1/5/21

2η Ομάδα Ασκήσεων

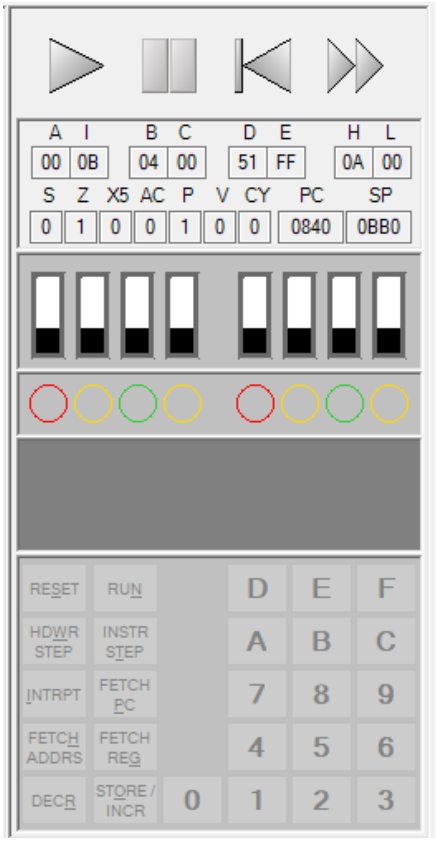
Συστήματα Μικροϋπολογιστών

Βαρδάκης Χριστόφορος el18883
Λυμπεράκης Γεώργιος el18881

1η ΑΣΚΗΣΗ*

Στην άσκηση αυτή, εκτελέσαμε τα τρία ερωτήματα παράλληλα. Δηλαδή το πρόγραμμα μας υπολογίζει τον επόμενο αριθμό, μετρά το πλήθος των 1 και το προσθέτει στο σύνολο, αυξάνει το πλήθος των αριθμών στο [10H , 60H] αν ανήκει στο εύρος αυτό και τέλος αποθηκεύει τον αριθμό στη μνήμη.

Το αποτέλεσμα των καταχωρητών και της μνήμης μετά την εκτέλεση του προγράμματος φαίνετε παρακάτω:



08F3	00	08F4	00	08F5	00	08F6	00	08F7	00	08F8	00	08F9	00	08FA	00	08FB	00
08FC	00	08FD	00	08FE	00	08FF	00	0900	00	0901	01	0902	02	0903	03	0904	04
0905	05	0906	06	0907	07	0908	08	0909	09	090A	0A	090B	0B	090C	0C	090D	0D
090E	0E	090F	0F	0910	10	0911	11	0912	12	0913	13	0914	14	0915	15	0916	16
0917	17	0918	18	0919	19	091A	1A	091B	1B	091C	1C	091D	1D	091E	1E	091F	1F
0920	20	0921	21	0922	22	0923	23	0924	24	0925	25	0926	26	0927	27	0928	28
0929	29	092A	2A	092B	2B	092C	2C	092D	2D	092E	2E	092F	2F	0930	30	0931	31
0932	32	0933	33	0934	34	0935	35	0936	36	0937	37	0938	38	0939	39	093A	3A
093B	3B	093C	3C	093D	3D	093E	3E	093F	3F	0940	40	0941	41	0942	42	0943	43
0944	44	0945	45	0946	46	0947	47	0948	48	0949	49	094A	4A	094B	4B	094C	4C
094D	4D	094E	4E	094F	4F	0950	50	0951	51	0952	52	0953	53	0954	54	0955	55
0956	56	0957	57	0958	58	0959	59	095A	5A	095B	5B	095C	5C	095D	5D	095E	5E
095F	5F	0960	60	0961	61	0962	62	0963	63	0964	64	0965	65	0966	66	0967	67
0968	68	0969	69	096A	6A	096B	6B	096C	6C	096D	6D	096E	6E	096F	6F	0970	70
0971	71	0972	72	0973	73	0974	74	0975	75	0976	76	0977	77	0978	78	0979	79
097A	7A	097B	7B	097C	7C	097D	7D	097E	7E	097F	7F	0980	80	0981	81	0982	82
0983	83	0984	84	0985	85	0986	86	0987	87	0988	88	0989	89	098A	8A	098B	8B
098C	8C	098D	8D	098E	8E	098F	8F	0990	90	0991	91	0992	92	0993	93	0994	94
0995	95	0996	96	0997	97	0998	98	0999	99	099A	9A	099B	9B	099C	9C	099D	9D
099E	9E	099F	9F	09A0	A0	09A1	A1	09A2	A2	09A3	A3	09A4	A4	09A5	A5	09A6	A6
09A7	A7	09A8	A8	09A9	A9	09AA	AA	09AB	AB	09AC	AC	09AD	AD	09AE	AE	09AF	AF
09B0	B0	09B1	B1	09B2	B2	09B3	B3	09B4	B4	09B5	B5	09B6	B6	09B7	B7	09B8	B8
09B9	B9	09BA	BA	09BB	BB	09BC	BC	09BD	BD	09BE	BE	09BF	BF	09C0	C0	09C1	C1
09C2	C2	09C3	C3	09C4	C4	09C5	C5	09C6	C6	09C7	C7	09C8	C8	09C9	C9	09CA	CA
09CB	CB	09CC	CC	09CD	CD	09CE	CE	09CF	CF	09D0	D0	09D1	D1	09D2	D2	09D3	D3
09D4	D4	09D5	D5	09D6	D6	09D7	D7	09D8	D8	09D9	D9	09DA	DA	09DB	DB	09DC	DC
09DD	DD	09DE	DE	09DF	DF	09E0	E0	09E1	E1	09E2	E2	09E3	E3	09E4	E4	09E5	E5
09E6	E6	09E7	E7	09E8	E8	09E9	E9	09EA	EA	09EB	EB	09EC	EC	09ED	ED	09EE	EE
09EF	EF	09F0	F0	09F1	F1	09F2	F2	09F3	F3	09F4	F4	09F5	F5	09F6	F6	09F7	F7
09F8	F8	09F9	F9	09FA	FA	09FB	FB	09FC	FC	09FD	FD	09FE	FE	09FF	FF	0A00	00
0A01	00	0A02	00	0A03	00	0A04	00	0A05	00	0A06	00	0A07	00	0A08	00	0A09	00

Τα αποτελέσματα είναι αναμενόμενα αφού το πλήθος των 1 είναι 1024 = 400H (αποθηκευμένο στο ζεύγος καταχβρητών BC) και το πλήθος των αριθμών στο [10H, 60H] είναι 60 – 50 + 1= 51H (αποθηκευμένο στον καταχωρητή D).

*Για την υλοποίηση της άσκησης βλ. ask1.8085

2η ΑΣΚΗΣΗ**

Στην άσκηση αυτή καλούμαστε να υλοποιήσουμε ένα σύστημα που διαχειρίζεται τα φώτα ενός χώρου.

Τα φώτα πρέπει να ανάβουν όταν ο διακόπτης (MSB των dip switches) ανοιγοκλείσει και να μένουν αναμμένα για 20 sec, εκτός αν επαναληφθεί το άνοιγμα και κλείσιμο του διακόπτη όπου ανανεώνεται ο χρόνος για ακόμα 20 sec.

Για την καθυστέρηση αποθηκεύουμε στον καταχωριτή D μία μεταβλητή που αρχικοποιείται στην τιμή C8H = 200 και μειώνετε κάθε 0,1 sec μέσω τεχνητής καθυστέρησης.

Το πρόγραμμα που υλοποιήσαμε λοιπόν έχει την εξής λειτουργία:

- Εάν ο διακόπτης είναι κλειστός περιμένει να ανοίξει (αν όχι εκτελεί το βήμα αυτό μόλις κλείσει)
- Όταν ανοίξει περιμένει να κλείσει για 2η φορά.
- Όταν κλείσει για 2η φορά ανάβει τα φώτα και ξεκινά αντίστροφη μέτρηση ελέγχοντας παράλληλα αν ο διακόπτης παραμένει κλειστός
- Αν ο διακόπτης ανοίξει όσο τα φώτα είναι αναμμένα συνεχίζει την αντίστροφη μέτρηση περιμένοντας ο διακόπτης να κλείσει.
- Αν ο διακόπτης κλείσει επαναφέρει το χρονόμετρο στα 20 sec και επαναλαμβάνει τη διαδικασία από το βήμα 3 και μετά.
- Όταν το χρονόμετρο τελειώσει, σβήνει τα φώτα ,επαναφέρει το χρονόμετρο και εκτελεί το πρόγραμμα από την αρχή.

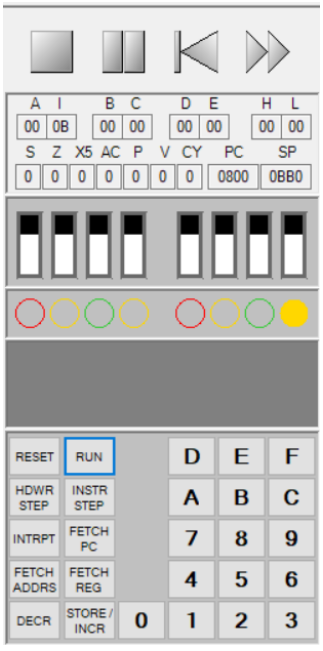
**Για την υλοποίηση της άσκησης βλ. ask.8085

3η ΑΣΚΗΣΗ***

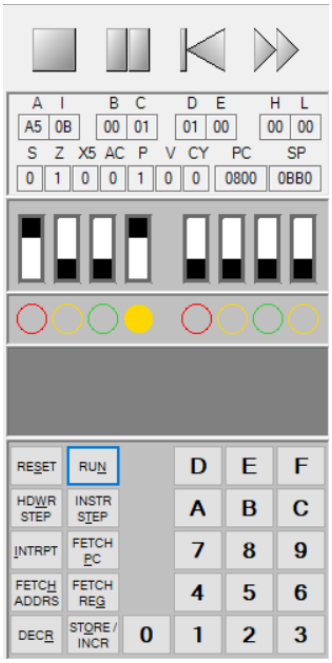
Στην άσκηση δημιουργούμε 3 διαφορετικά προγράμματα για τα οποία στην αναφορά θα παρουσιάσουμε αποκλειστικά μερικά ενδεικτικά παραδείγματα με στόχο την παρουσίαση της ορθότητας τους.

i. Το πρόγραμμα διαβάζει την είσοδο (μέσω των διακοπτών) και ανάλογα με το δεξιότερο σηκωμένο διακόπτη ανάβει και το αντίστοιχο LED.

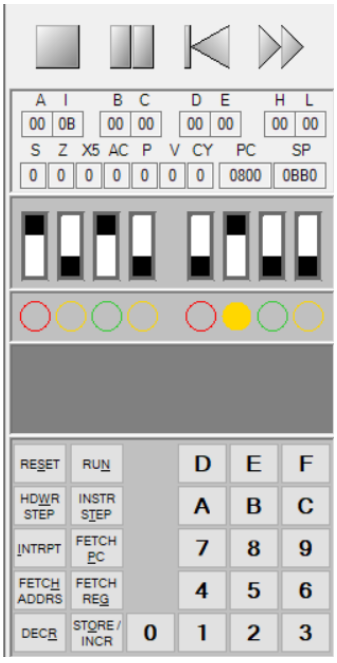
Επομένως, τρία ενδεικτικά παραδείγματα λειτουργίας είναι τα παρακάτω.



Είσοδος "11111111"



Είσοδος "10010000"

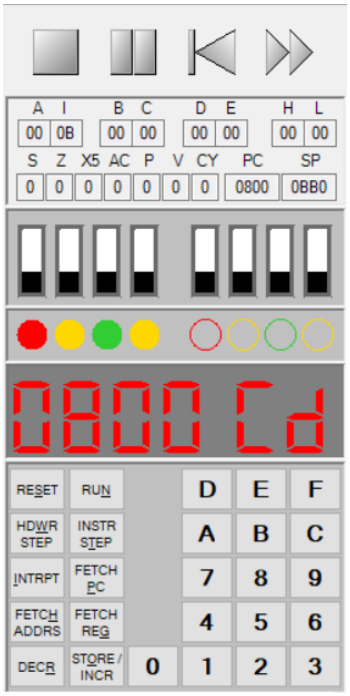


Είσοδος "10100100"

**Για την υλοποίηση των ερωτημάτων i ii iii βλ. αντίστοιχα στα ask3i.8085 ask3ii.8085 ask3iii.8085

ii. Το δεύτερο πρόγραμμα με βάσει τον αριθμό που πατάμε στο πληκτρολόγιο ανάβει το συγκεκριμένο LED και όλα τα επόμενα. Επομένως, παραθέτουμε τρία ενδεικτικά παραδείγματα .

Πάτημα Πλήκτρου "1"

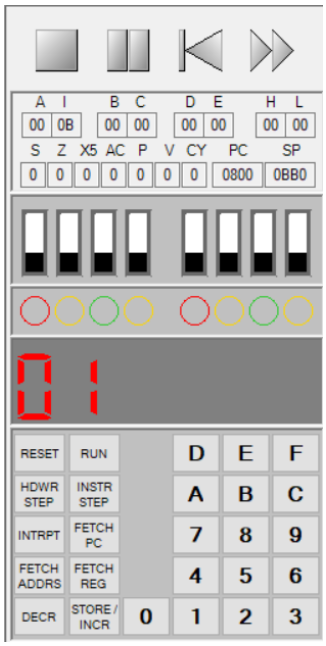


Πάτημα Πλήκτρου "5"

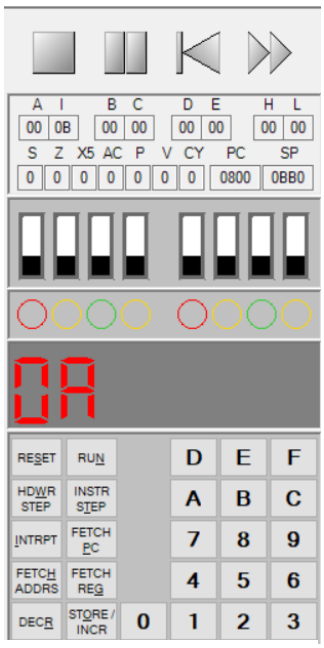
Πάτημα Πλήκτρου "8"

iii. Τέλος, στο τελευταίο πρόγραμμα πατάμε ένα πλήκτρο και εμφανίζεται στα δύο αριστερότερα 7 Segment Display.

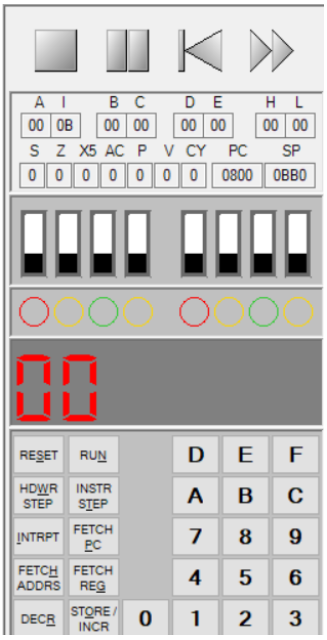
Οπότε έχουμε τα παρακάτω παραδείγματα.



Πάτημα Πλήκτρου "1"

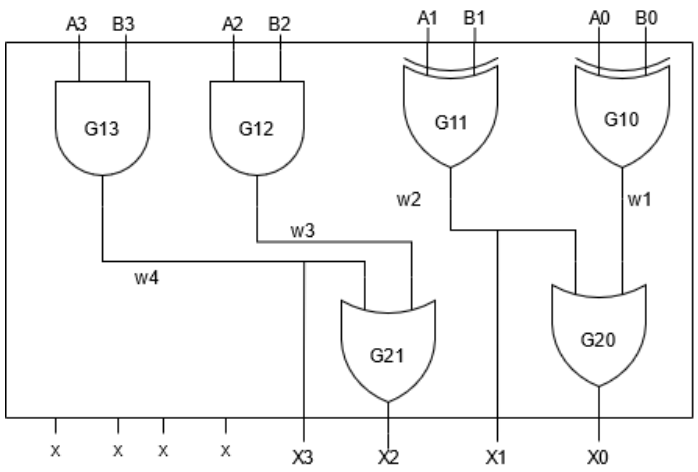


Πάτημα Πλήκτρου "A"



Πάτημα Πλήκτρου "0"

4η ΑΣΚΗΣΗ



Πρώτα ορίζουμε τις ενδιάμεσες μεταβλητές όπως φαίνονται και στο παραπάνω σχήμα. Οπότε έχουμε τα παρακάτω αποτελέσματα.

$$w_0 = A_0 \oplus B_0$$

$$w_1 = A_1 \oplus B_1$$

$$w_2 = A_2 + B_2$$

$$w_3 = A_3 + B_3$$

Και άρα τα Bit εξόδου ισούνται με

$$X_0 = w_0 \oplus w_1$$

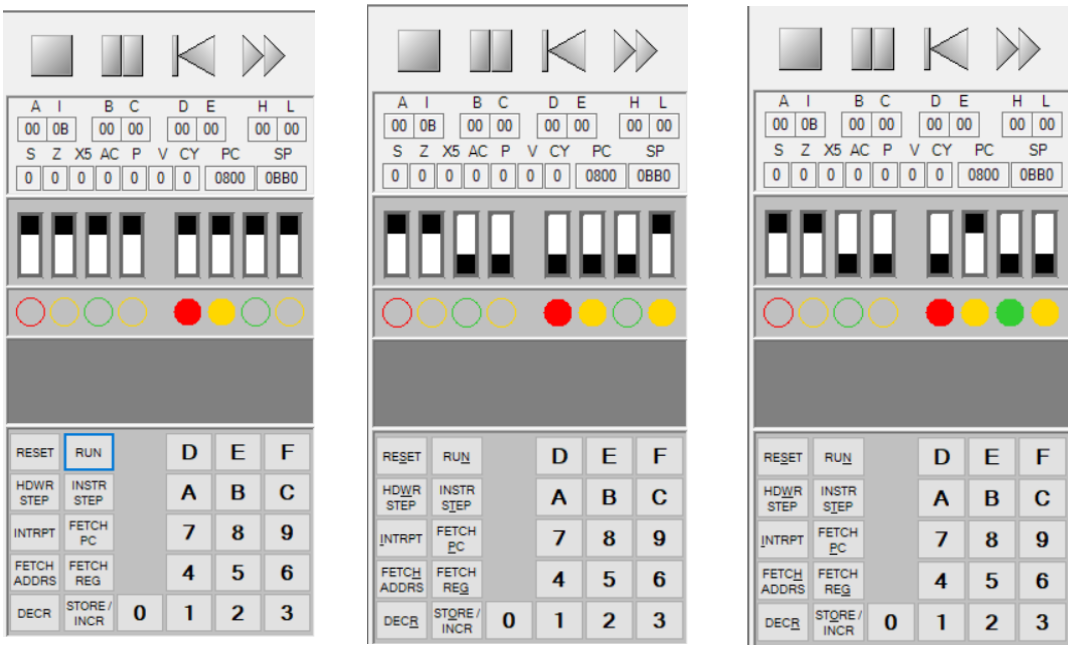
$$X_1 = w_2$$

$$X_2 = w_3 + w_4$$

$$X_3 = w_4$$

$$X_4 = X_5 = X_6 = X_7 = 0$$

Ενδεικτικά τώρα, θα παρουσιάσουμε κάποια ενδεικτικά παραδείγματα και το αποτέλεσμα το παραπάνω κώδικα ώστε να επιβεβαιώσουμε την ορθότητα του προγράμματος.



Παρατηρούμε ότι με βάσει την ανάλυση που προηγήθηκε τα αποτελέσματα συμβαδίζουν .

5η ΑΣΚΗΣΗ

Στην τρέχουσα άσκηση θα παρουσιαστεί η εσωτερική οργάνωση μιας μνήμης SRAM 256x4 bit και θα εξηγηθεί μέσω δύο παραδειγμάτων ο τρόπος που πραγματοποιείται η ανάγνωση και η εγγραφή στην συγκεκριμένη μνήμη.

Αρχικά, κάθε διεύθυνση αποτελείται από $\mathbf{\log_2 256 = 8\ bit = 1\ Byte}$ και το μήκος κάθε λέξης είναι ***4 Bit*** . Επιπρόσθετα , επιλέγουμε **4 bit** για την επιλογή της γραμμής $(A_0 - A_3)$ και της στήλης $(A_4 - A_7)$.

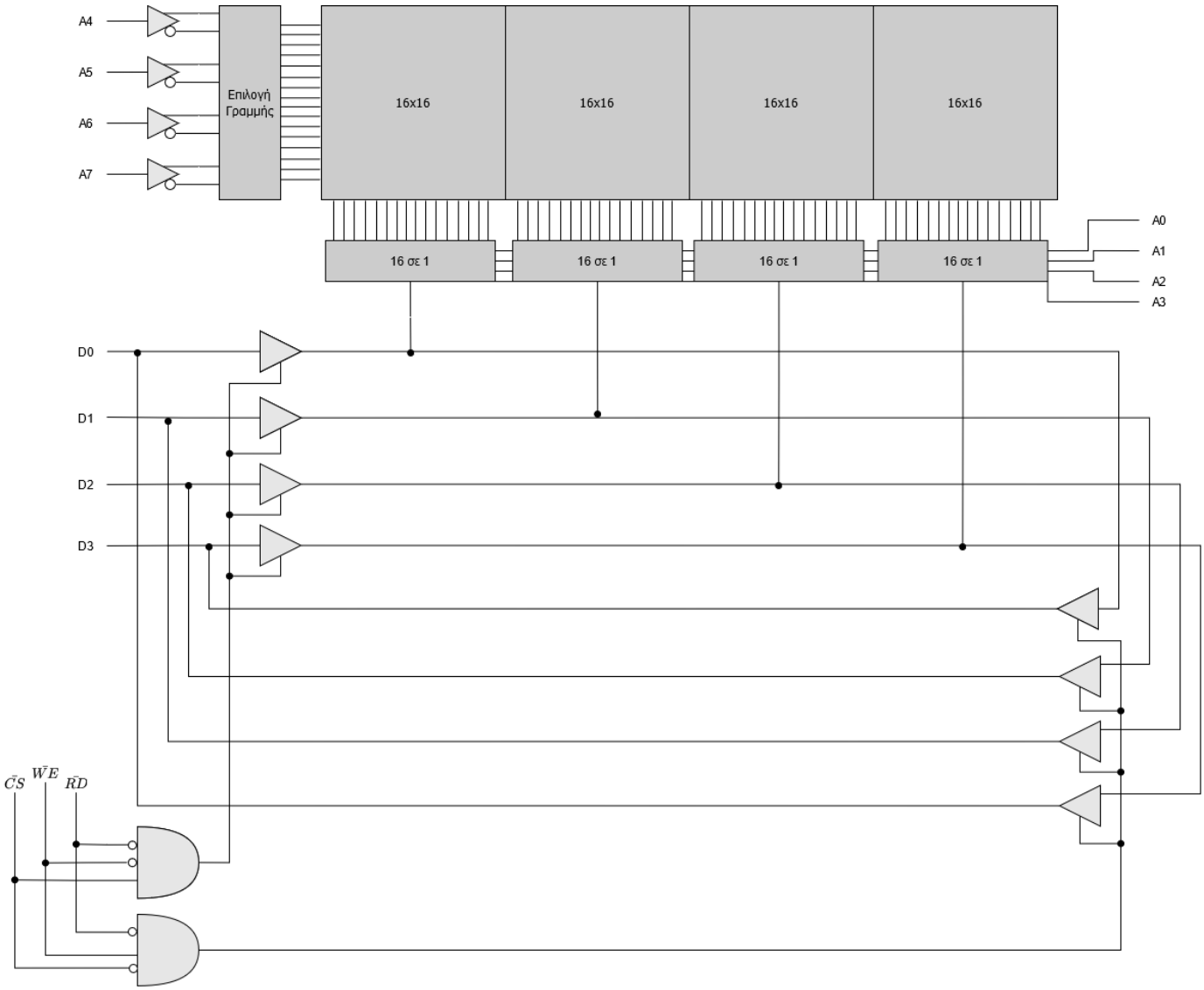
Επομένως, η μνήμη θα περιέχει 256 θέσεις για τα δεδομένα και θα την χωρίσουμε σε 4 υπό κομμάτια μεγέθους **1*256 bit (16X16)** το καθένα (τετράγωνα) .

Για την επιλογή της γραμμής θα γίνει χρήση ενός **αποκωδικοποιητή 4-16 (είσοδος $\boldsymbol{A_0 - A_3}$)** ενώ για την επιλογή στήλης θα τοποθετήσουμε **4 πολυπλέκτες-αποπλέκτες (είσοδος $\boldsymbol{A_4 - A_7}$)** έναν κάτω από κάθε υπό κομμάτι της μνήμης .

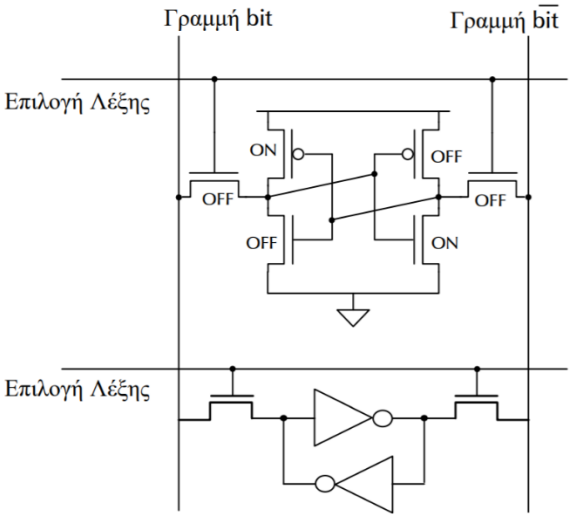
Για να ενεργοποιούμε το τσιπάκι μας έχουμε το σήμα $Cheap\ Select$ το οποίο ενεργοποιείται όταν λάβει το λογικό μηδέν (ανάστροφη λογική) καθώς και το σήμα $Write\ Enable$ και RD (ομοίως σε ανάστροφη λογική) .Ειδικότερα , σε περίπτωση ανάγνωσης του περιεχομένου μια διεύθυνσης της μνήμης στο σήμα RD' στέλνεται το λογικό μηδέν και στο WE' το λογικό ένα οπότε ενεργοποιείται η δεύτερη πύλη AND που με τη σειρά της ενεργοποιεί τους κατάλληλους απομονωτές ώστε να πραγματοποιηθεί η ανάγνωση. Στην περίπτωση που θέλουμε να γράψουμε μια λέξη στη μνήμη τότε αποστέλλονται το λογικό μηδέν στο WE' και το λογικό ένα στο σήμα RD' οπότε τώρα ενεργοποιείται η πρώτη AND.

Η μνήμης μας θα έχει **4 Bit** εισόδου/εξόδου τα $D_0 - D_3$. Σε περίπτωση εγγραφής στα 4 αυτά bit εισάγεται η τιμή που θέλουμε να αποθηκεύσουμε στη μνήμη ενώ σε περίπτωση ανάγνωσης τα bit αυτά μας δίνουν το περιεχόμενο της διεύθυνσης που προσπελάσαμε. Τα παραπάνω σήματα δεν συνδέονται άμεσα με τους 4 πολυπλέκτες - αποπλέκτες αλλά παρεμβάλλονται ανάμεσα τους απομονωτές (Voltage Buffer) για λόγους προστασίας/απομόνωσης της πληροφορίας και οι οποίοι δέχονται σήμα Enable ώστε ανάλογα με την λειτουργία να ενεργοποιούνται οι 4 από τους 8.

Τώρα θα παρουσιαστεί η οργάνωση της μνήμης μας .


A4
A5
A6
A7
Επιλογή Γραμμής
16x16
16x16
16x16
16x16
16 σε 1
16 σε 1
16 σε 1
16 σε 1
A0
A1
A2
A3
D0
D1
D2
D3
$\bar{C}S$
$\bar{W}E$
$\bar{R}D$

Για λόγους πληρότητας παρουσιάζουμε και το κύτταρο μνήμης SRAM το οποίο αποτελείται στην αρχή και στο τέλους του από δύο MOS τρανζίστορ τα οποία έχουν το ρόλο του διακόπτη και ο πυρήνας του κυττάρου είναι 2 αντιστροφείς οι οποίοι μανδαλώνουν -φυλακίζουν την πληροφορία 0 ή 1.



Οπότε, τώρα με δύο παραδείγματα θα εξηγηθεί αναλυτικά κάθε βήμα που λαμβάνει χώρα κατά τη λειτουργία της Μνήμης.

Αρχικά υποθέτουμε ότι με βάσει τη διεύθυνση που μας δίνει ο επεξεργαστής ενεργοποιείται το Cheap Select το όποιο είναι σε ανάστροφη μορφή και άρα γίνεται ίσο με το λογικό μηδέν ώστε να ενεργοποιηθεί το ολοκληρωμένο.

Παράδειγμα 1: Ανάγνωση το περιεχόμενο στη θέση μνήμης «01010101»

Αρχικά, θεωρούμε ότι το $\overline{WE}$ έχει τιμή 1 και το $\overline{RD}$ τιμή 0 ,τα οποία είναι σε ανάστροφη μορφή, και άρα ενεργοποιούνται οι 4 (δεξιοί) Buffer μέσω την δεύτερης πύλης AND . Με βάσει τα πρώτα 4 Bit της διεύθυνσης επιλέγεται η γραμμή $0101 \rightarrow 5_{10}$ μέσω του αποκωδικοποιητή ενώ με τα 4 τελευταία Bit $0101 \rightarrow 5_{10}$ επιλέγεται η σειρά που βρίσκεται η ζητούμενη προς ανάγνωση λέξη (ενεργοποιούνται τέσσερα διαφορετικά κύτταρα μνήμης ,ένα από κάθε πολυπλέκτη) ,έτσι οι πολυπλέκτες μεταφέρουν στην έξοδο τους το περιεχόμενο της όγδοης εισόδους τους η οποία και ενεργοποιήθηκε από τα Bit της διεύθυνσης εισόδου. Οπότε, τώρα στην έξοδο των πολυπλεκτών – αποπλεκτών είναι τα ζητούμενα 4 Bit $D_0 - D_3$ τα οποία είναι οι είσοδοι των Buffer – απομονωτών οι οποίοι οδηγούν το περιεχόμενο της διεύθυνσης στην έξοδο της μνήμης ,δηλαδή στα Bit $D_0 - D_3$.

Παράδειγμα 2: Εγγραφή της 4b λέξης «1010» στη θέση μνήμη «10101010»

Αρχικά, θεωρούμε ότι το $\overline{WE}$ έχει τιμή 0 και το $\overline{RD}$ τιμή 1 ,τα οποία είναι σε ανάστροφη μορφή, και άρα ενεργοποιούνται οι 4 (αριστεροί) Buffer μέσω την πρώτης πύλης AND .Επιπρόσθετα, στις εισόδους $D_0 - D_3$ βρίσκεται η λέξη που επιθυμούμε να γράψουμε ,την $1010 \rightarrow 10_{10}$ στην περίπτωση μας. Οπότε μέσω των απομονωτών τα 4 Bit της λέξης μεταφέρονται ένα σε κάθε είσοδο των αποπλεκτών. Με βάσει την διεύθυνση έχουμε ξανά τον υπολογισμό της γραμμής με τα πρώτα τέσσερα Bit και της σειράς με τα τέσσερα τελευταία Bit της διεύθυνσης (άρα τα 4 κύτταρα που θα γραφεί η λέξη).

Οπότε τώρα οι αποπλέκτες μεταφέρουν την είσοδο τους στην θέση που τους φανερώνει η διεύθυνση και αντικαθιστούν το περιεχόμενο των τεσσάρων κελιών με τις νέες τιμές που στο παράδειγμα μας είναι οι $1 - 0 - 1 - 0$.

6η ΑΣΚΗΣΗ

Στην άσκηση καλούμαστε να σχεδιάσουμε ένα σύστημα μνήμης που περιλαμβάνει $8KB\ ROM$ & $4KB\ RAM$.

Πιο συγκεκριμένα, για την υλοποίηση της ROM έχουμε στη διάθεση μας 3 ολοκληρωμένα κυκλώματα, $2 * (2KB)$ και $1 * (4KB)$ με τις χωρητικότητες να αναγράφονται στις παρενθέσεις.
Ενώ για την RAM μας δίνονται 2 ολοκληρωμένα κυκλώματα, τα $2 * (2KB)$.

Οι διευθύνσεις της μνήμης αρχίζουν από την διεύθυνση $000H$ και θεωρούμε ότι δεν έχουμε κενά διαστήματα μεταξύ των διευθύνσεων της ROM και της RAM.

Σε πρώτη φάση, σχεδιάζουμε το εύρος των διευθύνσεων που θα εξυπηρετούνται από κάθε ολοκληρωμένο ώστε να μπορέσουμε να σχεδιάσουμε στη συνέχεια τις συνδέσεις που απαιτούνται ώστε για κάθε διεύθυνση που ζητά να προσπελάσει ο επεξεργαστής να ενεργοποιείται το κατάλληλο τμήμα της μνήμης.

Επομένως, δημιουργείται ο παρακάτω πίνακας.

Απόλυτη Διεύθυνση	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	**Μνήμη**
0000H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ROM1 (2K)
07FFH	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	
0800H	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	ROM2 (2K)
0FFFH	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	
1000H	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	ROM3 (4K)
1FFFH	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	
2000H	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	SRAM1 (2K)
27FFH	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	
2800H	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	SRAM2 (2K)
2FFFH	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	

Για να μπορέσουμε να προσδιορίσουμε τις εισόδους των θυρών Cheap Select για κάθε στοιχείο μνήμης θα παρατηρήσουμε από τον παραπάνω πίνακα τα Bits βάσει των οποίων είναι εφικτή η αναγνώριση κάθε φορά για το ποιο στοιχείο μνήμης πρέπει να επισκεφτούμε με βάσει την διεύθυνση. Επομένως, θα επικεντρωθούμε στα bit A_{13} , A_{12} , A_{11}.

(α) Στο πρώτο ερώτημα μας παρέχεται ένας αποκωδικοποιητής 3 : 8 και έστω Y_i' η έξοδος του.

Οπότε παρατηρούμε τα παρακάτω,

1) **ROM 1** : $A_{13} = A_{12} = A_{11} = 0$ οπότε $\overline{CS_1} = \overline{Y}_0$.

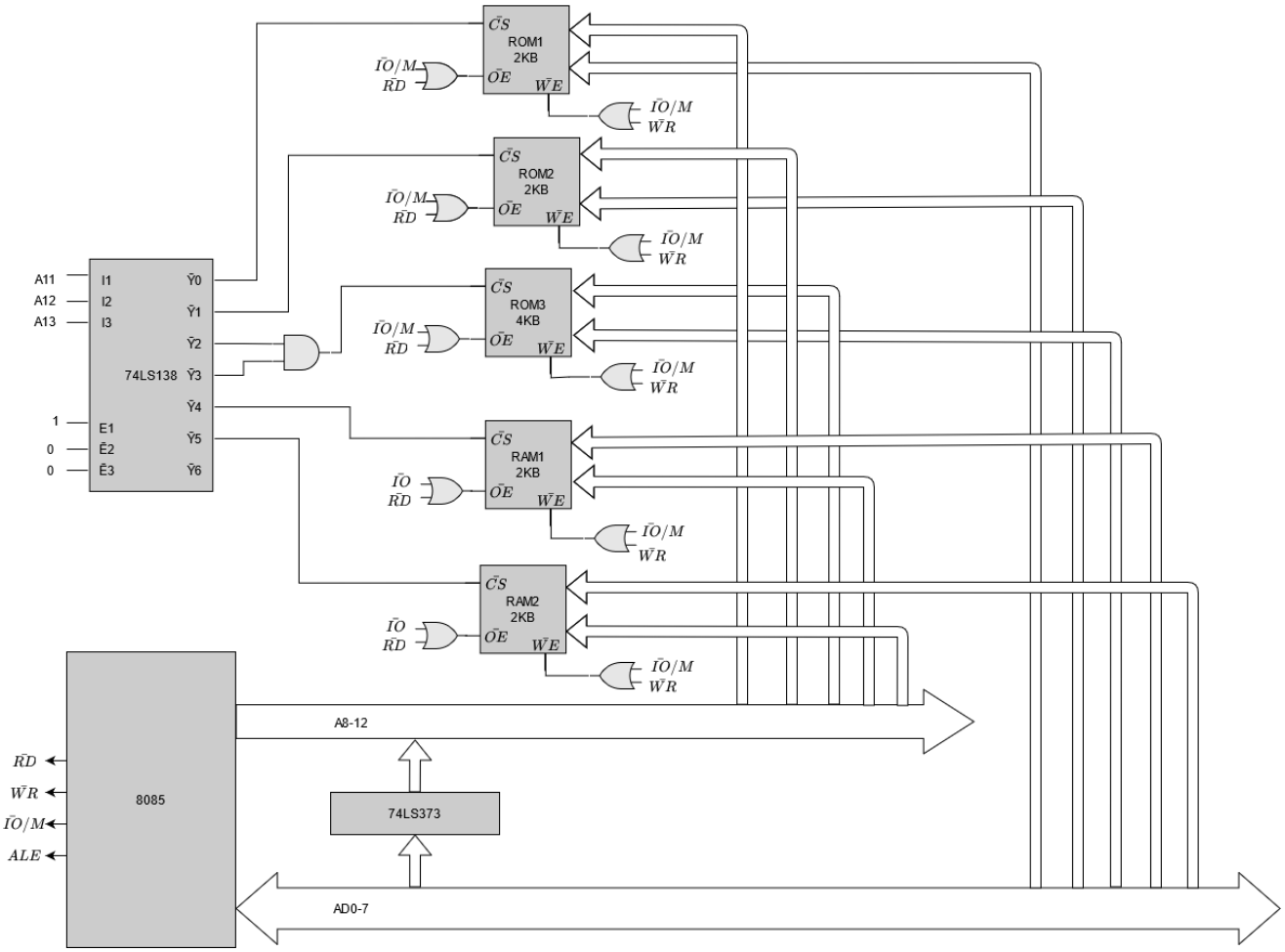
2) **ROM 2** : $A_{13} = A_{12} = 0,\ A_{11} = 1$ οπότε $\overline{CS_2} = \overline{Y}_1$.

3) **ROM 3** : $A_{13} = A_{11} = 0$, $A_{12} = 1$ ή $A_{12} = A_{11} = 1$, $A_{13} = 0$
οπότε $\overline{CS_3} = \overline{Y}_3 * \overline{Y}_2$.

4) **RAM 1** : $A_{11} = A_{12} = 0,\ A_{13} = 1$ οπότε $\overline{CS_1} = \overline{Y}_4$.

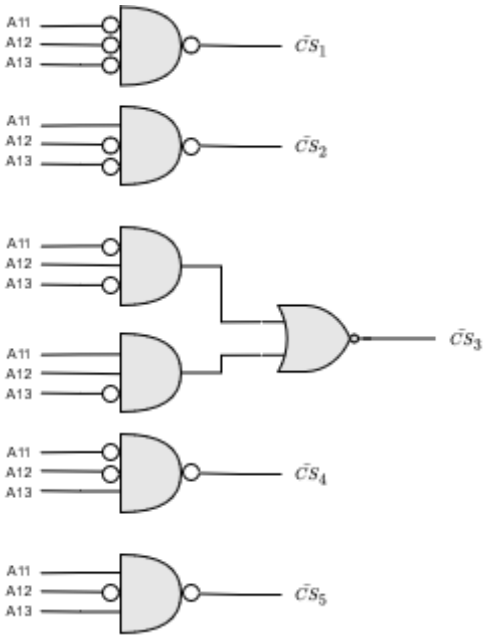
5) **RAM 2** : $A_{11} = A_{13} = 1,\ A_{12} = 0$ οπότε $\overline{CS_2} = \overline{Y}_5$.

Με βάσει όλα τα παραπάνω στο παρακάτω σχήμα παρουσιάζουμε το λογικό διάγραμμα της μνήμης συμπεριλαμβανομένου και των σημάτων του system bus.

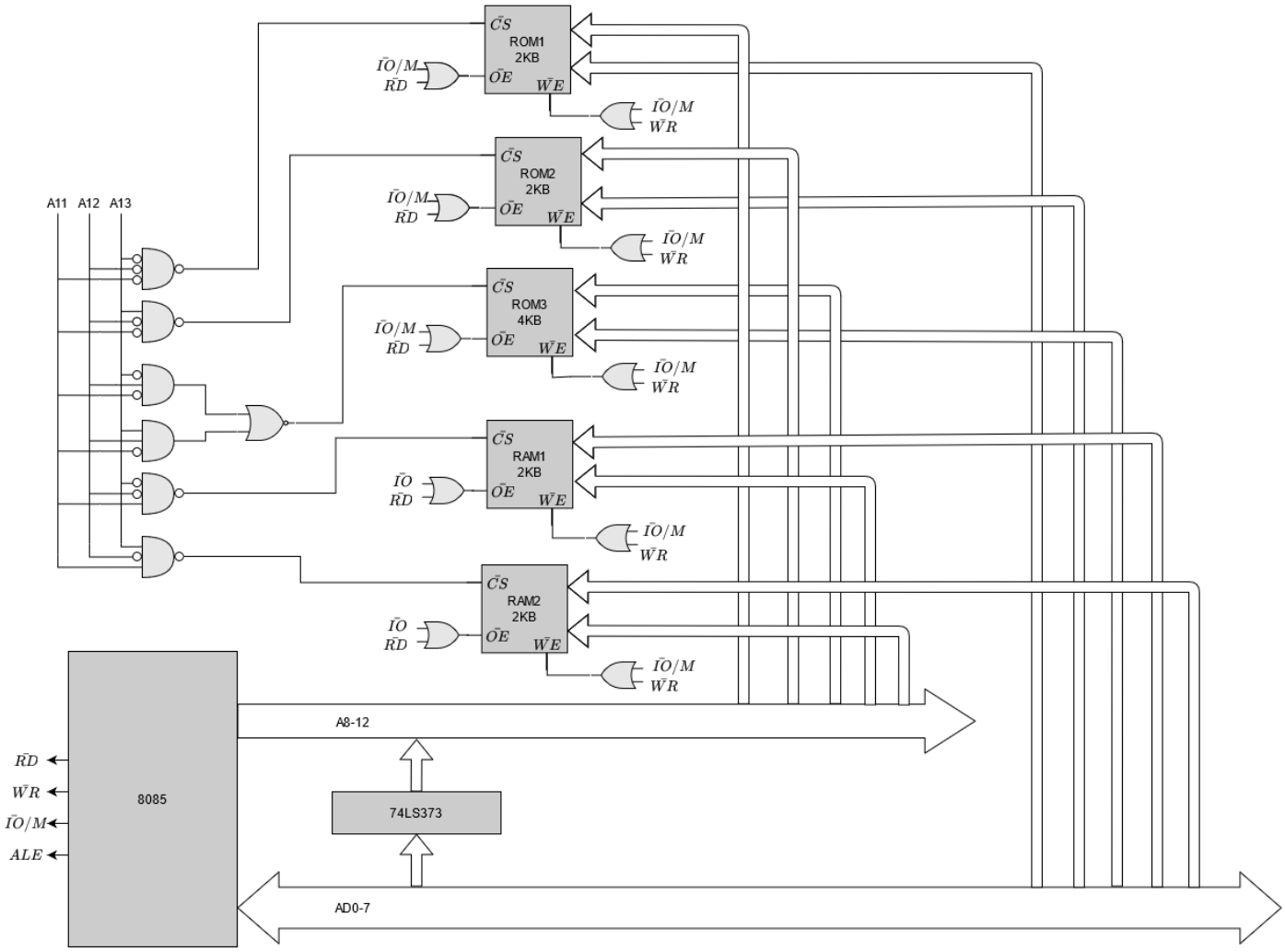


(β) Τώρα θα επαναλάβουμε το ίδιο πρόβλημα με μόνη διαφορά την αντικατάσταση του αποκωδικοποιητή με οποιοδήποτε συνδυασμό λογικών πυλών επιθυμούμε.

Με βάσει την προηγούμενη ανάλυση για το τι σήματα πρέπει να οδηγούνται στην θύρα Cheap Select καθενός ολοκληρωμένου προκύπτουν οι παρακάτω 4 πύλες που θα μας δώσουν ένα ισοδύναμο αποτέλεσμα.



Τέλος, παρουσιάζουμε το νέο χάρτη μνήμης σύμφωνα με την νέα υλοποίηση μας.



7η ΑΣΚΗΣΗ

Όμοια με το προηγούμενο ερώτημα έχουμε για τις εισόδους $\overline{CS}$ των μνημών:

Απόλυτη Διεύθυνση	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	**Μνήμη**
0000H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ROM
0000H	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	
1000H	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	ROM
1000H	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	
2000H	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	ROM
2FFFH	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	
3000H	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	RAM1 (4K)
3FFFH	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	
4000H	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RAM2 (4K)
4FFFH	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	
5000H	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	RAM3 (4K)
5FFFH	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	
6000H	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	ROM
6FFFH	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	

1) **ROM** : $A_{12} = A_{13} = A_{14} = 0$ **ή** $A_{12} = A_{14} = 0$, $A_{13} = 1$ **ή** $A_{12} = 0, A_{13} = A_{14} = 1$ οπότε $\overline{CS} = \overline{Y}_0 * \overline{Y}_1 * \overline{Y}_2 * \overline{Y}_6$.

2) **RAM 1** : $A_{13} = A_{12} = 1$, $A_{14} = 0$ οπότε $\overline{CS} = \overline{Y}_3$.

3) **RAM 2** : $A_{13} = A_{12} = 0$, $A_{14} = 1$ οπότε $\overline{CS} = \overline{Y}_4$.

4) **RAM 3** : $A_{12} = A_{14} = 1$, $A_{13} = 0$ οπότε $\overline{CS} = \overline{Y}_5$.

Για την πύλη εισόδου, επειδή η τιμή 70H αντιστοιχεί στην τιμή 0111000001110000 ενεργοποιεί την έξοδο $\bar{Y}_7$, άρα δεν μπορούμε να τη χρησιμοποιήσουμε για την θύρα εξόδου (7000H). Έτσι χρησιμοποιούμε την $\bar{Y}_7$ για την θύρα εισόδου, ενώ για την θύρα εξόδου χρησιμποιούμε μία πύλη and 16 εισόδων, στην οποία δίνουμε αντεστραμένες τις A_{12} A_{13} A_{14} και κανονικά όλες τις υπόλοιπες πύλες.

Έτσι έχουμε το λογικό διάγραμμα: