       MOV A,L       ; Move the value in register L to the accumulator
       RAR           ; Rotate accumulator right through carry
       DCR n         ; Decrement the value in n
       JMP LOOP      ; Jump to the LOOP label
END:
       POP H         ; Pop the register pair H from the stack, restoring its original
                     ; value
       POP PSW       ; Pop the PSW from the stack, restoring its original value
ENDM
```

4η Άσκηση

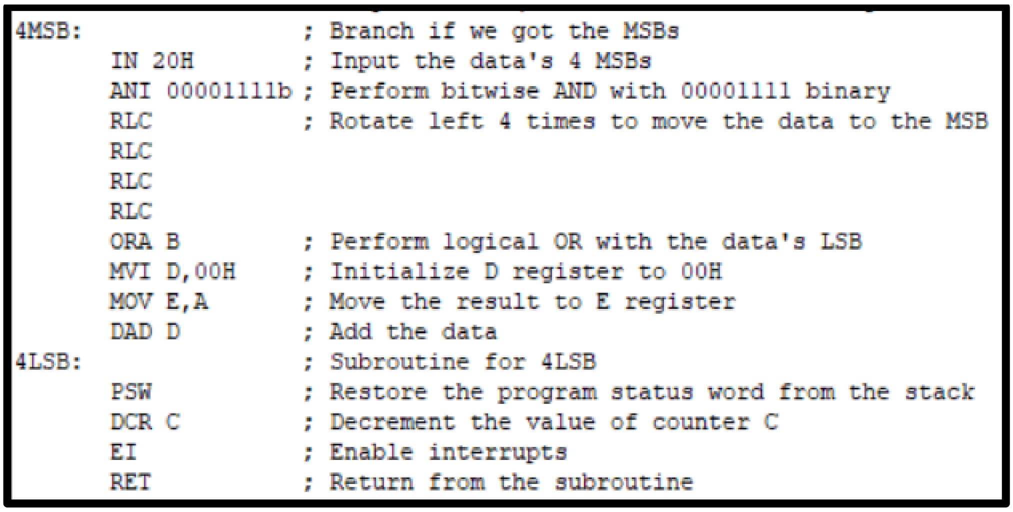
Ξεκινούμε με **αρχικές τιμές** για τον program counter **(PC)=0840H** και για τον stack pointer **(SP)=3000H**. Γίνεται η διακοπή RST 5.5 στο μέσο της εντολής CALL 0900H με αποτέλεσμα να συμβούν οι παρακάτω λειτουργίες:

1. **Εκτέλεση της εντολής CALL 0900H**: O (PC) παίρνει την διεύθυνση που υποδεικνύει η CALL άρα **(PC)=0900H**. Επιπλέον αποθηκεύει την διεύθυνση της επόμενης εντολής προς εκτέλεση(μετά την CALL) δηλαδή την 0843H(H CALL έχει μέγεθος 3 bytes) στην στοίβα μειώνοντας τον (SP) κατά 2 **(SP)=2FFEH**.Πιο συγκεκριμένα στην διεύθυνση 2FFEH θα μπει το 08H(high byte 0843H) και στην διεύθυνση 2FFFH θα μπει το 43H (low byte 0843H).
2. **Διακοπή RST 5.5**: O (PC) παίρνει την διεύθυνση της διακοπής RST 5.5 άρα **(PC)=002CH**. Αποθηκεύεται στην στοίβα η διεύθυνση 0900H(διεύθυνση που μας στέλνει η CALL) μειώνοντας τον (SP) κατά 2 **(SP)=2FFCH**. Πιο συγκεκριμένα στην διεύθυνση 2FFCH θα μπει το 09H(high byte 0900H) και στην διεύθυνση 2FFDH θα μπει το 00H(low byte 0900H).
3. **Τερματισμός Ρουτίνας RST 5.5**: Γίνεται POP από την στοίβα η τελευταία τιμή που κάναμε PUSH και ο (PC) παίρνει την τιμή αυτή άρα **(PC)=0900H**. Έτσι ο (SP) αυξάνεται κατά 2 **(SP)=2FFEH**.
4. **Τερματισμός Ρουτίνας στην διεύθυνση που μας έστειλε η CALL**: Γίνεται POP από την στοίβα η πρώτη τιμή που κάναμε PUSH(διεύθυνση συνέχειας του κύριου προγράμματος) και ο (PC) παίρνει την τιμή αυτή άρα **(PC)=0843H**. Έτσι ο (SP) αυξάνεται κατά 2 **(SP)=3000H**.
5. **Συνέχεια του κύριου προγράμματος**: Συνεχίζεται το κύριο πρόγραμμα μετά την εντολή CALL με **(PC)=0843H** και **(SP)=3000H**.

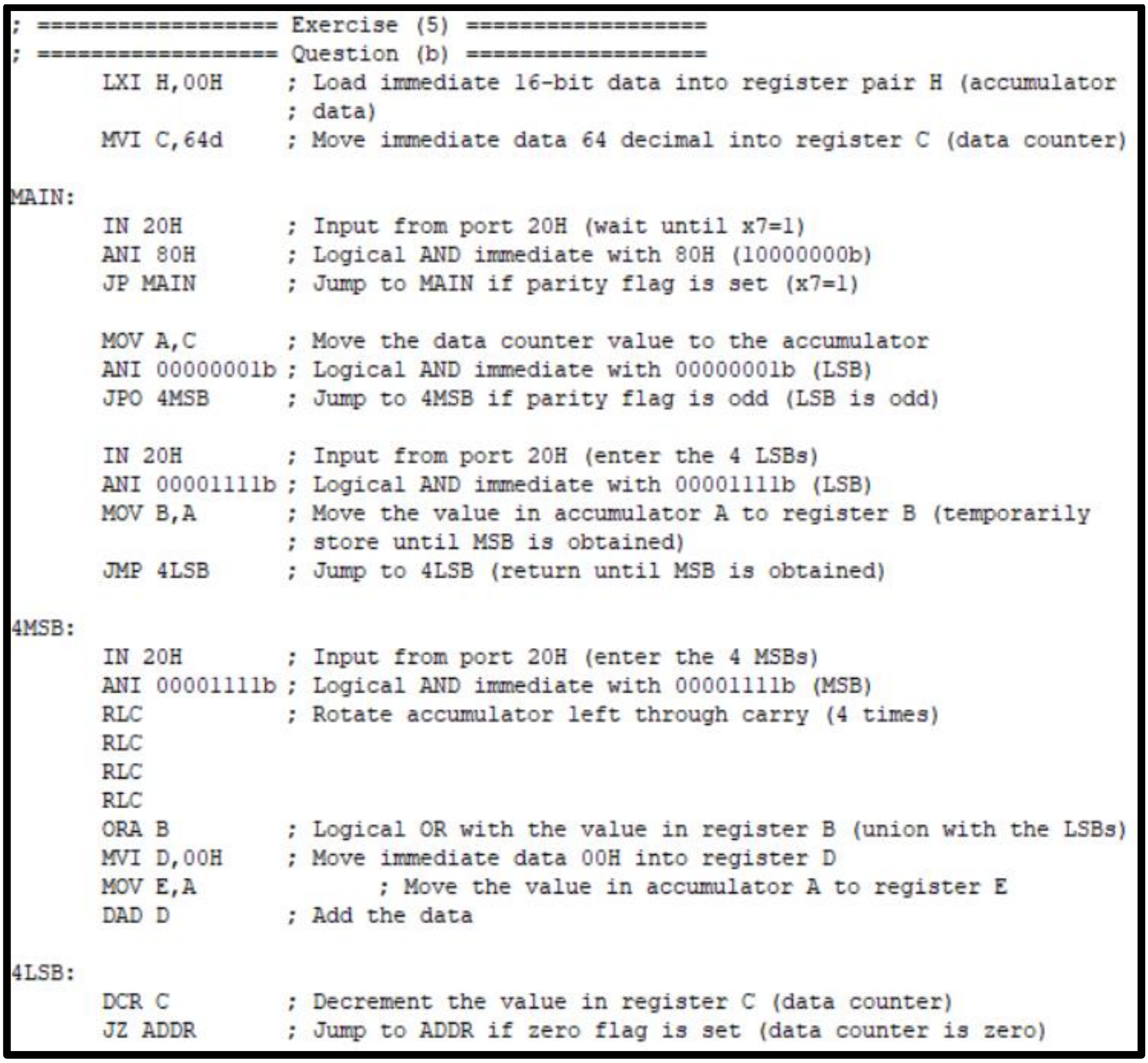
5η Άσκηση



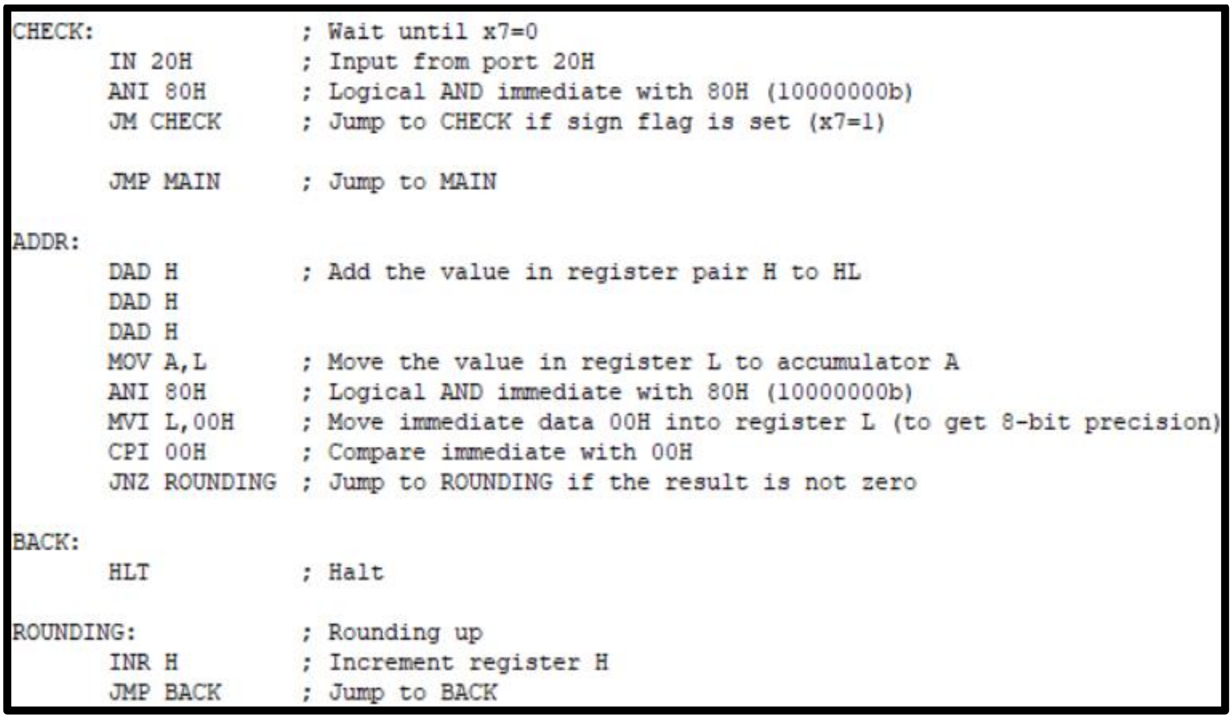
```
        MVI A,00H      ; Initialize register A with 4-switches mask
        SIM            ; Set the interrupt mask
        LXI H,00H      ; Initialize HL register pair as accumulator data
        MVI C,64d      ; Initialize counter C with decimal value 64
        EI             ; Enable interrupts (enable switches)
ADDR:                  ; Wait for data input
        MVI A,C        ; Move the value of counter C to A for comparison
        CPI 00H        ; Compare A with 00H
        JNZ ADDR       ; Jump to ADDR if A is not zero (check for complete input)
        DI             ; Disable interrupts (switch off switches)
        DAD H          ; Perform HL left rotation 3 times
        DAD H
        DAD H
        MOV A,L        ; Move the contents of L to A
        ANI 80H        ; Perform bitwise AND with 80H
        MVI L,00H      ; Set L to 00H for 8-bit precision
        CPI 00H        ; Compare A with 00H
        JNZ ROUNDING   ; Jump to ROUNDING if A is not zero (L's MSB = 1)
BACK:                  ; Infinite loop until interrupted
        HLT            ; Halt the processor
ROUNDING:              ; Round up
        INR H          ; Increment the value of H
        JMP BACK       ; Jump to BACK (infinite loop)
0034:                  ; Address label (assuming it represents a memory address)
        JMP RST6.5     ; Jump to RST6.5
RST6.5:                ; Subroutine for RST6.5
        PUSH PSW       ; Push the program status word onto the stack
        MOV A,C        ; Move the value of counter C to A
        ANI 00000001b  ; Perform bitwise AND with 00000001 binary (LSB)
        JPO 4MSB       ; Jump to 4MSB if parity is odd (checking if we got the LSBs or
                       ; MSBs)
        IN 20H         ; Input the data's 4 LSBs
        ANI 00001111b  ; Perform bitwise AND with 00001111 binary (door's 4 LSBs)
        MOV B,A        ; Temporarily store the result in register B
        JMP 4LSB       ; Jump to 4LSB (return to main until we get the data's MSBs)
```



```
4MSB:                  ; Branch if we got the MSBs
        IN 20H         ; Input the data's 4 MSBs
        ANI 00001111b  ; Perform bitwise AND with 00001111 binary
        RLC            ; Rotate left 4 times to move the data to the MSB
        RLC
        RLC
        RLC
        ORA B          ; Perform logical OR with the data's LSB
        MVI D,00H      ; Initialize D register to 00H
        MOV E,A        ; Move the result to E register
        DAD D          ; Add the data
4LSB:                  ; Subroutine for 4LSB
        PSW            ; Restore the program status word from the stack
        DCR C          ; Decrement the value of counter C
        EI             ; Enable interrupts
        RET            ; Return from the subroutine
```



```
; ================== Exercise (5) ==================
; ================== Question (b) ==================
       LXI H,00H     ; Load immediate 16-bit data into register pair H (accumulator
                     ; data)
       MVI C,64d     ; Move immediate data 64 decimal into register C (data counter)

MAIN:
       IN 20H        ; Input from port 20H (wait until x7=1)
       ANI 80H       ; Logical AND immediate with 80H (10000000b)
       JP MAIN       ; Jump to MAIN if parity flag is set (x7=1)

       MOV A,C       ; Move the data counter value to the accumulator
       ANI 00000001b ; Logical AND immediate with 00000001b (LSB)
       JPO 4MSB      ; Jump to 4MSB if parity flag is odd (LSB is odd)

       IN 20H        ; Input from port 20H (enter the 4 LSBs)
       ANI 00001111b ; Logical AND immediate with 00001111b (LSB)
       MOV B,A       ; Move the value in accumulator A to register B (temporarily
                     ; store until MSB is obtained)
       JMP 4LSB      ; Jump to 4LSB (return until MSB is obtained)

4MSB:
       IN 20H        ; Input from port 20H (enter the 4 MSBs)
       ANI 00001111b ; Logical AND immediate with 00001111b (MSB)
       RLC           ; Rotate accumulator left through carry (4 times)
       RLC
       RLC
       RLC
       ORA B         ; Logical OR with the value in register B (union with the LSBs)
       MVI D,00H     ; Move immediate data 00H into register D
       MOV E,A              ; Move the value in accumulator A to register E
       DAD D         ; Add the data

4LSB:
       DCR C         ; Decrement the value in register C (data counter)
       JZ ADDR       ; Jump to ADDR if zero flag is set (data counter is zero)
```



```
CHECK:               ; Wait until x7=0
       IN 20H        ; Input from port 20H
       ANI 80H       ; Logical AND immediate with 80H (10000000b)
       JM CHECK      ; Jump to CHECK if sign flag is set (x7=1)

       JMP MAIN      ; Jump to MAIN

ADDR:
       DAD H         ; Add the value in register pair H to HL
       DAD H
       DAD H
       MOV A,L       ; Move the value in register L to accumulator A
       ANI 80H       ; Logical AND immediate with 80H (10000000b)
       MVI L,00H     ; Move immediate data 00H into register L (to get 8-bit precision)
       CPI 00H       ; Compare immediate with 00H
       JNZ ROUNDING  ; Jump to ROUNDING if the result is not zero

BACK:
       HLT           ; Halt

ROUNDING:            ; Rounding up
       INR H         ; Increment register H
       JMP BACK      ; Jump to BACK
```