Νεκτάριος Μπούμπαλος 20441

Δανάη Σπέντζου 20237

1η Άσκηση

α)

```
IN 10H
  MVI A,OOH
  LXI H,0900H
  MOV M,A
STORE_LOOP:
  INR A
  INX H
  MOV M,A
  CPI 7FH
  JNZ STORE_LOOP
```

Η πρώτη εντολή IN 10H απενεργοποιεί την προστασία της μνήμης. Στη συνέχεια, η εντολή MVI A,OOH αποθηκεύει την τιμή 00H στον επεξεργαστή. Η εντολή LXI H,0900H αποθηκεύει τη διεύθυνση 0900H στους επεξεργαστές HL. Στη συνέχεια, η εντολή MOV M,A αποθηκεύει την τιμή του A στη διεύθυνση που κατέχουν οι επεξεργαστές HL.

Η διαδικασία αυτή επαναλαμβάνεται σε μια επαναληπτική δομή μέχρι να αποθηκευθούν όλες οι τιμές από το 0 έως το 255 με φθίνουσα σειρά στη διαδοχικές θέσεις της μνήμης από τη διεύθυνση 0900H και κάθε φορά αυξάνοντας κατά ένα τη διεύθυνση και την τιμή του A. Η εντολή CPI 7FH ελέγχει αν η τιμή του A είναι μικρότερη από τη μέγιστη τιμή 7FH (ή 127 στο δεκαδικό σύστημα) και η εντολή JNZ STORE_LOOP επαναλαμβάνει τη διαδικασία αποθήκευσης της τιμής στην επόμενη θέση μνήμης μέχρι να αποθηκευτούν όλες οι τιμές στην μνήμη.

β)

```
        LXI B,0000H
LOAD:
MOV A, M
MVI D,09H
ALL_DIGITS:
DCR D
JZ NEXT
RRC
JNC ALL_DIGITS
COUNT_ONES:
INX B
JMP ALL_DIGITS
NEXT:
DCR L
JNZ LOAD
```

γ)

```
MVI E, FFH
MVI D,00H
MOV A, M
STATEMENT:
CPI 10H
JC DONT_COUNT
CPI 61H
JNC DONT_COUNT
```

```
INR D
DONT_COUNT:
INR L
MOV A, M
DCR E
JZ CHECK_LAST
JMP STATEMENT
CHECK_LAST:
CPI 10H
JC END
CPI 61H
JNC END
INR D
END:
END
```

2η Άσκηση

```
LXI B,0064H    ; Load value 0064H into register pair B (B = 00H, C = 64H)
START:         ; Label for the start of the program
 LDA 2000H     ; Load input from dip switches into accumulator
 RLC           ; Rotate accumulator left, which places the value of the MSB in the carry flag
 JNC OFF       ; If the MSB is not set, jump to label OFF
 JMP START     ; Otherwise, jump back to the START label to check again

OFF:           ; Label for when the MSB switch is off
```

```
 LDA 2000H    ; Load input from dip switches into accumulator
 RLC          ; Rotate accumulator left, which places the value of the MSB in the carry flag
 JC ON1       ; If the MSB is set (i.e. the switch is turned on), jump to label ON1
 JMP OFF      ; Otherwise, continue waiting until the switch is turned on

ON1:          ; Label for when the MSB switch is on
 MVI D, C8H   ; Load the value C8H (200 decimal) into register D, which will be used for the delay loop
 LDA 2000H    ; Load input from dip switches into accumulator
 RLC          ; Rotate accumulator left, which places the value of the MSB in the carry flag
 JNC OPEN     ; If the MSB is turned off, the push-button (off-on-off) is activated, jump to label OPEN
 JMP ON1      ; Otherwise, continue waiting until the switch is turned off

OPEN:         ; Label for when the push-button is activated
 LDA 2000H    ; Load input from dip switches into accumulator
 RLC          ; Rotate accumulator left, which places the value of the MSB in the carry flag
 JC ON_AGAIN ; If the MSB is still on, the timer keeps on going, jump to label ON_AGAIN
 JMP OPEN     ; Otherwise, continue waiting until the switch is turned on again

ON_AGAIN:     ; Label for resetting the timer when the MSB switch is turned off
 LDA 2000H    ; Load input from dip switches into accumulator
 RLC          ; Rotate accumulator left, which places the value of the MSB in the carry flag
 JNC RESTART ; If the MSB is turned off, jump to label RESTART to reset the timer
 MVI A,00H    ; Load the value 00H into the accumulator
 STA 3000H    ; Store the value in the accumulator to the memory location 3000H, which turns on all
LEDs
 CALL DELB    ; Call subroutine DELB to delay for 1/10 seconds
 DCR D        ; Decrement the value of register D
 JNZ ON_AGAIN ; If D is not zero, jump back to label ON_AGAIN
 MVI A,FFH    ; Load the value FFH into the accumulator
```

```
  STA 3000H   ; Store the value in the accumulator to the memory location 3000H, which turns off all LEDs
  JMP OFF     ; Jump to label OFF to start waiting for the MSB switch to be turned on again

RESTART:     ; Label for resetting the timer when the MSB switch is turned off during LED display
  MVI D, C8H  ; Load the value C8H (200 decimal) into register D
  JMP OPEN   ;
```

3η Άσκηση

i)

```
START:
MVI D,08H        ;D = 8
LDA 2000H        ;Load input from dip switches to A
MVI B,00H        ;B = 0
CHECK:                          ;Starting from LSB to MSB we find the first
;dip switch that's on
RRC
DCR D            ;Decrease D
JZ TURNOFF       ;If D = 0 then no dip switch was on so turn off
;all LEDs and start again
INR B            ;Increase B (B is equal to the current position that
;we're checking)
JNC CHECK        ;If a dip switch is on then stop looping
MVI A,FEH
DCR B
TURN_ON:
RLC              ;Rotate left until we reach the correct position
```

```
DCR B
JNZ TURN_ON
STA 3000H       ;Turn on the LED
JMP START       ;Start checking again
TURNOFF:                    ;Getting here means that no dip switch was on
MVI A, FFH
STA 3000H       ;Turn off all LEDs
JMP START

END
```

ii)

```
START:
CALL KIND
CPI 00H ;If we press 0 then go to OFF
JZ OFF
CPI 09H ;If we press 9 then go to OFF
JNC OFF
MOV B,A         ;Save A to register B
MVI A,00H       ;A = 0
DCR B           ;Decrease B
JZ OPEN         ;If B = 0 then open all LEDs (We pressed  1)
INR A           ;Increase A
REPEAT:
DCR B
JZ OPEN         ;If B = 0 then go to OPEN with the current data of A
RLC             ;Rotate left
INR A
```

```
JMP REPEAT
OPEN:
STA 3000H       ;Open the LEDs starting from the number that we pressed
;up to the MSB
JMP START       ;Start checking again
OFF:                    ;If we pressed 0 or 9 then turn off all LEDs and go to START
MVI A,FFH
STA 3000H
JMP START

END
```

iii)

```
START:
IN 10H
LXI H,0A00H
MVI B,04H
L1:
MVI M,10H
INX H
DCR B
JNZ L1
LINE0:
MVI A,FEH
STA 2800H
LDA 1800H
ANI 07H
```

```
MVI C,86H
CPI 06H

JZ SHOW

MVI C,85H
CPI 05H

JZ SHOW

LINE1:
MVI A,FDH
STA 2800H
LDA 1800H
ANI 07H
MVI C,84H
CPI 06H ; RUN
JZ SHOW
MVI C,80H
CPI 05H ; FETCH_REG
JZ SHOW
MVI C,82H
CPI 03H ; FETCH_ADDRS
JZ SHOW
LINE2:
MVI A,FBH
STA 2800H
LDA 1800H
ANI 07H
```

```
MVI C,00H
CPI 06H ; 0
JZ SHOW
MVI C,83H
CPI 05H ; STORE/INCR
JZ SHOW
MVI C,81H
CPI 03H ; DECR
JZ SHOW
LINE3:
MVI A,F7H
STA 2800H
LDA 1800H
ANI 07H
MVI C,01H       ; 1
CPI 06H
JZ SHOW
MVI C,02H       ; 2
CPI 05H
JZ SHOW
MVI C,03H       ; 3
CPI 03H
JZ SHOW
LINE4:
MVI A,EFH
STA 2800H
LDA 1800H
ANI 07H
MVI C,04H
```

```
CPI 06H ; 4
JZ SHOW
MVI C,05H
CPI 05H ; 5
JZ SHOW
MVI C,06H
CPI 03H ; 6
JZ SHOW
LINE5:
MVI A,DFH
STA 2800H
LDA 1800H
ANI 07H
MVI C,07H
CPI 06H ; 7
JZ SHOW
MVI C,08H
CPI 05H ; 8
JZ SHOW
MVI C,09H
CPI 03H ; 9
JZ SHOW
LINE6:
MVI A,BFH
STA 2800H
LDA 1800H
ANI 07H
MVI C,0AH
CPI 06H ; A
```

```
JZ SHOW
MVI C,0BH
CPI 05H ; B
JZ SHOW
MVI C,0CH
CPI 03H ; C
JZ SHOW
LINE7:
MVI A,7FH
STA 2800H
LDA 1800H
ANI 07H
MVI C,0DH
CPI 06H ; D
JZ SHOW
MVI C,0EH
CPI 05H ; E
JZ SHOW
MVI C,0FH
CPI 03H ; F
JZ SHOW
JMP START
SHOW:
LXI H,0A04H
MOV A,C
ANI 0FH
MOV M,A

INX H
```

```
MOV A,C
ANI F0H
RLC
RLC
RLC
RLC
MOV M,A

LXI D,0A00H

CALL STDM
CALL DCD
        JMP START

END
```

4η Άσκηση

```
START:
LDA 2000H       ;Load input from dip switches to A
MOV B,A         ;Save A to register B
A0_B0:
ANI 01H         ;A = A AND 00000001
MOV C,A         ;C = A
MOV A,B         ;A is equal to the input of dip switches
ANI 02H         ;A = A AND 00000010
AND0:
```

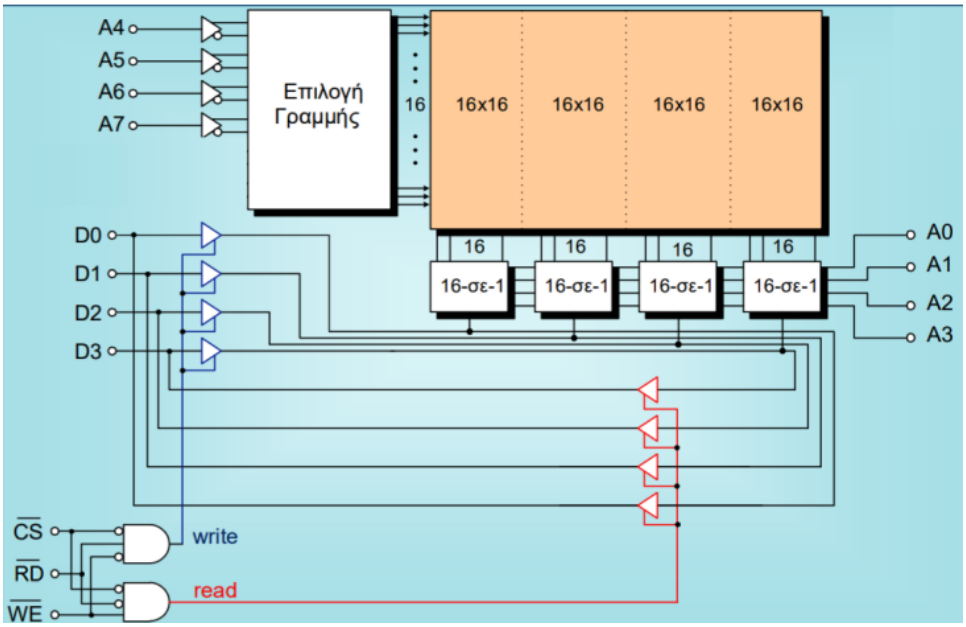
```
                RRC
                ANA C
                MOV D,A
A1_B1:
MOV A,B
ANI 04H         ;A = A AND 00000100
MOV C,A         ;C = A
MOV A,B
ANI 08H         ;A = A AND 00001000
AND1:
            RRC
                ANA C
                RRC
                MOV E,A
                RRC
                ORA E           ;Save X1 at 2nd LSB
MOV D,A         ;Save X0 at LSB


A2_B2:
MOV A,B
ANI 10H         ;A = A AND 00010000
MOV C,A         ;C = A
MOV A,B
ANI 20H         ;A = A AND 00100000
XOR0:
RRC
                XRA C
                MOV E,A
```

```
A3_B3:
MOV A,B
ANI 40H          ;A = A AND 01000000
MOV C,A          ;C = A
MOV A,B
ANI 80H          ;A = A AND 10000000
XOR1:
                 RRC
                 XRA B
                 RRC
                 RRC
                 MOV B,A
                 RRC
                 ORA D
                 MOV D,A   ;X3 αποθηκεύτηκε
OR_X2:
                  MOV A,B
                  ORA E
                  RRC
                  RRC
                  ORA D   ; X2 αποθηκεύτηκε στον A
LED:
CMA              ;Inverse logic
STA 3000H        ;Turn on the correct LEDs
JMP START        ;Start checking again

END
```

Άσκηση 5η



Παρατηρούμε την εσωτερική δομή ενός SRAM 256x4 bit (16x16x4), όπου η πρόσβαση στις διάφορες γραμμές του πίνακα μνήμης γίνεται με βάση τις γραμμές διεύθυνσης A4-A7. Τέσσερις πολυπλέκτες 16-σε-1 επιλέγουν μία από τις 16 τετράδες-στήλες του πίνακα, βασιζόμενοι στις γραμμές διευθύνσεων A0-A3 και την επιλεγμένη γραμμή του πίνακα. Έτσι, μπορούμε είτε να εγγράψουμε τα δεδομένα D0-D3 στις επιλεγμένες θέσεις του πίνακα, είτε να διαβάσουμε τα δεδομένα από αυτές.

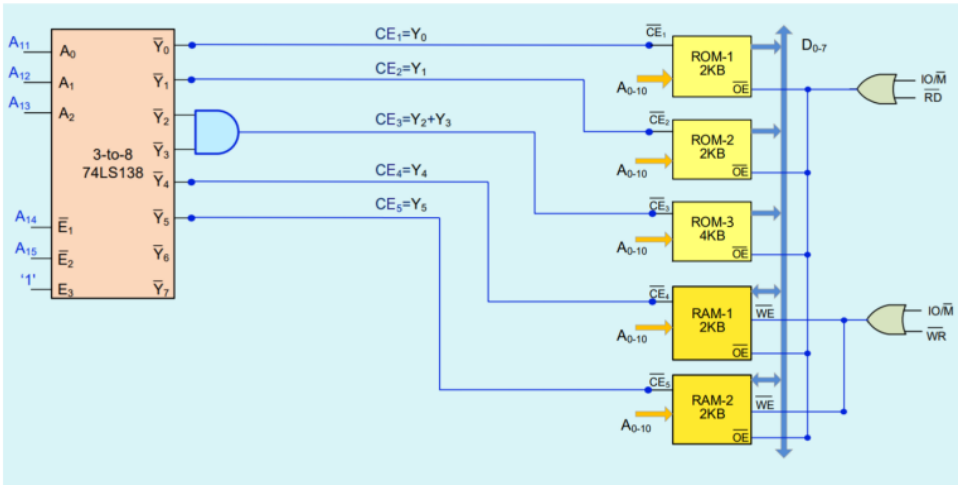
Για παράδειγμα, αν έχουμε μια διεύθυνση A0…A7 = 0110 1001, επιλέγουμε την 9η γραμμή και την 6η τετράδα του πίνακα μνήμης.

Όταν το σήμα CS (Chip Select) γίνει 0, ενεργοποιείται η λειτουργία της μνήμης. Στη συνέχεια, αν το σήμα WE (Write Enable) γίνει 0 και το σήμα RD (Read) γίνει 1, τότε ενεργοποιούνται οι απομονωτές με μπλε περίγραμμα και γίνεται εγγραφή στη μνήμη. Αντίθετα, αν το σήμα WE (Write Enable) γίνει 1 και το σήμα RD (Read) γίνει 0, τότε ενεργοποιούνται οι απομονωτές με κόκκινο περίγραμμα και γίνεται ανάγνωση από τη μνήμη.

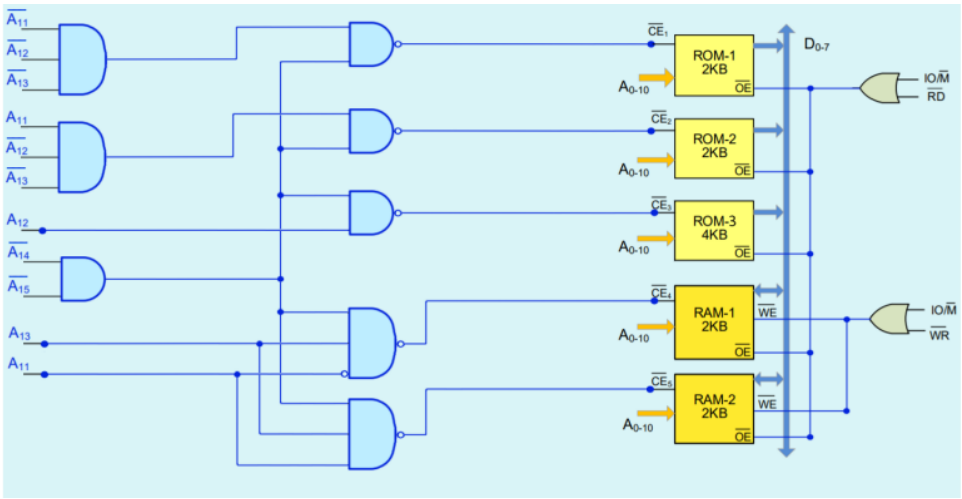
Οι απομονωτές με μπλε περίγραμμα επιτρέπουν την εγγραφή δεδομένων στην επιλεγμένη θέση της μνήμης, ενώ οι απομονωτές με κόκκινο περίγραμμα επιτρέπουν την ανάγνωση δεδομένων από την επιλεγμένη θέση της μνήμης. Με βάση την κατάσταση αυτών των σημάτων, επιλέγεται η σωστή λειτουργία (εγγραφή ή ανάγνωση) για τη μνήμη.

Άσκηση 6η

A)



B)



7Η ΑΣΚΗΣΗ

A)

Άσκηση 7η

Χαρτης Μνημης:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Memory
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000	ROM1-8K
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF	
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000	RAM1-1K
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	2FFF	
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	3000	RAM2-1K
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFF	
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000	RAM3-1K
0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	4FFF	
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	5000	ROM1-1K
0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	5FFF	
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	6000	ROM2-1K
0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	6FF	

B)

Συστήμα Μνήμης: