3η Ομάδα Ασκήσεων
Συστήματα Μικροϋπολογιστών

Ανδρέας Στάμος
Αριθμός Μητρώου: *03120****

Μάιος 2023

Περιεχόμενα

1 Άσκηση 1

```
IN 10H ;disable memory protection

MVI A,10H
STA 0900H ;zeroing the display message
STA 0901H
STA 0902H
STA 0903H
STA 0904H
STA 0905H

;enabling the RST6.5 interrupt
MVI A,0DH
SIM
```

```
14  EI
15
16  DO_NOTHING:
17  JMP DO_NOTHING
18
19  INTR_ROUTINE:
20  POP H ;the return address wont be required and is without reason
     ↪  filling the stack
21  EI ;reenabling the interrput
22  MVI A,00H
23  STA 3000H ;turn on the leds
24  MVI H,06H ;H stores the tens of seconds
25  MVI L,00H ;L stores one ten of seconds
26
27  SECOND_LOOP:
28  MOV A,L
29  DCR A ;decreasing by 1 sec
30  CPI FFH
31  JZ DCR_TENS ;if went to -1 sec then the tens need to be decreased
32  JMP L1
33
34  DCR_TENS:
35  MVI A,09H ;setting the units to 9
36  MOV L,A
37  STA 0900H
38  MOV A,H
39  DCR A
40  CPI FFH ;if tens went to -1 then 60secs are over
41  JZ EXIT_LOOP
42  MOV H,A
43  STA 0901H
44  JMP L2
45
46  L1:
47  MOV L,A
48  STA 0900H
49
50  L2:
51  PUSH H
52  LXI D,0900H ;address of display message
53  CALL STDM ;store display message to appropriate address
54  POP H
55  LXI B,0064H ;100ms
56
57  MVI A,0AH ;refresh display 10 times (1000ms / 100ms per refresh)
58
59  REFRESH_DISPLAY:
60  ;refresh display every 100ms
61  CALL DELB
62  CALL DCD ;display the display message
```

```
DCR A
JNZ REFRESH_DISPLAY
JMP SECOND_LOOP

EXIT_LOOP:
MVI A,FFH
STA 3000H ;turn off the leds;
JMP DO_NOTHING ;go back to waiting
END
```

2 Άσκηση 2

```
IN 10H ;disabling memory protection

;saving k1,k2 to memory
MVI A,10H ;K1 = 10H
STA 0906H
MVI A,20H ;K2 = 20H
STA 0907H

MVI A,10H ;initializing display to nothing
STA 0900H
STA 0901H
STA 0902H
STA 0903H
STA 0904H
STA 0905H

LXI D,0900H
CALL STDM

MVI A,0DH ;enabling interruputs
SIM
EI

DO_NOTHING: ;while no interrupt just refresh the display
CALL DCD
JMP DO_NOTHING

INTR_ROUTINE:

EI ;reenabling interrputs

;loading k1,k2 from memory to registers D,E
LDA 0906H
MOV D,A
LDA 0907H
MOV E,A

```

```
38  POP H ;the return address wont be required and is without reason
      ↪ filling the stack
39  CALL KIND ;inputting MSB hex digit from keyboard to A
40  STA 0905H ;saving for display
41  RLC ;taking the MSB hex digit to its correct position
42  RLC
43  RLC
44  RLC
45  MOV B,A
46  CALL KIND ;inputting LSB hex digit from keyboard
47  STA 0904H
48  ORA B ;building the inputted number in binary.
49
50  CMP E ;A > K2 ?
51  JNC CASE3
52
53  CMP D ; A > K1 ?
54  JNC CASE2
55
56  ;case1 (A <= K1 for sure and A>=0 by construction)
57  MVI A,7FH
58  JMP CONTINUE
59
60  CASE2:
61  MVI A,BFH
62  JMP CONTINUE
63
64  CASE3:
65  MVI A,DFH
66  JMP CONTINUE
67
68  CONTINUE:
69  STA 3000H ;output to leds
70
71  PUSH D
72  LXI D,0900H
73  CALL STDM ;moving display message to appropriate address
74  POP D
75
76  JMP DO_NOTHING ;start over
77
78  END
```

3 Άσκηση 3

3.1 Ερώτημα α

```
1  SWAP_Nible MACRO Q:
2
```

```
PUSH PSW

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

POP PSW

ENDM
```

3.2 Ερώτημα β

```
FILL MACRO RP,X,K

PUSH PSW
PUSH H

MOV H,R
MOV L,P
MVI A,X

BYTE:
MVI M,K
INX H
DCR A
JNZ BYTE

POP H
POP PSW

ENDM
```

3.3 Ερώτημα γ

```
RHLR MACRO

PUSH B ;using B to store A instead of push PSW in order to be able
 ↪  to update the CY flag
MOV B,A

MOV A,H
RRC ;lsb of H to CY
MOV A,L
RAR ;rotate right using MSB as CY (CY is the lsb of H). old lsb to
 ↪  CY.
MOV L,A
MOV A,H
```

```
12 RAR ;rotate right using MSB as CY (CY is the old lsb of L).
13 MOV H,A
14
15 RLC ;MSB of new H to CY
16
17 MOV A,B
18 POP B
19
20 ENDM
```

4 Άσκηση 4

Προτού γίνει jump στην ρουτίνα εξυπηρέτησης διακοπής, ολοκληρώνεται η τρέχουσα εντολή, δηλαδή η CALL.

Συγκεκριμένα στις θέσεις μνήμης 2FFFH, 2FFEH (SP-1, SP-2) τοποθετούνται οι αριθμοί 08H και 03H. Οι αριθμοί 08H, 03H ενωμένοι (0803H) αποτελούν την διεύθυνση της επόμενης εντολής της CALL , που επειδή η CALL έχει μέγεθος 3 bytes, είναι η 0803H (PC + 3). Επίσης ο καταχωρητής στοίβας SP αποκτά τιμή 2FFEH (SP-2) και επιπλέον ο καταχωρητής PC αποκτά τιμή 0900H.

Αφού συμβούν αυτά, δηλαδή έχει ολοκληρωθεί η εντολή CALL, ξεκινά η εξυπηρέτηση της διακοπής. Στις θέσεις μνήμης 2FFDH, 2FFCH (SP-1, SP-2) τοποθετούνται οι αριθμοί 09H και 00H (τρέχουσα τιμή του PC), ο καταχωρητής στοίβας SP αποκτά τιμή 2FFCH (SP-2) και επιπλεόν ο καταχωρητής PC αποκτά τιμή 002CH (διεύθυνση μνήμης που αντιστοιχεί στην ρουτίνα εξυπηρέτησης διακοπής RST 5.5).

Πλεόν η ΚΜΕ εκτελεί τον κώδικα της ρουτίνας εξυπηρέτησης της διακοπής. Ο προγραμματιστής οφείλει όταν ολοκληρωθεί η διακοπή να τοποθετήσει την εντολή RET που θα επιστρέψει πίσω στην κανονική ροή εκτέλεσης του προγραμματός. Συγκεκριμένα, όταν κληθεί η RET ο καταχωρητής PC λαμβάνει την τιμή 0900H, που ειναι η τιμή που έχει αποθηκευτεί στις θέσεις μνήμης SP, SP+1 και επιπλέον ο καταχωρητής στοίβας SP αποκτά τιμή 2FFEH (SP+2).

Η επόμενη εντολή που εκτελεί πλέον η ΚΜΕ είναι της κανονικής ροής του προγράμματος, δηλαδή στην διεύθυνση 0900H όπως είχε απαιτήσει η αρχική εντολή CALL.

5 Άσκηση 5

5.1 Ερώτημα α

```
1  IN 10H ;disable memory protection
2
3  LXI H,0000H ;HL stores the total sum
4
5  MOV B,40H ;data counter
6
7  MOV A,0BH ;enabling interrupt rst7.5
8  SIM
9  EI
10
```

```
MAIN:
MOV A,B
ANA A ;update the flags
JZ COMPLETE ;waiting until all data has been inputed
JMP MAIN

COMPLETE:
DI
MOV A,L
RLC ;checking if rounding up required
JC ROUND_UP
EXIT:
HLT ;result in H
ROUND_UP:
INR H
JMP EXIT

RST6.5:
PUSH PSW
;check if receiving MSBs or LSBs. (if d is even receive lsbs)
MOV A,B
RRC
JC MSBS

;receive lsbs
IN 20H
ANI 0FH
MOV A,E
DCR B
POP PSW
EI
RET

MSBS:
;receive msbs
IN 20H
ANI 0FH
RLC
RLC
RLC
RLC
ORA E ;building the number
MOV E,A
MOV D,00H
DAD D ;adding DE (D=0, E=input byte) to results
DCR B
POP PSW
EI
RET
```

5.2 Ερώτημα β

```
IN 10H ;disable memory protection

LXI H,0000H ;HL stores the total sum

MOV B,40H ;data counter

MAIN:
MOV A,B
ANA A ;update the flags
JZ COMPLETE ;checking if all data has been inputed
IN 20H
MOV C,A ;storing the input to C
RLC
JC RECEIVE
JMP MAIN

COMPLETE:
MOV A,L
RLC ;checking if rounding up required
JC ROUND_UP
EXIT:
HLT ;result in H
ROUND_UP:
INR H
JMP EXIT

RECEIVE:
;check if receiving MSBs or LSBs. (if d is even receive lsbs)
MOV A,B
RRC
JC MSBS

;receive lsbs
MOV A,C
ANI 0FH
MOV A,E
DCR B
JMP MAIN

MSBS:
;receive msbs
MOV A,C
ANI 0FH
RLC
RLC
RLC
RLC
ORA E ;building the number
```

```
49  MOV E,A
50  MOV D,00H
51  DAD D ;adding DE (D=0, E=input byte) to results
52  DCR B
53  JMP MAIN
```