|  |
| --- |
|  |
| Αρχιτεκτονική Υπολογιστών |
|  |
|  |
|  |
| **31/1/2018** |

|  |
| --- |
|  |

Άσκηση 1

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μεγέθους 2Κ, 4Κ, 8Κ θέσεων με 8 bits ανά θέση. Χρησιμοποιώντας τον μικρότερο δυνατό αριθμό από τα παραπάνω ΟΚ, καθώς και αποκωδικοποιητές και πύλες, σχεδιάστε αρχιτεκτονική συστήματος μνήμης μεγέθους 30ΚBytes. Ο δίαυλος διευθύνσεων του συστήματος έχει εύρος 16bits. Ακολουθήστε τα εξής στάδια κατά τη σχεδίαση:

Α. Καταστρώστε πίνακα με τις διευθύνσεις μνήμης που καταλαμβάνει το κάθε ΟΚ.

Β. Καταστρώστε πίνακα με τις τιμές των γραμμών των διευθύνσεων που χρησιμοποιούνται για τη δημιουργία των σημάτων επιλογής ΟΚ.

Γ. Σχεδιάστε το κύκλωμα της ζητούμενης αρχιτεκτονικής.

Άσκηση 2

Να γραφεί ένα πρόγραμμα σε γλώσσα του μικροεπεξεργαστή 8085 που να υπολογίζει το ακέραιο μέρος της τετραγωνικής ρίζα ενός ακεραίου αριθμού αποθηκευμένου στη θέση μνήμης (2030)16 και να το αποθηκεύει στη θέση μνήμης (2031)16 , με χρήση του αλγορίθμου των επιτυχών αφαιρέσεων των περιττών αριθμών αρχίζοντας από το 1 μέχρι να βγει το αποτέλεσμα 0(ή αρνητικός αριθμός).

Παράδειγμα 1: Δίνεται ο αριθμός 16   
16-1=15 1η  
15-3=12 2η

12-5=7 3η

7-7=0 4η

Σύνολο επιτυχών αφαιρέσεων 4 άρα τετραγωνική ρίζα του 16=4

Παράδειγμα 2: Δίνεται ο αριθμός 18

18-1=17 1η

17-3=14 2η

14-5=9 3η

9-7=2 4η

2-9=-7 5η

Σύνολο επιτυχών αφαιρέσεων 4 άρα τετραγωνική ρίζα του 18 είναι περίπου το 4.

**Άσκηση 1**

Παρουσιάζουμε δύο περιπτώσεις καθώς στην πρώτη ενώ χρησιμοποιούμε λιγότερα Ολοκληρωμένα Κυκλώματα (ΟΚ) δημιουργούμε μνήμη μεγέθους 32Kbytes οπότε έχουμε σπατάλη διευθύνσεων ενώ στη δεύτερη έχουμε περισσότερα ΟΚ αλλά ακριβώς 30 Kbytes μνήμη.

**1η περίπτωση**

Αρχικά καταγράφουμε τη διευθυνσιοδότηση της μνήμης:

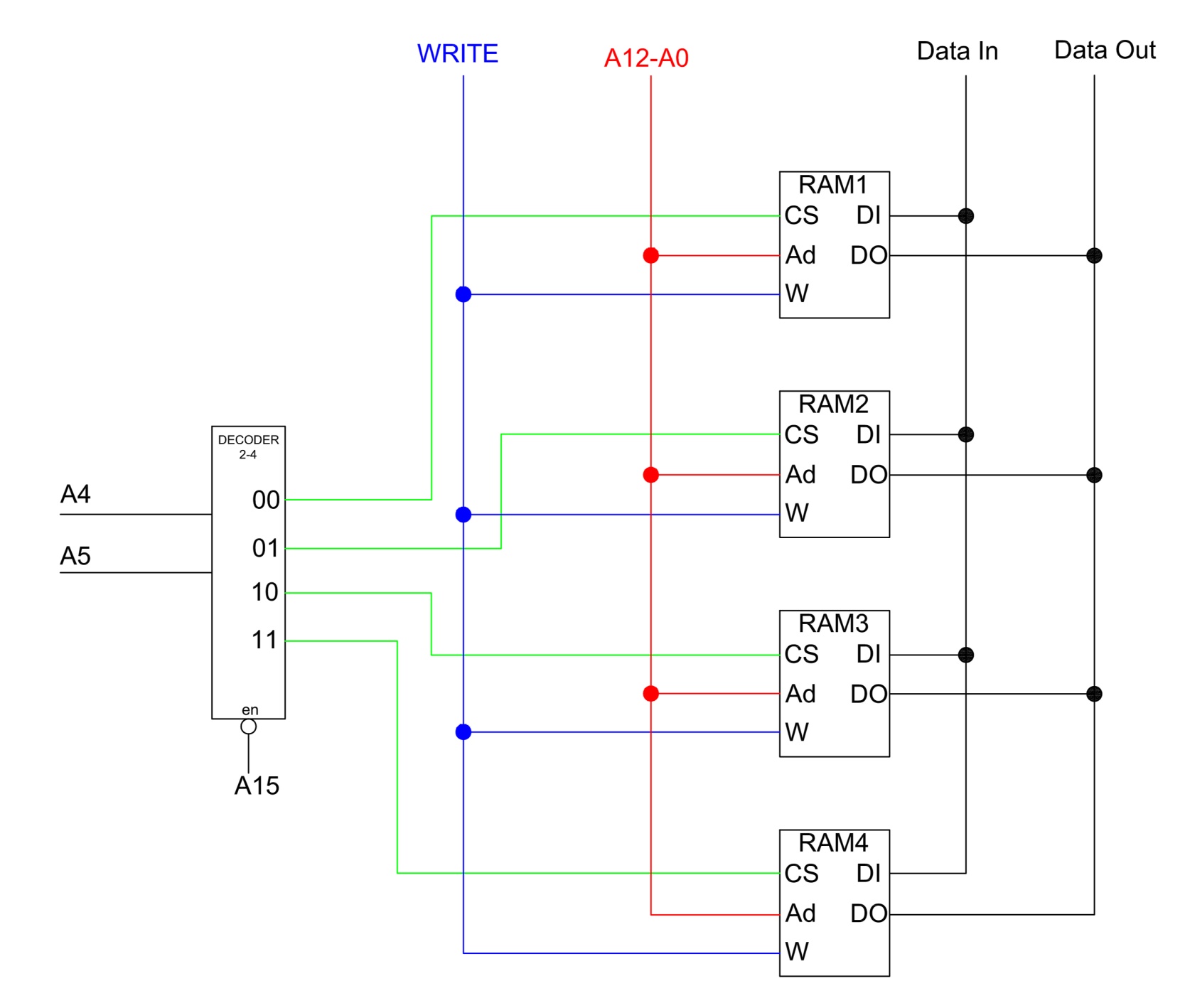
|  |  |  |  |
| --- | --- | --- | --- |
| Μνήμη | Μέγεθος | Πεδίο διεύθυνσης (HEX) | Διεύθυνση (BIN) |
| RAM1 | 8kByte | 0000-1FFF | 0000 0000 0000 0000 |
| 0001 1111 1111 1111 |
| RAM2 | 8kByte | 2000-3FFF | 0010 0000 0000 0000 |
| 0011 1111 1111 1111 |
| RAM3 | 8kByte | 4000-5FFF | 0100 0000 0000 0000 |
| 0101 1111 1111 1111 |
| RAM4 | 8kByte | 6000-7FFF | 0110 0000 0000 0000 |
| 0111 1111 1111 1111 |

Παρατηρούμε ότι:

1. Οι μνήμες έχουν 13 bit διεύθυνσης .
2. Τα 12 λιγότερο σημαντικά bit (Α12 - Α0) παίρνουν όλες τις δυνατές τιμές στο πεδίο διεύθυνσης του κάθε ολοκληρωμένου κυκλώματος. Άρα καθορίζουν τη θέση μνήμης στην οποία αναφέρεται η κάθε διεύθυνση.
3. Το A15 είναι 0 σε όλα τα ολοκληρωμένα κυκλώματα.
4. Από τις παραπάνω παρατηρήσεις συμπεραίνουμε ότι ο συνδυασμός των bit A14 –A13 θα μας βοηθήσει να επιλέξουμε ποια από τις μνήμες ενεργοποιείται κάθε φορά.

|  |  |  |
| --- | --- | --- |
| A14 | A13 | CS |
| 0 | 0 | RAM1 |
|
| 0 | 1 | RAM2 |
|
| 1 | 0 | RAM3 |
|
| 1 | 1 | RAM4 |
|

Επομένως για 2 σήματα εισόδου παίρνουμε 4 σήματα εξόδου, κάτι που επιτυγχάνεται με έναν 2-4 αποκωδικοποιητή. Εφόσον το A15 είναι 0 σε όλες τις δυνατές διευθύνσεις, ο αποκωδικοποιητής θα έχει ως σήμα επίτρεψης (enable) το A’15.



**2η περίπτωση**

Αρχικά καταγράφουμε τη διευθυνσιοδότηση της μνήμης:

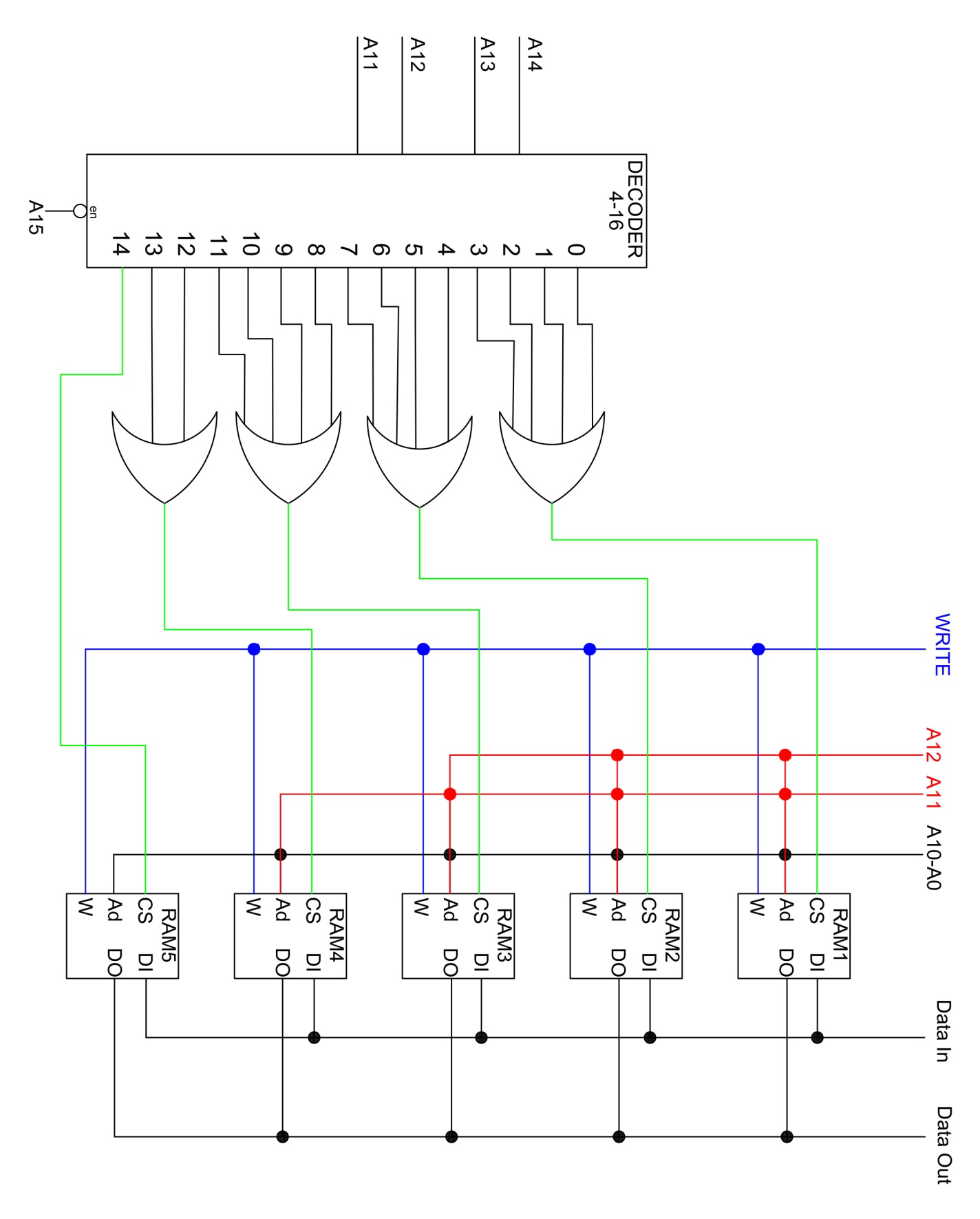
|  |  |  |  |
| --- | --- | --- | --- |
| Μνήμη | Μέγεθος | Πεδίο διεύθυνσης (HEX) | Διεύθυνση (BIN) |
| RAM1 | 8kByte | 0000-1FFF | 0000 0000 0000 0000 |
| 0001 1111 1111 1111 |
| RAM2 | 8kByte | 2000-3FFF | 0010 0000 0000 0000 |
| 0011 1111 1111 1111 |
| RAM3 | 8kByte | 4000-5FFF | 0100 0000 0000 0000 |
| 0101 1111 1111 1111 |
| RAM4 | 4kByte | 6000-6FFF | 0110 0000 0000 0000 |
| 0110 1111 1111 1111 |
| RAM5 | 2kByte | 7000-77FF | 0111 0000 0000 0000 |
| 0111 0111 1111 1111 |

Παρατηρούμε ότι:

1. Οι RAM1,RAM2,RAM3 έχουν 13 bit διεύθυνσης .
2. H RAM4 έχει 12 bit διεύθυνσης.
3. Η RAM5 έχει 11 bit διεύθυνσης
4. Τα 10 λιγότερο σημαντικά bit (Α10 - Α0) παίρνουν όλες τις δυνατές τιμές στο πεδίο διεύθυνσης του κάθε ολοκληρωμένου κυκλώματος. Άρα καθορίζουν τη θέση μνήμης στην οποία αναφέρεται η κάθε διεύθυνση.
5. Το A15 είναι 0 σε όλα τα ολοκληρωμένα κυκλώματα.
6. Από τις παραπάνω παρατηρήσεις συμπεραίνουμε ότι ο συνδυασμός των bit A14 –A11 θα μας βοηθήσει να επιλέξουμε ποια από τις μνήμες ενεργοποιείται κάθε φορά.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A14 | A13 | A12 | A11 | CS |
| 0 | 0 | 0 | 0 | RAM1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | RAM2 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | RAM3 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | RAM4 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | RAM5 |
| 1 | 1 | 1 | 1 |

Επομένως για 4 σήματα εισόδου παίρνουμε 16 σήματα εξόδου(χρησιμοποιούμε τα 15), κάτι που επιτυγχάνεται με έναν 4-16 αποκωδικοποιητή. Εφόσον το A15 είναι 0 σε όλες τις δυνατές διευθύνσεις, ο αποκωδικοποιητής θα έχει ως σήμα επίτρεψης (enable) το A’15.



**Άσκηση 2**

LDA 2030H ;loads accumulator

MOV B,A ; stores the number from accumulator into the register B

MVI C , 0H ; index used to find the total number of subtractions which coresponds to the sqrt we want to find

MVI D ,1H ;the number we use for subtraction -1 -3 -5 etc

MVI E ,0H ; we store zero in order to use it as comparison to check if our value is positive

MVI H, 1H ; used in order to create the step +1 and +2

SKP: MOV A,B ; loads acc with our number

SUB D ; subtracts the step B-D (-1,-2,-3)

JM NEG ; if the number is negative loop stops and jumps to neg(tag)

MOV B,A ; if the final number from subtraction is pos. it stores it back on reg B

MOV A,C ; loads index

ADD H ; index +1

MOV A,C ; stores index back on reg C

MOV A,D ; loads the step

ADD H

ADD H ; step+2

MOV D,A ; stores back the step

JMP SKP ; repeats the loop

NEG: MOV A,C ; transfers the result to acc

STA 2031H ; stores the result in memory location 2031H

HLT ; stops the execution