ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ, ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2021-22

ΑΝΑΦΟΡΑ 2$^{ΗΣ}$ ΟΜΑΔΑΣ ΑΣΚΗΣΕΩΝ

---

Ναταλία Μπούρδη

03119031

**1η ΑΣΚΗΣΗ**

(α) Το παρακάτω πρόγραμμα αποθηκεύει σε φθίνουσα σειρά τους αριθμούς 0 έως 255 σε διαδοχικές θέσεις μνήμης, αρχίζοντας από το 0900H.

```
      IN 10H

      LXI H,0900H
      MVI B,00H    ; limit
      MVI A,FFH    ; initialize to 255
DO:
      CMP B
      MOV M,A
      JZ DONE      ; (A) = 0 => DONE
      SUI 01H
      INX H        ; next memory address
      JMP DO
DONE:
      END
```

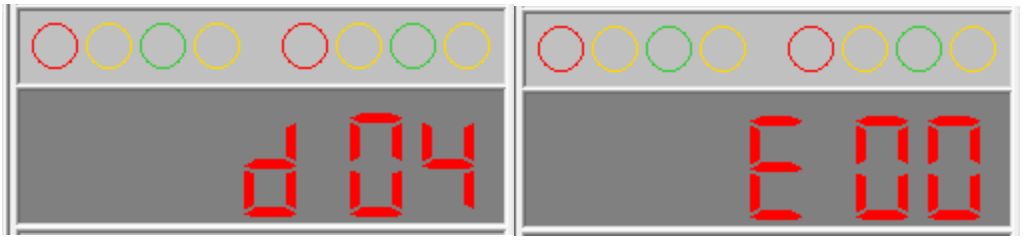
Τρέχουμε το πρόγραμμα στον προσομοιωτή μLab και, πράγματι, στις θέσεις 0900H έως 09FFH βρίσκονται οι δεκαεξαδικοί αριθμοί FFH με 00H.

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 08FC | 00 | 08FD | 00 | 08FE | 00 | 08FF | 00 | 0900 | FF | 0901 | FE | 0902 | F | 0903 | F | 0904 | FB |
| 0905 | FA | 0906 | F9 | 0907 | F8 | 0908 | F7 | 0909 | F6 | 090A | F5 | 090B | F4 | 090C | F3 | 090D | F2 |
| 090E | F1 | 090F | F0 | 0910 | EF | 0911 | E | 0912 | E | 0913 | E | 0914 | E | 0915 | E | 0916 | E9 |
| 0917 | E8 | 0918 | E7 | 0919 | E6 | 091A | E5 | 091B | E4 | 091C | E3 | 091D | E2 | 091E | E1 | 091F | E0 |
| 0920 | D | 0921 | D | 0922 | D | 0923 | D | 0924 | D | 0925 | D | 0926 | D9 | 0927 | D8 | 0928 | D7 |
| 0929 | D6 | 092A | D5 | 092B | D4 | 092C | D3 | 092D | D2 | 092E | D1 | 092F | D0 | 0930 | C | 0931 | C |
| 0932 | C | 0933 | C | 0934 | C | 0935 | C | 0936 | C9 | 0937 | C8 | 0938 | C7 | 0939 | C6 | 093A | C5 |
| 093B | C4 | 093C | C3 | 093D | C2 | 093E | C1 | 093F | C0 | 0940 | BF | 0941 | B | 0942 | B | 0943 | B |
| 0944 | B | 0945 | B | 0946 | B9 | 0947 | B8 | 0948 | B7 | 0949 | B6 | 094A | B5 | 094B | B4 | 094C | B3 |
| 094D | B2 | 094E | B1 | 094F | B0 | 0950 | AF | 0951 | A | 0952 | A | 0953 | A | 0954 | A | 0955 | A |
| 0956 | A9 | 0957 | A8 | 0958 | A7 | 0959 | A6 | 095A | A5 | 095B | A4 | 095C | A3 | 095D | A2 | 095E | A1 |
| 095F | A0 | 0960 | 9F | 0961 | 9E | 0962 | 9D | 0963 | 9C | 0964 | 9B | 0965 | 9A | 0966 | 99 | 0967 | 98 |
| 0968 | 97 | 0969 | 96 | 096A | 95 | 096B | 94 | 096C | 93 | 096D | 92 | 096E | 91 | 096F | 90 | 0970 | 8F |
| 0971 | 8E | 0972 | 8D | 0973 | 8C | 0974 | 8B | 0975 | 8A | 0976 | 89 | 0977 | 88 | 0978 | 87 | 0979 | 86 |
| 097A | 85 | 097B | 84 | 097C | 83 | 097D | 82 | 097E | 81 | 097F | 80 | 0980 | 7F | 0981 | 7E | 0982 | 7D |
| 0983 | 7C | 0984 | 7B | 0985 | 7A | 0986 | 79 | 0987 | 78 | 0988 | 77 | 0989 | 76 | 098A | 75 | 098B | 74 |
| 098C | 73 | 098D | 72 | 098E | 71 | 098F | 70 | 0990 | 6F | 0991 | 6E | 0992 | 6D | 0993 | 6C | 0994 | 6B |
| 0995 | 6A | 0996 | 69 | 0997 | 68 | 0998 | 67 | 0999 | 66 | 099A | 65 | 099B | 64 | 099C | 63 | 099D | 62 |
| 099E | 61 | 099F | 60 | 09A0 | 5F | 09A1 | 5E | 09A2 | 5D | 09A3 | 5C | 09A4 | 5B | 09A5 | 5A | 09A6 | 59 |
| 09A7 | 58 | 09A8 | 57 | 09A9 | 56 | 09AA | 55 | 09AB | 54 | 09AC | 53 | 09AD | 52 | 09AE | 51 | 09AF | 50 |
| 09B0 | 4F | 09B1 | 4E | 09B2 | 4D | 09B3 | 4C | 09B4 | 4B | 09B5 | 4A | 09B6 | 49 | 09B7 | 48 | 09B8 | 47 |
| 09B9 | 46 | 09BA | 45 | 09BB | 44 | 09BC | 43 | 09BD | 42 | 09BE | 41 | 09BF | 40 | 09C0 | 3F | 09C1 | 3E |
| 09C2 | 3D | 09C3 | 3C | 09C4 | 3B | 09C5 | 3A | 09C6 | 39 | 09C7 | 38 | 09C8 | 37 | 09C9 | 36 | 09CA | 35 |
| 09CB | 34 | 09CC | 33 | 09CD | 32 | 09CE | 31 | 09CF | 30 | 09D0 | 2F | 09D1 | 2E | 09D2 | 2D | 09D3 | 2C |
| 09D4 | 2B | 09D5 | 2A | 09D6 | 29 | 09D7 | 28 | 09D8 | 27 | 09D9 | 26 | 09DA | 25 | 09DB | 24 | 09DC | 23 |
| 09DD | 22 | 09DE | 21 | 09DF | 20 | 09E0 | 1F | 09E1 | 1E | 09E2 | 1D | 09E3 | 1C | 09E4 | 1B | 09E5 | 1A |
| 09E6 | 19 | 09E7 | 18 | 09E8 | 17 | 09E9 | 16 | 09EA | 15 | 09EB | 14 | 09EC | 13 | 09ED | 12 | 09EE | 11 |
| 09EF | 10 | 09F0 | 0F | 09F1 | 0E | 09F2 | 0D | 09F3 | 0C | 09F4 | 0B | 09F5 | 0A | 09F6 | 09 | 09F7 | 08 |
| 09F8 | 07 | 09F9 | 06 | 09FA | 05 | 09FB | 04 | 09FC | 03 | 09FD | 02 | 09FE | 01 | 09FF | 00 | 0A00 | 00 |

(β) Επεκτείνουμε το πρόγραμμα ώστε να αποθηκεύεται το πλήθος των δυαδικών μηδενικών των παραπάνω δεδομένων στο ζεύγος καταχωρητών D-E.

```
      IN 10H

      LXI H,0900H
      LXI D,0000H ; initialize counter of zeros
      MVI B,00H   ; limit
      MVI A,FFH   ; initialize to 255
DO:
      CMP B
      MOV M,A
      JZ LB       ; (A) = 0 => LB
      SUI 01H
      INX H       ; next memory address
      JMP DO
LB:               ; (b)
      MOV A,M
      MVI C,09H  ; 8 bits -> 8 iterations
ROT:
      DCR C
      JZ DONE    ; (C)=0 => next number
      RRC
      JC ROT     ; IF CY = 0 COUNT IT
COUNT:
      INX D      ; count a zero
      JMP ROT
DONE:
      DCR L      ; previous memory address
      JNZ LB
      RST 1      ; interrupt to check D-E register

      END
```

Με την εντολή `RST 1` διακόπτουμε την εκτέλεση του προγράμματος ώστε με το κουμπί `FETCH REG` και `STORE/INCR` να δούμε το περιεχόμενο όλων των καταχωρητών.

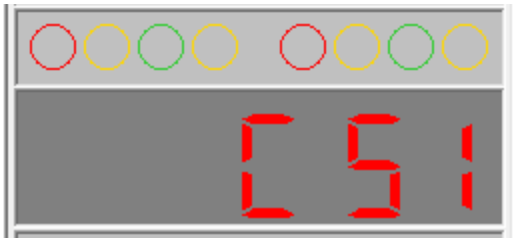



Το ζεύγος καταχωρητών DE περιέχει τον αριθμό `0400H = 1024DEC` οπότε επιβεβαιώνεται η ορθότητα του προγράμματος μας.

(γ) Επεκτείνουμε περαιτέρω το πρόγραμμα ώστε να αποθηκεύεται στον καταχωρητή C το πλήθος των αριθμών που είναι μεταξύ των 20H και 70H. Το τελικό πρόγραμμα:

```
      IN 10H

      LXI H,0900H
      LXI D,0000H ; initialize counter of zeros
      MVI B,00H   ; limit
      MVI A,FFH   ; initialize to 255
DO:
      CMP B
      MOV M,A
      JZ LB       ; (A) = 0 => LB
      SUI 01H
      INX H       ; next memory address
      JMP DO

LB:               ; (b)
      MOV A,M
      MVI C,09H  ; 8 bits -> 8 iterations
ROT:
      DCR C
      JZ DONE    ; (C)=0 => next number
      RRC
      JC ROT     ; IF CY = 0 COUNT IT
COUNT:
      INX D      ; count a zero
      JMP ROT
DONE:
      DCR L      ; previous memory address
      JNZ LB
      ;RST 1     ; interrupt to check D-E register


      ;(g)
      MVI C,00H  ; initialize counter
      MVI B,FFH
LG:
      MOV A,M    ; at first: M = 0900 => A=255 and L = 00
      INR L
      DCR B
      JZ FIN
      CPI 71H
      JNC LG     ; (A)>70H => next number
      CPI 20H
      JC LG      ; (A)<20H => next number
      INR C      ; within limits, count it
      JMP LG

FIN:
      ;RST 1    ; interrupt to check register C: expect 51H
      END
```

Με τον ίδιο τρόπο όπως στο ερώτημα (β), ελέγχουμε το περιεχόμενο του καταχωρητή C. Το περιεχόμενό του είναι αυτό που περιμέναμε. Δηλαδή 51H αριθμούς μεταξύ των 20H και 70H.



**2η ΑΣΚΗΣΗ**

Το πρόγραμμα σε Assembly 8085:

```
      LXI B,00CBH      ; 200ms delb

START:
      LDA 2000H        ; load dip switches
      RAR              ; to get lsb
      JC START         ; if on, go to start
L1:    ; its on after off
      LDA 2000H
      RAR
      JNC L1           ; if still off, go to L1 looking again
L2:
      LDA 2000H
      RAR
      JC L2            ; if it's on after on
      ;if we reached here it's OFF-ON-OFF
TIMER_ON:
      MVI D,4BH        ; 75x200ms = 15s
      MVI H,05H        ; counter to reverse leds every second
      MVI E,FFH
TSTART:
      CALL DELB        ; introduce 200ms delay
      DCR D            ; decrease the counter
      JZ START         ; 15 seconds passed, go to START
      DCR H
      JNZ CONT
      MOV A,E          ; reverse leds if a second has passed
      CMA
      MOV E,A
      STA 3000H
      MVI H,05H
CONT:
      LDA 2000H        ; load dip switches
      RAR              ; to get lsb
      JC TSTART        ; if on, go to start
TL1:
      CALL DELB        ; introduce 200ms delay
      DCR D            ; decrease the timer counter
```

```
      JZ L1
      DCR H
      JNZ CONT2
      MOV A,E
      CMA
      MOV E,A
      STA 3000H
      MVI H,05H
CONT2:
      LDA 2000H
      RAR
      JNC TL1          ; if still off, go to TL1 looking again
TL2:
      CALL DELB
      DCR D
      JZ L2
      DCR H
      JNZ CONT3
      MOV A,E
      CMA
      MOV E,A
      STA 3000H
      MVI H,05H
CONT3:
      LDA 2000H
      RAR
      JC TL2           ; if it's on after on
      JMP TIMER_ON     ; restart timer

      END
```

**3η ΑΣΚΗΣΗ**

i. Πρόγραμμα συνεχούς λειτουργίας που διαβάζει την πόρτα εισόδου των dip switches και με βάση το 1ο αριστερότερο ON, ανάβει το αντίστοιχης τάξης led και όλα τα υψηλότερης τάξης led μετά από αυτό.

```
START:
      LDA 2000H
      MVI B,09H
ROT:                   ; loop to find the first ON switch
      DCR B
      JZ NOLEDS        ; there are no switches on
      RAL
      JNC ROT

      MVI A,09H
      SUB B
      MOV B,A          ;(B) = how many leds we want ON
      MVI A,00H
```

```
OUTP:                ; we add (B) ones on the left
     RRC
     ORI 80H         ; 80H is 10000000
     DCR B
     JNZ OUTP

     CMA             ; reverse LED logic
     STA 3000H
     JMP START
NOLEDS:              ; turn off all leds if no switch is ON
     MVI A,FFH
     STA 3000H
     JMP START

     END
```

ii. Πρόγραμμα που αναμένει πάτημα του πληκτρολογίου και μόνο των αριθμών 1 έως 8. Για αριθμούς από 1 έως 4 αναβοσβήνουν (με ρυθμό ½ sec) 4 φορές και τα 4 LSB led ενώ για αριθμούς από 5 έως 8 αναβοσβήνουν 4 φορές και τα 4 MSB led.

```
     LXI B,01F4H     ; 500ms delay
START:
     MVI D,04H       ; 4 iterations
     CALL KIND
     CPI 00H         ; no response for pressing of 0
     JZ START
     CPI 05H         ; 0 < (A) < 5 => LSBS
     JC LSBS
     CPI 09H         ; 5 <= (A) < 9 => MSBS
     JNC START
MSBS:
     MVI A,0FH       ; 4 MSBs ON (reverse logic) for 1/2 sec
     STA 3000H
     CALL DELB
     MVI A,FFH       ; all OFF for 1/2 sec
     STA 3000H
     CALL DELB
     DCR D           ; decrease counter of iterations
     JNZ MSBS
     JMP START
LSBS:
     MVI A,F0H       ; 4 LSBs ON for 1/2 sec
     STA 3000H
     CALL DELB
     MVI A,FFH
     STA 3000H
     CALL DELB
     DCR D
     JNZ LSBS
     JMP START
END
```

iii. Πρόγραμμα που κάνει απευθείας ανάγνωση του πληκτρολογίου χωρίς χρήση της ρουτίνας KIND. Το αποτέλεσμα του κωδικού εμφανίζεται στα 2 αριστερότερα 7-segment display.

```
      IN 10H
SCAN0:
      MVI A,FEH        ; line 0
      STA 2800H        ; scan line0
      LDA 1800H        ; load the results to A
      ANI 07H          ; isolate 3 lsbs
      MVI B,86H        ; the char code of INSTR STEP
      CPI 06H
      JZ   OUTP        ; INSTR STEP
      MVI B,85H
      CPI 05H
      JZ OUTP          ; FETCH PC
SCAN1:
      MVI A,FDH
      STA 2800H
      LDA 1800H
      ANI 07H          ; isolate 3 lsbs
      MVI B,82H
      CPI 03H
      JZ OUTP          ; FETCH ADDRESS
      MVI B,80H
      CPI 05H
      JZ OUTP          ; FETCH REG
      MVI B,84H
      CPI 06H
      JZ OUTP          ; RUN
SCAN2:
      MVI A,FBH
      STA  2800H
      LDA 1800H
      ANI 07H          ; isolate 3 lsbs
      MVI B,81H
      CPI 03H
      JZ    OUTP       ; DECR
      MVI B,83H
      CPI 05H
      JZ   OUTP        ; STINCR
      MVI B,00H
      CPI 06H
      JZ   OUTP        ; 0
SCAN3:
      MVI A,F7H
      STA  2800H
      LDA 1800H
      ANI 07H          ; isolate 3 lsbs
      MVI B,03H
      CPI 03H
      JZ OUTP          ; 3
      MVI B,02H
```

```
      CPI 05H
      JZ OUTP          ; 2
      MVI B,01H
      CPI 06H
      JZ   OUTP        ; 1
SCAN4:
      MVI A,EFH
      STA  2800H
      LDA 1800H
      ANI 07H          ; isolate 3 lsbs
      MVI B,06H
      CPI 03H
      JZ OUTP          ; 6
      MVI B,05H
      CPI 05H
      JZ   OUTP        ; 5
      MVI B,04H
      CPI 06H
      JZ   OUTP        ; 4
SCAN5:
      MVI A,DFH
      STA  2800H
      LDA 1800H
      ANI 07H          ; isolate 3 lsbs
      MVI B,09H
      CPI 03H
      JZ   OUTP        ; 9
      MVI B,08H
      CPI 05H
      JZ   OUTP        ; 8
      MVI B,07H
      CPI 06H
      JZ   OUTP        ; 7
SCAN6:
      MVI A,BFH
      STA 2800H
      LDA 1800H
      ANI 07H          ; isolate 3 lsbs
      MVI B,0CH
      CPI 03H
      JZ   OUTP        ; C
      MVI B,0BH
      CPI 05H
      JZ   OUTP        ; B
      MVI B,0AH
      CPI 06H
      JZ   OUTP        ; A
SCAN7:
      MVI A,7FH
      STA 2800H
      LDA 1800H
      ANI 07H          ; isolate 3 lsbs
```

```
      MVI B,0FH
      CPI 03H
      JZ   OUTP       ; F
      MVI B,0EH
      CPI 05H
      JZ   OUTP       ; E
      MVI B,0DH
      CPI 06H
      JZ   OUTP       ; D
      JMP SCAN0

OUTP:
      MOV A,B         ; character code in A
      ANI 0FH         ; isolate the 4 LSBs
      STA 0B00H       ; put them in the 5th display slot
      MOV A,B
      ANI F0H         ; isolate the 4 MSBs
      RLC
      RLC
      RLC
      RLC
      STA 0B01H       ; put them in the 6th display slot
      MVI A,10H       ; spaces in the 4 rightmost slots
      STA 0AFCH
      STA 0AFDH
      STA 0AFEH
      STA 0AFFH
      LXI D,0AFCH     ; our data is at 0AFCH - 0B001H
      CALL STDM
      CALL DCD

      JMP SCAN0
      END
```

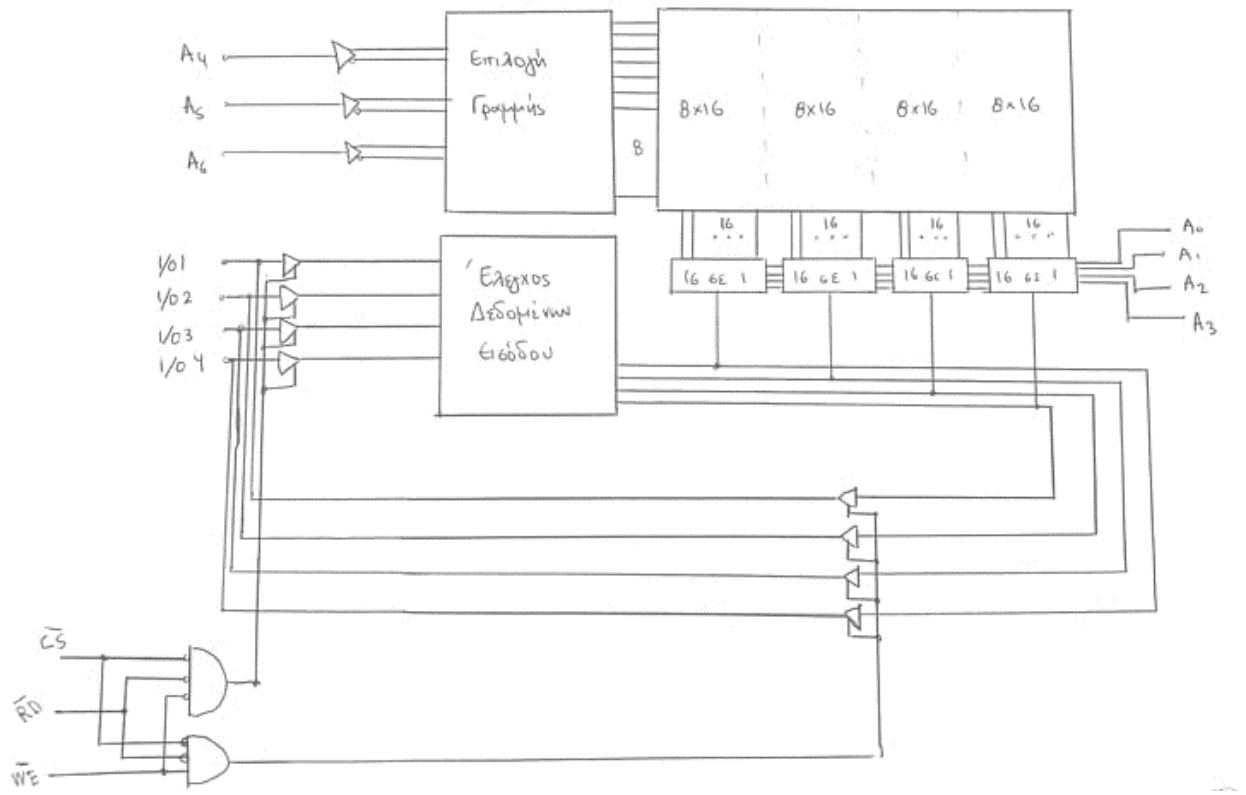
**4η ΑΣΚΗΣΗ**

```
START:
      LDA 2000H
      MOV B,A   ; B holds dip switches state
      RAR
      ORA B     ; the 2 OR gates
      MOV D,A   ; X1 in bit 2
      RAR
      RAR
      ANA D     ; x1 AND (A0 or B0)
      MOV E,A   ; X0 in lsb
      MOV A,B   ; dip switch state in A again
      RAR
      ANA B     ; the 2 AND gates on the left
```

```
MOV C,A    ; X2 in bit 4
RAR
RAR
XRA C      ; the XOR gate
MOV B,A    ; X3 in bit 4

MVI A,01H
ANA E      ; isolate X0
MOV E,A
MVI A,04H
ANA D      ; isolate X1
RAR        ; put it in bit 1
ORA E
MOV D,A

MVI A,10H
ANA C      ; isolate X2
RAR
RAR
ORA D      ; put it in bit 2
MOV C,A

MVI A,10H
ANA B      ; isolate X3
RAR
ORA C      ; put it in bit 4
CMA        ; LEDs in reverse logic
STA 3000H

JMP START
END
```

**5η ΑΣΚΗΣΗ**

Η εσωτερική δομή μια; μνήμης SRAM 128x4 bit:




Η τετραγωνική διάταξη αποτελεί τον πίνακα της μνήμης με 8x64 στοιχεία μνήμης. Βάσει τις γραμμές A4-A6 της διεύθυνσης επιλέγεται μία από τις 8 γραμμές του πίνακα. Τα bits σε κάθε γραμμή είναι οργανωμένα κατά τετράδες. Η επιλογή μεταξύ των 16, συνολικά, τετράδων γίνεται με 4 πολυπλέκτες 16 σε 1 σύμφωνα με τις γραμμές διεύθυνσης A0-A3. Οι γραμμές I/O1 - I/O4 είτε καθορίζουν τα δεδομένα που θα εγγραφούν στη μνήμη (λειτουργία εγγραφής) είτε λαμβάνουν το περιεχόμενο της μνήμης (λειτουργία ανάγνωσης).

Έστω ότι θέλουμε να διαβάσουμε από την διεύθυνση 0011 1100.

A6-4: 0011 $\Longrightarrow$ επιλογή της $3^{ης}$ γραμμής

A3-0: 1100 $\Longrightarrow$ επιλογή της $12^{ης}$ τετράδας

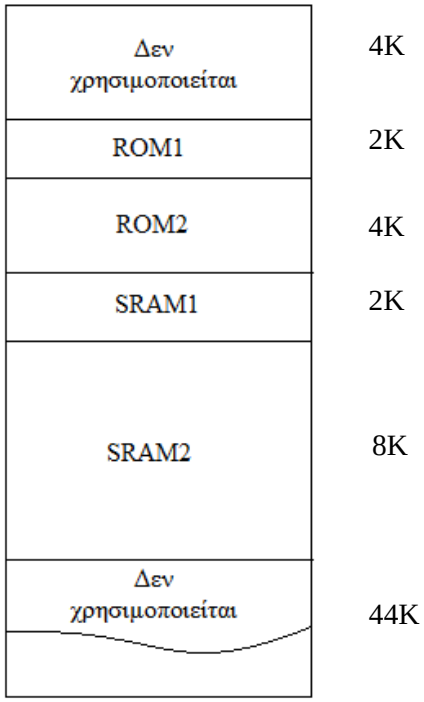
Για ανάγνωση τα σήματα είναι

- $\overline{CE} = 0$
- $\overline{RD} = 0$
- $\overline{WE} = 1$

Οπότε η πύλη AND που ενεργοποιεί τους απομονωτές της εξόδου παίρνει τη τιμή 1 και η πύλη AND που ενεργοποιεί τους απομονωτές εισόδου παίρνει τη τιμή 0. Έτσι, το περιεχόμενο της διεύθυνσης 3CH οδηγείται στα I/O1 – 4.

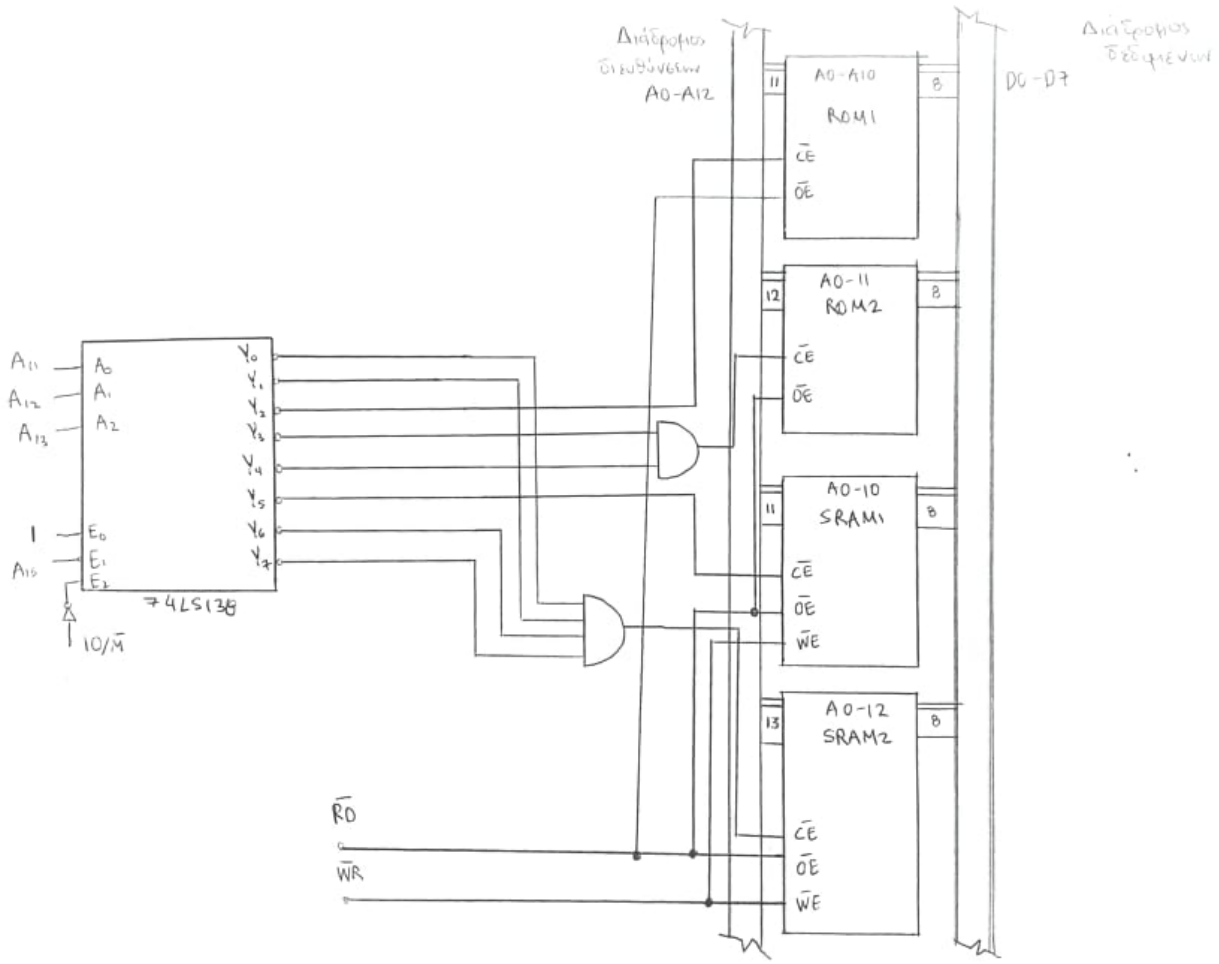
**$6^η$ ΑΣΚΗΣΗ**

Ο χάρτης μνήμης του συστήματος:

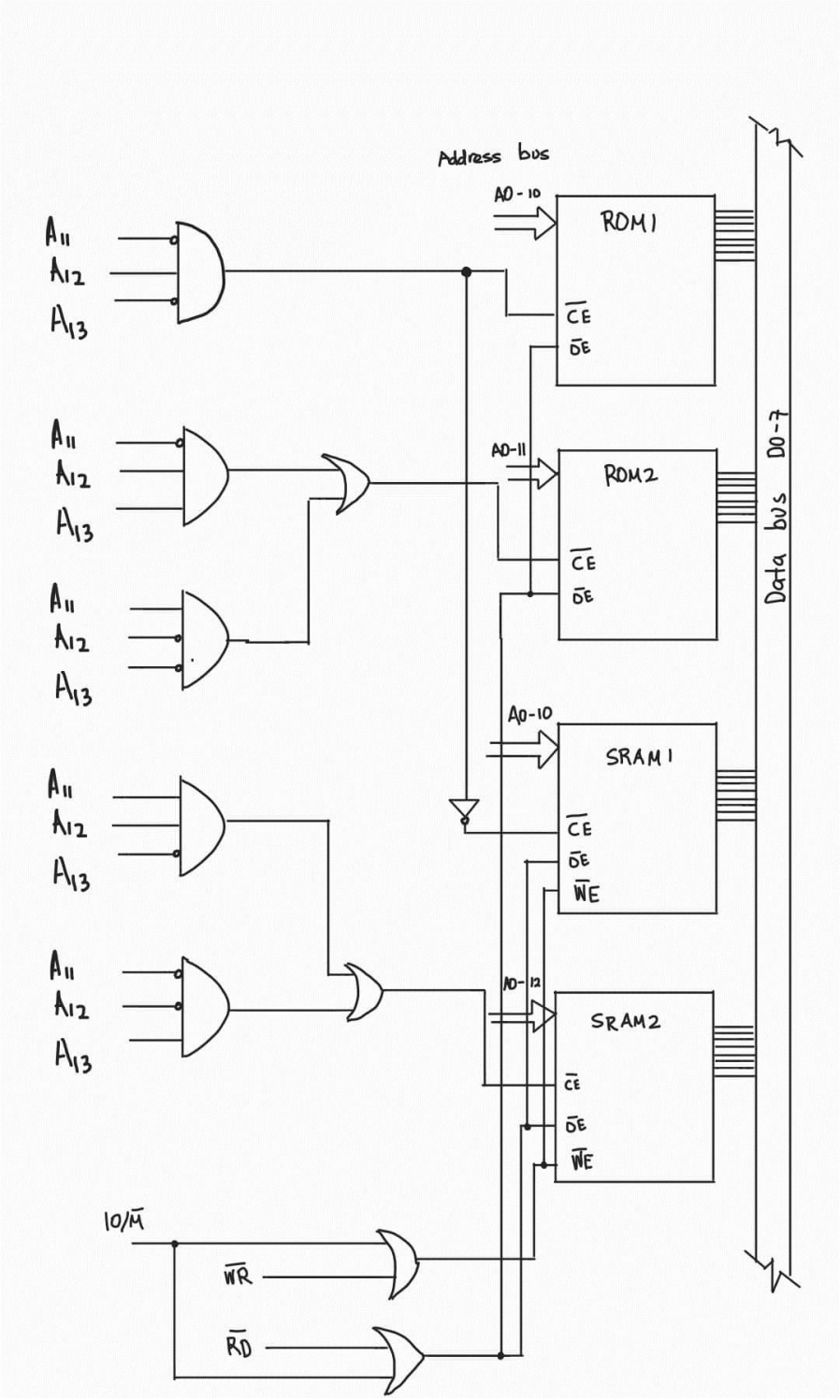



| | Διευθύνσεις | A15-12 | A11-8 | A7-4 | A3-0 |
|---|---|---|---|---|---|
| ROM1 | 1000H | 0001 | 0000 | 0000 | 0000 |
| | 17FFH | 0001 | 0111 | 1111 | 1111 |
| ROM2 | 1800H | 0001 | 1000 | 0000 | 0000 |
| | 27FFH | 0010 | 0111 | 1111 | 1111 |
| SRAM1 | 2800H | 0010 | 1000 | 0000 | 0000 |
| | 2FFFH | 0010 | 1111 | 1111 | 1111 |
| SRAM2 | 3000H | 0011 | 0000 | 0000 | 0000 |
| | 4FFFH | 0100 | 1111 | 1111 | 1111 |

α) Υλοποίηση με αποκωδικοποιητή 3 σε 8 και λογικές πύλες:




β) Υλοποίηση αποκλειστικά με λογικές πύλες:




**7η ΑΣΚΗΣΗ**

Διαθέτουμε:

- ROM των 16KB,
- 2 RAM των 4KB και των 8KB,

- mP 8085,
- καταχωρητές και απομονωτές των 8 bits,
- κωδικοποιητές 3 σε 8 και
- βασικές λογικές πύλες.

Και επιχειρούμε να σχεδιάσουμε ένα μικροϋπολογιστικό σύστημα με τον εξής χάρτη μνήμης:

1000-1FFF Hex : ROM (4KB)

2000-4FFF Hex : RAM (12KB)

5000-7FFF Hex : ROM (12KB)

8000 Hex : θύρα εισόδου (Memory map I/O)

80 Hex : θύρα εξόδου (Standard I/O)

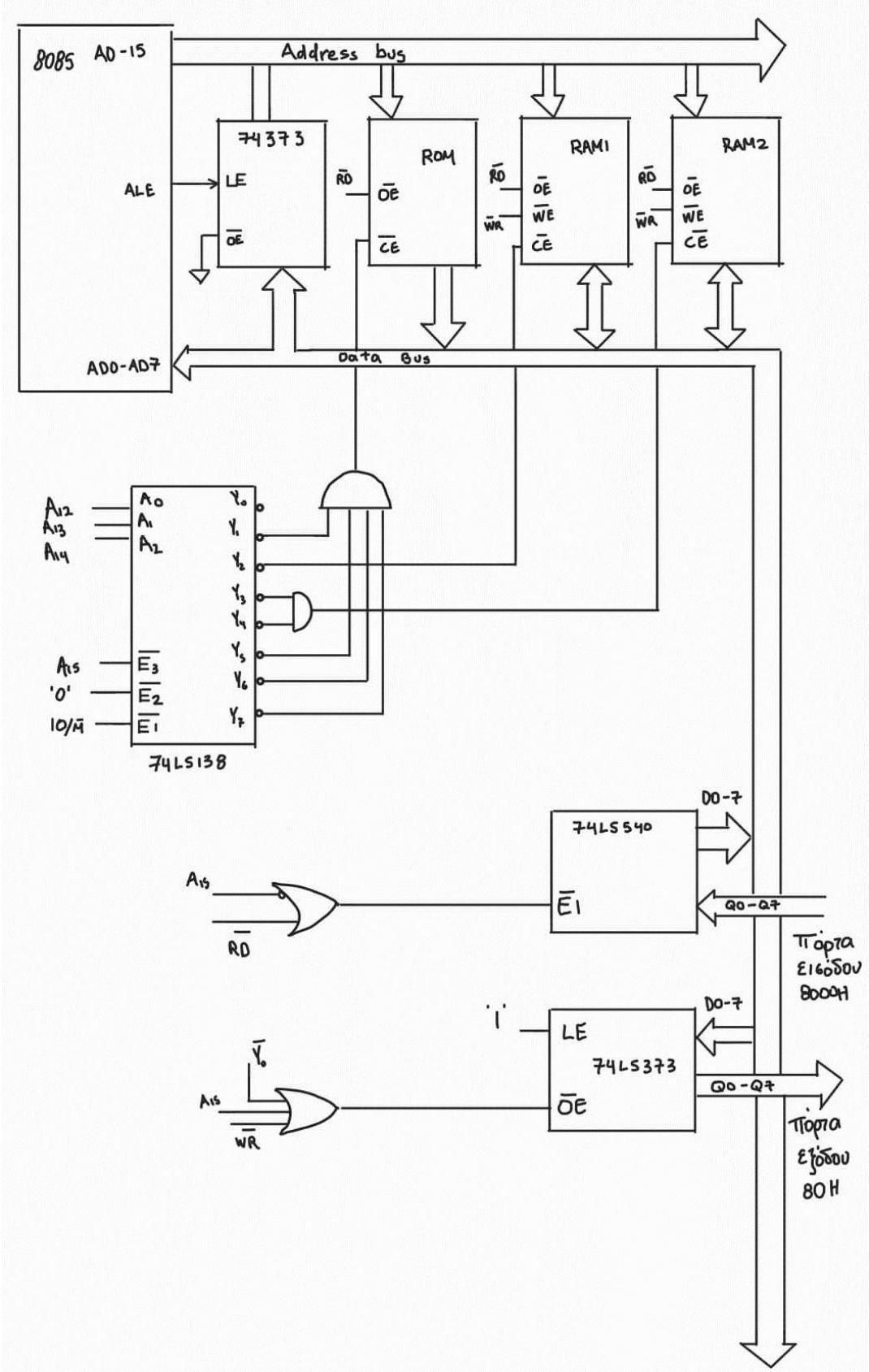
Αναλυτικά ο χάρτης μνήμης χρησιμοποιώντας 1 ROM των 16K, 1 RAM 4K και 1 RAM 8K:

| | Διευθύνσεις | A15-12 | A11-8 | A7-4 | A3-0 |
|---|---|---|---|---|---|
| ROM | 1000H | 0001 | 0000 | 0000 | 0000 |
| | 1FFFH | 0001 | 1111 | 1111 | 1111 |
| RAM1 (4K) | 2000H | 0010 | 0000 | 0000 | 0000 |
| | 2FFFH | 0010 | 1111 | 1111 | 1111 |
| RAM1 (8K) | 3000H | 0011 | 0000 | 0000 | 0000 |
| | 4FFFH | 0100 | 1111 | 1111 | 1111 |
| ROM | 5000H | 0101 | 0000 | 0000 | 0000 |
| | 7FFFH | 0111 | 1111 | 1111 | 1111 |

| | | | | | |
|---|---|---|---|---|---|
| Mem map I/O | 8000H | 1000 | 0000 | 0000 | 0000 |
| Standard I/O | 80H | 0000 | 0000 | 1000 | 0000 |

Το μΥ-Σ:

*(Στη συγκεκριμένη σχεδίαση λόγω της ύπαρξης μόνο δύο θυρών, δε γίνεται πλήρης αποκωδικοποίηση)*


8085 AD-15
Address bus
74373
ROM
RAM1
RAM2
ALE
LE
OE
RD
WR
CE
WE
AD0-AD7
Data Bus
A12
A13
A14
A15
'0'
IO/M
74LS138
74LS540
74LS373
DO-7
Q0-Q7
Πόρτα Εισόδου 8000H
Πόρτα Εξόδου 80H