Συστήματα Μικροϋπολογιστών
2η σειρά ασκήσεων

Νικόλαος Παγώνας, el18175
Αναστάσιος Παπαζαφειρόπουλος, el18079

1η άσκηση

Ο κώδικας για την 1η άσκηση, μαζί με τα κατάλληλα σχόλια, έχει συμπεριληφθεί στο αρχείο `ex1.8085`.

2η άσκηση

Ο κώδικας για την 2η άσκηση, μαζί με τα κατάλληλα σχόλια, έχει συμπεριληφθεί στο αρχείο `ex2.8085`.

3η άσκηση

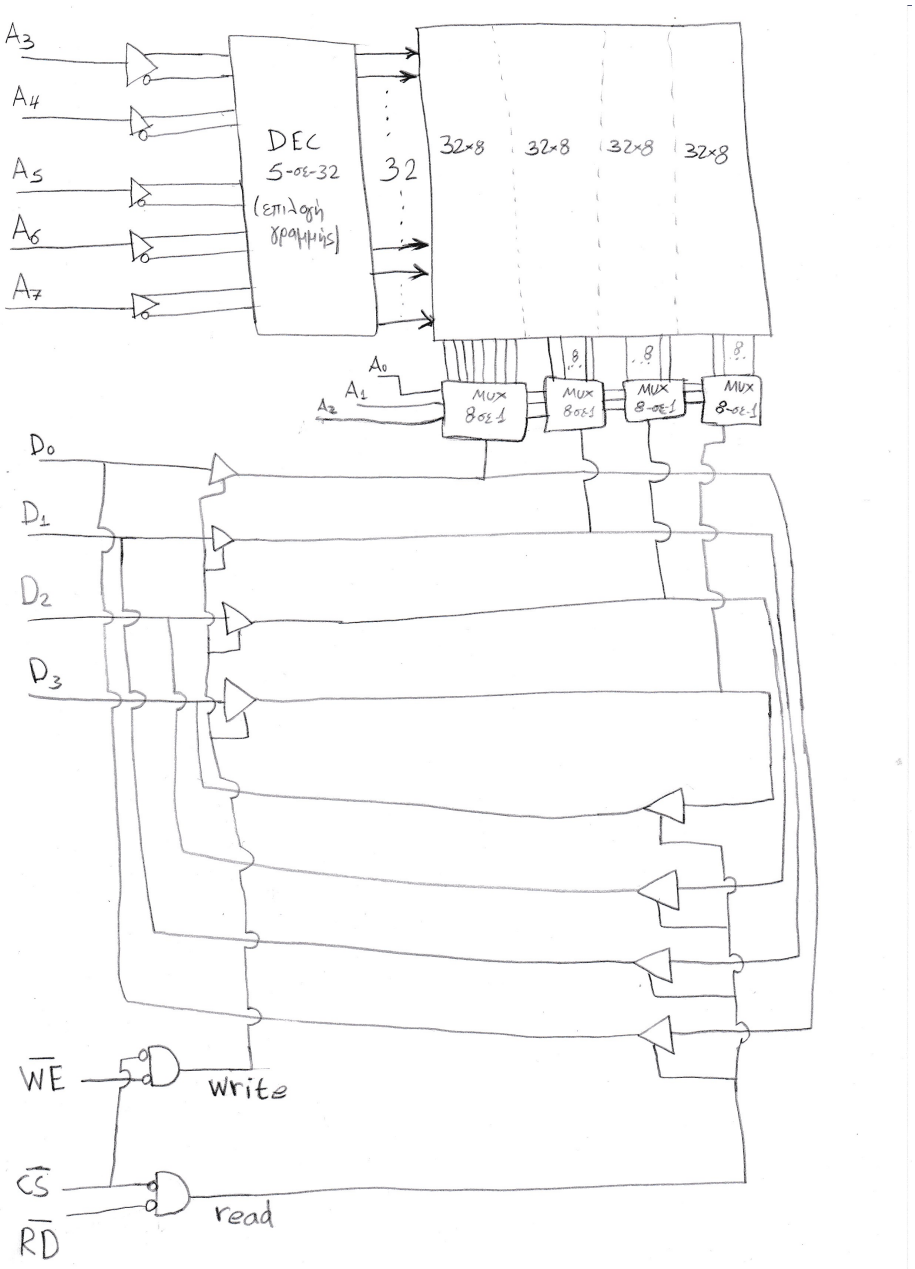
Ο κώδικας για την 3η άσκηση, μαζί με τα κατάλληλα σχόλια, έχει συμπεριληφθεί στα αρχεία `ex3i.8085`, `ex3ii.8085` και `ex3iii.8085` αντίστοιχα.

4η άσκηση

Ο κώδικας για την 4η άσκηση, μαζί με τα κατάλληλα σχόλια, έχει συμπεριληφθεί στο αρχείο `ex4.8085`.

5η άσκηση

Η ζητούμενη μνήμη SRAM 256x4 bits έχει χώρο για 256 λέξεις μεγέθους 4 bits. Χωρίζουμε λοιπόν τη μνήμη σε 4 τμήματα μεγέθους 256 bits, ώστε η κάθε λέξη να μοιράζει από ένα bit σε καθένα από τα τέσσερα τμήματα. Ουσιαστικά, το κάθε τμήμα είναι ένας πίνακας με διαστάσεις α και β με $\alpha \cdot \beta = 256$. Για να πετύχουμε τετραγωνικό σχήμα επιλέγουμε διαστάσεις 32x8. Η επιλογή γραμμής απαιτεί: $log_2 32 = 5$ bits οπότε για τον σκοπό αυτό χρησιμοποιούνται οι ακροδέκτες διευθύνσεων $A_0 - A_2$ με πολυπλέκτες 8-σε-1. Αντίστοιχα, η επιλογή στήλης απαιτεί: $log_2 8 = 3$ bits και για το σκοπό αυτό χρησιμοποιούνται οι ακροδέκτες διευθύνσεων $A_3 - A_7$ με αποκωδικοποιητή 5-σε-32. Οι ακροδέκτες $D_0 - D_3$ είναι οι ακροδέκτες εισόδου/εξόδου των 4 bit και τα σήματα $\overline{CS}$ και $\overline{RD}$ μέσω των αντίστοιχων ακροδεκτών επιτρέπουν ή αποτρέπουν τις λειτουργίες εγγραφής και ανάγνωσης. Τα τρία σήματα ελέγχου έχουν αρνητική πολικότητα, δηλαδή ενεργοποιούν τη λειτουργία τους, το καθένα, όταν είναι μηδέν. Το σήμα $\overline{CS}$ (αρνητικό ChipSelect) ενεργοποιεί ή αδρανοποιεί ολόκληρο το chip και προορίζεται για χρήση όταν φτιάχνουμε μια μεγάλη μνήμη από πολλά chips, για να επιλέγουμε σε ποιο chip απευθυνόμαστε κάθε φορά. Όταν $\overline{CS} = 0$ (ενεργό chip), το σήμα $\overline{WE}$ (αρνητικό WriteEnable) ενεργοποιεί την εγγραφή ενώ το σήμα $\overline{RD}$ ενεργοποιεί την ανάγνωση.



Ανάγνωση από τη μνήμη:

Αρχικά εφαρμόζουμε στις γραμμές $A_0 - A_7$ τη διεύθυνση από την οποία θέλουμε να διαβάσουμε. Το σήμα $\overline{CS}$ τίθεται στο λογικό 0, μέσω αρνητικού παλμού και σταματάει η απομόνωση εισόδου και εξόδου της μνήμης. Έρχεται αρνητικός παλμός στον ακροδέκτη $\overline{RD}$, ενώ στον $\overline{WE}$ έρχεται θετικός και ξενικάει η ανάγνωση από τη μνήμη αφού η έξοδος της αντίστοιχης πύλης AND (read) γίνεται 1, σε

αντίθεση με την έξοδος της πύλης AND (write) που γίνεται 0. Οι ακροδέκτες $A_3 - A_7$ καθορίζουν ποια γραμμή θα επιλεχθεί για την ανάγνωση των 4 bits, ενώ οι ακροδέκτες $A_0 - A_2$ επιτρέπουν στα bit της επιλεγμένης στήλης να φτάσουν στις εξόδους των πολυπλεκτών. Στη συνέχεια το σύρμα read, οδηγείται στο enable των τρισταθών buffers που κοιτούν προς τα αριστερά και έτσι επιτρέπεται το πέρασμα πληροφορίας μέσα από τους D_0, D_1, D_2, D_3. Με αυτόν τον τρόπο η πληροφορία τους περνάει στην έξοδο των τρισταθών buffers με αποτέλεσμα διαβάζουμε την επιθυμητή διεύθυνση. Οι τρισταθείς buffers που κοιτούν προς τα δεξιά τίθενται σε κατάσταση υψήλης αντίστασης. Έπειτα, τελειώνει ο αρνητικός παλμός $\overline{RD}$ και επανέρχεται το $\overline{CS}$ στο λογικό 1.

Εγγραφή στη μνήμη:

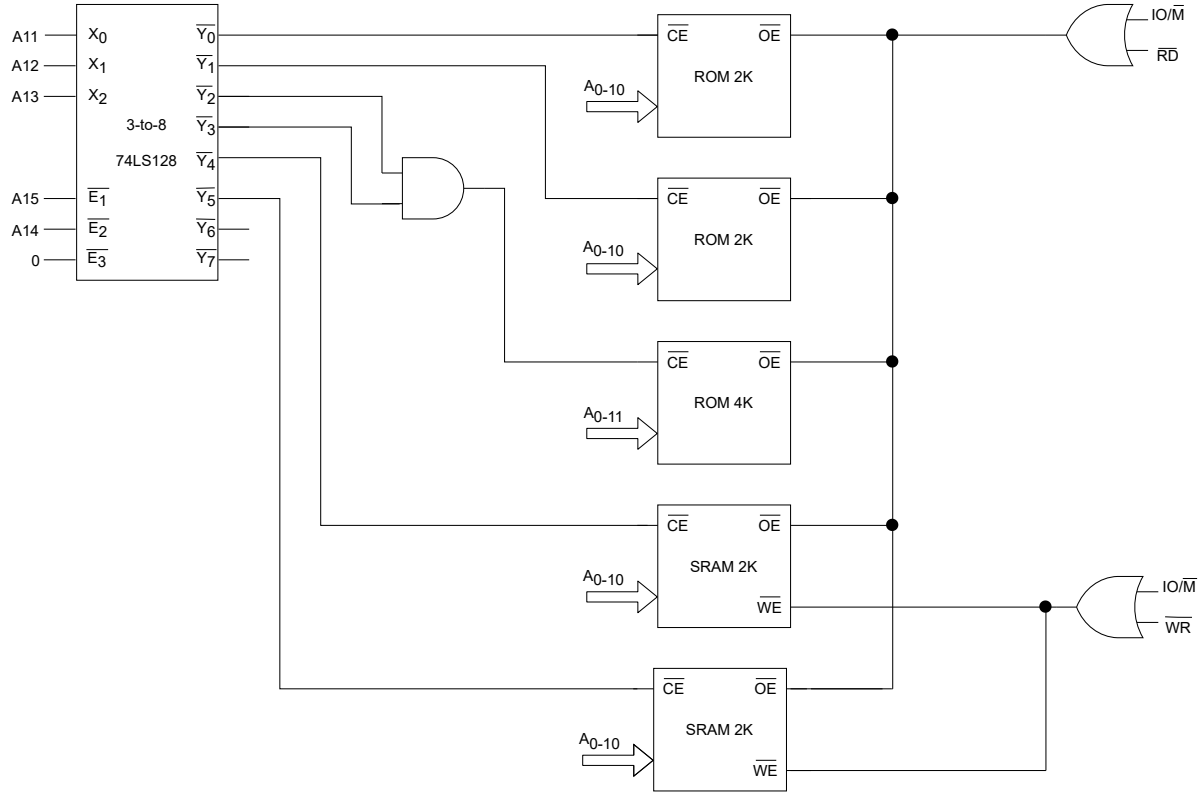
Αρχικά εφαρμόζουμε στις γραμμές $A_0 - A_7$ τη διεύθυνση στην οποία θέλουμε να γράψουμε. Το σήμα $\overline{CS}$ τίθεται στο λογικό 0, μέσω αρνητικού παλμού και σταματάει η απομόνωση εισόδου και εξόδου της μνήμης. Έρχεται αρνητικός παλμός στον ακροδέκτη $\overline{WE}$, ενώ στον $\overline{RD}$ έρχεται θετικός και ξενικάει η εγγραφή στη μνήμη αφού η έξοδος της αντίστοιχης πύλης AND (write) γίνεται 1, σε αντίθεση με την έξοδος της πύλης AND (read) που γίνεται 0. Οι $A_0 - A_2$ προσδιορίζουν σε ποια στήλη από τις 8 να οδηγήσουν την είσοδο και οι $A_3 - A_7$ σε ποια γραμμή θα γίνει η εγγραφή των 4 bits. Στη συνέχεια το σύρμα write, οδηγείται στο enable των τρισταθών buffers που κοιτούν προς τα δεξιά ελέγχοντας το πέρασμα πληροφορίας από τις εισόδους D_0, D_1, D_2, D_3 οι οποίες ενεργοποιούνται. Με αυτόν τον τρόπο η πληροφορία τους περνάει στην έξοδο των τρισταθών buffers και ανανεώνει το περιεχόμενο της κατάλληλης θέσης μνήμης η οποία έχει επιλεχθεί. Οι τρισταθείς buffers που κοιτούν προς τα αριστερά τίθενται σε κατάσταση υψήλης αντίστασης. Έπειτα, τελειώνει ο αρνητικός παλμός $\overline{WE}$ και επανέρχεται το $\overline{CS}$ στο λογικό 1.

6η άσκηση

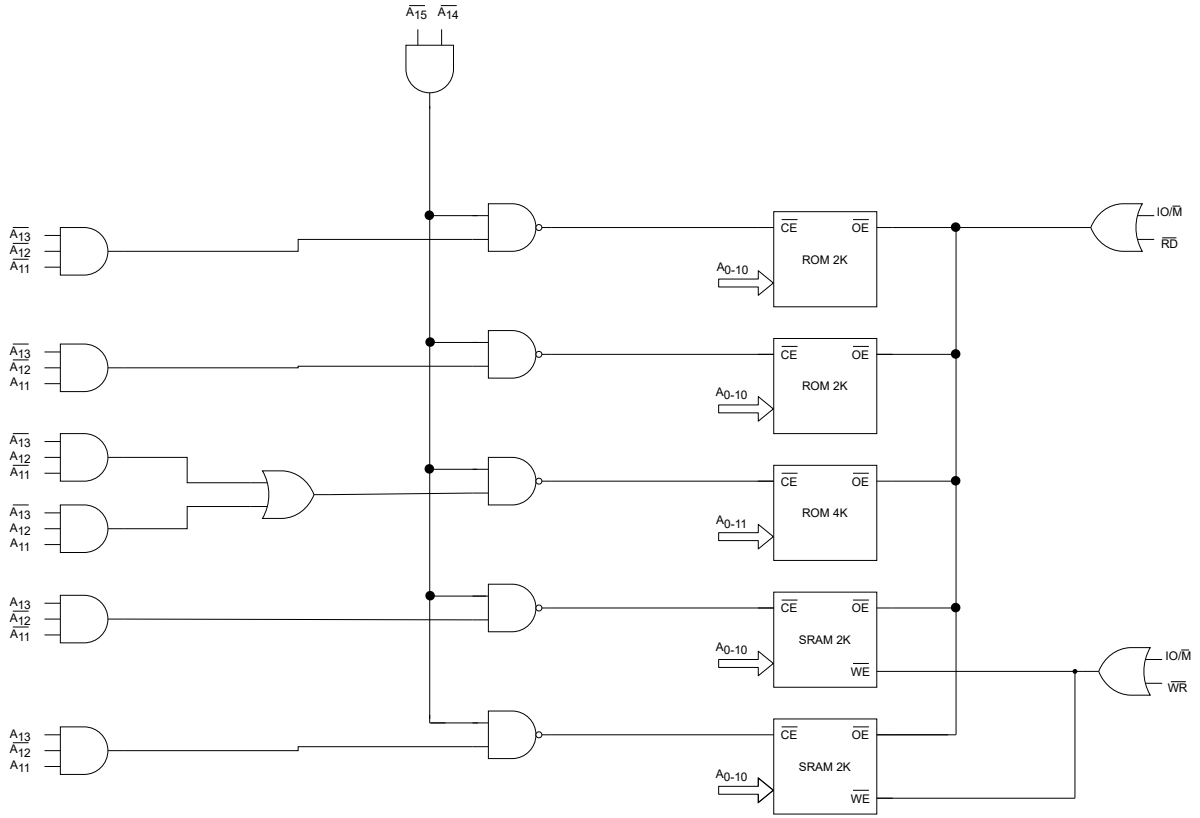
Ο χάρτης μνήμης που προκύπτει από τα δεδομένα της εκφώνησης είναι ο εξής:

Memory	**Bit**																**Hex**
	15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
ROM1 (2K)	0	0	**0**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0000
	0	0	**0**	**0**	**0**	1	1	1	1	1	1	1	1	1	1	1	07FF
ROM2 (2K)	0	0	**0**	**0**	**1**	0	0	0	0	0	0	0	0	0	0	0	0800
	0	0	**0**	**0**	**1**	1	1	1	1	1	1	1	1	1	1	1	0FFF
ROM3 (4K)	0	0	**0**	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0	1000
	0	0	**0**	**1**	**1**	1	1	1	1	1	1	1	1	1	1	1	1FFF
SRAM1 (2K)	0	0	**1**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	2000
	0	0	**1**	**0**	**0**	1	1	1	1	1	1	1	1	1	1	1	27FF
SRAM2 (2K)	0	0	**1**	**0**	**1**	0	0	0	0	0	0	0	0	0	0	0	2800
	0	0	**1**	**0**	**1**	1	1	1	1	1	1	1	1	1	1	1	2FFF

Υλοποίηση με αποκωδικοποιητή 3x8 και λογικές πύλες:



Υλοποίηση μόνο με λογικές πύλες (πλήρης αποκωδικοποίηση):



7η άσκηση

Χάρτης μνήμης:

- 0000H-2FFFH: ROM (12KB)
- 3000H-5FFFH: RAM (12KB)
- 6000H-6FFFH: ROM (4KB)
- 7000H: θύρα εξόδου (Memory map I/O)
- 70H: θύρα εισόδου (Standard I/O)

Address	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0000H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0FFFH	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
1000H	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1FFFH	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
2000H	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
2FFFH	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0
3000H	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
3FFFH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
4000H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
4FFFH	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	0
5000H	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
5FFFH	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0
6000H	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
6FFFH	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0

Από τον παραπάνω πίνακα γίνεται φανερό ότι η διαφοροποίηση των θέσεων μνήμης που καταλαμβάνουν οι ROM, RAM γίνεται μέσω των θέσεων 12, 13, 14. Για αυτό, τα συγκεκριμένα bits χρησιμοποιούνται ως είσοδοι προκειμένου να επιλεχθεί το κατάλληλο ολοκληρωμένο μνήμης. Επιπρόσθετα, οι προδιαγραφές του προβλήματος ορίζουν την κάλυψη των θέσεων μνήμης με αλληλουχία ROM, RAM, ROM. Δηλαδή, η ενιαία μνήμη ROM που διαθέτουμε και έχει μέγεθος 16KB ίσο με το άθροισμα των μεγεθών των επιμέρους 4 KB και οι 3 RAM των 4 KB με συνολικό μέγεθος 12 KB θα πρέπει να παρεμβάλλονται. Οπότε το πρόβλημα ανάγεται στο πως θα μπορέσουμε να διαχωρίσουμε τις διευθύνσεις που αντιστοιχούν σε κάθε κομμάτι της ROM. Αναλυτικά έχουμε ότι λέξεις που αποθηκεύονται στην ROM είναι σε πλήθος $16K = 2^{14}$ και άρα απαιτούνται 14 bits $(A_0 - A_{13})$ για την ορθή διευθυνσιοδότησή τους.

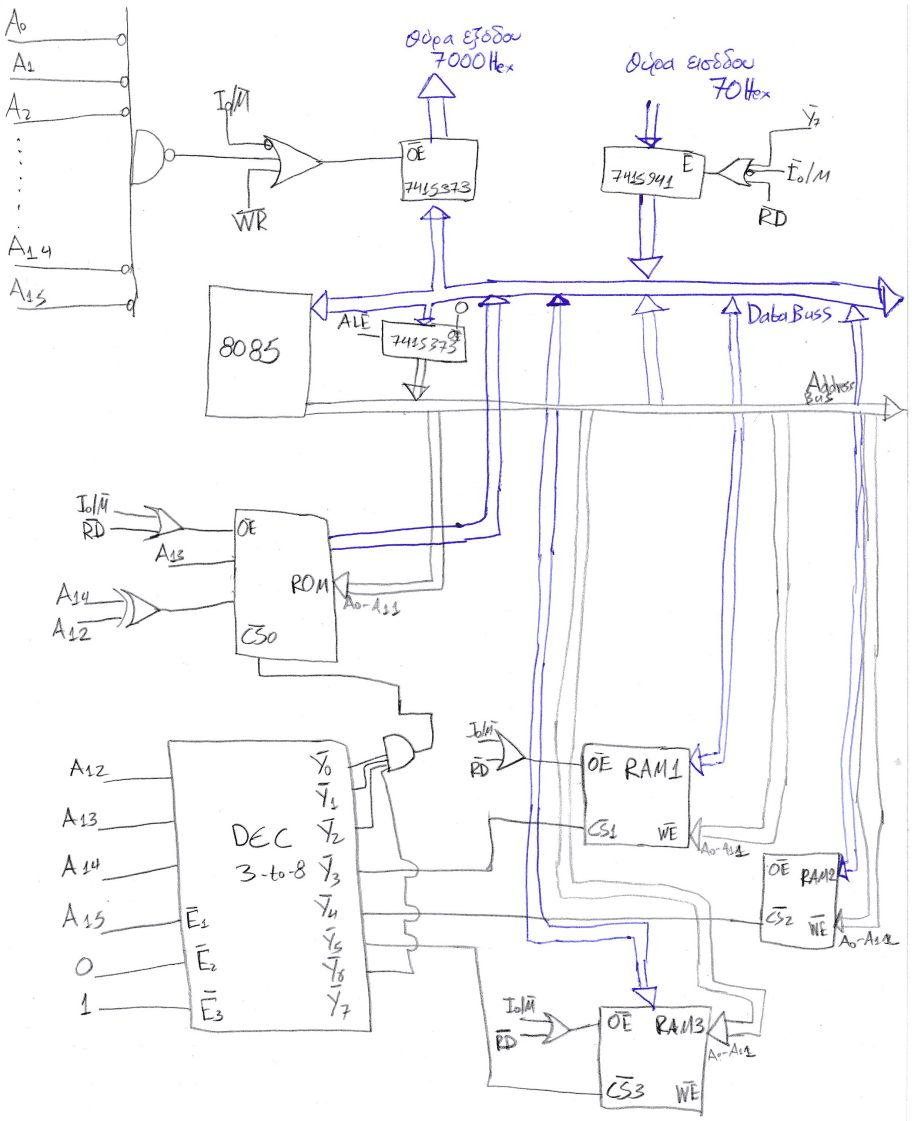
Επίσης, οι λέξεις που αποθηκεύονται στις RAM 1, RAM 2, RAM 3 μεγέθους 4KB είναι σε πλήθος $4K = 2^{12}$ και άρα απαιτούνται 12 bits $(A_0 - A_{11})$ για την ορθή διευθυνσιοδότησή τους.

Τα bits A_{12}, A_{13}, A_{14} χρησιμοποιούνται για την επιλογή του επιθυμητού ολοκληρωμένου (ROM, RAM1, RAM2 ή RAM3) καθώς ένας ή περισσότεροι συνδυασμοί αυτών προσδιορίζουν μοναδικά τις περιοχές μνήμης που αντιστοιχούν σε κάθε ολοκληρωμένο.

Πιο Συγκεκριμένα:

- ROM: $A_{14}A_{13}A_{12} = 000$ και $A_{14}A_{13}A_{12} = 001$ και $A_{14}A_{13}A_{12} = 010$ και $A_{14}A_{13}A_{12} = 110$
- RAM 1: $A_{14}A_{13}A_{12} = 011$
- RAM 2: $A_{14}A_{13}A_{12} = 100$
- RAM 3: $A_{14}A_{13}A_{12} = 101$

Η μνήμη ROM λαμβάνει τα bits $A_0 - A_{11}$ από το address bus ενώ το bit A_{12} από την έξοδο της πύλης XOR και το A_{13} αυτούσιο όπως φαίνεται στο παρακάτω σχήμα. Έτσι τα bits A_{12} και A_{13} μετατρέπουν τις διευθύνσεις του χάρτη μνήμης που αντιστοιχούν σε θέσεις της μνήμης ROM που δεν είναι στο πρώτο τμήμα της (0000H-2FFFH) σε συνεχόμενες θέσεις εσωτερικά στο ολοκληρωμένο. Τέλος, για την επίτρεψη του latch της θύρας εξόδου 7000H χρησιμοποιείται πύλη AND με 16 εισόδους (γιατί είναι memory map I/O), ενώ για την επίτρεψη του latch εισόδου χρησιμοποιείται το $\overline{Y_7}$ (αφού είναι από το standard I/O) που δεν χρησιμοποιείται κάπου αλλού.


A0
A1
A2
A14
A15
IO/M
WR
Θύρα εξόδου
7000 Hex
OE
74LS373
Θύρα εισόδου
70 Hex
74LS541
E
Y7
IO/M
RD
8085
ALE
74LS373
Data Buss
Address Bus
IO/M
RD
A13
A14
A12
OE
ROM
CS0
A0-A11
DEC
3-to-8
A12
A13
A14
A15
0
1
E1
E2
E3
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
IO/M
RD
OE
RAM1
CS1
WE
A0-A11
OE
RAM2
CS2
WE
A0-A12
IO/M
RD
OE
RAM3
CS3
WE
A0-A11