12/5/21

3η Ομάδα Ασκήσεων

Συστήματα Μικροϋπολογιστών

Βαρδάκης Χριστόφορος el18883
Λυμπεράκης Γεώργιος el18881

1η ΑΣΚΗΣΗ

Στην πρώτη άσκηση θα δημιουργήσουμε ένα πρόγραμμα σε Assembly 8085 με το οποίο μέσω της διακοπής RST 6.5 θα ελέγχουμε την ενεργοποίηση των LED της πόρτας εξόδου για διάρκεια ενός λεπτού καθώς και την ανανέωση του χρόνου παραμονής της λειτουργίας μέσω της χρήσης ίδιας διακοπής . Σε αυτή τη λειτουργία συμβάλουν και οι segment displays που μας ενημερώνουν για τον υπολειπόμενο χρόνο που τα LED θα είναι ανοιχτά.

```
;Exercise 1

;Main Function
MAIN:
      IN 10H

      MVI A,0DH
      SIM
      EI

;Wait for an Interrupt
WAIT:
      JMP WAIT

INTR ROUTINE:
      JMP RST6.5

;Routine that service the RST 6.5
RST6.5:
      PUSH PSW
      EI
      MVI A,00H
      STA 3000H

      MVI H,06H
      MVI L,00H

;Make the time and refresh it
DISPLAY_TIME:

      MVI C,AFH
L1:
      CALL MAKE_OUTPUT
      CALL DELA
      DCR C
      JNZ L1

      ;LXI B,05FFH
      ;CALL DELB

      MOV A,L
      CPI 00H
      JZ REFRESH
      DCR L
```

```
CONT:
      MOV A,H
      CPI 00H
      JZ CHECK
      JMP DISPLAY_TIME

;refresh the 2 counters
REFRESH:
      MOV L,A
      MOV A,H
      CPI 00H
      JZ EXIT
      MOV H,A
      MVI L,09H
      DCR H
      JMP CONT

;Check if the remainder time
;is zero
CHECK:
      MOV A,L
      CPI 00H
      JNZ DISPLAY_TIME

;If the remainder time is zero
;then it's time to switch off
;the leds and leave
EXIT:
      CMA
      STA 3000H
      POP PSW
      RET

;Print the output in the
;Seven Segment Displays
MAKE OUTPUT:
      PUSH PSW
      PUSH H

      LXI D,0A00H
      MVI A,10H
      STA 0A00H
      STA 0A01H
      STA 0A04H
      STA 0A05H

      MOV A,H
      STA 0A03H
      MOV A,L
      STA 0A02H

      CALL STDM
      CALL DCD

      POP H
      POP PSW

      RET
END
```

2η ΑΣΚΗΣΗ

Σε αυτή την άσκηση θα δημιουργήσουμε ένα πρόγραμμα σε Assembly 8085 με το οποίο μέσω της διακοπής RST 6.5 θα διαβάζει από το πληκτρολόγιο τα 2 ψηφία ενός αριθμού, και αφού τον απεικονήσει στην 7-segment display, ανάβει το πρώτο lsb αν ο αριθμος βρίσκετε στο [0,K1) το 2ο lsb αν βρίσκετε στο (K1,K2] και το 3ο lsb αν βρίσκετε στο (K2,FFH]. Για λόγους επαλήθευσης ορίσαμε ως K1 = 50H και K2 = A0H που χωρίζουν το [0,FFH] σε δύο περίπου ισόποσα μέρη. Ο Κώδικας φαίνετε παρακάτω:

```
;Exercise 2

;Main Function
MAIN:
      IN 10H
      MVI A,0DH
      SIM
      EI

      LXI D,0A00H ; empty 7segment display
      MVI A,10H
      STA 0A00H
      STA 0A01H
      STA 0A02H
      STA 0A03H
      STA 0A04H
      STA 0A05H
      CALL STDM
      CALL DCD

      MVI D,50H   ; set ?1 = 50H (sample value)
      MVI E,A0H   ; set K2 = A0H (sample value)

;Wait for an Interrupt
WAIT:
      JMP WAIT

INTR ROUTINE:
      JMP RST6.5

;Routine that service the RST 6.5
RST6.5:
      CALL KIND   ; read lsb
      MOV B,A     ; move lsb to B
      STA 0A00H
      CALL KIND   ; read msb
      STA 0A01H
      RLC
      RLC
      RLC
      RLC
      ADD B       ; create 2 digit number
      CMP D
      JC ZEROTOK1
      JZ ZEROTOK1
      JMP K1TOFFH
```

```
ZEROTOK1:          ;0 <= x <= K1
       MVI A,01H
       JMP PRINT

K1TOFFH:           ;K1 < x <=FFH
       CMP E
       JC K1TOK2
       JZ K1TOK2
       JMP K2TOFFH

K1TOK2:            ;K1 < x <=K2
       MVI A,02H
       JMP PRINT

K2TOFFH:           ;K1 < x <= FFH
       MVI A,04H
PRINT:
       CMA
       STA 3000H

       PUSH D
       LXI D,0A00H

       CALL STDM
       POP D
       EI

       CALL DCD

END
```

3η ΑΣΚΗΣΗ

α) Παρακάτω ακολουθεί η υλοποίηση της μακροεντολής SWAP Nible Q :

```
SWAP Nible MACRO Q

       PUSH PSW

       MOV A,Q
       RLC
       RLC
       RLC
       RLC
       MOV Q,A

       MOV A,M
       RLC
       RLC
       RLC
       RLC
       MOV M,A

       POP PSW

ENDM
```

β) Παρακάτω ακολουθεί η υλοποίηση της μακροεντολής FILL RP,X,K :

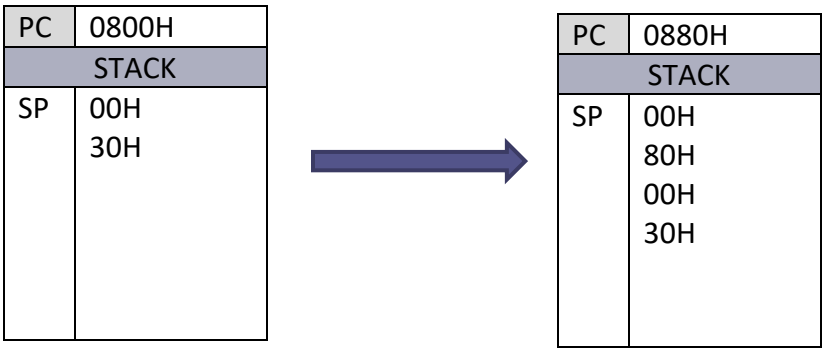
```
FILL MACRO RP, X, K

       PUSH PSW
       PUSH H
       PUSH R
       POP H
LOO:
       MOV A,K
       MOV M,A
       INX H
       DCR X
       JNZ LOO
       POP H
       POP PSW

ENDM
```

γ) Παρακάτω ακολουθεί η υλοποίηση της μακροεντολής RHLR n :

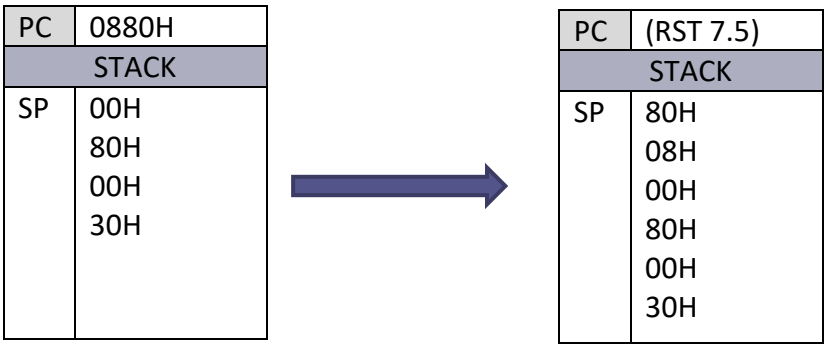
```
RHLR MACRO n

       PUSH PSW
       PUSH B

       MVI A,n
       CPI 00H
       JZ FINISH
       MVI B,n
LOO:
       MOV A,L
       RAR
       MOV L,A
       MOV A,H
       RAR
       MOV H,A
       DCR B
       JNZ LOO

FINISH:
       POP B
       POP PSW

ENDM
```

4η ΑΣΚΗΣΗ

Αφού η διακοπή συμβαίνει στο μέσο της εντολής CALL 0880H θα ολοκληρωθεί πρώτα η εκτέλεση της εντολής αυτής. Έτσι η τιμή του PC (0800H) θα αποθηκευτεί στο σωρό, ο SP θα ανέβει 2 θέσεις πάνω και στον PC θα καταχωρηθεί η τιμή 0880H .



Έπειτα για να εκτελεστεί η ρουτίνα εξυπηρέτησης της διακοπής RST 7.5 αποθηκεύεται ξανά ο PC στο σωρό και ο SP ανεβαίνει άλλες 2 θέσεις.



Αφού εκτελεστεί η ρουτίνα εξυπηρέτησης της διακοπής, η τιμή που βρίσκετε στην κορυφή του σωρού επανέρχεται στον PC και ο SP κατεβαίνει κατά 2 θέσεις.

PC	(RST 7.5)
STACK	
SP	80H
	08H
	00H
	80H
	00H
	30H

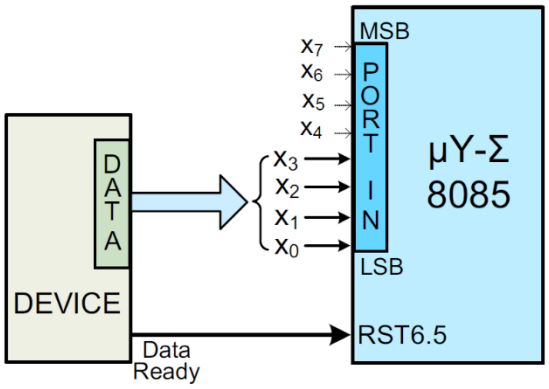
PC	0880H
STACK	
	~~80H~~
	~~08H~~
SP	00H
	80H
	00H
	30H

Τέλος, αφού εκτελεστεί η ρουτίνα που καλεί η CALL 0880H, η τιμή που βρίσκετε στην κορυφή του σωρού επανέρχεται στον PC και ο SP κατεβαίνει κατά 2 θέσεις. Έτσι το πρόγραμμα έχει επανέλθει στην αρχική του κατάσταση.

PC	0880H
STACK	
	~~80H~~
	~~08H~~
SP	00H
	80H
	00H
	30H

PC	0800H
STACK	
	~~80H~~
	~~08H~~
	~~00H~~
	~~80H~~
SP	00H
	30H

5η ΑΣΚΗΣΗ



Στόχος της παρακάτω προγράμματος είναι η μεταφορά 32 Byte από μια εξωτερική συσκευή μέσω της θύρας εισόδου 20^H χρησιμοποιώντας κάθε φορά μόνα τα 4 Bit της και με την χρήση της διακοπής RST 6.5 ώστε να ενημερωνόμαστε για την άφιξη νέου δεδομένου .

Επομένως, έχουμε τον παρακάτω κώδικα

α)

```
;Exercise 5

;Main Function
START:
      MVI A,0DH
      SIM
      MVI C,40H
      LXI H,0000H
      EI

;Wait for the 64 interrupts
;and after find the average value
;and pause
ADDR:
      MOV A,C
      CPI 00H
      JNZ ADDR

      MOV A,L
      ANI 1FH
      MOV L,A

      DI
      DAD H
      DAD H
      DAD H

      HLT
```

```
INTR ROUTINE:
      JMP RST6.5

;Routine for the RST6.5
RST6.5:
      PUSH PSW

      MOV A,C
      ANI 01H
      CPI 00H
      JNZ READ 4MSB
      JMP READ_4LSB

;Read the 4 MSB
READ 4MSB:
      IN 20H
      ANI 0FH
      RLC
      RLC
      RLC
      RLC

      ORA B
      MVI D,00H
      MOV E,A
      DAD D

      JMP EXIT

;Read the 4 LSB
READ 4LSB:
      IN 20H
      ANI 0FH
      MOV B,A

;Decrease the counter and return to
;waiting loop
EXIT:
      DCR C

      POP PSW
      EI
      RET
END
```

β) Τώρα, θα αντιμετωπίσουμε το ίδιο πρόβλημα αντικαθιστώντας τις διακοπές με τον έλεγχο απλά του τελευταίου Bit της θύρας εισόδου .Οπότε, ο κώδικας Assembly που υλοποιεί την παραπάνω απαίτηση είναι ο κάτωθι

```
;Exercise 5

START:
      MVI C,40H

READING:
      MOV A,C
      CPI 00H
      JZ EXIT

WAIT ONE:          ;wait for bit = 1
      LDA 2000H
      MOV B,A
      ANI 80H
      CPI 80H
      JNZ WAIT_ONE

      MOV A,C
      ANI 01H
      CPI 00H
      JNZ READ 4MSB
      JMP READ_4LSB

;Read the 4 MSB
READ 4MSB:
      LDA 20H
      ANI 0FH
      RLC
      RLC
      RLC
      RLC

      ORA B
      MVI D,00H
      MOV E,A
      DAD D

      JMP WAIT_ZERO

;Read the 4 LSB
READ 4LSB:
      LDA 20H
      ANI 0FH
      MOV B,A

WAIT ZERO:              ;wait for bit = 0
      LDA 20H
      ANI 80H
      CPI 00H
      JNZ WAIT_ZERO

      DCR C
      JMP READING

EXIT:
      MOV A,L
      ANI 1FH
      MOV L,A

      DAD H
      DAD H
      DAD H
      HLT

END
```