Ροή Υ :: Υπολογιστικά Συστήματα

Ακαδημαϊκό Έτος 2020-2021.

Βικέντιος Βιτάλης el18803, Αριάδνη Καζδάγλη el18838

2η Σειρά Ασκήσεων Συστήματα Μικροϋπολογιστών

Άσκηση 1:

Ex1 - TSIK

```
        IN 10H
        MVI A,00H     ; Numbers for A from 0 to 255
        LXI H,0900H   ; Adresses
        LXI B,0000H   ; Units
        MVI D,00H     ; Numbers 10Hex to 60Hex
BEGIN:
        MOV M,A       ; Store in memory
        MOV E,A       ; Temporary storage
        JMP ONES
ONESCOUNTER:
        MOV A,E
        JMP NUMBERS
COUNTINGNUMBERS:
        INX H         ; Next position in memory
        INR A         ; Next number
        CPI 00H       ; Is A accumulator equal to 00H?
        JZ FINISH     ; Check if the procedure is over
        JMP BEGIN

ONES:                 ; Couning units (binary ones)
        STC           ; CY=0
        CMC
        RAR           ; Bit by bit on CY
        JNC ONE       ; Is this bit a unit?
        INX B
ONE:
        CPI 00H         ; Is A accumulator equal to 00H?
        JZ ONESCOUNTER  ; Check if the procedure is over
        JMP ONES

NUMBERS:                    ; Counting numbers 10Hex to 60Hex
        CPI 10H
        JC COUNTINGNUMBERS  ; Is A smaller than 10H?
        CPI 60H
        JC NUMBER           ; Is A smaller than 60H?
        JNZ COUNTINGNUMBERS  ; Is A equal to 60H?
NUMBER:
        INR D
        JMP COUNTINGNUMBERS
FINISH:
        END
```

Άσκηση 2:

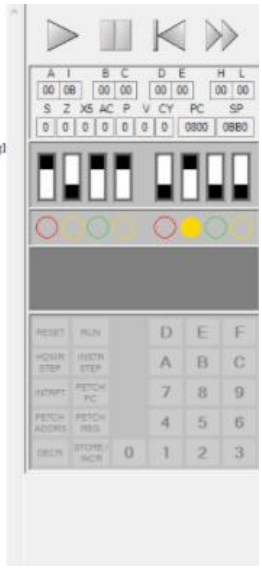
Ex2 - TSIK

```
       MVI A,FFH
       STA 3000H
       MVI D,64H            ; 100 Decimal = 64 Hex and 100 * 0,2 = 20 sec
       LXI B,0064H          ; 1/10 sec = 100 msec
BEGIN:, de
       LDA 2000H            ; Diabasma diakoptwn
       ANI 80H              ; MSB
       CPI 00H              ; Is the switch off;
       JZ  TURNEDOFF1
       JMP BEGIN
TURNEDOFF1:                 ; Switch first off
       LDA 2000H
       ANI 80H
       CPI 80H              ; Is the switch on?
       JZ  TURNEDON1
       JMP TURNEDOFF1
TURNEDON1:                  ; Switch first On
       LDA 2000H
       ANI 80H
       CPI 00H              ; Is the switch off?
       JZ  TURNEDOFF2
       JMP TURNEDON1
TURNEDOFF2:                 ; The shiwtch was turned off
       LDA 2000H
       ANI 80H
       CPI 80H              ; Is the switch on;
       JZ  TURNEDON2
       MVI A,00H            ; Turn on the switches
       STA 3000H            ;
       CALL DELB
       DCR D                ; Time
       MOV A,D
       CPI 00H              ; Did the time pass?
       JNZ TURNEDOFF2       ; Check if the time has passed
       MVI A,FFH            ; Light off
       STA 3000H
       MVI D,64H            ; Restart timer
       JMP TURNEDOFF1
TURNEDON2:
       LDA 2000H
       ANI 80H
       CPI 00H              ; Is the switch off?
       JZ  AGAIN            ; Check restart of the timer
       MVI A,00H
       STA 3000H
       CALL DELB
       DCR D
       MOV A,D
       CPI 00H
       JNZ TURNEDON2
       MVI A,FFH
       STA 3000H
       MVI D,64H            ; Restart timer
       JMP TURNEDOFF1
AGAIN:
       MVI D,64H            ; Restart timer
       JMP TURNEDOFF2
       END
```

Άσκηση 3:

i.

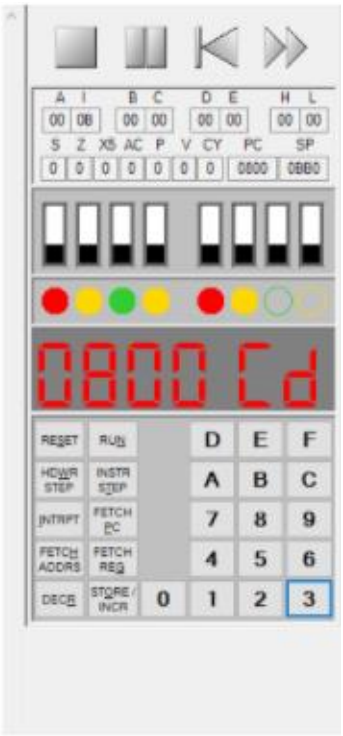
```
BEGIN:
      LDA 2000H
      CPI 00H
      JZ ZERO ;for zero input print zero
      MVI B,00H

ROTATE:
      RAR
      INR B ;final value of B will be the position of the rig
      JNC ROTATE


      MVI D,80H ;D is 2^7

FINDPOWER:
      MOV A,B
      CPI 00H
      JZ FINISH
      DCR B ;will add powers of 2 B times
      MOV A,D
      RLC ;multiply D by 2 to get next power of 2
      MOV D,A
      JMP FINDPOWER

FINISH:
      MOV A,D


ZERO:
      CMA
      STA 3000H
      JMP BEGIN
END
```



ii. Το πρόγραμμα καλεί την ρουτίνα KIND για να διαβάσει το πάτημα ενός πλήκτρου του δεκαεξαδικού πληκτρολογίου και:

- Αν η τιμή του αριθμού που πατήθηκε είναι 1 έως 8 ανάβει το led αντίστοιχης θέση και όλα τα led υψηλότερης τάξης μετά από αυτό.
- Αν δεν βρίσκεται στο εύρος 1-8, κάνει άλμα στο START

```
START:
CALL KIND
CPI 00H
JZ START ;if A=0 jump start
MVI C,80H
MVI D,80H
MOV B,A
CPI 09H
JC ADDPOWERS ;if A>0 & A<9 jump ADDPOWERS
JMP START

ADDPOWERS:
MOV A,B
CPI 00H
JZ FINISH
MOV A,C
ADD D ;add a power of 2
MOV C,A
DCR B ;
MOV A,D
RLC ;left shift register
MOV D,A
JMP ADDPOWERS

FINISH:
MOV A,C
STA 3000H
JMP START
END
```



iii. Το πρόγραμμα ελέγχει κάθε φορά πιο πλήκτρο είναι πατημένο, ελέγχοντας ξεχωριστά την κάθε γραμμή του πληκτρολογίου και προσπαθώντας να βρει την στήλη στην οποία βρίσκεται το πατημένο πλήκτρο και φορτώνει στη μνήμη τον κωδικό του, ο οποίος τυπώνεται στα 7 segment-displays με την βοήθεια των ρουτίνων STDM και DCD. το πρόγραμμα είναι συνεχούς λειτουργίας και επομένως τυπώνει τον κωδικό του κάθε πλήκτρου για όσο είναι αυτό πατημένο. Επίσης στην αρχή το πρόγραμμα τυπώνει τον κωδικό της εντολής run()84 , μόλις αυτό πατιέται για την εκκίνηση ενός προγράμματος. Σε περίπτωση που θέλαμε να το αποφύγουμε αυτό, θα μπορούσαμε να βάλουμε μια CALL DELB στην αρχή του προγράμματος,για να αγνοηθεί το αρχικό run.

Άσκηση 4:

Για την εξομοίωση της λειτουργίας του I.C που δόθηκε αρχικά θα απομονώσουμε τα ψηφία που θέλουμε χρησιμοποιώντας την εντολή ANI, αφού τα φέρουμε στην επιθυμητή θέση με τη χρήση της εντολής RRC. Με τις εντολές ANA, XRA και ORA θα εκτελέσουμε τις κατάλληλες λογικές πράξεις και τέλος θα προσθέσουμε τα επιμέρους αποτελέσματα με τη χρήση της εντολής ADD.

```
START:
LDA 2000H
MOV B,A ;b contains input
RRC
MOV C,A ;c contains input rotated once

ANA B ;B and C
RRC
RRC
MOV D,A
ANI 04H
MOV E,A ;E contains A2 and B2 in right position
MOV A,D
RRC
ANI 08H
MOV D,A ;D contains A3 and B3 in right position
RRC
ORA E
MOV E,A ;E contains D or E

MOV A,C
XRA B ;B XOR C
MOV C,A
ANI 01H
MOV B,A ;B containts A0 xor B0 in right position
MOV A,C
RRC
ANI 02H
MOV C,A ;C contains A1 xor B1 in right position
ADD E
ADD D
MOV D,A
MOV A,C
RRC
XRA B ;A contains C xor B
ADD D

CMA
STA 3000H
JMP START
END
```

Άσκηση 5:

Οι διαφορές με το βιβλίο είναι τα λιγότερα bits για τις γραμμές διευθύνσεων και τη γραμμή του πίνακα μετά το κύκλωμα αποκωδικοποίησης και το σήμα RD για την ανάγνωση. Η λειτουργία της μνήμης πρέπει να ενεργοποιηθεί με το CS. Έπειτα, επιλέγουμε να διαβάσουμε τη μνήμη με το RD, που μέσω των κόκκινων πυλών οδηγεί τα δεδομένα από τους πολυπλέκτες στην έξοδο (D0-D3) ή να γράψουμε σε αυτήν με το WE, που μέσω των μπλε πυλών οδηγεί τα 4 bits της εισόδου (D0-D3) στους πολυπλέκτες για να επιλέξουν την επιθυμητή τετράδα του πίνακα. Η επιλογή από τις 16 τετράδες γίνεται με τα πρώτα 4 bits της διεύθυνσης (A0-A3), ενώ η γραμμή του πίνακα στην οποία ανήκει η τετράδα επιλέγεται με τα τελευταία 4 bits της διεύθυνσης (A4-A7).

Για παράδειγμα, για να γίνει εγγραφή στη μνήμη:

1)Αρχικά εφαρμόζουμε στις γραμμές διεύθυνσης A0-A6 τη διεύθυνση στην οποία θα γράψουμε.

2)Η είσοδος (CS)' τίθεται στο λογικό 0, παύοντας την απομόνωση εισόδου και εξόδου της μνήμης.

3)Τα 4 bits προς αποθήκευση εφαρμόζονται στις εισόδους D0-D3.

4)Έρχεται αρνητικός παλμός (μικρής διάρκειας) στην είσοδο (WE)' και η εγγραφή ξεκινά. Οι γραμμές διεύθυνσης A0-A3 λένε στους πολυπλέκτες (που τώρα εκτελούν αντίστροφη λειτουργία) σε ποια από τις 8 εξόδους τους να οδηγήσουν την είσοδο. Τέλος οι γραμμές A4-A7 επιλέγουν την κατάλληλη γραμμή που θα γίνει η εγγραφή των 4 bits.

5)Τέλος αρνητικού παλμού (WE)'

6)Επαναφορά (CS)' στο λογικό 1, απομόνωση εισόδου και εξόδου της μνήμης.

Για να γίνει ανάγνωση από τη μνήμη:

1)Αρχικά εφαρμόζουμε στις γραμμές διεύθυνσης A0-A6 τη διεύθυνση την οποία θα διαβάσουμε.

2)Η είσοδος (CS)' τίθεται στο λογικό 0, παύοντας την απομόνωση εισόδου και εξόδου της μνήμης.

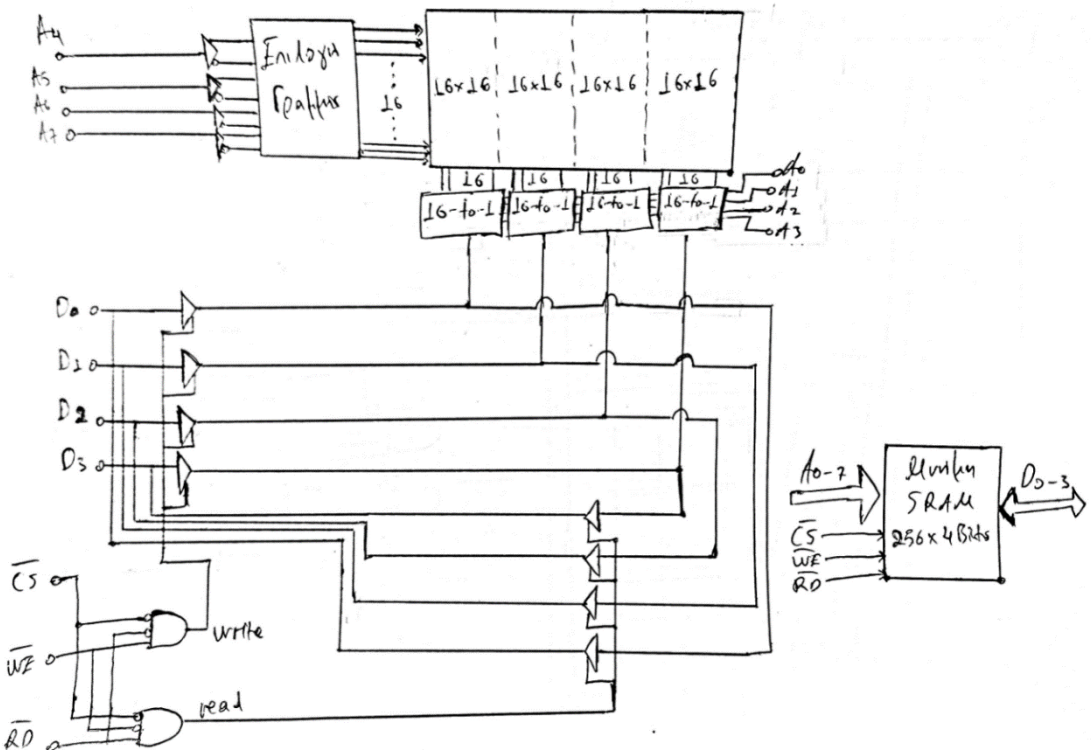
3)Έρχεται αρνητικός παλμός (μικρής διάρκειας) στην είσοδο (RD)' και η ανάγνωση ξεκινά. Οι γραμμές διεύθυνσης A0-A3 λένε στους πολυπλέκτες (που τώρα εκτελούν ορθή λειτουργία) ποια από τις 16 εισόδους τους να οδηγήσουν στην έξοδο. Τέλος οι γραμμές A4-A7 επιλέγουν την κατάλληλη γραμμή από όπου θα γίνει η ανάγνωση των 4 bits.

4)Μετά από πάροδο χρόνου όσος κι ο χρόνος προσπέλασης μνήμης, εμφανίζονται τα δεδομένα (4 bits προς ανάγνωση) στις εξόδους DO-D3

5)Τέλος αρνητικού παλμού (RD)

6)Επαναφορά (CS)' στο λογικό 1, απομόνωση εισόδου και εξόδου της μνήμης.

Η εσωτερική οργάνωση αυτής της μνήμης παρουσιάζεται παρακάτω:

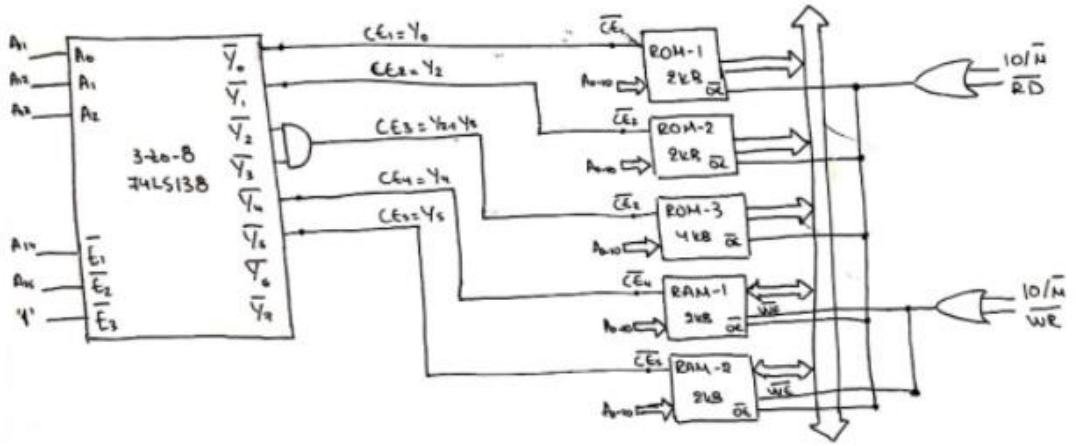


Άσκηση 6:

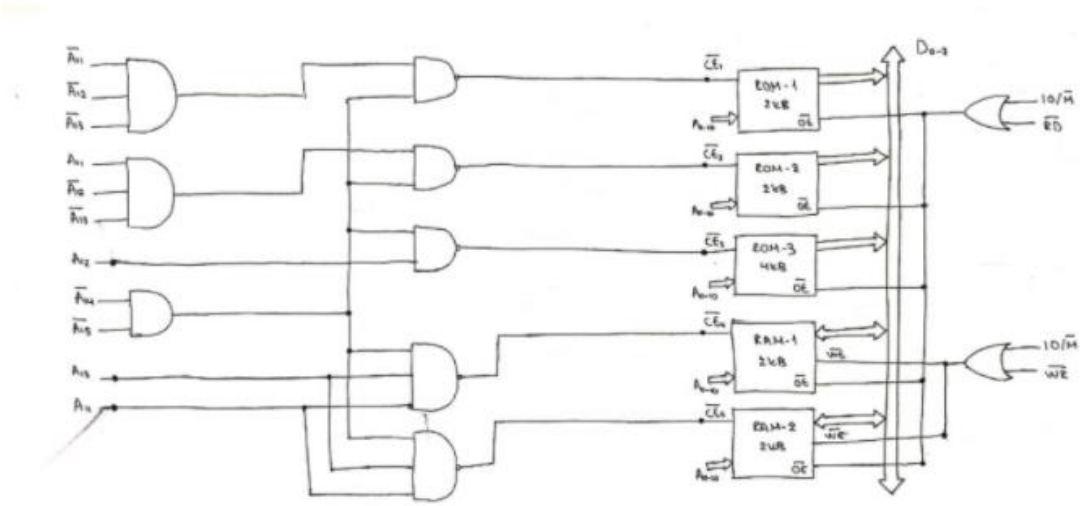
Ο χάρτης μνήμης του συστήματος είναι:

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	ROM1-2K
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	07FF	
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0800	ROM2-2K
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0FFF	
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1000	ROM3-4K
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	RAM1-2K
0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	27FF	
0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	2800	RAM2-2K
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	2FFF	

α) Το σύστημα μνήμης υλοποιημένο με αποκωδικοποιητή και λογικές πύλες:



β)Το σύστημα μνήμης υλοποιημένο με λογικές πύλες:



Άσκηση 7:

	A15	A14	A13	A12	A11	A10	A9	A8	A7	A7	A6	A5	A4	A3	A2	A1	A0	
ROM1-4K	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H
	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2FFFH
RAM1-4K	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3000H
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFFH
RAM2-4K	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000H
	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4FFFH
RAM3-4K	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5000H
	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	5FFFH
ROM2-12K	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6000H
	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFFH
Mem map	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	7000H

Το κύκλωμα απεικονίζεται παρακάτω: