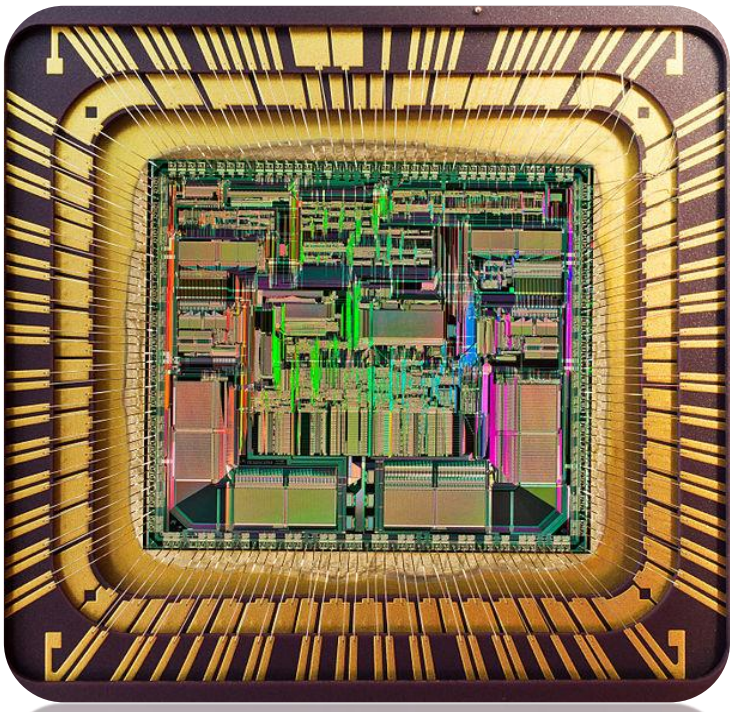
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ

2η Ομάδα Ασκήσεων

Βαβουλιώτης Γεώργιος
ΑΜ : 03112083
6ο Εξάμηνο

Γιαννόπουλος Αναστάσης
Α.Μ. : 03112176
6ο Εξάμηνο



ΑΣΚΗΣΗ 1

Ο κώδικας σε assembly τις άσκησης αυτής φαίνεται παρακάτω :

```
BEGIN:
IN 10H

;A ERWTHMA
LXI H,0900H      ;ARXIKH ADDRESS 0900
MVI A,00H        ;A THA PERIEXEI APO 0 EWS 255

LOOP1:
MOV M,A          ;(HL)=CURRENT NUMBER
INR A            ;NEXT NUMBER
INX H            ;NEXT ADDRESS
MOV M,A          ;HL)=NEXT NUMBER
CPI FFH          ;CURRENT NUMBER = 255
JNZ LOOP1        ;AN OXI SUNEXISE NA VAZEIS ARITHMOUS
                 ;ALLIWS TELOS A ERWTHMATOS

;B ERWTHMA
LXI B,0008H      ;ARXIKOPOIW TO PLHTHOS TWN ASWN ME 8 POU EINAI
                 ;OI ASOI TOY 255
MVI A,00H        ;KRATA TOUS ARITHMOUS O EWS 255
MVI D,00H        ;O D THA XREIASTEI GIA NA SWZEI TON A

NEXT_NUM:
MOV A,D          ;EPANAFORA A(TIN 1H FORA A=00H)
MVI E,00H        ;KRATA TO PLHTHOS TWN BIT POU CHECKARISTHKAN
                 ;GIA TO AN EINAI 1 H 0
INR A            ;NEXT NUMBER
MOV D,A          ;SAVE A
CPI FFH          ;IF NUMBER = 255 THEN FINISH
JNZ CHECK        ;ELSE CHECK THE NUMBER
JZ FINISH        ;TELOS B ERWTHMATOS

CHECK:
INR E            ;E=THESI CURRENT BIT
MOV A,E          ;(A)<--(E)
CPI 09H          ;IF E = 09H THEN GO TO THE NEXT NUMBER
MOV A,D          ;EPANAFORA A
JNZ CONTINUE     ;ELSE CONTINUE
JZ NEXT_NUM

CONTINUE:
RAR              ;CY = LSB
JNC CHECK        ;IF CY = 0 TOTE TSEKARE TO EPOMENO BIT
INX B            ;ELSE ANOTHER ONE ACE
JC CHECK         ;CHECK THE NEXT BIT

;G ERWTHMA
FINISH:
MVI E,0FH        ;O E THA PAREI TOUS ARITHMOUS APO 10H EWS 60H
MVI D,00H        ;PLHTHOS ARITHMWN APO 10H EWS 60H
```

```
CALCULATE:
INR E              ;NEXT NUMBER
INR D              ;PLHTHOS=PLHTHOS+1
MOV A,E            ;(A)<--(E)
CPI 60H            ;IF E=60H THEN SHOW THE RESULT
JNZ CALCULATE      ;ELSE CONTINUE CALCULATING
JZ RESULT

;D ERWTHMA
RESULT:
STC
CMC                ;CY = 0 GIA SIGOURIA
LDA 2000H          ;READ INPUT FROM SWITCHES
RAL                ;ME 6 SHIFT LEFT EXW CY = 3o LSB FROM SWITCHES
RAL                ;KANW RAL GIA NA IKANOPOIEITAI H EKS ARISTERWN
PROTERAIOTHTA
RAL
RAL
RAL
RAL
JC SHOW_D          ;IF 3o LSB = 1 THEN SHOW D
RAL
JC SHOW_C          ;IF 2o LSB = 1 THEN SHOW C
RAL ;ELSE
JC SHOW_B          ;IF 1o LSB = 1 THEN SHOW B
MVI A,FFH          ;ELSE ALL LED ARE OFF
STA 3000H          ;OUTPUT
JMP RESULT         ;CONTINUE TO READ INPUT

SHOW_B:
MOV A,B            ;(A)<--(B)
CMA                ;SUMPLHRWSH TWN LED
STA 3000H          ;OUTPUT
JMP RESULT         ;CONTINUE TO READ INPUT

SHOW_C:
MOV A,C            ;(A)<--(C)
CMA                ;SUMPLHRWSH TWN LED
STA 3000H          ;OUTPUT
JMP RESULT         ;CONTINUE TO READ INPUT

SHOW_D:
MOV A,D            ;(A)<--(D)
CMA                ;SUMPLHRWSH TWN LED
STA 3000H          ;OUTPUT
JMP RESULT         ;CONTINUE TO READ INPUT

END: END
```

<u>Παρατηρήσεις σχετικά με τον κώδικα</u>:

- Στον κώδικα έχω κάνει **bold** τις ετικέτες του προγράμματος στις οποίες γίνονται τα διάφορα jumps για να είναι εμφανή τα διάφορα μέρη του κώδικα.

ΑΣΚΗΣΗ 2

Ο κώδικας σε assembly τις άσκησης αυτής φαίνεται παρακάτω :

```
MVI B,03H
MVI C,E8H              ;(BC)=03E8H FOR TIME DELAY WITH ROUTINE
                       ;DELB
BEGIN:

      LDA 2000H        ;READ SWITCHES
      RAL              ;TAKE THE MSB OF THE DIP-SWITCHES AND
                       ;CHECK IT
      JC BEGIN         ;IF MSB=1 THE SWITCH IS UP.
                       ;THEN WE HAVE TO READ INPUT AGAIN,
                       ;UNTIL THE MSB SWITCH IS DOWN

DOWN_TO_UP:            ;NOW MSB IS DOWN FOR SURE

      LDA 2000H        ;READ INPUT
      RAL              ;IF MSB IS 1 --> C=1
      JNC DOWN_TO_UP   ;READ INPUT UNTIL SWITCH IS UP


UP_TO_DOWN:            ;NOW MSB IS UP FOR SURE

      LDA 2000H        ;READ INPUT
      RAL              ;IF MSB=0 --> C=0 AND LED HAVE TO OPEN
      JC UP_TO_DOWN    ;READ INPUT UNTIL SWITCH IS DOWN


MSBLED_ON:             ;THE LED IS ON FOR SURE BECAUSE WE HAD
                       ;OFF-ON-OFF
      MVI D,00H        ;(D)=0 --> MSB IS DOWN
      MVI E,C8H        ;(E)=200 IN DECIMAL,ITS A COUNTER FOR
                       ;SECS
      MVI A,7FH        ;(A)=01111111 IN BINARY
      STA 3000H        ;ANAVEI MONO TO MSB
                       ;EPEIDH EINAI ANTISTROFHS LOGIKHS

DELAY:
      CALL DELB        ;DELB DELAYS 1 SEC
      DCR E            ;(E)=(E)-1
      MOV A,E          ;(A)<--(E)
      CPI 00H          ;COMPARE A WITH 00H
      JZ LED_OFF       ;IF (A)=00H THEN MSB_LED HAVE TO BE OFF
      MOV A,D          ;(A)<--(D)
      CPI 00H          ;CHECK THE STATE WHERE I AM
      JZ STATE1        ;IF Z=0 THE DIP-SWITCH IS DOWN YET
      CPI 01H          ;COMPARE (A) WITH 01H
      JZ STATE2        ;IF Z=0 THE DIP-SWITCH WENT FROM
                       ;OFF TO ON. CHECK NEXT MOVE
```

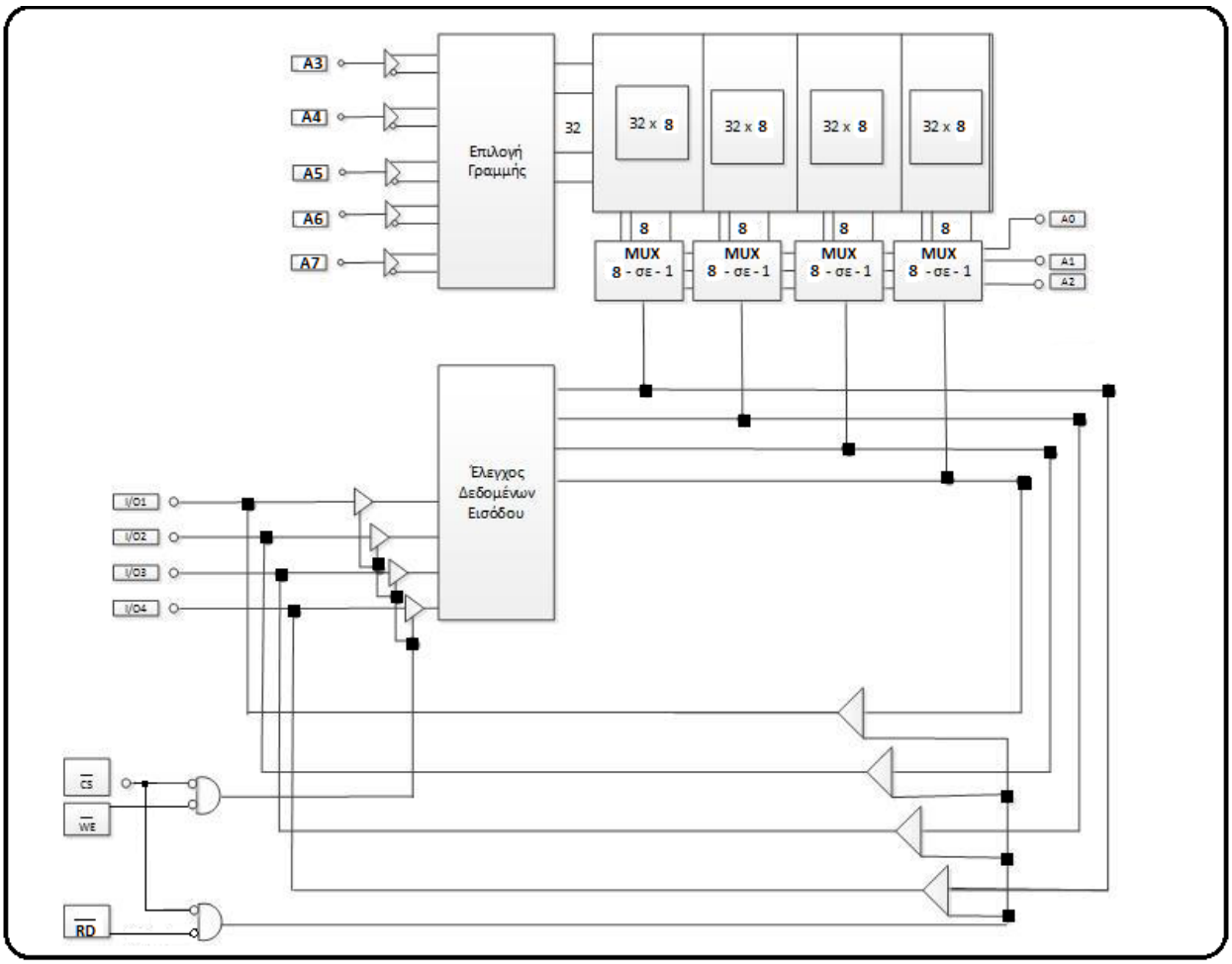
```
LED_OFF:              ;CLOSE THE MSB_LED

      MVI A,FFH       ;(A)=11111111(BINARY)
      STA 3000H       ;ALL LEDS OFF
      JMP BEGIN


STATE1:

      LDA 2000H       ;READ INPUT AGAIN
      RAL             ;TAKE THE MSB OF SWITCHES
      JNC DELAY       ;IF MSB=0 THEN ITS DOWN YET
                      ;THERE ARE NO CHANGES
      INR D           ;ELSE (D)=01H AND THAT MEANS
                      ;THAT MSB WENT FROM OFF TO ON
      JMP DELAY


STATE2:

      LDA 2000H       ;READ INPUT AGAIN
      RAL             ;TAKE THE MSB OF SWITCHES
      JC DELAY        ;IF MSB=1 CONTINUE AS YOU ARE
      JMP REFRESH     ;IF MSB=0 HAVE TO REFRESH DELB TIME


REFRESH:              ;MSB_LED HAVE TO BE OPEN
                      ;FOR ANOTHER 20 SECS

      MVI E,C8H       ;(E)=200 IN DECIMAL
      MVI D,00H       ;(D)=00H, WE MAKE A NEW START
      JMP DELAY

END: END
```

<u>Παρατηρήσεις σχετικά με τον κώδικα και τον τρόπο υλοποίησης</u> :

- Στον κώδικα έχω κάνει **bold** τις ετικέτες του προγράμματος στις οποίες γίνονται τα διάφορα jumps για να είναι εμφανή τα διάφορα μέρη του κώδικα.
- Εισάγω καθυστέρηση στο πρόγραμμα με χρήση της συνάρτησης DELB η οποία εισάγει καθυστέρηση ίση με (1ms)*(BC). Το γεγονός αυτό δικαιολογεί και την παρουσία των δυο εντολών ανάθεσης στην αρχή του κώδικα. Θα πρέπει να τονιστεί ότι η καθυστέρηση που έχει εισαχθεί είναι 20 secs αλλά αυτό ισχύει για τον υπολογιστή που υλοποιήθηκε η άσκηση. Σε ένα άλλο υπολογιστή ανάλογα με τις δυνατότητές του η καθυστέρηση αυτή αλλάζει και πολλές φορές η απόκλιση από τα ζητούμενα 20 secs είναι αρκετά μεγάλη. Επίσης βάζω να γίνουν 200 επαναλήψεις για να περάσουν τα ζητούμενα 20 secs λόγω της διακριτικής ικανότητας του συστήματος που είναι 1/10 sec.

ΑΣΚΗΣΗ 3

Παρακάτω παρουσιάζεται η εσωτερική οργάνωση μιας μνήμης **SRAM 256×4bits**:



Εξήγηση λειτουργίας

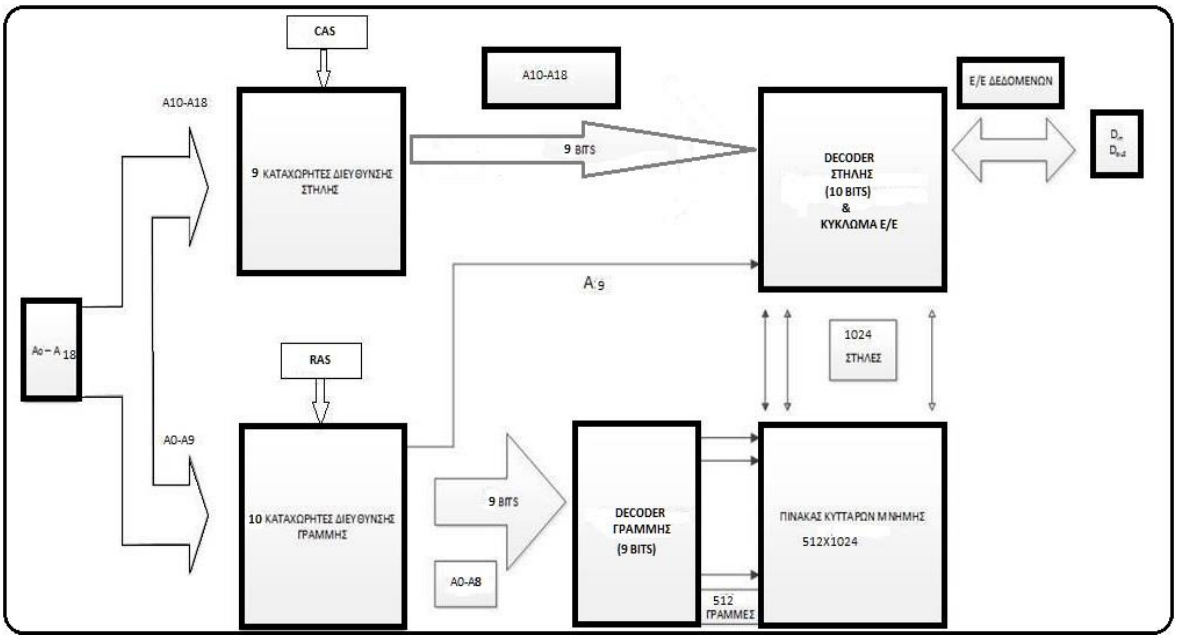
- **Ανάγνωση:** Η είσοδος $\overline{CS} = 0$, η είσοδος $\overline{RD} = 0$ και η είσοδος $\overline{WE} = 1$ → η πάνω πύλη AND έχει έξοδο 0 και η κάτω πύλη AND έχει έξοδο 1. Λόγω της πρώτης εμποδίζεται το σήμα εισόδου να καταγραφεί στη μνήμη, ενώ λόγω της δεύτερης επιτρέπεται η ανάγνωση από τη μνήμη. Η διεύθυνση της οποίας το περιεχόμενο θέλουμε να διαβάσουμε εφαρμόζεται στις εισόδους A_0-A_7. Ας υποθέσουμε για παράδειγμα ότι θέλουμε να διαβάσουμε το περιεχόμενο της μνήμης που βρίσκεται στη διεύθυνση 00111101 (αυτή εφαρμόζεται στα A_0-A_7). Τα περισσότερο 5 σημαντικά ψηφία 00111 (=7) εφαρμόζονται ως είσοδος στον επιλογέα γραμμής, ο οποίος είναι ένας decoder 5 σε 32. Έτσι, επιλέγεται η $7^η$ γραμμή της μνήμης, η οποία έχει μήκος 32 bits (=8 τετράδες). Από αυτές, επιλέγεται μόνο η $5^η$ τετράδα λόγω των ελάχιστα σημαντικών bits 101 (=5) τα οποία εισάγονται ως επιλογείς στους πολυπλέκτες. Ως αποτέλεσμα, επειδή οι 4 απομονωτές έχουν επίτρεψη 1, η τετράδα που επιλέγεται περνάει στις 4 γραμμές I/O.

- **<u>Εγγραφή:</u>** Η είσοδος $\overline{CS}$ = 0, η είσοδος $\overline{WE}$ = 0 και η είσοδος $\overline{RD}$ = 1 → η πάνω πύλη AND έχει έξοδο 1 και η κάτω πύλη AND έχει έξοδο 0. Έτσι ενεργοποιούνται οι απομονωτές που επιτρέπουν τη διέλευση των γραμμών I/O, ενώ απενεργοποιούνται οι απομονωτές μέσω των οποίων γίνεται διέλευση των δεδομένων της μνήμης. Η επιλογή της μνήμης με διεύθυνση 00111101 γίνεται με ακριβώς ίδιο τρόπο αλλά η μόνη διαφορά είναι ότι εδώ, αποθηκεύεται στη μνήμη μέσω των πολυπλεκτών, η πληροφορία των εισόδων I/O.

<u>Παρατηρήσεις:</u>

1. Είναι προφανές ότι τα σήματα RD' και WE' πρέπει να είναι πάντα συμπληρωματικά, αφού δεν γίνεται να ζητάει μια άλλη δραστηριότητα εγγραφή και ανάγνωση του περιεχομένου μιας διεύθυνσης την ίδια στιγμή.
2. Οι πολυπλέκτες έχουν την δυνατότητα αμφίδρομης ροής (από και προς τον πίνακα).

Αντίστοιχη είναι και η εσωτερική οργάνωση μίας μνήμης **DRAM 512K×1bit**:



<u>Εξήγηση λειτουργίας</u>

Η συγκεκριμένη μνήμη έχει διεύθυνση μήκους 19bits και είναι οργανωμένη σε λέξεις των 1bit. Επιλέγουμε τη δημιουργία του πίνακα κυττάρων μνήμης με διαστάσεις (512 ×1024). Επομένως, για τον προσδιορισμό της διεύθυνσης στήλης απαιτούνται τα 9 MSB (2^9=512) και για τον προσδιορισμό γραμμής απαιτούνται τα 10 LSB (2^{10}=1024). Η διαδικασία ανάγνωσης και εγγραφής περιλαμβάνει την μεταφορά της διεύθυνσης στην είσοδο. Τα ψηφία A_0-A_9 που αποτελούν τη διεύθυνση γραμμής, κρατούνται στο TMS4416 με τον παλμό $\overline{RAS}$ και στη συνέχεια τα ψηφία A_{10}-A_{18} κρατούνται στο ίδιο chip με τον παλμό $\overline{CAS}$.

ΑΣΚΗΣΗ 4

Ζητούμενο της παρούσας άσκησης είναι η κατασκευή ενός συστήματος μνήμης για μΥ σύστημα του μΕ 8085. Διαθέσιμες μνήμες:

- EPROM 10kB (χρησιμοποιούνται δύο ολοκληρωμένα κυκλώμαφα 4Kx8bit και ένα 2Kx8bit)
- RAM 6kB (ένα 4Kx8bit και ένα 2Kx8bit)

Η ROM θέλουμε να ξεκινάει από τη θέση 0000H και αμέσως μετά να βρίσκεται ο χώρος που χρησιμοποιεί η μνήμη RAM. Επομένως θα τοποθετήσουμε τα ολοκληρώμενα της EPROM πρώτα και της RAM στη συνέχεια. Επιλέγουμε για κάθε είδος μνήμης να τοποθετήσουμε πρώτα το chip μεγαλύτερης χωρητικότητας και έπειτα το μικρότερο, χωρίς να άλλαζει η λειτουργία του συστήματος μνήμης αν γίνει ανάποδη τοποθέτηση.

Χάρτες Μνημών

Άρχικα κατάσκευάζουμε το χάρτη μνήμης. Το πρώτο chip EPROM που χρησιμοποιούμε έχει διαστάσεις 4Kx8bit = 2^{12}x8bit, δηλαδή χρειάζομαστε 12bits διεύθυνσης για να προσδιορίσουμε ποιο τμήμα της σελίδας της EPROM θέλουμε να προσπελάσουμε και επίσης από την έξοδο του συγκεκριμένου chip μπορούμε να παίρνουμε κάθε φορά μία πλήρη λέξη των 8bit. Ο χάρτης μνήμης για το συγκεκριμένο chip είναι:

EPROM 1 (4Kx8bit)				
Αρχή διευθύνσεων				
BIN	0000	0000	0000	0000
HEX	0	0	0	0
Τέλος διευθύνσεων				
BIN	0000	1111	1111	1111
HEX	0	F	F	F

Αντίστοιχα το επόμενο chip EPROM ξεκινάει από την αμέσως επόμενη διεύθυνση μνήμης και λόγω της μίας διάστασής του (4K = 2^{12}) χρειάζεται 12bit διεύθυνσης για να προσδίοριστει το τμήμα της μνήμης που θέλουμε από αυτό. Έτσι, ο χάρτης μνήμης στην περίπτωση αυτή είναι:

EPROM 2 (4Kx8bit)				
Αρχή διευθύνσεων				
BIN	0001	0000	0000	0000
HEX	1	0	0	0
Τέλος διευθύνσεων				
BIN	0001	1111	1111	1111
HEX	1	F	F	F

Στη συνέχεια, ακολουθεί το επόμενο chip της μνήμης EPROM. Αυτό το chip της EPROM, επειδή θα έχει μία διάσταση 2K = 2^{11}, χρειάζονται 11 bits για την παράσταση μιας διεύθυνσης τους. Οι χάρτες μνήμης για τα τρίτο chip της EPROM είναι:

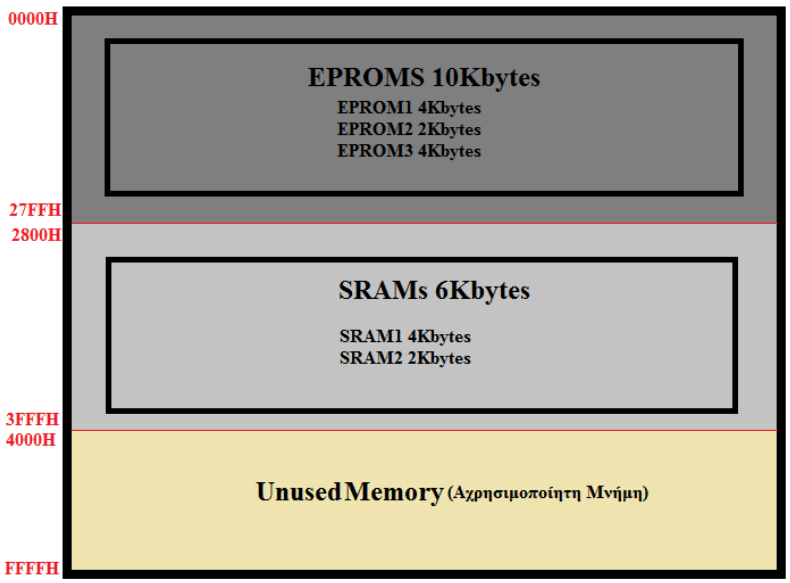
EPROM 3 (2Kx8bit)				
Αρχή διευθύνσεων				
BIN	0010	0000	0000	0000
HEX	2	0	0	0
Τέλος διευθύνσεων				
BIN	0010	0111	1111	1111
HEX	2	7	F	F

Στη συνέχεια ακολουθούν τα chip της μνήμης RAM. Το πρώτο chip της RAM (SRAM) θα έχει μία διάσταση 4K = 2^{12}, δηλ. χρειάζονται 12 bits για την παράσταση μιας διεύθυνσης τους, ενώ το δεύτερο 11 bits. Οι χάρτες μνήμης για τα 2 chip της RAM είναι:

SRAM 1 (4KX8bit)				
Αρχή διευθύνσεων				
BIN	0010	1000	0000	0000
HEX	2	8	0	0
Τέλος διευθύνσεων				
BIN	0011	0111	1111	1111
HEX	3	7	F	F

SRAM 2 (2KX8bit)				
Αρχή διευθύνσεων				
BIN	0011	1000	0000	0000
HEX	3	8	0	0
Τέλος διευθύνσεων				
BIN	0011	1111	1111	1111
HEX	3	F	F	F

Ο συνολικός χάρτης μνημών φαίνεται παρακάτω :



Τώρα, ο στόχος μας είναι να εντοπίσουμε ένα μοτίβο με το οποίο θα επιλέγεται το κάθε chip μνήμης (πότε δηλαδή η υποδοχή CS' του κάθε chip θα πρέπει να γίνεται 0). Για το σκοπό αυτό τοποθέτουμε σε ένα πίνακα την αρχική και την τελική διεύθυνση κάθε chip. Έτσι:

EPROM 1	0000	0000	0000	0000
	0000	1111	1111	1111
EPROM 2	0001	0000	0000	0000
	0001	1111	1111	1111
EPROM 3	0010	0000	0000	0000
	0010	0111	1111	1111
SRAM 1	0010	1000	0000	0000
	0011	0111	1111	1111
SRAM 2	0011	1000	0000	0000
	0011	1111	1111	1111

Παρατηρώντας τον παραπάνω πίνακα συμπεραίνουμε ότι τα 5 MSB είναι ικανά να μας προσδιορίσουνε το επιθυμητό chip. Επίσης βλέπουμε ότι είτε στην αρχική είτε στην τελική διεύθυνση του κάθε chip τα ψηφία A15-A14 (πράσινα ψηφία) είναι 0. Άρα διαφοροποιήση υπάρχει στα ψηφία A13-A12-A11 (κόκκινα ψηφία). Έτσι, για την επιλογή του chip θα χρησιμοποιήσουμε τον αποκωδικοποιητή 3 σε 8 (74LS138), ο οποίος έχει 3 εισόδους ενεργοποίησης (E1-E3) και 3 εισόδους που δίνουν τις 8 εξόδους. Στις δύο εισόδους ενεργοποίησης θα συνδεθούν τα bit A15-A14, αφού περάσουν από μία πύλη αντιστροφέα, ώστε η μνήμη να μην είναι προσπελάσιμη στην περίπτωση διευθύνσεων με πιο σημαντικά ψηφία τα 01,10,11 (απ' το datasheet του 74138 ισχύει $E_{1,2} = E_1$ OR E_2).

Σχεδίαση Διάταξης

Για να επιλέξουμε ποια έξοδος του αποκωδικοποιητή ενεργοποιεί ποιο chip κατασκεύαζουμε τον ακόλουθο πίνακα:

A_{13}	A_{12}	A_{11}	EPROM1	EPROM2	EPROM3	SRAM1	SRAM2
0	0	0	1	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	1	0	0	0
0	1	1	0	1	0	0	0
1	0	0	0	0	1	0	0
1	0	1	0	0	0	1	0
1	1	0	0	0	0	1	0
1	1	1	0	0	0	0	1

Όταν δύο τριάδες διευθύνσεων ενεργοποιούν το ίδιο chip, τότε εισάγονται σε μία AND, της οποίας η έξοδος οδήγειται στο επιθυμητό chip. Αυτό συμβαίνει επειδή ο αποκωδικοποιητής έχει συμπληρωματικές εξόδους σε δεδομένη είσοδο και στο CS πρέπει να εισαχθεί 0 για να ενεργοποιηθεί. Επίσης, σύμφωνα με τον πίνακα με τις αρχικές και τελικές διευθύνσεις, πρέπει να κυκλοφορούν στο ADDRESS BUS τα bits A0-A13 για να μπορούν να προσδιορίστουν πλήρως όλες οι λέξεις που υπάρχουν στην μνήμη (αυτό συμβαίνει επειδή στην περίπτωση της RAM1 παρατηρούμε διαφοροποίηση της αρχικής και τελικής διεύθυνσης στα 14 λιγότερο σημαντικά bit). Προκύπτει η διάταξη: