ΕΛΛΗΝΙΚΟ ΑΝΟΙΚΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΘΕΜΑΤΙΚΗ ΕΝΟΤΗΤΑ: ΠΛΗ-21

ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΕΣ

ΑΣΚΗΣΕΙΣ
ΓΡΑΠΤΩΝ ΕΡΓΑΣΙΩΝ
&
ΘΕΜΑΤΩΝ ΕΞΕΤΑΣΕΩΝ

ΣΥΝΤΕΛΕΣΤΕΣ (ΚΑΤ' ΑΛΦΑΒΗΤΙΚΗ ΣΕΙΡΑ):

Γ. ΑΛΕΞΙΟΥ
Χ. ΒΕΡΓΟΣ
Κ. ΕΥΣΤΑΘΙΟΥ
Γ. ΘΕΟΔΩΡΙΔΗΣ
Χ. ΚΑΒΟΥΣΙΑΝΟΣ
Ο. ΚΟΥΦΟΠΑΥΛΟΥ
Κ. ΛΑΜΠΡΙΝΟΥΔΑΚΗΣ
Φ. ΛΙΟΤΟΠΟΥΛΟΣ
Α. ΜΟΣΧΟΒΟΣ
Δ. ΜΠΑΚΑΛΗΣ
Σ. ΝΙΚΟΛΑΪΔΗΣ
Δ. ΝΙΚΟΛΟΣ
Β. ΠΑΛΙΟΥΡΑΣ
Ι. ΠΑΠΑΕΥΣΤΑΘΙΟΥ
Δ. ΠΑΠΑΚΩΣΤΑΣ
Α. ΣΤΟΥΡΑΙΤΗΣ
Α. ΣΚΟΔΡΑΣ
Β. ΦΩΤΟΠΟΥΛΟΣ
Α. ΧΑΤΖΟΠΟΥΛΟΣ

ΕΠΙΜΕΛΕΙΑ ΕΚΔΟΣΗΣ: Φ. ΛΙΟΤΟΠΟΥΛΟΣ – Δ. ΜΠΑΚΑΛΗΣ – Χ. ΚΑΒΟΥΣΙΑΝΟΣ

ΠΑΤΡΑ 2008

Το παρόν υλικό αποτελεί το κύριο τμήμα των ασκήσεων που δόθηκαν προς επίλυση στους φοιτητές του Ελληνικού Ανοικτού Πανεπιστημίου στα πλαίσια της Θεματικής Ενότητας **ΠΛΗ-21: Ψηφιακά Συστήματα** του Προγράμματος Σπουδών της Πληροφορικής κατά τα ακαδημαϊκά έτη 2001 - 2008.

ΠΕΡΙΕΧΟΜΕΝΑ

Δίαυλοι

ΑΣΚΗΣΗ 1

Ένας επεξεργαστής διαθέτει 64 εσωτερικούς καταχωρητές με μήκος λέξης 8 bytes. Πόσες γραμμές πρέπει να έχει ο εσωτερικός δίαυλος διευθύνσεων για να μπορεί να επιλέγει έναν από αυτούς τους καταχωρητές;
Ο εξωτερικός δίαυλος διευθύνσεων έχει 48 γραμμές. Πόσα bits μνήμης μπορεί να υποστηρίξει αν κάθε διεύθυνση αναφέρεται: 1) σε ένα byte 2) σε μία λέξη των 4 bytes 3) σε μία λέξη των 8 bytes;

Λύση:
Αφού ο επεξεργαστής διαθέτει 64 εσωτερικούς καταχωρητές απαιτείται ο εσωτερικός δίαυλος διευθύνσεων να έχει τόσες γραμμές ώστε να υποστηρίζονται 64 διαφορετικές διευθύνσεις. Άρα κατ' ελάχιστον το πλήθος των γραμμών πρέπει να είναι 6 αφού $2^6 = 64$.
Ο εξωτερικός δίαυλος διευθύνσεων έχει 48 γραμμές. Άρα μπορεί να προσπελάσει 2^{48} διαφορετικές διευθύνσεις.

1) Στην πρώτη περίπτωση ο αριθμός διευθύνσεων αντιστοιχεί σε αριθμό bytes. Άρα μπορεί να υποστηρίξει $8 \times 2^{48} = 2^3 \times 2^{48} = 2^{51}$ bits.

2) Στη δεύτερη περίπτωση ο αριθμός διευθύνσεων αντιστοιχεί σε αριθμό λέξεων των 4 bytes. Άρα μπορεί να υποστηρίξει $4 \times 8 \times 2^{48} = 32 \times 2^{48} = 2^5 \times 2^{48} = 2^{53}$ bits.

3) Στην τρίτη περίπτωση ο αριθμός διευθύνσεων αντιστοιχεί σε αριθμό λέξεων των 8 bytes. Άρα μπορεί να υποστηρίξει $8 \times 8 \times 2^{48} = 64 \times 2^{48} = 2^6 \times 2^{48} = 2^{54}$ bits.

ΑΣΚΗΣΗ 2

Ο επεξεργαστής MIPS 3000 διαθέτει 32 εσωτερικούς καταχωρητές με μήκος λέξης 4 bytes. Πόσες γραμμές πρέπει να έχει ο εσωτερικός δίαυλος διευθύνσεων για να μπορεί να επιλέγει έναν από αυτούς τους καταχωρητές;

Ο εξωτερικός δίαυλος διευθύνσεων έχει 32 γραμμές. Πόσα bits μνήμης μπορεί να υποστηρίξει, αν κάθε διεύθυνση αναφέρεται: 1) σε ένα byte, 2) σε μία λέξη των 4 bytes;

Λύση:
Αφού ο επεξεργαστής MIPS 3000 διαθέτει 32 εσωτερικούς καταχωρητές απαιτείται ο εσωτερικός δίαυλος διευθύνσεων να έχει τόσες γραμμές ώστε να υποστηρίζονται 32 διαφορετικές διευθύνσεις. Άρα κατά ελάχιστον το πλήθος των γραμμών πρέπει να είναι 5 αφού $32=2^5$.

Ο εξωτερικός δίαυλος διευθύνσεων έχει 32 γραμμές. Άρα μπορεί να προσπελάσει 2^{32} = 4.294.967.296 διαφορετικές διευθύνσεις.

1) Στην πρώτη περίπτωση ο αριθμός διευθύνσεων αντιστοιχεί σε αριθμό bytes. Άρα μπορεί να υποστηρίξει $8 \times 2^{32} = 8 \times 4.294.967.296 = 34.359.738.368$ bits.

2) Στη δεύτερη περίπτωση ο αριθμός διευθύνσεων αντιστοιχεί σε αριθμό λέξεων των 4 bytes. Άρα μπορεί να υποστηρίξει $4 \times 8 \times 2^{32} = 4 \times 8 \times 4.294.967.296 = 137.438.953.472$ bits.

ΑΣΚΗΣΗ 3

Ένας επεξεργαστής διαθέτει 48 εσωτερικούς καταχωρητές με μήκος λέξης 8 bytes. Πόσες γραμμές πρέπει να έχει ο εσωτερικός δίαυλος διευθύνσεων για να μπορεί να επιλέγει έναν από αυτούς τους καταχωρητές;

Ο εξωτερικός δίαυλος διευθύνσεων έχει 64 γραμμές. Πόσα bits μνήμης μπορεί να υποστηρίξει, αν κάθε διεύθυνση αναφέρεται: 1) σε ένα byte, 2) σε μία λέξη των 4 bytes, 3) σε μία λέξη των 8 bytes;

Λύση:
Αφού ο επεξεργαστής διαθέτει 48 εσωτερικούς καταχωρητές απαιτείται ο εσωτερικός δίαυλος διευθύνσεων να έχει τόσες γραμμές ώστε να υποστηρίζονται 48 διαφορετικές διευθύνσεις. Άρα κατά ελάχιστον το πλήθος των γραμμών πρέπει να είναι 6 αφού 32<48<64 δηλ $2^5<48<2^6$.

Ο εξωτερικός δίαυλος διευθύνσεων έχει 64 γραμμές. Άρα μπορεί να προσπελάσει 2^{64} διαφορετικές διευθύνσεις.

1) Στην πρώτη περίπτωση ο αριθμός διευθύνσεων αντιστοιχεί σε αριθμό bytes. Άρα μπορεί να υποστηρίξει $8 \times 2^{64} = 2^3 \times 2^{64} = 2^{67}$ bits.

2) Στη δεύτερη περίπτωση ο αριθμός διευθύνσεων αντιστοιχεί σε αριθμό λέξεων των 4 bytes. Άρα μπορεί να υποστηρίξει $4 \times 8 \times 2^{64} = 32 \times 2^{64} = 2^5 \text{ x } 2^{64} = 2^{69}$ bits.
3) Στην τρίτη περίπτωση ο αριθμός διευθύνσεων αντιστοιχεί σε αριθμό λέξεων των 8 bytes. Άρα μπορεί να υποστηρίξει $8 \times 8 \times 2^{64} = 64 \times 2^{64} = 2^6 \text{ x } 2^{64} = 2^{70}$ bits.

ΑΣΚΗΣΗ 4

Ένας επεξεργαστής διαθέτει ένα δίαυλο δεδομένων 48 bits. Μία εντολή ανάγνωσης μνήμης χρειάζεται για να ολοκληρωθεί 7 κύκλους μηχανής. Οι 2 πρώτοι κύκλοι μηχανής χρειάζονται 4 κύκλους ρολογιού έκαστος, οι δύο επόμενοι κύκλοι μηχανής 3 κύκλους ρολογιού και οι 3 τελευταίοι κύκλοι μηχανής 2 κύκλους ρολογιού έκαστος. Ποιος είναι ο μέγιστος ρυθμός ανάγνωσης δεδομένων μνήμης (bits/sec) από τον επεξεργαστή αν ένας κύκλος ρολογιού διαρκεί 40 nsec.

Λύση:
Κατά αρχήν υπολογίζουμε το συνολικό αριθμό κύκλων ρολογιού που απαιτείται για να ολοκληρωθεί μία εντολή ανάγνωσης μνήμης του επεξεργαστή. Πρέπει να προσθέσουμε τους κύκλους ρολογιού όλων των κύκλων μηχανής. Άρα ο συνολικός αριθμός κύκλων ρολογιού που απαιτούνται για την εκτέλεση μιας εντολής ανάγνωσης είναι 2x4 + 2x3 + 3x2 = 20 κύκλοι ρολογιού. Η χρονική διάρκεια των 20 κύκλων ρολογιού είναι 20 X 40 nsec = 800 nsec. Αφού ο δίαυλος δεδομένων είναι 48 bits τότε στη διάρκεια των 20 κύκλων ρολογιού ο μικροεπεξεργαστής μπορεί να διαβάσει 48 bits. Δηλαδή ο ρυθμός ανάγνωσης δεδομένων μνήμης είναι 48 / 800 (bits/nsec) = 60 Mbits/sec.

ΑΣΚΗΣΗ 5

Ένας επεξεργαστής διαθέτει ένα δίαυλο δεδομένων 32 bits. Μία εντολή ανάγνωσης μνήμης χρειάζεται για να ολοκληρωθεί 5 κύκλους μηχανής. Οι δύο πρώτοι κύκλοι μηχανής χρειάζονται 3 κύκλους ρολογιού έκαστος, ο τρίτος 2 κύκλους ρολογιού και οι 2 τελευταίοι 4

κύκλους ρολογιού έκαστος. Ποιος είναι ο μέγιστος ρυθμός ανάγνωσης δεδομένων μνήμης (bits/sec) από τον επεξεργαστή αν ένας κύκλος ρολογιού διαρκεί 100 nsec.

Λύση:
Κατά αρχήν υπολογίζουμε το συνολικό αριθμό κύκλων ρολογιού που απαιτείται για να ολοκληρωθεί μία εντολή ανάγνωσης μνήμης του επεξεργαστή. Πρέπει να προσθέσουμε τους κύκλους ρολογιού όλων των κύκλων μηχανής. Άρα ο συνολικός αριθμός κύκλων ρολογιού που απαιτούνται για την εκτέλεση μιας εντολής ανάγνωσης είναι 3 + 3 + 2 + 4 + 4 = 16 κύκλοι ρολογιού. Η χρονική διάρκεια των 16 κύκλων ρολογιού είναι 16 X 100 nsec = 1600 nsec. Αφού ο δίαυλος δεδομένων είναι 32 bits τότε στη διάρκεια των 16 κύκλων ρολογιού ο μικροεπεξεργαστής μπορεί να διαβάσει 32 bits. Δηλαδή ο ρυθμός ανάγνωσης δεδομένων μνήμης είναι 32 / 1600 (bits/nsec) = 20 Mbits/sec.

ΑΣΚΗΣΗ 6

Ένας επεξεργαστής διαθέτει ένα δίαυλο δεδομένων 32 bits. Μία εντολή ανάγνωσης μνήμης χρειάζεται για να ολοκληρωθεί 8 κύκλους μηχανής. Οι 3 πρώτοι κύκλοι μηχανής χρειάζονται 3 κύκλους ρολογιού έκαστος, οι δύο επόμενοι κύκλοι μηχανής 4 κύκλους ρολογιού και οι 3 τελευταίοι κύκλοι μηχανής 5 κύκλους ρολογιού έκαστος. Ποιος είναι ο μέγιστος ρυθμός ανάγνωσης δεδομένων μνήμης (bits/sec) από τον επεξεργαστή αν ένας κύκλος ρολογιού διαρκεί 50 nsec.

Λύση:
Κατά αρχήν υπολογίζουμε το συνολικό αριθμό κύκλων ρολογιού που απαιτείται για να ολοκληρωθεί μία εντολή ανάγνωσης μνήμης του επεξεργαστή. Πρέπει να προσθέσουμε τους κύκλους ρολογιού όλων των κύκλων μηχανής. Άρα ο συνολικός αριθμός κύκλων ρολογιού που απαιτούνται για την εκτέλεση μιας εντολής ανάγνωσης είναι 3x3 + 2x4 + 3x5 = 32 κύκλοι ρολογιού. Η χρονική διάρκεια των 32 κύκλων ρολογιού είναι 32 X 50 nsec = 1600 nsec. Αφού ο δίαυλος δεδομένων είναι 32 bits τότε στη διάρκεια των 32 κύκλων ρολογιού ο μικροεπεξεργαστής μπορεί να διαβάσει 32 bits. Δηλαδή ο ρυθμός ανάγνωσης δεδομένων μνήμης είναι 32 / 1600 (bits/nsec) = 20 Mbits/sec.

ΑΣΚΗΣΗ 7

Θεωρείστε επεξεργαστή με δίαυλο διευθύνσεων 16 ψηφίων. Διαθέτετε τα Ολοκληρωμένα Κυκλώματα (ΟΚ) μνήμης, με οργάνωση ένα byte ανά θέση μνήμης, του παρακάτω πίνακα.

Μνήμη	Μέγεθος (bytes)
ROM1	2K
ROM2	2K
RAM1	4K
RAM2	4K
RAM3	8K
RAM4	8K

Χρησιμοποιώντας όλα τα ΟΚ που αναφέρονται στον πίνακα και επιπλέον κυκλώματα, να σχεδιάσετε την αρχιτεκτονική του συστήματος μνήμης που προτείνετε και να δώσετε το πεδίο διευθύνσεων που αντιστοιχεί σε κάθε ΟΚ μνήμης. Να καλυφθούν όλες οι διευθύνσεις μνήμης από την αρχική (0000Η) έως εκείνη που ορίζεται από τη χωρητικότητα των ΟΚ που διαθέτετε.

Λύση
Τοποθετώντας σε διαδοχικές διευθύνσεις τα ΟΚ μνήμης που διαθέτουμε προκύπτει η χαρτογράφηση μνήμης που δίνεται στον παρακάτω πίνακα:

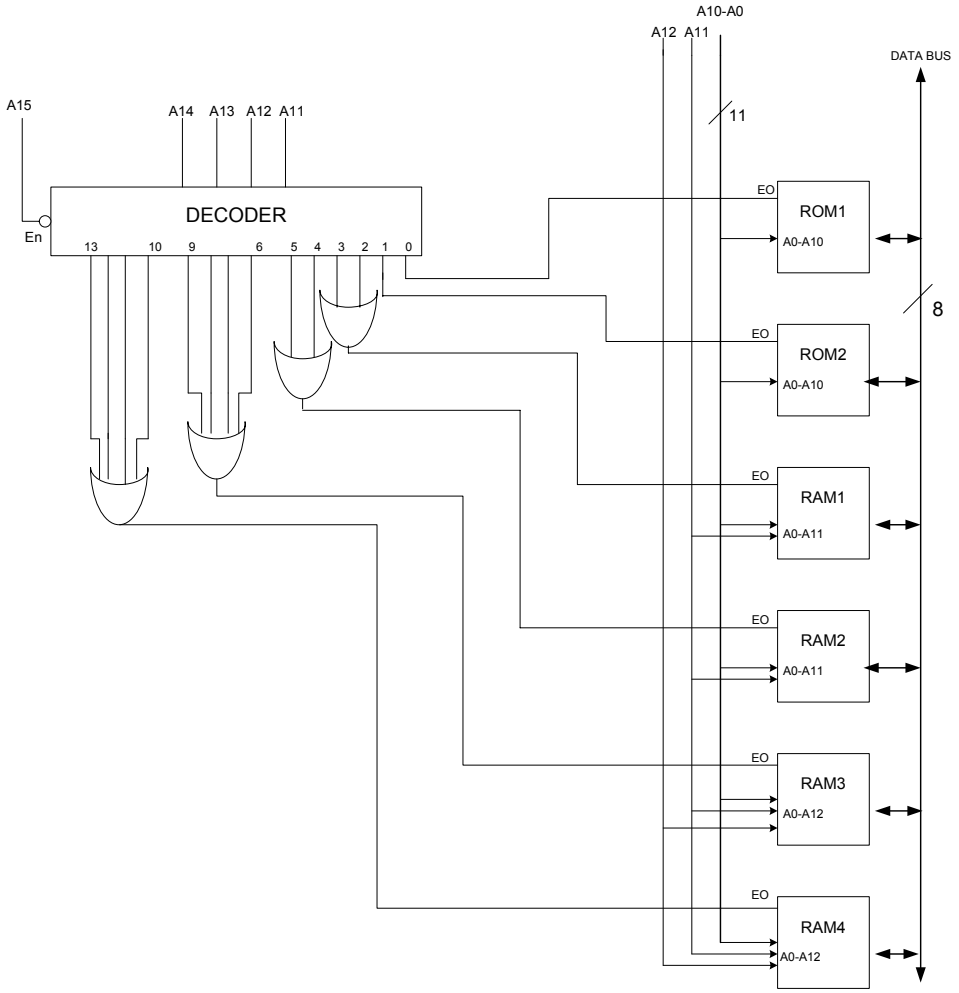
Μνήμη	**Μέγεθος (bytes)**	**Ψηφία Διεύθυνσης**	**Πεδίο Διευθύνσεων**	**Διευθύνσεις σε δυαδική μορφή** A_{15} A_{14} … A_2 A_1 A_0
ROM1	2K	11	0000 – 07FF	0000 0000 0000 0000 0000 0111 1111 1111
ROM2	2K	11	0800 – 0FFF	0000 1000 0000 0000 0000 1111 1111 1111
RAM1	4K	12	1000 – 1FFF	0001 0000 0000 0000 0001 1111 1111 1111
RAM2	4K	12	2000 – 2FFF	0010 0000 0000 0000 0010 1111 1111 1111
RAM3	8K	13	3000 – 4FFF	0011 0000 0000 0000 0100 1111 1111 1111
RAM4	8K	13	5000 – 6FFF	0101 0000 0000 0000 0110 1111 1111 1111

Παρατηρούμε ότι τα 11 λιγότερο σημαντικά ψηφία (A_0 – A_{10}) του δίαυλου διευθύνσεων παίρνουν όλες τις δυνατές τιμές σε κάθε πεδίο διεύθυνσης ΟΚ μνήμης. Συνεπώς τα ψηφία αυτά εφαρμόζονται σε κάθε ΟΚ μνήμης συντελώντας στον καθορισμό της θέσης μνήμης του ΟΚ που αναφέρεται στη λέξη διεύθυνσης. Το ψηφίο A_{15} είναι σταθερά 0 για όλο το σύστημα μνήμης. Το ψηφίο αυτό χρησιμοποιείται στην επιλογή του συστήματος μνήμης που σχεδιάζουμε. Αυτό επιτυγχάνεται θέτοντας το ψηφίο αυτό ως είσοδο ενεργοποίησης (ενεργοποίηση στο 0) για τον αποκωδικοποιητή που θα χρησιμοποιηθεί στην παραγωγή των σημάτων επιλογής των ΟΚ μνήμης. Επειδή οι RAM1 και RAM2 έχουν δίαυλο διευθύνσεων 12 ψηφίων θα εφαρμοστεί σε αυτές και το A_{11}. Επειδή οι RAM3 και RAM4 έχουν δίαυλο διευθύνσεων 13 ψηφίων θα εφαρμοστεί σε αυτές και το A_{11} και το A_{12}.

Όπως παρατηρούμε από τον παραπάνω πίνακα για τον καθορισμό των σημάτων επιλογής των ΟΚ μνήμης απαιτείται η χρήση των ψηφίων A_{11}, A_{12}, A_{13} και A_{14}. Τα σήματα επιλογής (ΕΟ) για κάθε ΟΚ όπως προκύπτουν από τους συνδυασμούς των τεσσάρων ψηφίων δίνονται στον παρακάτω πίνακα.

A_{14}	A_{13}	A_{12}	A_{11}	**ΕΟ**
0	0	0	0	ROM1
0	0	0	1	ROM2
0	0	1	0	RAM1
0	0	1	1	RAM1
0	1	0	0	RAM2
0	1	0	1	RAM2
0	1	1	0	RAM3
0	1	1	1	RAM3
1	0	0	0	RAM3
1	0	0	1	RAM3
1	0	1	0	RAM4
1	0	1	1	RAM4
1	1	0	0	RAM4
1	1	0	1	RAM4

Επομένως, για την επιλογή των ΟΚ μνήμης χρησιμοποιείται 4-σε-16 αποκωδικοποιητής με είσοδο ενεργοποίησης. Αυτός παρέχει τα σήματα επιλογής στις μονάδες μνήμης. Η αρχιτεκτονική του συστήματος μνήμης φαίνεται στο παρακάτω σχήμα.



ΑΣΚΗΣΗ 8

Θεωρείστε επεξεργαστή με δίαυλο δεδομένων 8 ψηφίων και με δίαυλο διευθύνσεων 16 ψηφίων. Διαθέτετε τα Ολοκληρωμένα Κυκλώματα (ΟΚ) μνήμης, με οργάνωση μιας ψηφιολέξης (byte) ανά θέση μνήμης, του παρακάτω πίνακα.

Μνήμη	Μέγεθος (bytes)
ROM1	4K
ROM2	4K
RAM1	2K
RAM2	2K
RAM3	4K
RAM4	8K

Χρησιμοποιώντας όλα τα ΟΚ που αναφέρονται στον πίνακα και επιπλέον κυκλώματα, να σχεδιάσετε αρχιτεκτονική συστήματος μνήμης έτσι ώστε να καλυφθούν όλες οι διευθύνσεις

μνήμης από την αρχική (0000Η) έως εκείνη που ορίζεται από τη χωρητικότητα των ΟΚ που διαθέτετε.
Τα ΟΚ να τοποθετηθούν στο πεδίο διευθύνσεων με τη σειρά που εμφανίζονται στον πίνακα. Δηλαδή, στο πρώτο πεδίο διευθύνσεων που ξεκινά από τη διεύθυνση 0000_{16} να τοποθετηθεί η ROM1, στο αμέσως επόμενο η ROM2, κ.ο.κ..
Συγκεκριμένα:
Α. Δώστε πίνακα με τις διευθύνσεις μνήμης που καταλαμβάνονται από κάθε ΟΚ.
Β. Δώστε πίνακα με τις τιμές των ψηφίων που χρησιμοποιούνται για τη δημιουργία των σημάτων επιλογής των ΟΚ.
Γ. Σχεδιάστε την αρχιτεκτονική του συστήματος μνήμης.

Λύση:
Α. Τοποθετώντας σε διαδοχικές διευθύνσεις τα ΟΚ μνήμης που διαθέτουμε προκύπτει η χαρτογράφηση μνήμης που δίνεται στον παρακάτω πίνακα:

Μνήμη	**Μέγεθος (bytes)**	**Ψηφία Διεύθυνσης**	**Πεδίο Διευθύνσεων**	**Διευθύνσεις σε δυαδική μορφή** $A_{15}\ A_{14}\ \ldots\ A_2\ A_1\ A_0$
ROM1	4K	12	0000 – 0FFF	0000 0000 0000 0000 0000 1111 1111 1111
ROM2	4K	12	1000 – 1FFF	0001 0000 0000 0000 0001 1111 1111 1111
RAM1	2K	11	2000 – 27FF	0010 0000 0000 0000 0010 0111 1111 1111
RAM2	2K	11	2800 – 2FFF	0010 1000 0000 0000 0010 1111 1111 1111
RAM3	4K	12	3000 – 3FFF	0011 0000 0000 0000 0011 1111 1111 1111
RAM4	8K	13	4000 – 5FFF	0100 0000 0000 0000 0101 1111 1111 1111

Β. Παρατηρούμε ότι τα 11 λιγότερο σημαντικά ψηφία (A_0 – A_{10}) του δίαυλου διευθύνσεων παίρνουν όλες τις δυνατές τιμές σε κάθε πεδίο διεύθυνσης ΟΚ μνήμης. Συνεπώς τα ψηφία αυτά εφαρμόζονται σε κάθε ΟΚ μνήμης συντελώντας στον καθορισμό της θέσης μνήμης του ΟΚ που αναφέρεται στη λέξη διεύθυνσης.

Το ψηφίο A_{15} είναι σταθερά 0 για όλο το σύστημα μνήμης. Το ψηφίο αυτό χρησιμοποιείται στην επιλογή του συστήματος μνήμης που σχεδιάζουμε. Αυτό επιτυγχάνεται θέτοντας το ψηφίο αυτό ως είσοδο ενεργοποίησης (ενεργοποίηση στο 0) για τον αποκωδικοποιητή που θα χρησιμοποιηθεί στην παραγωγή των σημάτων επιλογής των ΟΚ μνήμης.

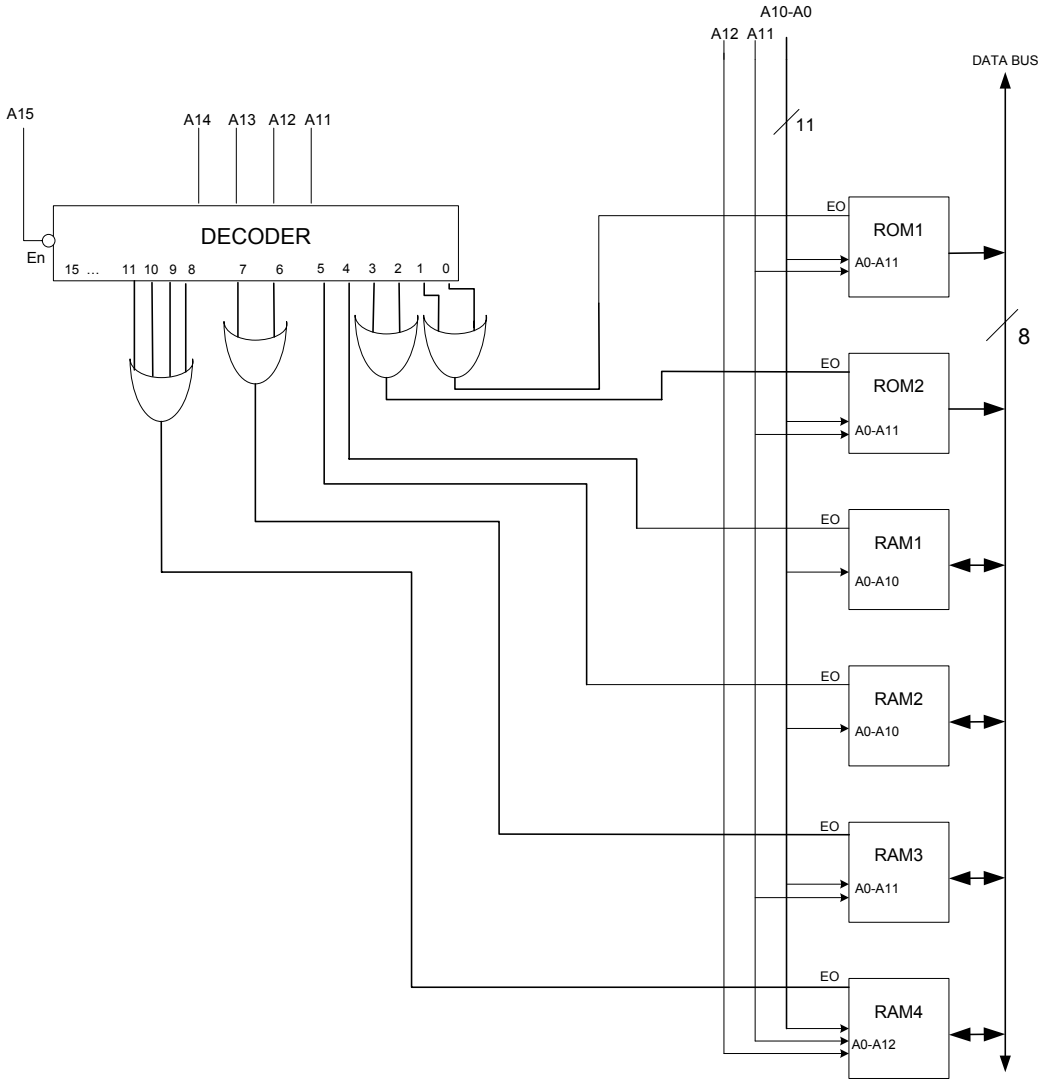
Επειδή οι ROM1, ROM2 και RAM3 έχουν δίαυλο διευθύνσεων 12 ψηφίων θα εφαρμοστεί σε αυτές και το A_{11}. Επειδή η RAM4 έχει δίαυλο διευθύνσεων 13 ψηφίων θα εφαρμοστεί σε αυτή και το A_{11} και το A_{12}.

Όπως παρατηρούμε από τον παραπάνω πίνακα για τον καθορισμό των σημάτων επιλογής των ΟΚ μνήμης απαιτείται η χρήση των ψηφίων A_{11}, A_{12}, A_{13} και A_{14}. Τα σήματα επιλογής (ΕΟ) για κάθε ΟΚ όπως προκύπτουν από τους συνδυασμούς των τεσσάρων ψηφίων δίνονται στον παρακάτω πίνακα.

A_{14}	A_{13}	A_{12}	A_{11}	**ΕΟ**
0	0	0	0	ROM1

0	0	0	1	ROM1
0	0	1	0	ROM2
0	0	1	1	ROM2
0	1	0	0	RAM1
0	1	0	1	RAM2
0	1	1	0	RAM3
0	1	1	1	RAM3
1	0	0	0	RAM4
1	0	0	1	RAM4
1	0	1	0	RAM4
1	0	1	1	RAM4

Γ. Όπως προκύπτει από τον προηγούμενο πίνακα, για την επιλογή των ΟΚ απαιτείται η χρήση 4-σε-16 αποκωδικοποιητή με είσοδο ενεργοποίησης. Αυτός παρέχει τα σήματα επιλογής στις μονάδες μνήμης.
Η αρχιτεκτονική του συστήματος μνήμης φαίνεται στο παρακάτω σχήμα:



ΑΣΚΗΣΗ 9

Θεωρείστε επεξεργαστή με δίαυλο **δεδομένων 8** ψηφίων και δίαυλο **διευθύνσεων 16** ψηφίων ($A_{15}...A_0$). Να σχεδιάσετε σύστημα μνήμης που στην αρχική διεύθυνση του πεδίου διευθύνσεων να περιέχει μνήμη **ROM 4K×8 bits** και ακολούθως να περιέχει μνήμη **RAM 4K×8 bits**. Η αρχική διεύθυνση του συστήματος μνήμης είναι η $\mathbf{0000_{16}}$.

Έχετε στη διάθεσή σας τους ακόλουθους τύπους Ολοκληρωμένων Κυκλωμάτων (Ο.Κ.): (α) ROM **2K×8 bits**, (β) RAM **4K×4 bits**, και (γ) όλους τους τύπους των αποκωδικοποιητών και των λογικών πυλών. Οι αποκωδικοποιητές έχουν σήμα ενεργοποίησης σε **χαμηλή** στάθμη. Μπορείτε να χρησιμοποιήσετε περισσότερα από ένα Ο.Κ. από κάθε τύπο.

Πιο συγκεκριμένα:

α. Καθορίστε το πλήθος και τον τύπο των Ο.Κ. που απαιτούνται. Δώστε πίνακα με τις διευθύνσεις μνήμης που καταλαμβάνονται από κάθε Ο.Κ. μνήμης που χρησιμοποιήσατε.

β. Δώστε πίνακα με τις τιμές των σημάτων διευθύνσεων που χρησιμοποιούνται για τη δημιουργία των σημάτων επιλογής των Ο.Κ. μνήμης.

γ. Σχεδιάστε την αρχιτεκτονική του συστήματος μνήμης.

Λύση

α. Επειδή η πρώτη μνήμη ROM, που τοποθετείται στην αρχή του πεδίου διευθύνσεων, έχει μέγεθος 4K×8 bits και λαμβάνοντας υπόψη τα διαθέσιμα Ο.Κ. θα χρησιμοποιηθούν δύο ΟΚ (ΟΚ1, ΟΚ2) τύπου ROM1 2K×8 bits με συνεχόμενες περιοχές διευθύνσεων.

Με βάση τα διαθέσιμα ΟΚ, για τη μνήμη RAM θα χρησιμοποιηθεί ένα ζεύγος ΟΚ (ΟΚ3, ΟΚ4) τύπου RAM 4K×4. Η οργάνωση σε ζεύγος είναι απαραίτητη καθώς ο επεξεργαστής έχει δίαυλο δεδομένων 8 ψηφίων ενώ κάθε θέση μνήμης των ΟΚ3, ΟΚ4 έχει εύρος τεσσάρων ψηφίων.

Με βάση τα παραπάνω και λαμβάνοντας υπόψη ότι η αρχική διεύθυνση του συστήματος μνήμης είναι 0000Η, προκύπτει ο ακόλουθος πίνακας με τις περιοχές διευθύνσεων του κάθε ΟΚ μνήμης.

ΟΚ	**Μέγεθος (bytes)**	**Ψηφία Διεύθυνσης**	**Πεδίο Διευθύνσεων**	**Δυαδική αναπαράσταση $A_{15}A_{14}$ ..... A_1A_0**
ΟΚ1 (ROM1 2K×8)	2K	11	0000-07FF	0000 0000 0000 0000 0000 0111 1111 1111
ΟΚ2 (ROM1 2K×8)	2K	11	0800-0FFF	0000 1000 0000 0000 0000 1111 1111 1111
ΟΚ3, ΟΚ4 (RAM 4K×4, RAM 4K×4)	4K, 4K	12, 12	1000-1FFF	0001 0000 0000 0000 0001 1111 1111 1111

β. Παρατηρούμε ότι τα 11 λιγότερο σημαντικά ψηφία (A_{10}-A_0) του διαύλου διευθύνσεων παίρνουν όλες τις δυνατές τιμές σε κάθε πεδίο διεύθυνσης για κάθε ΟΚ. Επομένως, τα ψηφία αυτά εφαρμόζονται σε όλα τα ΟΚ για τον καθορισμό της εκάστοτε διεύθυνσης. Επίσης, για τα ΟΚ3, ΟΚ4 επειδή απαιτούνται 12 ψηφία για διευθυνσιοδότηση θα χρησιμοποιηθεί και το A_{11}.

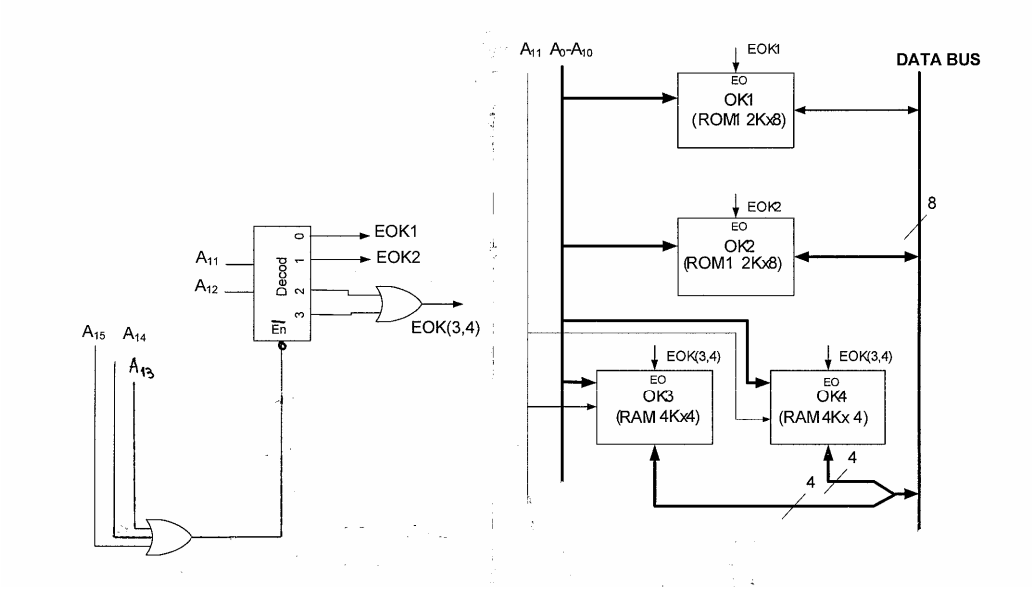
Όσον αφορά τη δημιουργία των σημάτων επιλογής για κάθε Ο.Κ. αυτή θα γίνει με βάση τα σήματα A_{15}-A_{11} χρησιμοποιώντας τον παρακάτω πίνακα.

ΕΟ	**A_{15}**	**A_{14}**	**A_{13}**	**A_{12}**	**A_{11}**
ΟΚ1	0	0	0	0	0
ΟΚ2	0	0	0	0	1
ΟΚ3, ΟΚ4	0	0	0	1	0
	0	0	0	1	1

γ. Για το σχεδιασμό του συστήματος μνήμης πρέπει να δημιουργηθούν τα σήματα επιλογής για κάθε ΟΚ μέσω κυκλώματος αποκωδικοποίησης λαμβάνοντας υπόψη ότι έχουμε αποκωδικοποιητή 2 σε 4. Πρέπει να σημειωθεί ότι τα ΟΚ3, ΟΚ4 έχουν κοινό σήμα επιλογής.

Εξετάζοντας τον πίνακα του ερωτήματος Β συμπεραίνουμε ότι για τα ΟΚ1-ΟΚ4 τα ψηφία $A_{15} = A_{14} = A_{13} = 0$ ενώ τα A_{12}, A_{11} παίρνουν όλους του δυνατούς συνδυασμούς. Συγκεκριμένα, όταν $A_{12} = A_{11} = 0$ επιλέγεται το ΟΚ1, όταν $A_{12} = 0$ και $A_{11} = 1$ επιλέγεται το ΟΚ2 κ.λ.π. Επομένως, θα χρησιμοποιήσουμε έναν αποκωδικοποιητή 2 σε 4 με εισόδους τα A_{12}, A_{11}, ενώ ο αποκωδικοποιητής θα ενεργοποιείται όταν $A_{15} = A_{14} = A_{13} = 0$.

Με βάση τα παραπάνω η αρχιτεκτονική του συστήματος μνήμης είναι η ακόλουθη:

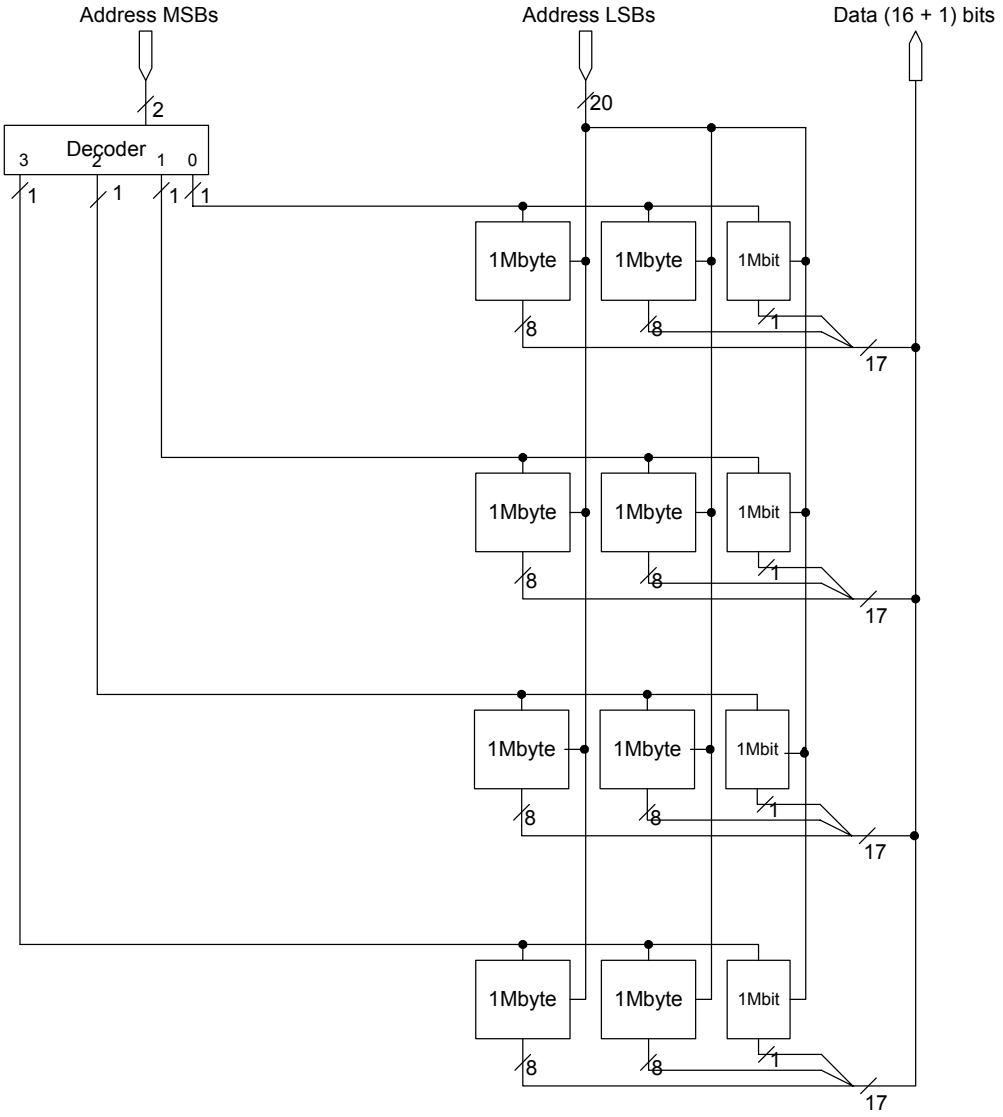


ΑΣΚΗΣΗ 10

Να σχεδιαστεί σύστημα κύριας μνήμης χωρητικότητας **2^{22} θέσεων με οργάνωση 16 δυαδικών** ψηφίων και **ένα bit ισοτιμίας ανά θέση μνήμης**. Έχετε στη διάθεσή σας ολοκληρωμένα κυκλώματα (chips) 1MByte (με οργάνωση 8 δυαδικών ψηφίων ανά θέση μνήμης) και ολοκληρωμένα κυκλώματα 1 Mbit (με οργάνωση ενός δυαδικού ψηφίου ανά θέση μνήμης).

Λύση:

Για κάθε θέση μνήμης απαιτούνται 16 δυαδικά ψηφία συν το ψηφίο ισοτιμίας, σύνολο 17 δυαδικά ψηφία. Άρα κάθε θέση μνήμης απαιτεί 2 bytes και 1 bit, γιατί $2 \times 8 + 1 = 17$.
Έτσι με μία δομή που αποτελείται από δύο ολοκληρωμένα του 1Mbyte (2^{20} θέσεις x 8 bits) και ένα ολοκληρωμένο του 1Mbit (2^{20} θέσεις x 1 bit) μπορούμε να έχουμε 2^{20} θέσεις x 17 bits. Αφού η ζητούμενη μνήμη απαιτεί 2^{22} θέσεις των 17 bits, πρέπει να χρησιμοποιήσουμε 2^2=4 επαναλήψεις αυτής της δομής και έναν αποκωδικοποιητή 2 σε 4 ως εξής:



ΑΣΚΗΣΗ 11

Δίνεται μικροϋπολογιστικό σύστημα με δίαυλο δεδομένων 8 ψηφίων και δίαυλο διευθύνσεων 16 ψηφίων. Στο σύστημα αυτό οι διευθύνσεις μνήμης 0000-0FFF καλύπτονται από μία μνήμη ROM ενώ οι διευθύνσεις μνήμης 2000-3FFF και 5000-5FFF από τα ολοκληρωμένα κυκλώματα μνήμης RAM1 και RAM2, αντίστοιχα.

Α. Πόσο είναι το μέγεθος (Kbits) κάθε ολοκληρωμένου κυκλώματος μνήμης (ROM, RAM1, RAM2);

Β. Θεωρείστε ότι διαθέτετε επιπλέον ολοκληρωμένα κυκλώματα μνήμης RAM των 16Kbits με οργάνωση 8 ψηφίων ανά λέξη και των 32Kbits με οργάνωση 4 ψηφίων ανά λέξη. Να επεκτείνετε το παραπάνω σύστημα μνήμης ώστε να καλυφθούν πλήρως τα πρώτα 32KB της μνήμης (διευθύνσεις 0000-7FFF). Εκτός των κυκλωμάτων μνήμης διαθέτετε αποκωδικοποιητές 2 σε 4 με είσοδο επίτρεψης και βασικές πύλες. Στη λύση που θα δώσετε να φροντίσετε ώστε να γίνεται πλήρης χρήση του μεγέθους μνήμης κάθε ολοκληρωμένου κυκλώματος μνήμης που χρησιμοποιείτε (να μην υπάρχουν δηλαδή αχρησιμοποίητες περιοχές μνήμης στα ολοκληρωμένα κυκλώματα).

Λύση

Α.

Καταρχάς, αφού ο δίαυλος δεδομένων είναι 8 ψηφίων αυτό θα είναι και το μήκος λέξης κάθε κυκλώματος μνήμης.

Η μνήμη ROM καλύπτει τις διευθύνσεις 0000-0FFF και συνεπώς περιέχει 2^{12} λέξεις των 8 ψηφίων. Δηλαδή περιέχει $2^{12}=2^2*2^{10}$=4Κλέξεις ή 32Kbits.

Η μνήμη RAM1 καλύπτει τις διευθύνσεις 2000-3FFF, δηλαδή συνολικά 1FFF διευθύνσεις, που αντιστοιχούν σε 2^{13} λέξεις των 8 ψηφίων. Δηλαδή περιέχει $2^{13}=2^3*2^{10}$=8Κλέξεις ή 64Kbits.

Η μνήμη RAM2 καλύπτει τις διευθύνσεις 5000-5FFF, δηλαδή συνολικά 0FFF διευθύνσεις, που αντιστοιχούν σε 2^{12} λέξεις των 8 ψηφίων. Δηλαδή περιέχει $2^{12}=2^2*2^{10}$=4Κλέξεις ή 32Kbits.

Συνολικά, το μικροϋπολογιστικό σύστημα διαθέτει μνήμη των 128Kbits.

Β.

Θα προσπαθήσουμε να καλύψουμε τα πρώτα 32KB του συστήματος μνήμης χρησιμοποιώντας τα ολοκληρωμένα κυκλώματα μνήμης RAM που μας δίνονται. Αφού το μήκος λέξης είναι 8 ψηφία τα 32KB αντιστοιχούν σε 32Κλέξεις.

Κάθε ολοκληρωμένο κύκλωμα μνήμης RAM των 16Kbits με οργάνωση 8 ψηφίων ανά λέξη περιέχει 2Κλέξεις των 8 ψηφίων, ενώ κάθε ολοκληρωμένο κύκλωμα μνήμης RAM των 32Kbits με οργάνωση 4 ψηφίων ανά λέξη περιέχει 8Κλέξεις των 4 ψηφίων.

Αρχικά πρέπει να καλύψουμε τις διευθύνσεις 1000-1FFF που αντιστοιχούν σε 4Κλέξεις των 8 ψηφίων. Θα χρησιμοποιήσουμε δύο μνήμες των 2Κλέξεων (RAM3, RAM4 στο σχήμα). Η μνήμη RAM3 καλύπτει τις διευθύνσεις 1000-17FF ενώ η RAM4 τις διευθύνσεις 1800-1FFF.

Κατόπιν πρέπει να καλύψουμε τις διευθύνσεις 4000-4FFF που και αυτές αντιστοιχούν σε 4Κλέξεις των 8 ψηφίων και άρα θα χρειαστούν 2 μνήμες των 2Κλέξεων (RAM5, RAM6). Η RAM5 καλύπτει τις διευθύνσεις 4000-47FF ενώ η RAM6 τις διευθύνσεις 4800-4FFF.

Τέλος πρέπει να καλύψουμε τις διευθύνσεις 6000-7FFF που αντιστοιχούν σε 8Κλέξεις των 8 ψηφίων και άρα θα χρησιμοποιήσουμε δύο μνήμες (RAM7, RAM8) των 8Κλέξεων των 4 ψηφίων ανά λέξη όπως φαίνεται στο σχήμα.

Με την τοποθέτηση που κάναμε προκύπτει η χαρτογράφηση μνήμης που δίνεται στον παρακάτω πίνακα:

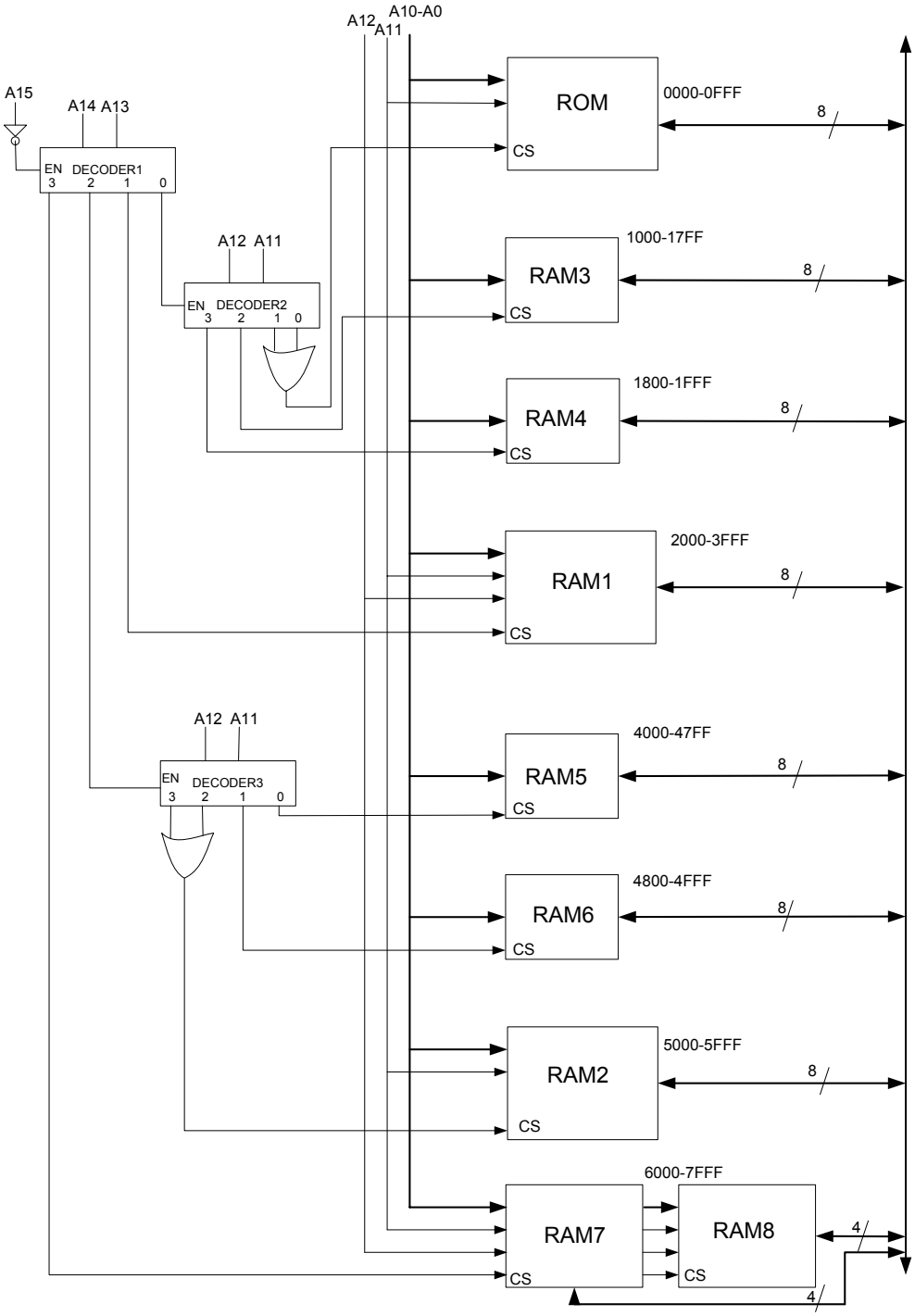
Μνήμη	**Μέγεθος (λέξεις)**	**Ψηφία Διεύθυνσης**	**Πεδίο Διευθύνσεων**	**Διευθύνσεις σε δυαδική μορφή** A_{15} A_{14} … A_2 A_1 A_0
ROM	4K	12	0000 – 0FFF	0000 0000 0000 0000 0000 1111 1111 1111
RAM3	2K	11	1000 – 17FF	0001 0000 0000 0000 0001 0111 1111 1111
RAM4	2K	11	1800 – 1FFF	0001 1000 0000 0000 0001 1111 1111 1111
RAM1	8K	13	2000 – 3FFF	0010 0000 0000 0000 0011 1111 1111 1111

RAM5	2K	11	4000 – 47FF	0100 0000 0000 0000 0100 0111 1111 1111
RAM6	2K	11	4800 – 4FFF	0100 1000 0000 0000 0100 1111 1111 1111
RAM2	4K	12	5000-5FFF	0101 0000 0000 0000 0101 1111 1111 1111
RAM7,RAM8	8K	13	6000-7FFF	0110 0000 0000 0000 0111 1111 1111 1111

Τα ολοκληρωμένα κυκλώματα της μνήμης πρέπει να συνδεθούν στα κατάλληλα ψηφία του δίαυλου διευθύνσεων. Έτσι για εσωτερική διευθυνσιοδότηση μνήμες 2Κλέξεων συνδέονται με τα ψηφία του δίαυλου διευθύνσεων A_{10}-A_0, μνήμες των 4Κλέξεων συνδέονται με τα ψηφία A_{11}-A_0 και μνήμες των 8Κλέξεων συνδέονται με τα ψηφία A_{12}-A_0.

Για να δημιουργήσουμε τα σήματα επιλογής (CS) στις μνήμες χρησιμοποιούμε τα ψηφία A_{15}-A_{11} και μια δομή από 3 αποκωδικοποιητές όπως φαίνεται στο σχήμα. Ο DECODER1 δέχεται ως εισόδους τα ψηφία A_{14}, A_{13} και οι τέσσερις έξοδοί του αντιστοιχούν σε τμήματα μνήμης των 8Κλέξεων. Οι μνήμες RAM1 και RAM7, RAM8 επιλέγονται κατευθείαν από εξόδους του αποκωδικοποιητή αυτού. Οι άλλες δυο έξοδοι χρησιμοποιούνται ως σήματα επίτρεψης σε άλλους αποκωδικοποιητές που παράγουν σήματα για τις μικρότερου μεγέθους μνήμες. Πρόκειται για τους αποκωδικοποιητές DECODER 2 και 3 που λαμβάνουν ως είσοδο τα A_{12} και A_{11}. Το ψηφίο A_{15} αφού αντιστραφεί αποτελεί είσοδο επίτρεψης για τον DECODER1 ώστε το σύστημα μνήμης να επιλέγεται με A_{15}=0 (όπως φαίνεται και στον πίνακα) και με αυτό τον τρόπο να αντιστοιχεί στις πρώτες 32K διευθύνσεις.

Παρατήρηση
Οι διευθύνσεις μνήμης 6000-7FFF θα μπορούσαν, σε μια εναλλακτική υλοποίηση, να καλυφθούν και από 4 μνήμες των 2Κλέξεων. Τότε θα χρειαζόμασταν έναν επιπλέον αποκωδικοποιητή 2-σε-4 με εισόδους τα A_{11} και A_{12}. Κάθε μία από τις 4 εξόδους του θα αποτελούσε το σήμα ελέγχου μνήμης των 2Κλέξεων. Στην είσοδο επίτρεψης του αποκωδικοποιητή θα συνδέονταν η έξοδος 3 του DECODER1.


A15
A14 A13
EN DECODER1
3 2 1 0
A12 A11
EN DECODER2
3 2 1 0
A12 A11
EN DECODER3
3 2 1 0
A12
A11
A10-A0
ROM
0000-0FFF
8
CS
RAM3
1000-17FF
RAM4
1800-1FFF
RAM1
2000-3FFF
RAM5
4000-47FF
RAM6
4800-4FFF
RAM2
5000-5FFF
RAM7
6000-7FFF
RAM8
4

ΑΣΚΗΣΗ 12

Θεωρείστε μικροϋπολογιστικό σύστημα με δίαυλο δεδομένων 8 ψηφίων και δίαυλο διευθύνσεων 16 ψηφίων. Να σχεδιάσετε το σύστημα μνήμης του έτσι ώστε να καλυφθούν τα πρώτα 48KBytes. Οι πρώτες 8K διευθύνσεις να καλυφθούν από μνήμη ROM. Έχετε στη διάθεσή σας ένα ολοκληρωμένο κύκλωμα (ΟΚ) μνήμης ROM 8KBytes και ολοκληρωμένα κυκλώματα (περισσότερα από ένα) μνήμης RAM των 4KBytes και 32KBytes. Όλες οι μνήμες έχουν οργάνωση 8 ψηφίων ανά λέξη. Σχεδιάστε το σύστημα μνήμης με το μικρότερο δυνατό αριθμό ολοκληρωμένων κυκλωμάτων μνήμης.

Διαθέτετε επίσης δύο αποκωδικοποιητές 2 σε 4 με είσοδο επίτρεψης και πύλες Or 2 εισόδων για τη δημιουργία των σημάτων ελέγχου.
Στη λύση που θα δώσετε να φροντίσετε ώστε να γίνεται πλήρης χρήση του μεγέθους μνήμης κάθε ολοκληρωμένου κυκλώματος μνήμης που χρησιμοποιείτε (να μην υπάρχουν δηλαδή αχρησιμοποίητες περιοχές μνήμης στα ολοκληρωμένα κυκλώματα).

Λύση

Το ολοκληρωμένο κύκλωμα μνήμης ROM των 8KBytes θα καλύπτει τις διευθύνσεις 0000 έως 1FFF.

Η χρήση ενός αποκωδικοποιητή 2 σε 4 με είσοδο τα σήματα A_{15} και A_{14} χωρίζει το συνολικό πεδίο διευθύνσεων των 64KBytes σε 4 ομάδες των 16KBytes. Για την πρώτη ομάδα υπολείπονται 8KBytes που θα καλυφθούν με δυο ολοκληρωμένα των 4KBytes. Οι επόμενες δυο ομάδες μπορούν να καλυφθούν με ένα ολοκληρωμένο των 32KBytes. Έτσι καλύπτονται τα πρώτα 48KBytes του συστήματος μνήμης.

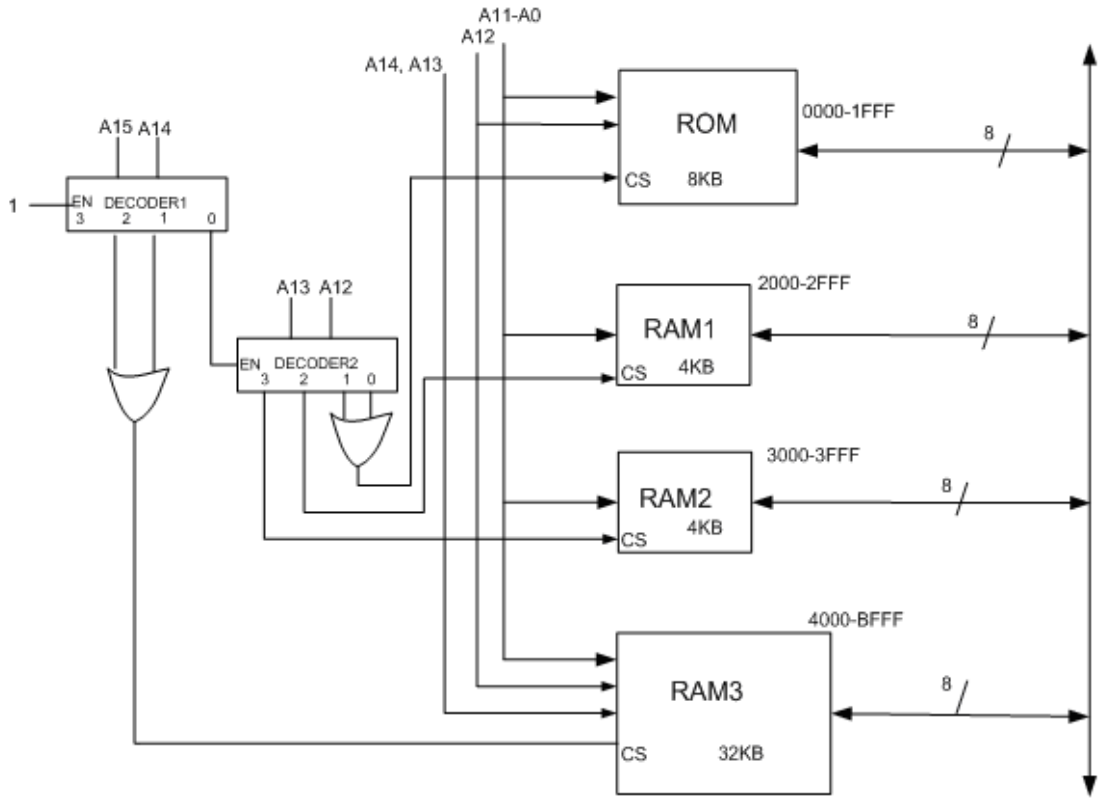
Με την τοποθέτηση που κάναμε προκύπτει η χαρτογράφηση μνήμης που δίνεται στον παρακάτω πίνακα:

Μνήμη	**Μέγεθος (bytes)**	**Ψηφία Διεύθυνσης**	**Πεδίο Διευθύνσεων**	**Διευθύνσεις σε δυαδική μορφή A_{15} A_{14} … A_2 A_1 A_0**
ROM	8K	13	0000 – 1FFF	0000 0000 0000 0000 0001 1111 1111 1111
RAM1	4K	12	2000 – 2FFF	0010 0000 0000 0000 0010 1111 1111 1111
RAM2	4K	12	3000 – 3FFF	0011 0000 0000 0000 0011 1111 1111 1111
RAM3	32K	15	4000 – BFFF	0100 0000 0000 0000 1011 1111 1111 1111

Τα ολοκληρωμένα κυκλώματα της μνήμης πρέπει να συνδεθούν στα κατάλληλα ψηφία του δίαυλου διευθύνσεων. Έτσι για εσωτερική διευθυνσιοδότηση στις μνήμες των 4KBytes συνδέονται τα ψηφία A_{11}-A_0 του διαύλου διευθύνσεων, στη μνήμη των 8KBytes συνδέονται τα ψηφία A_{12}-A_0 και στη μνήμη των 32KBytes συνδέονται τα ψηφία A_{14}-A_0.

Για να δημιουργήσουμε τα σήματα επιλογής (Chip Select - CS) στις μνήμες χρησιμοποιούμε τα ψηφία A_{15}-A_{12} και μια δομή από 2 αποκωδικοποιητές και πύλες OR όπως φαίνεται στο σχήμα. Ο DECODER1 δέχεται ως εισόδους τα ψηφία A_{15}, A_{14} και οι τέσσερις έξοδοί του αντιστοιχούν σε τμήματα μνήμης των 16KBytes. Η πρώτη έξοδος (έξοδος 0) του

DECODER1 αποτελεί είσοδο επίτρεψης για τον δεύτερο αποκωδικοποιητή DECODER2 ο οποίος λαμβάνει ως εισόδους τα A_{13} και A_{12}. Ο DECODER2 παράγει τα σήματα επιλογής για τα πρώτα 16KBytes του συστήματος μνήμης. Η είσοδος επίτρεψης του DECODER1 τίθεται στο '1' (συνεχώς ενεργοποιημένο).



Προγραμματισμός σε Assembly

ΑΣΚΗΣΗ 13

Ένας μικροεπεξεργαστής διαθέτει τις παρακάτω εντολές:

LOADA# : εντολή άμεσης φόρτωσης του καταχωρητή Α με αριθμό
INCRA : εντολή αύξησης του περιεχομένου του καταχωρητή Α κατά 1
STOREA: εντολή αποθήκευσης του περιεχομένου του συσσωρευτή Α

Διαθέτει δε τους παρακάτω τρόπους διευθυνσιοδότησης :

Όνομα εντολής	Κωδικός Λειτουργίας	Παράδειγμα	Τρόπος Προσπέλασης
LOADA#	0A	LOADA#A5	Άμεση (immediate)
INCRA	1A	INCRA	Υπονοούμενη
STOREA	20	STOREA $00A5	Απόλυτη διευθυνσιοδότηση
	21	STOREA $A5	Διευθυνσιοδότηση μηδενικής σελίδας
	22	STOREA ($A5A5)	Εμμεση διευθυνσιοδότηση

Το παρακάτω πρόγραμμα είναι αποθηκευμένο στις διευθύνσεις μνήμης 2300_{16} μέχρι και $230D_{16}$

```
2300   LOADA #B4
2302   INCRA
2303   INCRA
2304   STOREA $04
2306   INCRA
2307   STOREA ($1313)
230A   INCRA
230B   STOREA $0008
```

Συμπληρώστε τους παρακάτω πίνακες. Δείξτε ποιες θέσεις μνήμης θα αλλάξουν το περιεχόμενο τους (και ποιο θα είναι το καινούριο περιεχόμενο τους) μετά την εκτέλεση του προγράμματος.

Υποθέτουμε ότι το περιεχόμενο της θέσης μνήμης 1313_{16} είναι το $000B_{16}$.

Διεύθυνση Μνήμης	Περιεχόμενο μνήμης
0000	
0001	
0002	
0003	
0004	
0005	
0006	
0007	
0008	
0009	
000A	
000B	

Διεύθυνση Μνήμης	Κωδικός Λειτουργίας	Εντολή	Περιεχόμενο του Α μετά την εκτέλεση της εντολής
2300			
2301			
2302			
2303			
2304			
2305			
2306			
2307			
2308			
2309			
230A			
230B			
230C			
230D			

Λύση:

Διεύθυνση Μνήμης	Κωδικός Λειτουργίας	Εντολή	Περιεχόμενο του Α μετά την εκτέλεση της εντολής
2300	0A	LOADA#	B4
2301	B4	B4	
2302	1A	INCRA	B5
2303	1A	INCRA	B6
2304	21	STOREA	
2305	04	$04	
2306	1A	INCRA	B7
2307	22	STOREA	
2308	13	($13	
2309	13	13)	
230A	1A	INCRA	B8
230B	20	STOREA	
230C	00	$00	
230D	08	08	

Διεύθυνση Μνήμης	Περιεχόμενο μνήμης
0000	
0001	
0002	
0003	
0004	B6
0005	
0006	
0007	
0008	B8
0009	
000A	
000B	B7

ΑΣΚΗΣΗ 14

Ο μικροεπεξεργαστής SFAX διαθέτει τις παρακάτω εντολές:
INCA: εντολή αύξησης του περιεχομένου του καταχωρητή Α κατά 1
FOB: εντολή αποθήκευσης του περιεχομένου του συσσωρευτή Α
Διαθέτει τους παρακάτω τρόπους διευθυνσιοδότησης:

Όνομα εντολής	Κωδικός Λειτουργίας	Παράδειγμα	Τρόπος Προσπέλασης
INCA	25	INCA	Υπονοούμενη
FOB	30	FOB $0023	Απόλυτη διευθυνσιοδότηση
	31	FOB $23	Διευθυνσιοδότηση μηδενικής σελίδας
	32	FOB ($2323)	Εμμεση διευθυνσιοδότηση

Το παρακάτω πρόγραμμα είναι αποθηκευμένο στις διευθύνσεις μνήμης 1000_{16} μέχρι και $100A_{16}$:
1000 INCA
1001 FOB $0007
1004 INCA
1005 FOB $03
1007 INCA
1008 FOB ($2424)

Συμπληρώστε τους παρακάτω πίνακες. Δείξτε ποιες θέσεις μνήμης θα αλλάξουν το περιεχόμενο τους (και ποιο θα είναι το καινούργιο περιεχόμενο τους) μετά την εκτέλεση του προγράμματος.

Υποθέτουμε ότι το περιεχόμενο της θέσης μνήμης 2424_{16} είναι το $000A_{16}$. Το αρχικό περιεχόμενο του Α είναι 23.

Διεύθυνση Μνήμης	Κωδικός Λειτουργίας	Εντολή	Περιεχόμενο του Α μετά την εκτέλεση της εντολής
1000			
1001			

1002			
1003			
1004			
1005			
1006			
1007			
1008			
1009			
100A			

Διεύθυνση Μνήμης	Περιεχόμενο μνήμης
0000	
0001	
0002	
0003	
0004	
0005	
0006	
0007	
0008	
0009	
000A	
000B	

Λύση:

Διεύθυνση Μνήμης	Κωδικός Λειτουργίας	Εντολή	Περιεχόμενο του Α μετά την εκτέλεση της εντολής
1000	25	INCA	24
1001	30	FOB $0007	24
1002	00		24
1003	07		24
1004	25	INCA	25
1005	31	FOB $03	25
1006	03		25
1007	25	INCA	26
1008	32	FOB ($2424)	26
1009	24		26
100A	24		26

Διεύθυνση Μνήμης	Περιεχόμενο μνήμης
0000	
0001	
0002	
0003	25
0004	
0005	
0006	
0007	24

0008	
0009	
000A	26
000B	

ΑΣΚΗΣΗ 15

Ένας μικροεπεξεργαστής διαθέτει τις παρακάτω εντολές:
LDA# : εντολή άμεσης φόρτωσης του καταχωρητή Α με αριθμό
INCA : εντολή αύξησης του περιεχομένου του καταχωρητή Α κατά 1
STA: εντολή αποθήκευσης του περιεχομένου του συσσωρευτή Α
Διαθέτει δε τους παρακάτω τρόπους διευθυνσιοδότησης:

Όνομα εντολής	Κωδικός Λειτουργίας	Παράδειγμα	Τρόπος Προσπέλασης
LDA#	0A	LDA#A5	Άμεση (immediate)
INCA	1A	INCA	Υπονοούμενη
STA	30	STA $00A5	Απόλυτη διευθ/δότηση
	31	STA $A5	Διευθυνσιοδότηση μηδενικής σελίδας
	32	STA ($A5A5)	Εμμεση διευθ/δότηση

Το παρακάτω πρόγραμμα είναι αποθηκευμένο στις διευθύνσεις μνήμης 1000_{16} μέχρι και $100C_{16}$:

```
1000   LDA#A5
1002   INCA
1003   STA $03
1005   INCA
1006   STA ($2424)
1009   INCA
100A   STA $0007
```

Συμπληρώστε τους παρακάτω πίνακες. Δείξτε ποιες θέσεις μνήμης θα αλλάξουν το περιεχόμενο τους (και ποιο θα είναι το καινούργιο περιεχόμενο τους) μετά την εκτέλεση του προγράμματος. Υποθέτουμε ότι το περιεχόμενο της θέσης μνήμης 2424_{16} είναι το $000A_{16}$.

Διεύθυνση Μνήμης	Κωδικός Λειτουργίας	Εντολή	Περιεχόμενο του Α μετά την εκτέλεση της εντολής
1000			
1001			
1002			
1003			
1004			
1005			
1006			
1007			
1008			
1009			
100A			
100B			
100C			

Διεύθυνση Μνήμης	Περιεχόμενο μνήμης
0000	
0001	
0002	
0003	
0004	
0005	
0006	
0007	
0008	
0009	
000A	
000B	

Λύση:

Διεύθυνση Μνήμης	Κωδικός Λειτουργίας	Εντολή	Περιεχόμενο του Α μετά την εκτέλεση της εντολής
1000	0A	LDA#	A5
1001	A5	A5	
1002	1A	INCA	A6
1003	31	STA	
1004	03	$03	
1005	1A	INCA	A7
1006	32	STA	
1007	24	($24	
1008	24	24)	
1009	1A	INCA	A8
100A	30	STA	
100B	00	$00	
100C	07	07	

Διεύθυνση Μνήμης	Περιεχόμενο μνήμης
0000	
0001	
0002	
0003	A6
0004	
0005	
0006	
0007	A8
0008	
0009	
000A	A7
000B	

ΑΣΚΗΣΗ 16

Δίδεται το παρακάτω πρόγραμμα, το οποίο έχει γραφτεί για το μικροεπεξεργαστή 8085 της Intel.

```
        MVI B,56
        MVI A,05
        MOV C,A
K:      INR B
        DCR C
        MOV A,C
        JNZ K
        HLT
```

(α) Υποθέτοντας ότι το πρόγραμμα θα αποθηκεύεται στην κύρια μνήμη ξεκινώντας από την θέση $213A_{16}$ βρείτε τα περιεχόμενα της μνήμης από τη θέση $213A_{16}$ έως και την 2145_{16} . Ποιο είναι το μέγεθος του προγράμματος σε bytes;

Διεύθυνση Μνήμης	Περιεχόμενο Μνήμης	Εντολή
213A		
213B		
213C		
213D		
213E		
213F		
2140		
2141		
2142		
2143		
2144		
2145		

(β) Εάν εκτελέσουμε το πρόγραμμα, πόσες φορές θα εκτελεστεί η κάθε εντολή του προγράμματος;

Εντολή	Φορές Εκτέλεσης
MVI B,56	
MVI A,05	
MOV C,A	
INR B	
DCR C	
MOV A,C	
JNZ K	
HLT	

(γ) Ποια θα είναι τα περιεχόμενα των καταχωρητών B και C μετά την εκτέλεση του κάθε γύρου του προγράμματος;

	Περιεχόμενο του καταχωρητή B	Περιεχόμενο του καταχωρητή C

Αρχικό	56	05
Τέλος 1ου γύρου		
Τέλος 2ου γύρου		
Τέλος 3ου γύρου		
Τέλος 4ου γύρου		
Τέλος 5ου γύρου		

Λύση:
(α)

Διεύθυνση Μνήμης	Περιεχόμενο Μνήμης	Εντολή
213A	06	MVI B,56
213B	56	
213C	3E	MVI A,05
213D	05	
213E	4F	MOV C,A
213F	04	INR B
2140	0D	DCR C
2141	79	MOV A,C
2142	C2	JNZ 213F
2143	3F	
2144	21	
2145	76	HLT

Το μέγεθος του προγράμματος είναι 12 bytes.

(β)

Εντολή	Φορές Εκτέλεσης
MVI B,56	1
MVI A,15	1
MOV C,A	1
INR B	5
DCR C	5
MOV A,C	5
JNZ K	5
HLT	1

(γ) Τα περιεχόμενα των δύο καταχωρητών φαίνονται στο παρακάτω πίνακα:

	Περιεχόμενο του καταχωρητή B	Περιεχόμενο του καταχωρητή C
Αρχικό	56	05
Τέλος 1ου γύρου	57	04
Τέλος 2ου γύρου	58	03
Τέλος 3ου γύρου	59	02
Τέλος 4ου γύρου	5A	01
Τέλος 5ου γύρου	5B	00

ΑΣΚΗΣΗ 17

Δίδεται το παρακάτω πρόγραμμα, το οποίο έχει γραφτεί για το μικροεπεξεργαστή 8085 της Intel.

```
      MVI B, 09
      MVI C, 48
K:    INR B
      INR C
      MOV A,B
      SUI 0F
      JNZ K
      HLT
```

(α) Υποθέτοντας ότι το πρόγραμμα θα αποθηκεύεται στην κύρια μνήμη ξεκινώντας από την θέση 1024_{16} βρείτε τα περιεχόμενα της μνήμης από την θέση 1024_{16} έως και την 1030_{16} . Ποιο είναι το μέγεθος του προγράμματος σε bytes;

Διεύθυνση Μνήμης	Περιεχόμενο Μνήμης	Εντολή
1024		
1025		
1026		
1027		
1028		
1029		
102A		
102B		
102C		
102D		
102E		
102F		
1030		

(β) Εάν εκτελέσουμε το πρόγραμμα, πόσες φορές θα εκτελεστεί η κάθε εντολή του προγράμματος;

Εντολή	Φορές Εκτέλεσης
MVI B, 09	
MVI C, 48	
INR B	
INR C	
SUI 0F	
JNZ K	
HLT	

(γ) Ποια θα είναι τα περιεχόμενα των καταχωρητών B και C μετά την εκτέλεση του κάθε γύρου του προγράμματος;

	Περιεχόμενο του καταχωρητή B	Περιεχόμενο του καταχωρητή C
Αρχικό	09	48
Τέλος 1ου γύρου		
Τέλος 2ου γύρου		

(δ) Για να εκτελεστεί ο βρόχος 13 (δεκαδικό) φορές ακριβώς ποιο πρέπει να είναι το αρχικό περιεχόμενο του καταχωρητή B (θεωρείστε ότι δεν υπάρχει η αρχική εντολή MVI B, 09) ;

Λύση:
(α)

Διεύθυνση Μνήμης	Περιεχόμενο Μνήμης	Εντολή
1024	06	MVI B, 09
1025	09	
1026	0E	MVI C, 48
1027	48	
1028	04	INR B
1029	0C	INR C
102A	78	MOV A,B
102B	D6	SUI 0F
102C	0F	
102D	C2	JNZ K
102E	28	
102F	10	
1030	76	HLT

Το μέγεθος του προγράμματος είναι 13 bytes.

(β)

Εντολή	Φορές Εκτέλεσης
MVI B, 09	1
MVI C, 48	1
INR B	6
INR C	6
SUI 0F	6
JNZ K	6
HLT	1

(γ) Τα περιεχόμενα των δύο καταχωρητών φαίνονται στο παρακάτω πίνακα:

	Περιεχόμενο του καταχωρητή B	Περιεχόμενο του καταχωρητή C
Αρχικό	09	48
Τέλος 1ου γύρου	0A	49
Τέλος 2ου γύρου	0B	4A
Τέλος 3ου γύρου	0C	4B
Τέλος 4ου γύρου	0D	4C
Τέλος 5ου γύρου	0E	4D
Τέλος 6ου γύρου	0F	4E

(δ) Το αρχικό περιεχόμενο του καταχωρητή B πρέπει να είναι $B=02_{16}$.

ΑΣΚΗΣΗ 18

Δίδεται το παρακάτω πρόγραμμα, το οποίο έχει γραφτεί για το μικροεπεξεργαστή 80196 της Intel. Μπορείτε να βρείτε την σημασία κάθε εντολής στον Γ τόμο.

```
        LDB 80, #08
        LDB 82, #64
K:      INCB 80
        INCB 82
        CMPB 80, #0F
        JNE K
        BRK
```

(α) Υποθέτοντας ότι το πρόγραμμα θα αποθηκεύεται στην κύρια μνήμη ξεκινώντας από τη θέση 1024_{16} βρείτε τα περιεχόμενα της μνήμης από τη θέση 1024_{16} έως και την 1035_{16}. Ποιο είναι το μέγεθος του προγράμματος σε bytes;

(β) Εάν εκτελέσουμε το πρόγραμμα, πόσες φορές θα εκτελεστεί η κάθε εντολή του προγράμματος;

Εντολή	Φορές Εκτέλεσης
LDB 80, #08	
LDB 82, #64	
INCB 80	
INCB 82	
CMPB 80, #0F	
JNE K	
BRK	

(γ) Ποια θα είναι τα περιεχόμενα των καταχωρητών 80 και 82 μετά την εκτέλεση του κάθε γύρου του προγράμματος;

	Περιεχόμενο του καταχωρητή 80	Περιεχόμενο του καταχωρητή 82
Αρχικό		
Τέλος 1ου γύρου		

Τέλος 2ου γύρου		

Λύση:
(α)

Διεύθυνση Μνήμης	Περιεχόμενο Μνήμης	Εντολή
1024	B1	LDB 80, #08
1025	08	
1026	80	
1027	B1	LDB 82, #64
1028	64	
1029	82	
102A	17	INCB 80
102B	80	
102C	17	INCB 82
102D	82	
102E	99	CMPB 80, #0F
102F	0F	
1030	80	
1031	D7	JNE K
1032	F7	
1033	E7	BRK
1034	F2	
1035	DF	

Το μέγεθος του προγράμματος είναι 18 bytes.

(β)

Εντολή	Φορές Εκτέλεσης
LDB 80, #08	1
LDB 82, #64	1
INCB 80	7
INCB 82	7
CMPB 80, #0F	7
JNE K	7
BRK	1

(γ) Τα περιεχόμενα των δύο καταχωρητών φαίνονται στον παρακάτω πίνακα:

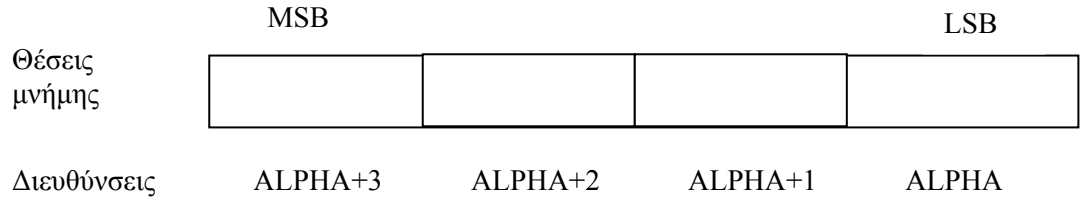
	Περιεχόμενο του καταχωρητή 80	Περιεχόμενο του καταχωρητή 82
Αρχικό	08	64
Τέλος 1ου γύρου	09	65
Τέλος 2ου γύρου	0A	66
Τέλος 3ου γύρου	0B	67
Τέλος 4ου γύρου	0C	68
Τέλος 5ου γύρου	0D	69

Τέλος 6ου γύρου	0E	6A
Τέλος 7ου γύρου	0F	6B

ΑΣΚΗΣΗ 19

Γράψτε πρόγραμμα για τον επεξεργαστή της INTEL 8085 που να προσθέτει δυο δεκαεξαδικούς αριθμούς Α και Β, 8 δεκαεξαδικών ψηφίων.
Ο κάθε αριθμός καταλαμβάνει 4 θέσεις μνήμης καθώς το μήκος λέξης της μνήμης είναι ένα byte. Ο αριθμός Α τοποθετείται στις διευθύνσεις μνήμης ALPHA έως ALPHA+3. Ο αριθμός Β τοποθετείται στις διευθύνσεις μνήμης BETA έως BETA+3. Η τοποθέτηση γίνεται όπως φαίνεται στο παρακάτω σχήμα. Στη διεύθυνση με τη μικρότερη τιμή γράφονται τα ελάχιστης σημαντικότητας ψηφία των αριθμών.

Το αποτέλεσμα της πρόσθεσης να αποθηκευτεί στη θέση μνήμης που καταλαμβάνει ο αριθμός Α. Αγνοείστε την περίπτωση ύπαρξης τελικού κρατουμένου. Επιλέξτε δικές σας τιμές για τα ALPHA, BETA καθώς και για τη θέση του προγράμματος στη μνήμη.



Λύση:

Καθώς ο επεξεργαστής περιέχει αθροιστή 8 δυαδικών ψηφίων η άθροιση των αριθμών θα γίνει τμηματικά. Τα δεκαεξαδικά ψηφία των αριθμών θα προστίθενται ανά δύο (τα δύο του ενός με τα δύο του άλλου). Κατά την πρόσθεση όμως των ψηφίων είναι πιθανό να προκύψει κρατούμενο το οποίο πρέπει να προστεθεί στα επόμενης σημαντικότητας ψηφία. Αυτό γίνεται με το να εκτελείται εντολή άθροισης με χρήση κρατουμένου.

Το πρόγραμμα έχει ως εξής:

```
LXI  D, ALPHA  : Φόρτωσε τη διεύθυνση των λιγότερο σημαντικών ψηφίων του αριθμού Α
                 στους καταχωρητές D, E
LXI  H, BETA   : Φόρτωσε τη διεύθυνση των λιγότερο σημαντικών ψηφίων του αριθμού Β
                 στους καταχωρητές H, L
MVI  C, 04     : Θέσε την τιμή 4 στον καταχωρητή C
XRA  A         : Μηδένισε το κρατούμενο εκτελώντας την πράξη XOR  του A με τον
                 εαυτό του
LOOP: LDAX  D  : Φόρτωσε τον accumulator με τα δυο ψηφία του αριθμού Α
MOV B,M        : Μετέφερε τα αντίστοιχα ψηφία του αριθμού Β στον καταχωρητή Β
ADC  B         : Πρόσθεσε με χρήση κρατουμένου τα αντίστοιχα δυο ψηφία του αριθμού
                 Β με αυτά του Α
STAX  D        : Αποθήκευσε τα δυο ψηφία του αποτελέσματος στη μνήμη
INX            : Αύξησε τους καταχωρητές D,E
INX  H         : Αύξησε τους καταχωρητές H,L
DCR  C         : Ελάττωσε τον καταχωρητή C
JNZ  LOOP      : Είναι C=0; Αν όχι πήγαινε στη διεύθυνση LOOP
HLT            : Σταμάτησε τον επεξεργαστή
```

Σημείωση: Θα μπορούσε να χρησιμοποιηθεί και η εντολή ADC M. Η εντολή αυτή είναι όμοια με την ADC r, με τη διαφορά ότι αντί για τον καταχωρητή r έχουμε θέση μνήμης που καθορίζουν οι καταχωρητές H, L. Στην περίπτωση αυτή οι εντολές MOV B,M και ADC B αντικαθίστανται από την ADC M.

ΑΣΚΗΣΗ 20

Να γραφεί ένα πρόγραμμα σε γλώσσα του μικροϋπολογιστή 8085 που να υπολογίζει το συμπλήρωμα ως προς 2 ενός αριθμού Α αποθηκευμένου στην μνήμη (διεύθυνση $(2030)_H$) και να το αποθηκεύει στην διεύθυνση $(2031)_H$.

Μπορείτε να χρησιμοποιήσετε τις εντολές:
CMA; INR A; INX H; HLT; LXI H,addr; MOV A,M; MOV M,A;

Η χρήση αυτών των εντολών επεξηγείται στο φυλλάδιο ασκήσεων της 4ης ΟΣΣ.

Λύση:

```
LXI H,2030H   ; Φόρτωσε στο ζεύγος καταχωρητών HL τη διεύθυνση 2030H
MOV A,M       ; Φέρε από την μνήμη το δεδομένο στο συσσωρευτή A
CMA           ; Υπολόγισε το συμπλήρωμα του περιεχομένου του A
INR A         ; Αύξησε κατά ένα το περιεχόμενο του A. Έτσι βρήκαμε το
              συμπλήρωμα του α ως προς 2
INX H         ; Αύξησε κατά ένα τη διεύθυνση που περιέχει το ζεύγος HL (δηλαδή
              από 2030 να γίνει 2031)
MOV M,A       ; Μετάφερε το συμπλήρωμα ως προς 2 του α στη διεύθυνση 2031
HLT           ; Σταμάτησε τον επεξεργαστή
```

ΑΣΚΗΣΗ 21

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

Παράδειγμα 1: Δίνεται ο αριθμός 16
16-1=15 1η αφαίρεση
15-3=12 2η αφαίρεση
12-5=7 3η αφαίρεση
7-7=0 4η αφαίρεση
Σύνολο επιτυχών αφαιρέσεων 4 άρα τετραγωνική ριζα του 16=4

Παράδειγμα 2: Δίνεται ο αριθμός 18
18-1= 17 1η αφαίρεση
17-3= 14 2η αφαίρεση
14-5= 9 3η αφαίρεση
9-7= 2 4η αφαίρεση
2-9= -7 5η αφαίρεση

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

Λύση:

```
MVI B,01      ; Το περιεχόμενο του byte 2 (01) της εντολής μετακινείται στον καταχωρητή B
MVI C,00      ; Το περιεχόμενο του byte 2 (00) της εντολής μετακινείται στον καταχωρητή C
LXI H,2030    ; Το byte 3 (20) της εντολής μετακινείται στον καταχωρητή H. Το byte 2 (30) της εντολής μετακινείται στον καταχωρητή L.
MOV A,M       ; Το περιεχόμενο της θέσης μνήμης, της οποίας η διεύθυνση περιέχεται στους καταχωρητές H και L (2030), μετακινείται στον καταχωρητή A
SUB B         ; Το περιεχόμενο του καταχωρητή B αφαιρείται από το περιεχόμενο του συσσωρευτή A και το αποτέλεσμα τοποθετείται στο συσσωρευτή A
JC 2012       ; Εάν η σημαία (flag) carry (κρατούμενο) είναι ενεργοποιημένη (δηλαδή C = 1) τότε ο έλεγχος μεταφέρεται στην εντολή της οποίας η διεύθυνση καθορίζεται στα byte 2 και 3 της τρέχουσας εντολής (2012)
INR B         ; Το περιεχόμενο του καταχωρητή B αυξάνεται κατά ένα
INR B         ; Το περιεχόμενο του καταχωρητή B αυξάνεται κατά ένα
INR C         ; Το περιεχόμενο του καταχωρητή C αυξάνεται κατά ένα
JMP 2008      ; Ο έλεγχος μεταφέρεται στην εντολή της οποίας η διεύθυνση καθορίζεται στα byte 2 και 3 της τρέχουσας εντολής (2008)
INX H         ; Το περιεχόμενο του ζεύγους καταχωρητών H αυξάνεται κατά ένα
MOV M,C       Το περιεχόμενο του καταχωρητή C μετακινείται στη θέσης μνήμης, της οποίας η διεύθυνση περιέχεται στους καταχωρητές H και L (2031)
HLT           ; Ο επεξεργαστής σταματά. Οι καταχωρητές και οι σημαίες μένουν ανεπηρέαστα
```

ΑΣΚΗΣΗ 22

Για το μικροεπεξεργαστή 8085 η εντολή ADD r εκτελεί την πράξη A=A+r όπου r κάποιος από τους άλλους καταχωρητές του επεξεργαστή (πχ B, C, D κλ.π). Δηλαδή προσθέτει το περιεχόμενο του συσσωρευτή A με το περιεχόμενο του καταχωρητή r.

Έστω ότι το περιεχόμενο του καταχωρητή A είναι $7B_{16}$ και τα περιεχόμενα των καταχωρητών B και C είναι $8F_{16}$ και $F3_{16}$. Δώστε τις τιμές των σημαιών κατάστασης του καταχωρητή κατάστασης επεξεργαστή μετά την εκτέλεση κάθε εντολής του παρακάτω προγράμματος.

```
ADD B
ADD C
ADD B
```

Εντολή	Τιμή του καταχ. A	Τιμή του καταχ. B	Τιμή του καταχ. C	Σημαία				
				Zero	Sign	Parity	Carry	Auxil. Carry
	$7B_{16}$ 01111011	$8F_{16}$ 10001111	$F3_{16}$ 11110011					
ADD B								
ADD C								
ADD B								

Λύση:

Εντολή	Τιμή του καταχ. A	Τιμή του καταχ. B	Τιμή του καταχ. C	Σημαία				
				Zero	Sign	Parity	Carry	Auxil. Carry
	$7B_{16}$ 01111011	$8F_{16}$ 10001111	$F3_{16}$ 11110011					
ADD B	$0A_{16}$ 00001010	$8F_{16}$ 10001111	$F3_{16}$ 11110011	0	0	1	1	1
ADD C	FD_{16} 11111101	$8F_{16}$ 10001111	$F3_{16}$ 11110011	0	1	0	0	0
ADD B	$8C_{16}$ 10001100	$8F_{16}$ 10001111	$F3_{16}$ 11110011	0	1	0	1	1

ΑΣΚΗΣΗ 23

Να γραφεί πρόγραμμα σε γλώσσα του μικροεπεξεργαστή 8085 που να ανιχνεύει το μέγιστο μεταξύ είκοσι μη ίσων θετικών αριθμών που βρίσκονται αποθηκευμένοι στις διευθύνσεις μνήμης 2000 έως 2013 και να τον αποθηκεύει στη θέση μνήμης με διεύθυνση 3000. Οι διευθύνσεις μνήμης δίνονται σε δεκαεξαδική μορφή.

Λύση:
Σύμφωνα με τον αλγόριθμο που ακολουθείται οι αριθμοί συγκρίνονται ανά δυο και ο μεγαλύτερος από αυτούς, αφού γραφτεί στον καταχωρητή A, συγκρίνεται με τον επόμενο. Στο τέλος της διαδικασίας ο μέγιστος αριθμός θα βρίσκεται στον A και συνεπώς θα μπορεί να εγγραφεί στη διεύθυνση 3000 της μνήμης. Το πρόγραμμα περιλαμβάνει ένα βρόχο που εκτελείται 19 (ή 13 στο δεκαεξαδικό) φορές. Ο καταχωρητής C χρησιμοποιείται για την αποθήκευση του αριθμού αυτού. Σε κάθε επανάληψη το περιεχόμενο του καταχωρητή C μειώνεται κατά ένα. Η διαδικασία συνεχίζεται έως ότου ο καταχωρητής C να μηδενιστεί.

Το πρόγραμμα ξεκινά από τη θέση μνήμης 0000 και έχει ως εξής:

0000: MVI C 13	Φόρτωσε τον καταχωρητή C με το $(13)_{16}=(19)_{10}$
LDA 2000	Φόρτωσε τον accumulator (καταχωρητή A) με την τιμή της θέσης μνήμης με διεύθυνση 2000
LXI H 2001	Φόρτωσε τους καταχωρητές H, L με τη τιμή 2001
0008: CMP M	Αφαίρεσε από τον A τον αριθμό M που είναι αποθηκευμένος στη θέση μνήμης με διεύθυνση το περιεχόμενο των H, L
JNC 000D	Αν A > M (CY=0) πήγαινε στη θέση 000D, διαφορετικά προχώρα στην επόμενη εντολή
MOV A M	Φόρτωσε στον A τον αριθμό M
000D: INX H	Αύξησε την τιμή των καταχωρητών H,L κατά ένα

DCR C	Μείωσε την τιμή του καταχωρητή C κατά ένα
JNZ 0008	Αν C≠0 πήγαινε στη θέση 0008, διαφορετικά προχώρα
STA 3000	Αποθήκευσε την τιμή του A στη θέση μνήμης 3000
HLT	Σταμάτησε τον επεξεργαστή.

ΑΣΚΗΣΗ 24

Για το μικροεπεξεργαστή 8080 η εντολή ADD B εκτελεί την πράξη A=A+B. Δηλαδή προσθέτει το περιεχόμενο του συσσωρευτή A με το περιεχόμενο του καταχωρητή B. Έστω ότι το περιεχόμενο του καταχωρητή A είναι $7A_{16}$ και τα περιεχόμενα των καταχωρητών B και C είναι $9F_{16}$ και $E7_{16}$. Δώστε τις τιμές των σημαιών κατάστασης του καταχωρητή κατάστασης επεξεργαστή μετά την εκτέλεση κάθε εντολής του παρακάτω προγράμματος.

ADD B
ADD C
ADD B

Εντολή	Τιμή του καταχ. A	Τιμή του καταχ. B	Τιμή του καταχ. C	Σημαία				
				Zero	Sign	Parity	Carry	Auxil. Carry
	$7A_{16}$ 01111010	$9F_{16}$ 10011111	$E7_{16}$ 11100111					
ADD B								
ADD C								
ADD B								

Λύση:

Εντολή	Τιμή του καταχ. A	Τιμή του καταχ. B	Τιμή του καταχ. C	Σημαία				
				Zero	Sign	Parity	Carry	Auxil. Carry
	$7A_{16}$ 01111010	$9F_{16}$ 10011111	$E7_{16}$ 11100111					
ADD B	19_{16} 00011001	$9F_{16}$ 1001111	$E7_{16}$ 11100111	0	0	0	1	1
ADD C	00_{16} 00000000	$9F_{16}$ 1001111	$E7_{16}$ 11100111	1	0	1	1	1
ADD B	$9F_{16}$ 10011111	$9F_{16}$ 1001111	$E7_{16}$ 11100111	0	1	1	0	0

ΑΣΚΗΣΗ 25

Διαθέτετε επεξεργαστή που έχει τις παρακάτω εντολές

Όνομα εντολής	Κώδικας λειτουργίας	Τρόπος διευθυνσιοδότησης
ADA	AB	Άμεσος
	BB	Απευθείας/κατ' ευθείαν
	CB	Μηδενικής σελίδας
	DB	Σχετικός
STA	CC	Απευθείας/κατ' ευθείαν
	DC	Σχετικός
	EC	Δεικτοδοτημένος

Η εντολή ADA προσθέτει στο περιεχόμενο του καταχωρητή Α το δεδομένο που βρίσκεται στη θέση που δηλώνει ο τρόπος διευθυνσιοδότησης. Το αποτέλεσμα τίθεται στον καταχωρητή Α. Η εντολή STA αποθηκεύει το περιεχόμενο του καταχωρητή Α στη θέση μνήμης που δηλώνεται από τον τρόπο διευθυνσιοδότησης.

Θεωρείστε ότι ο καταχωρητής Α περιέχει την τιμή 15_{16} . Η θέση μνήμης με διεύθυνση 2000_{16} την τιμή $3D_{16}$, η θέση μνήμης με διεύθυνση 1022_{16} την τιμή $9A_{16}$, η θέση μνήμης με διεύθυνση $00B3_{16}$ την τιμή 46_{16} και ο καταχωρητής δείκτη Χ την τιμή $104A_{16}$.

Για τα παρακάτω μέρη προγράμματος να δοθεί η διεύθυνση μνήμης όπου θα εγγραφεί το αποτέλεσμα καθώς επίσης και η τιμή που θα εγγραφεί.

Α)
1000 ADA $2000 (κωδικός λειτουργίας BB)
1003 STA $3000 (κωδικός λειτουργίας CC)

Β)
1000 ADA $B3 (κωδικός λειτουργίας CB)
1002 STA $54,X (κωδικός λειτουργίας EC)

Γ)
1000 ADA #25 (κωδικός λειτουργίας AB)
1002 STA $10 (κωδικός λειτουργίας DC)
1004 NOP

Δ)
1000 ADA $20 (κωδικός λειτουργίας DB)
1002 STA $30,X (κωδικός λειτουργίας EC)

Λύση
Α)
Η εντολή «ADA $2000 (κωδικός λειτουργίας BB)» είναι απευθείας διευθυνσιοδότησης και επομένως στην τιμή του καταχωρητή Α προστίθεται το περιεχόμενο της θέσης μνήμης με διεύθυνση 2000_{16}, δηλαδή $A = 15_{16} + 3D_{16} = 52_{16}$.
Καθώς η εντολή «STA $3000 (κωδικός λειτουργίας CC)» είναι απευθείας διευθυνσιοδότησης η τιμή 52_{16} θα γραφεί στη θέση μνήμης με διεύθυνση 3000_{16}.

Β)

Η εντολή «ADA $B3 (κωδικός λειτουργίας CB)» είναι τρόπου διευθυνσιοδότησης μηδενικής σελίδας και επομένως στην τιμή του καταχωρητή Α προστίθεται το περιεχόμενο της θέσης μνήμης με διεύθυνση $00B3_{16}$, δηλαδή $A = 15_{16} + 46_{16} = 5B_{16}$.
Καθώς η εντολή «STA $54,X (κωδικός λειτουργίας EC)» είναι δεικτοδοτημένης διευθυνσιοδότησης η ενεργός διεύθυνση της θέσης μνήμης όπου θα αποθηκευτεί το αποτέλεσμα προκύπτει από το άθροισμα του περιεχομένου του καταχωρητή δείκτη με το 54_{16}. Συνεπώς η ενεργός διεύθυνση είναι $104A_{16} + 54_{16} = 109E_{16}$.
Άρα στη θέση μνήμης με διεύθυνση $109E_{16}$ θα γραφεί η τιμή $5B_{16}$.

Γ)
Η εντολή «ADA #25 (κωδικός λειτουργίας AB)» είναι άμεσης διευθυνσιοδότησης και συνεπώς στην τιμή του καταχωρητή Α θα προστεθεί το 25_{16}, $A = 15_{16} + 25_{16} = 3A_{16}$.
Καθώς η εντολή «STA $10 (κωδικός λειτουργίας DC)» είναι σχετικής διευθυνσιοδότησης η ενεργός διεύθυνση προκύπτει από το άθροισμα της τιμής του μετρητή προγράμματος στην τιμή 10_{16}. Όταν εκτελείται η εντολή ο μετρητής προγράμματος δείχνει την επόμενη εντολή και συνεπώς έχει τιμή 1004_{16}. Επομένως η ενεργός διεύθυνση είναι $1004_{16} + 10_{16} = 1014_{16}$.
Άρα στη θέση μνήμης με διεύθυνση 1014_{16} θα γραφεί η τιμή $3A_{16}$.

Δ)
Η εντολή «ADA $20 (κωδικός λειτουργίας DB)» είναι σχετικής διευθυνσιοδότησης και συνεπώς στην τιμή του καταχωρητή Α θα προστεθεί το περιεχόμενο της μνήμης που βρίσκεται στη διεύθυνση $1002_{16} + 20_{16} = 1022_{16}$. Στη διεύθυνση 1022_{16} υπάρχει η τιμή $9A_{16}$.
Άρα $A = 15_{16} + 9A_{16} = AF_{16}$.
Καθώς η εντολή «STA $30,X (κωδικός λειτουργίας EC)» είναι δεικτοδοτημένης διευθυνσιοδότησης η ενεργός διεύθυνση προκύπτει ως $104A_{16} + 30_{16} = 107A_{16}$.
Άρα στη θέση μνήμης με διεύθυνση $107A_{16}$ θα γραφεί η τιμή AF_{16}.

ΑΣΚΗΣΗ 26

Δίνεται το παρακάτω πρόγραμμα το οποίο έχει γραφτεί για τον μικροεπεξεργαστή 8085 της Intel.

```
   LXI H,2000
   MVI B,9A
   MVI C,1E
K: MOV A,B
   INX H
   SUB C
   MOV M,A
   MOV B,A
   JNC K
   HLT
```

Α) Υποθέστε ότι το πρόγραμμα θα αποθηκευτεί στην κύρια μνήμη ξεκινώντας από τη θέση 1000Η. Δώστε στον παρακάτω πίνακα τα περιεχόμενα της μνήμης από τη διεύθυνση 1000Η και ως το τέλος του προγράμματος. Τι αλγόριθμος εκτελείται; Πόσες φορές εκτελείται ο βρόχος;

Διεύθυνση Μνήμης	**Περιεχόμενο Μνήμης**	**Εντολή**
1000		
1001		
1002		
. . .	. . .	. . .

Β) Να συμπληρώσετε στον παρακάτω πίνακα τις διευθύνσεις μνήμης στις οποίες το πρόγραμμα πραγματοποιεί εγγραφή καθώς επίσης και τα περιεχόμενα αυτών.

Διεύθυνση Μνήμης	**Περιεχόμενο Μνήμης**
...	...

Γ) Όπως θα παρατηρήσετε στις εγγραφές στη μνήμη εμφανίζεται μία αρνητική τιμή (το πλέον σημαντικό δυαδικό ψηφίο του αριθμού είναι 1). Πραγματοποιείστε μία μικρή τροποποίηση στο παραπάνω πρόγραμμα ώστε να αποφευχθεί η εγγραφή του αριθμού αυτού στη μνήμη χωρίς να επηρεαστεί κανένα άλλο αποτέλεσμα.

Λύση
Α)

Διεύθυνση Μνήμης	**Περιεχόμενο Μνήμης**	**Εντολή**
1000	21	LXI H,2000
1001	00	
1002	20	
1003	06	MVI B,9A
1004	9A	
1005	0E	MVI C,1E
1006	1E	
1007	78	MOV A,B
1008	23	INX H
1009	91	SUB C
100A	77	MOV M,A
100B	47	MOV B,A
100C	D2	JNC 1007
100D	07	
100E	10	
100F	76	HLT

Το πρόγραμμα αφαιρεί επαναληπτικά την τιμή 1ΕΗ που βρίσκεται στον καταχωρητή C από την τιμή στον καταχωρητή B. Ο βρόχος εκτελείται 6 φορές, έως ότου εμφανιστεί αρνητική τιμή μετά από τις διαδοχικές αφαιρέσεις.

Β)

Διεύθυνση Μνήμης	**Περιεχόμενο Μνήμης**
2000	
2001	7C
2002	5E
2003	40
2004	22
2005	04
2006	E6

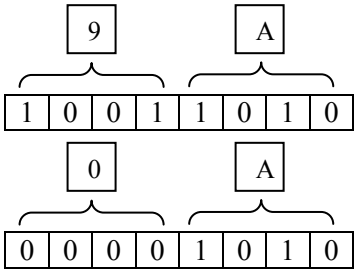
Γ) Ο αρνητικός αριθμός εμφανίζεται γιατί μετά την αφαίρεση πρώτα γίνεται εγγραφή του αποτελέσματος στη μνήμη και κατόπιν ελέγχεται το πρόσημο του αποτελέσματος (με την εντολή JNC). Η εγγραφή του αρνητικού αριθμού θα αποφευχθεί αν εισάγουμε μία εντολή ελέγχου του πρόσημου της πράξης (JC) πριν την εγγραφή στη μνήμη. Κάνοντας αυτό, αφού η νέα εντολή αποτελεί την εντολή εξαγωγής από το βρόχο η JNC εντολή του αρχικού προγράμματος μετατρέπεται σε εντολή διακλάδωσης χωρίς συνθήκη, JMP.

Το νέο πρόγραμμα έχει ως εξής:

```
8085 Machine Code File
1000 LXI H 2000   21 ;        Φόρτωσε στο ζεύγος καταχωρητών H, L την τιμή 2000H
1001              00 ;
1002              20 ;
1003 MVI B 9A     06 ;        Φόρτωσε στον καταχωρητή B την τιμή 9A
1004              9A ;
1005 MVI C 1E     0E ;        Φόρτωσε στον καταχωρητή C την τιμή 1E
1006              1E ;
1007 MOV A B      78 ;        Φόρτωσε στο συσσωρευτή A την τιμή του B
1008 INX H        23 ;        Αύξησε κατά 1 την τιμή του ζεύγους H, L
1009 SUB C        91 ;        Αφαίρεσε τη τιμή του C από τον A
100A JC  1012     DA ;        Αν υπάρχει κρατούμενο (A<C) μετέφερε το πρόγραμμα
                              στη διεύθυνση 1012
100B              12 ;
100C              10 ;
100D MOV M A      77 ;        Φόρτωσε την τιμή του A στη θέση μνήμης με διεύθυνση
                              την τιμή του ζεύγους H, L
100E MOV B A      47 ;        Φόρτωσε στον καταχωρητή B την τιμή του A
100F JMP 1007     C3 ;        Μετέφερε το πρόγραμμα στη διεύθυνση 1007
1010              07 ;
1011              10 ;
1012 HLT          76 ;        Σταμάτησε την εκτέλεση του προγράμματος
Data
2001 7C ;
2002 5E ;
2003 40 ;
2004 22 ;
2005 04 ;
```

ΑΣΚΗΣΗ 27

Θεωρείστε ότι σε μια περιοχή μνήμης 16 θέσεων που ξεκινούν από τη διεύθυνση 2000H είναι αποθηκευμένοι ανά ζεύγη θετικοί δυαδικοί αριθμοί των 4 ψηφίων οι οποίοι δεν είναι ίσοι. Κάθε ζεύγος καταλαμβάνει μία θέση μνήμης των 8 ψηφίων (ένα byte), έτσι ώστε ο ένας αριθμός να καταλαμβάνει τα 4 πλέον σημαντικά ψηφία της λέξης και ο άλλος τα υπόλοιπα. Γράψτε πρόγραμμα σε assembly του 8085 που να μηδενίζει σε κάθε ένα από τα παραπάνω ζεύγη (bytes) τον μικρότερο από τους δυο αριθμούς του ζεύγους.
(Π.χ. Αν πριν την εκτέλεση του προγράμματος στη θέση μνήμης με διεύθυνση 2000H ήταν αποθηκευμένοι οι αριθμοί 9 και A τότε μετά την εκτέλεσή του θα πρέπει να είναι αποθηκευμένοι οι αριθμοί 0 και A, όπως φαίνεται στο σχήμα.)



*Περιγράφονται οι παρακάτω εντολές που πιθανόν να χρειαστούν στην υλοποίηση του προγράμματος της άσκησης αυτής ή άλλων και δεν βρίσκονται στο υποστηρικτικό υλικό που σας δόθηκε:

ANI byte : Πράξη AND του περιεχομένου του συσσωρευτή A με την τιμή του byte, ψηφίο προς ψηφίο. Το αποτέλεσμα τοποθετείται στον A.

CMP r : Σύγκριση περιεχόμενου του A με αυτό του καταχωρητή r. Εκτελείται η πράξη (A)-(r) αλλά το περιεχόμενο του A μένει ανεπηρέαστο. Αν (A)=(r) τότε γίνεται Z=1. Αν (A)<(r) τότε γίνεται C=1.

CPI byte : Σύγκριση περιεχόμενου του A με αυτό του byte. Εκτελείται η πράξη (A)-(byte) αλλά το περιεχόμενο του A μένει ανεπηρέαστο. Αν (A)=(byte) τότε γίνεται Z=1. Αν (A)<(byte) τότε γίνεται C=1.

Λύση

Μεταφέρουμε το περιεχόμενο κάθε θέσης μνήμης στον συσσωρευτή (A) και εκτελώντας λογική AND με τους αριθμούς 0F και F0 (μάσκες) αποκαλύπτουμε τους δυο τετραψήφιους αριθμούς και τους αποθηκεύουμε στους καταχωρητές B και C, αντίστοιχα. Συνεπώς στον καταχωρητή B ο αριθμός καταλαμβάνει τα τέσσερα λιγότερο σημαντικά ψηφία ενώ τα υπόλοιπα είναι 0. Στον καταχωρητή C ο αριθμός καταλαμβάνει τα τέσσερα περισσότερο σημαντικά ψηφία ενώ τα υπόλοιπα είναι 0. Κατόπιν ολισθαίνουμε προς τα δεξιά (λογική ολίσθηση) τον αριθμό που είναι στον καταχωρητή C κατά τέσσερα ψηφία ώστε να μπορεί να πραγματοποιηθεί σύγκριση με αυτόν στον καταχωρητή B. Μετά τη σύγκριση ο μεγαλύτερος, στη μορφή που είχε αρχικά αποθηκευτεί στον B ή στον C, τοποθετείται στην αντίστοιχη θέση μνήμης. Η διαδικασία επαναλαμβάνεται και για τους 16 (10_{16}) αριθμούς.

Ένα ενδεικτικό πρόγραμμα για το παραπάνω πρόβλημα έχει ως εξής:

```
8085 Machine Code File
0000 LXI H 2000  21 ;        Φόρτωσε στο ζεύγος καταχωρητών H, L την τιμή 2000H
0001             00 ;
0002             20 ;
0003 MVI D 10    16 ;        Φόρτωσε στον καταχωρητή D την τιμή 10H
0004             10 ;
0005 MOV A M     7E ;        Φόρτωσε την τιμή της θέσης μνήμης με διεύθυνση
                             την τιμή του ζεύγους H, L στον συσσωρευτή A
0006 ANI 0F      E6 ;        Πράξη AND του A με το 0F και το αποτέλεσμα στον A
0007             0F ;
0008 MOV B A     47 ;        Φόρτωσε στον καταχωρητή B την τιμή του A
0009 MOV A M     7E ;        Φόρτωσε την τιμή της θέσης μνήμης με διεύθυνση
                             την τιμή του ζεύγους H, L στον συσσωρευτή A
000A ANI F0      E6 ;        Πράξη AND του A με το F0 και το αποτέλεσμα στον A
000B             F0 ;
000C MOV C A     4F ;        Φόρτωσε στον καταχωρητή C την τιμή του A
000D RRC         0F ;        Λογική ολίσθηση του A προς τα δεξιά κατά 1 ψηφίο
000E RRC         0F ;        Λογική ολίσθηση του A προς τα δεξιά κατά 1 ψηφίο
```

```
000F RRC          0F ;      Λογική ολίσθηση του A προς τα δεξιά κατά 1 ψηφίο
0010 RRC          0F ;      Λογική ολίσθηση του A προς τα δεξιά κατά 1 ψηφίο
0011 CMP B        B8 ;      Σύγκριση των περιεχομένων του A και του B
0012 JNC 0019     D2 ;      Εάν δεν υπάρχει κρατούμενο (A>B) μετέφερε το
                            πρόγραμμα στη διεύθυνση 0019
0013              19 ;
0014              00 ;
0015 MOV M B      70 ;      Φόρτωσε την τιμή του B στη θέση μνήμης με διεύθυνση
                            την τιμή του ζεύγους H, L
0016 JMP 001A     C3 ;      Μετέφερε το πρόγραμμα στη διεύθυνση 001A
0017              1A ;
0018              00 ;
0019 MOV M C      71 ;      Φόρτωσε την τιμή του C στη θέση μνήμης με διεύθυνση
                            την τιμή του ζεύγους H, L
001A INX H        23 ;      Αύξησε κατά 1 την τιμή του ζεύγους H, L
001B DCR D        15 ;      Μείωσε κατά 1 την τιμή του D
001C JNZ 0005     C2 ;      Εάν η τιμή στον D δεν είναι μηδενική μετέφερε το
                            πρόγραμμα στη διεύθυνση 0005
001D              05 ;
001E              00 ;
001F HLT          76 ;      Σταμάτησε την εκτέλεση του προγράμματος
```

ΑΣΚΗΣΗ 28

Στην 8-δυαδικών ψηφίων έξοδο μιας θύρας επεξεργαστή Intel 8085 με διεύθυνση 10H έχει τοποθετηθεί αντίστοιχος αριθμός φωτεινών στοιχείων (led), ένα ανά ψηφίο. Τα στοιχεία αυτά εκπέμπουν όταν το αντίστοιχο ψηφίο της εξόδου έχει τιμή «1». Γράψτε πρόγραμμα ώστε τα στοιχεία να εκπέμπουν το ένα μετά το άλλο ξεκινώντας από το λιγότερο σημαντικό ψηφίο προς το περισσότερο σημαντικό. Ο χρόνος παραμονής ενός στοιχείου σε εκπομπή να είναι **περίπου** 30ms. Αφού φωταγωγήσει και το 8^{ο} led η διαδικασία να σταματήσει. Θεωρείστε ότι ο επεξεργαστής λειτουργεί σε συχνότητα 100KHz (κύκλος ρολογιού 10μs).

Υπόδειξη: Για τη επίτευξη του χρόνου παραμονής δημιουργείστε βρόχο καθυστέρησης πχ. μειώνοντας την τιμή ενός καταχωρητή. Πρέπει να βρείτε την κατάλληλη αρχική τιμή του καταχωρητή(Οι κύκλοι ρολογιού που απαιτούνται για την εκτέλεση ενός προγράμματος εμφανίζονται στον προσομοιωτή ως “States”) Οι θύρες εξόδου ορίζονται από την επιλογή “Options/port options”.

Λύση

Απαιτείται αρχικά ο καθορισμός ενός τρόπου για την επίτευξη της επιθυμητής διάρκειας εκπομπής κάθε στοιχείου. Αυτό μπορεί εύκολα να γίνει με ένα βρόχο ελεγχόμενου αριθμού εκτελέσεων. Ένας τρόπος για την πραγματοποίησή του είναι να θέσουμε έναν καταχωρητή σε μια αρχική τιμή και κατόπιν να τον μειώνουμε κατά 1 έως ώσπου να μηδενιστεί. Η αρχική τιμή του καταχωρητή καθορίζει τον αριθμό επανάληψης των εντολών του βρόχου και συνεπώς τη συνολική καθυστέρηση εκτέλεσης.

Παράδειγμα υλοποίησης αντίστοιχου βρόχου:

```
  MVI D N     : θέτουμε στον καταχωρητή D αρχική τιμή N
K: DCR D      : μειώνουμε τον D κατά 1
  JNZ K       : εάν δεν έχει μηδενιστεί ο D το πρόγραμμα επανέρχεται στην προηγούμενη
                εντολή.
```

Ο παραπάνω βρόχος των δυο εντολών DCR, JNZ εκτελείται τόσες φορές όσες αντιστοιχούν στην τιμή του *N*. Αν ξέρουμε πόσοι κύκλοι ρολογιού χρειάζονται για την εκτέλεση των δύο αυτών εντολών μπορούμε να υπολογίσουμε τον αριθμό *N* που απαιτείται για την επίτευξη συγκεκριμένης καθυστέρησης. Τον αριθμό των κύκλων τον παίρνουμε από την ένδειξη "States" του προσομοιωτή. Για τις συγκεκριμένες εντολές είναι 15. Αφού ο κύκλος ρολογιού διαρκεί 10μs η κάθε μία εκτέλεση των εντολών του βρόχου αντιστοιχεί σε 150μs. Επομένως για την επίτευξη καθυστέρησης 30ms θα πρέπει *N*=200 ή C8H.

Στην πραγματικότητα η συνολική διάρκεια εκπομπής θα είναι λίγο μεγαλύτερη λόγω της παρουσίας των άλλων εντολών του προγράμματος όπως π.χ. της αρχικοποίησης του καταχωρητή και της εκτέλεσης της λογικής του αλγορίθμου (ολίσθηση). Μπορούν να ληφθούν υπόψη και οι καθυστερήσεις των εντολών αυτών στην προσπάθεια επίτευξης μεγαλύτερης ακρίβειας στο χρόνο εκπομπής αλλά μια τέτοια προσπάθεια θα ήταν υπερβολική. Μόνο όταν απαιτείται από τις προδιαγραφές μιας εφαρμογής μεγάλη ακρίβεια τότε θα πρέπει να έχουμε αντίστοιχη αντιμετώπιση.

Το πρόγραμμα έχει ως εξής:

```
8085 Machine Code File
0000 MVI A 01    3E ;        Φόρτωσε στον συσσωρευτή A την τιμή 01
0001             01 ;
0002 MVI D C8    16 ;        Φόρτωσε στον καταχωρητή D την αρχική τιμή C8
0003             C8 ;
0004 OUT 10      D3 ;        Γράψε στη θύρα 10 την τιμή του A
0005             10 ;
0006 DCR D       15 ;        Μείωσε κατά 1 την τιμή του D
0007 JNZ 0006    C2 ;        Αν D≠0 πήγαινε στην διεύθυνση προγράμματος 0006,
                             << βρόχος υλοποίηση καθυστέρησης>>
0008             06 ;
0009             00 ;
000A CPI 80      FE ;        Σύγκριση της τιμής του A με το 80
000B             80 ;
000C JZ  0013    CA ;        Αν A=80 μετέφερε το πρόγραμμα στη διεύθυνση 0013
000D             13 ;
000E             00 ;
000F RLC         07 ;        Ολίσθηση προς τα αριστερά του περιεχομένου του A
0010 JMP 0002    C3 ;        Μετέφερε το πρόγραμμα στη διεύθυνση 0002
0011             02 ;
0012             00 ;
0013 HLT         76 ;        Σταμάτησε την εκτέλεση του προγράμματος
```

Το παραπάνω πρόγραμμα εκτελείται σε 24330 κύκλους ρολογιού και συνεπώς κάθε led θα φωτοβολεί για (24330/8)x10μs≈30,41ms.

ΑΣΚΗΣΗ 29

Θεωρείστε ότι σε μια περιοχή μνήμης 15 θέσεων που ξεκινούν από τη διεύθυνση 2000H είναι αποθηκευμένοι μη προσημασμένοι δυαδικοί αριθμοί των 8 ψηφίων. Γράψτε πρόγραμμα σε συμβολική γλώσσα (assembly) του 8085 που να αποθηκεύει μόνο τους περιττούς από αυτούς, με τη σειρά που ανιχνεύονται, σε περιοχή μνήμης που ξεκινά από τη διεύθυνση 3000H.
Π.χ.

Διεύθυνση	Περιεχόμενο
2000	A9
2001	24
2002	20
2003	23
2004	11
…	….

Διεύθυνση	Περιεχόμενο
3000	A9
3001	23
3002	11
…	…

Λύση

Για την υλοποίηση ενός τέτοιου προγράμματος απαιτείται έλεγχος για το αν ένας αριθμός είναι περιττός ή άρτιος. Αυτό μπορεί να γίνει με χρήση της εντολής RRC που ολισθαίνει το πλέον δεξιό ψηφίο του αριθμού, που είναι αποθηκευμένος στον καταχωρητή Α, στη σημαία CY ώστε να μπορεί να γίνει χρήση της τιμής της σημαίας αυτής για αλλαγή ροής προγράμματος. Έτσι αν η τιμή του CY είναι «1» γράφεται ο περιττός αριθμός στην κατάλληλη διεύθυνση και αυξάνεται κατά 1 η διεύθυνση που απευθύνεται στην αρχική λίστα των αριθμών καθώς και αυτής της νέας λίστας των αριθμών. Αν η τιμή του CY είναι «0» δεν γίνεται εγγραφή στη νέα λίστα και επίσης δεν αυξάνεται η αντίστοιχη διεύθυνση. Η διαδικασία αυτή επαναλαμβάνεται για 15 φορές ώστε να ελεγχθούν όλοι οι αριθμοί της αρχικής λίστας.

Μια ενδεικτική υλοποίηση είναι η παρακάτω:

```
8085 Machine Code File
0000 LXI H 2000  21 ;
0001             00 ;
0002             20 ;
0003 LXI D 3000  11 ;
0004             00 ;
0005             30 ;
0006 MVI B 0F    06 ;
0007             0F ;
0008 MOV A M     7E ;
0009 RRC         0F ;
000A JNC 0010    D2 ;
000B             10 ;
000C             00 ;
000D MOV A M     7E ;
000E STAX D      12 ;
000F INX D       13 ;
0010 INX H       23 ;
0011 DCR B       05 ;
0012 JNZ 0008    C2 ;
0013             08 ;
0014             00 ;
0015 HLT         76 ;
```

ΑΣΚΗΣΗ 30

Θεωρείστε ότι σε μια περιοχή μνήμης 15 θέσεων που ξεκινούν από τη διεύθυνση 2000H είναι αποθηκευμένοι προσημασμένοι δυαδικοί αριθμοί των 8 ψηφίων σε αναπαράσταση συμπληρώματος ως προς 2. Γράψτε πρόγραμμα σε assembly του 8085 που να αντικαθιστά κάθε έναν από τους παραπάνω αριθμούς με την απόλυτη τιμή του.
Π.χ.

Διεύθυνση (πριν την εκτέλεση του προγράμματος)	Περιεχόμενο
2000	A9
2001	EE
2002	23
2003	11
2004	98
…	….

Διεύθυνση (μετά την εκτέλεση του προγράμματος)	Περιεχόμενο
2000	57
2001	12
2002	23
2003	11
2004	68
…	

Λύση
Για την υλοποίηση του προγράμματος θα πρέπει να ανιχνευθούν οι αρνητικοί αριθμοί της λίστας και να μετατραπούν σε θετικούς. Η ανίχνευση του πρόσημου ενός αριθμού μπορεί να πραγματοποιηθεί με την εντολή RLC που ολισθαίνει το πλέον αριστερό ψηφίο (πρόσημο) του αριθμού στον καταχωρητή A στη σημαία CY ώστε να μπορεί να γίνει χρήση της τιμής της σημαίας αυτής για αλλαγή ροής προγράμματος. Έτσι αν η τιμή του CY είναι «1» ο αριθμός είναι αρνητικός και αντιστρέφεται με την εντολή CMA (αντιστρέφει την τιμή κάθε ψηφίου) ώστε να προκύψει το συμπλήρωμα ως προς 1 του αριθμού στο οποίο προστίθεται η μονάδα ώστε να προκύψει η αντίστοιχη θετική τιμή η οποία και εγγράφεται στη μνήμη. Αν ο αριθμός είναι θετικός δεν πραγματοποιείται καμιά ενέργεια σε αυτόν. Η παραπάνω διαδικασία επαναλαμβάνεται για 15 φορές ώστε να ελεγχθούν όλοι οι αριθμοί της αρχικής λίστας.

Μια ενδεικτική υλοποίηση είναι η παρακάτω:

```
8085 Machine Code File
0000 LXI H 2000  21 ;
0001             00 ;
0002             20 ;
0003 MVI B 0F    06 ;
0004             0F ;
0005 MOV A M     7E ;
0006 RLC         07 ;
0007 JNC 000E    D2 ;
0008             0E ;
0009             00 ;
000A MOV A M     7E ;
000B CMA         2F ;
000C INR A       3C ;
000D MOV M A     77 ;
000E INX H       23 ;
000F DCR B       05 ;
0010 JNZ 0005    C2 ;
0011             05 ;
0012             00 ;
0013 HLT         76 ;
```

ΑΣΚΗΣΗ 31

Διαθέτετε επεξεργαστή που έχει τις παρακάτω εντολές

Όνομα εντολής	Κωδικός λειτουργίας	Τρόπος διευθυνσιοδότησης
ADA	AA	Άμεσος
	AB	Απευθείας/κατ' ευθείαν
	AC	Μηδενικής σελίδας
	AD	Σχετικός
STA	BB	Απευθείας/κατ' ευθείαν
	BD	Σχετικός
	BE	Δεικτοδοτημένος
LDA	CB	Απευθείας/κατ' ευθείαν
	CD	Σχετικός
	CE	Δεικτοδοτημένος

Η εντολή ADA προσθέτει στο περιεχόμενο του καταχωρητή Α το δεδομένο που βρίσκεται στη θέση που προκύπτει από τον τρόπο διευθυνσιοδότησης. Το αποτέλεσμα αποθηκεύεται στον καταχωρητή Α. Η εντολή STA αποθηκεύει το περιεχόμενο του καταχωρητή Α στη θέση μνήμης που προκύπτει από τον τρόπο διευθυνσιοδότησης. Η εντολή LDA αποθηκεύει στον καταχωρητή Α το περιεχόμενο της θέσης μνήμης που προκύπτει από τον τρόπο διευθυνσιοδότησης

Θεωρείστε ότι η θέση μνήμης με διεύθυνση 2000_{16} περιέχει την τιμή $5C_{16}$, η θέση μνήμης με διεύθυνση 1022_{16} περιέχει την τιμή $3B_{16}$, η θέση μνήμης με διεύθυνση $006B_{16}$ την τιμή $A1_{16}$, η θέση μνήμης με διεύθυνση 2114_{16} την τιμή $1A_{16}$ και ο καταχωρητής δείκτη Χ την τιμή $20F4_{16}$.

Για τα παρακάτω μέρη προγράμματος να δοθεί η διεύθυνση μνήμης όπου θα εγγραφεί το αποτέλεσμα καθώς επίσης και η τιμή που θα εγγραφεί. Εξηγείστε το σκεπτικό σας.

Α)
1000 LDA $2000 (κωδικός λειτουργίας CB)
1003 ADA $6B (κωδικός λειτουργίας AC)
1005 STA $74, X (κωδικός λειτουργίας BE)
1007 STOP

Β)
1000 LDA $20 (κωδικός λειτουργίας CD)
1002 ADA $2000 (κωδικός λειτουργίας AB)
1005 STA $2500 (κωδικός λειτουργίας BB)
1008 STOP

Γ)
1000 LDA $20, X (κωδικός λειτουργίας CE)
1002 ADA #33 (κωδικός λειτουργίας AA)
1004 STA $55 (κωδικός λειτουργίας BD)
1006 STOP

Λύση
Α)

Η εντολή «LDA $2000 (κωδικός λειτουργίας CB)» είναι απευθείας διευθυνσιοδότησης και επομένως στον καταχωρητή Α αποθηκεύεται το περιεχόμενο της θέσης μνήμης με διεύθυνση 2000_{16}, δηλαδή $A = 5C_{16}$.
Η εντολή «ADA $6B (κωδικός λειτουργίας AC)» αντιστοιχεί σε διευθυνσιοδότηση μηδενικής σελίδας και επομένως στην τιμή του καταχωρητή Α προστίθεται το περιεχόμενο της θέσης μνήμης με διεύθυνση $006B_{16}$, δηλαδή $A = 5C_{16} + A1_{16} = FD_{16}$.
Η εντολή «STA $74, X (κωδικός λειτουργίας BE)» είναι δεικτοδοτούμενης διευθυνσιοδότησης και συνεπώς η ενεργός διεύθυνση της θέσης μνήμης όπου θα αποθηκευτεί το αποτέλεσμα προκύπτει από το άθροισμα του περιεχομένου του καταχωρητή δείκτη Χ με το 74_{16}. Άρα η ενεργός διεύθυνση είναι $20F4_{16} + 74_{16} = 2168_{16}$ όπου γράφεται η τιμή FD_{16}.

Β)
Καθώς η εντολή «LDA $20 (κωδικός λειτουργίας CD)» είναι σχετικής διευθυνσιοδότησης, η ενεργός διεύθυνση προκύπτει από το άθροισμα της τιμής του μετρητή προγράμματος στην τιμή 20_{16}. Όταν εκτελείται η εντολή ο μετρητής προγράμματος δείχνει την επόμενη εντολή και συνεπώς έχει τιμή 1002_{16}. Επομένως, η ενεργός διεύθυνση είναι $1002_{16} + 20_{16} = 1022_{16}$ και στον καταχωρητή Α αποθηκεύεται η τιμή $3B_{16}$.
Η εντολή «ADA $2000 (κωδικός λειτουργίας AB)» είναι απευθείας διευθυνσιοδότησης και επομένως στην τιμή του καταχωρητή Α προστίθεται το περιεχόμενο της θέσης μνήμης με διεύθυνση 2000_{16}, δηλαδή $A = 3B_{16} + 5C_{16} = 97_{16}$.
Καθώς η εντολή «STA $2500 (κωδικός λειτουργίας BB)» είναι απευθείας διευθυνσιοδότησης η τιμή 97_{16} θα αποθηκευτεί στη θέση μνήμης με διεύθυνση 2500_{16}.

Γ)
Η εντολή «LDA $20,X (κωδικός λειτουργίας CE)» είναι δεικτοδοτούμενης διευθυνσιοδότησης και συνεπώς η ενεργός διεύθυνση προκύπτει από το άθροισμα της τιμής του καταχωρητή δείκτη με την τιμή 20_{16}. Άρα, η ενεργός διεύθυνση είναι $20F4_{16} + 20_{16} = 2114_{16}$ και στον καταχωρητή Α αποθηκεύεται η τιμή $1A_{16}$.
Η εντολή «ADA #33 (κωδικός λειτουργίας AA)» είναι άμεσης διευθυνσιοδότησης και συνεπώς στην τιμή του καταχωρητή Α θα προστεθεί το 33_{16}, άρα $A = 1A_{16} + 33_{16} = 4D_{16}$.
Καθώς η εντολή «STA $55 (κωδικός λειτουργίας BD)» είναι σχετικής διευθυνσιοδότησης η ενεργός διεύθυνση προκύπτει από το άθροισμα της τιμής του μετρητή προγράμματος στην τιμή 55_{16}. Όταν εκτελείται η εντολή ο μετρητής προγράμματος δείχνει την επόμενη εντολή και συνεπώς έχει τιμή 1006_{16}. Επομένως η ενεργός διεύθυνση είναι $1006_{16} + 55_{16} = 105B_{16}$.
Άρα στη θέση μνήμης με διεύθυνση $105B_{16}$ θα γραφεί η τιμή $4D_{16}$.

ΑΣΚΗΣΗ 32

Δίνεται το παρακάτω πρόγραμμα σε γλώσσα Assembly του επεξεργαστή Intel 8085.

```
   LXI SP, F000
  MVI A,15
  STA 1000
  MVI A,20
  STA 1001
  CALL K
  MOV D,A
  HLT

K: LXI H,1000
  MOV A,M
```

```
INX H
RAR
MOV B,A
MOV A,M
ADD B
MOV C,A
RAL
RET
```

Α) Υποθέστε ότι το πρόγραμμα θα αποθηκευτεί στην κύρια μνήμη ξεκινώντας από τη θέση $00FD_{16}$. Επίσης υποθέστε ότι η διεύθυνση Κ (διεύθυνση αρχής της ρουτίνας) αντιστοιχεί στην επόμενη διεύθυνση του τέλους του κυρίως προγράμματος. Δώστε στον παρακάτω πίνακα τα περιεχόμενα της μνήμης από τη διεύθυνση 0100Η και ως το τέλος του συνολικού προγράμματος.

Διεύθυνση Μνήμης	**Περιεχόμενο Μνήμης**	**Εντολή**
0100		
0101		
0102		
. . .	. . .	. . .

Β) Να προσδιορίσετε χρησιμοποιώντας μόνο τις τέσσερις βασικές πράξεις (πρόσθεση, αφαίρεση, πολλαπλασιασμό, διαίρεση) τη συνάρτηση που υλοποιείται από τη ρουτίνα που καλείται από το κυρίως πρόγραμμα. Τι τιμή έχουν οι καταχωρητές που χρησιμοποιούνται στο πρόγραμμα, μετά το τέλος του προγράμματος; Γιατί εμφανίζεται σφάλμα στην τιμή της συνάρτησης σε σχέση με την αναμενόμενη τιμή;

Γ) Τροποποιήστε τη συνάρτηση και γράψτε νέα ρουτίνα που να υπολογίζει την τιμή της χωρίς την εμφάνιση σφάλματος.

Λύση

Α)

Διεύθυνση Μνήμης	**Περιεχόμενο Μνήμης**	**Εντολή**
0100	3E	MVI A,15
0101	15	
0102	32	STA 1000
0103	00	
0104	10	
0105	3E	MVI A,20
0106	20	
0107	32	STA 1001
0108	01	
0109	10	
010A	CD	CALL 010F
010B	0F	
010C	01	
010D	57	MOV D,A
010E	76	HLT
010F	21	LXI H,1000

0110	00	
0111	10	
0112	7E	MOV A,M
0113	23	INX H
0114	1F	RAR
0115	47	MOV B,A
0116	7E	MOV A,M
0117	80	ADD B
0118	4F	MOV C,A
0119	17	RAL
011A	C9	RET

Β)
Η ρουτίνα εκτελεί πράξεις μεταξύ των τιμών που βρίσκονται αποθηκευμένες στις διευθύνσεις μνήμης 1000_{16} και 1001_{16}. Αν θέσουμε ως X την τιμή της διεύθυνσης 1000_{16} και ως Y την τιμή της διεύθυνσης 1001_{16} τότε εκτελείται η συνάρτηση

$$2 \cdot \left(Y + \frac{X}{2} \right).$$

Το αποτέλεσμα αποθηκεύεται στον καταχωρητή D. Το πρόγραμμα λειτουργεί για θετικούς αριθμούς.

Περιεχόμενα καταχωρητών μετά το τέλος του προγράμματος

Καταχωρητής	**Περιεχόμενα**
A	54
B	0A
C	2A
D	54
HL	1001
SP	F000

Επειδή $X=15_{16}$ και $Y=20_{16}$, το αποτέλεσμα της παραπάνω συνάρτησης έπρεπε να είναι 55_{16} ενώ παίρνουμε τελικά 54_{16}. Εμφανίζεται σφάλμα, το οποίο δημιουργείται κατά την δεξιά ολίσθηση του αριθμού X με την εντολή RAR που χρησιμοποιείται για την πραγματοποίηση της διαίρεσης με το 2. Εμφανίζεται κλασματικό ψηφίο το οποίο και χάνεται.

Γ)
Για να αποφευχθεί η δημιουργία σφάλματος τροποποιούμε τη μορφή της παραπάνω συνάρτησης σε $2 \cdot Y + X$. Με αυτό τον τρόπο αποφεύγεται η δεξιά ολίσθηση και άρα η δημιουργία κλασματικού μέρους που είναι και η πηγή του σφάλματος.
Η ρουτίνα που αποφεύγει την εμφάνιση του σφάλματος έχει ως εξής:

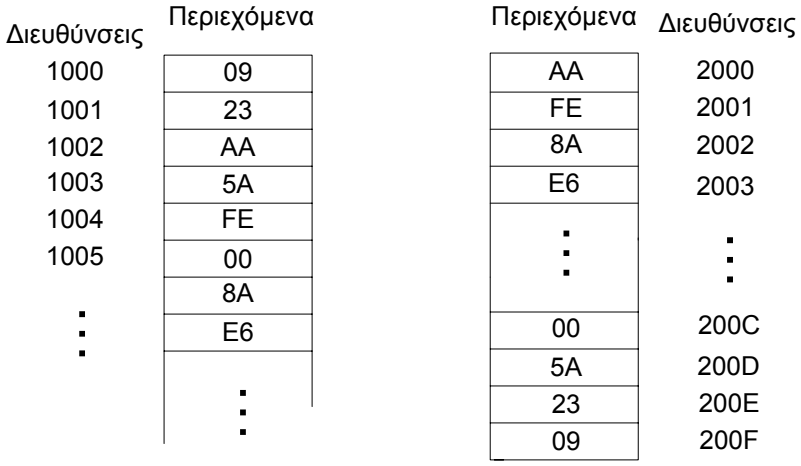
```
LXI H,1000
MOV A,M
MOV B,A
INX H
MOV A,M
RAL
ADD B
RET
```

ΑΣΚΗΣΗ 33

Θεωρείστε ότι σε περιοχή μνήμης που ξεκινά από τη διεύθυνση 1000_{16} είναι αποθηκευμένοι δεκαέξι προσημασμένοι δυαδικοί αριθμοί της μίας ψηφιολέξης (8 bits). Γράψτε πρόγραμμα σε γλώσσα Assembly του επεξεργαστή Intel 8085 που να τοποθετεί τους αριθμούς αυτούς σε αντίστοιχη περιοχή μνήμης που να ξεκινά από τη διεύθυνση 2000_{16} έτσι ώστε οι αρνητικοί αριθμοί να καταλαμβάνουν τις πρώτες θέσεις της περιοχής και οι θετικοί αριθμοί τις τελευταίες. Το μηδέν να αντιμετωπιστεί όπως και οι θετικοί αριθμοί.

Υπόδειξη: Εξηγείστε το σκεπτικό σας, δώστε το διάγραμμα ροής, δώστε το πρόγραμμα όπως προκύπτει από τον προσομοιωτή που χρησιμοποιήσατε.

Παράδειγμα:



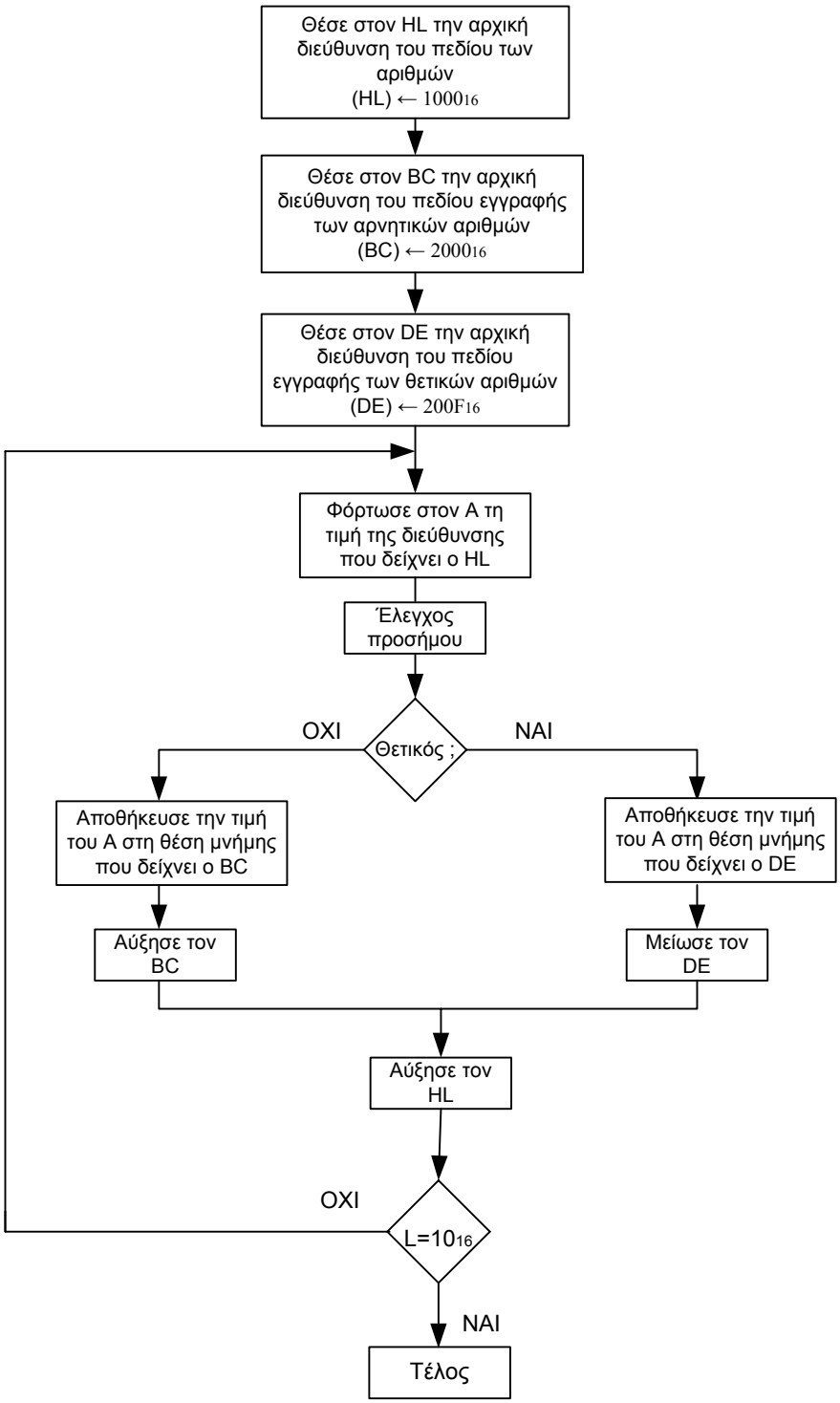
Λύση

Για την ανάπτυξη ενός προγράμματος Assembly που να εκτελεί την επιθυμητή διαδικασία στην αρχή θέτουμε τις αρχικές διευθύνσεις των πεδίων όπου θα γραφούν αντίστοιχα οι θετικοί και οι αρνητικοί αριθμοί. Κατόπιν, καθώς διαβάζουμε έναν αριθμό ελέγχουμε το πρόσημό του και τον γράφουμε στο κατάλληλο πεδίο. Κάθε φορά που προκύπτει αριθμός για εγγραφή σε ένα από τα πεδία, θα τροποποιείται η τρέχουσα διεύθυνση στο πεδίο αυτό κατά ένα (είτε με αύξηση είτε με μείωση του αντίστοιχου καταχωρητή διεύθυνσης) ώστε να μην υπάρξει επικάλυψη στις εγγραφές.

Αφού έχουμε δεκαέξι αριθμούς, η αρχική διεύθυνση του πεδίου εγγραφής των αρνητικών αριθμών είναι 2000_{16} και των θετικών μπορεί να είναι $200F_{16}$. Κάθε φορά που γράφουμε αρνητικό αριθμό στη μνήμη θα αυξάνουμε τον αντίστοιχο καταχωρητή διεύθυνσης ενώ κάθε φορά που γράφουμε θετικό αριθμό θα μειώνουμε τον αντίστοιχο καταχωρητή διεύθυνσης.

Η διαδικασία σταματά όταν έχουν διαβαστεί και γραφτεί στο κατάλληλο πεδίο και οι δεκαέξι αριθμοί.

Παρακάτω δίνεται ένα διάγραμμα ροής για τη διαδικασία αυτή.



Ένα ενδεικτικό πρόγραμμα που υλοποιεί την απαιτούμενη διαδικασία δίνεται παρακάτω:
(Η εντολή ADI μπήκε για να ενεργοποιήσει το ψηφίο σημαία S, ώστε να γίνει ο έλεγχος θετικών/αρνητικών αριθμών που ουσιαστικά πραγματοποιείται με την εντολή JP.)

```
8085 Machine Code File
0000  LXI H 1000  21 ;
0001                00 ;
```

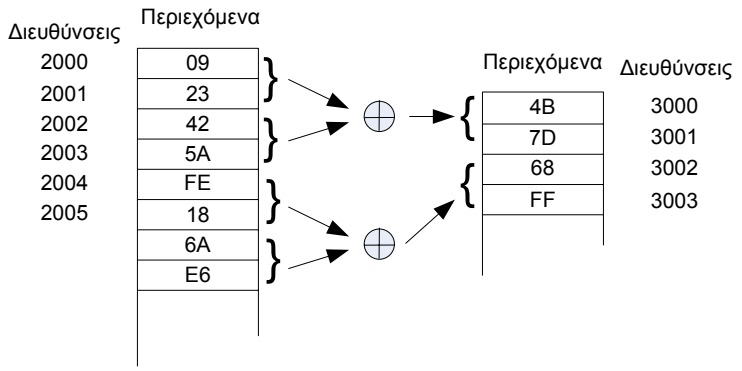
```
0002                10 ;
0003  LXI B 2000  01 ;
0004                00 ;
0005                20 ;
0006  LXI D 200F  11 ;
0007                0F ;
0008                20 ;
0009  MOV A M   7E ;
000A  ADI 00      C6 ;
000B                00 ;
000C  JP  0014     F2 ;
000D                14 ;
000E                00 ;
000F  STAX B      02 ;
0010  INX B       03 ;
0011  JMP 0016    C3 ;
0012                16 ;
0013                00 ;
0014  STAX D      12 ;
0015  DCX D       1B ;
0016  INX H       23 ;
0017  MVI A 10   3E ;
0018                10 ;
0019  SUB L       95 ;
001A  JNZ 0009   C2 ;
001B                09 ;
001C                00 ;
001D HLT          76 ;
```

ΑΣΚΗΣΗ 34

Θεωρείστε ότι σε περιοχή μνήμης που ξεκινά από τη διεύθυνση 2000_{16} είναι αποθηκευμένοι δεκαέξι δεκαεξαδικοί αριθμοί των τεσσάρων ψηφίων. Κάθε αριθμός καταλαμβάνει δύο θέσεις μνήμης με τα λιγότερα σημαντικά ψηφία να καταλαμβάνουν τη θέση με τη μικρότερη διεύθυνση. Γράψτε πρόγραμμα σε γλώσσα Assembly του επεξεργαστή Intel 8085 που να αθροίζει τους αριθμούς ανά ζεύγη και να αποθηκεύει τα αποτελέσματα κατά τον ίδιο τρόπο σε περιοχή μνήμης που ξεκινά από τη διεύθυνση 3000_{16}. Συνολικά θα πρέπει να γίνουν δηλαδή 8 προσθέσεις των 16 δυαδικών ψηφίων η κάθε μία.

Υπόδειξη: Εξηγείστε το σκεπτικό σας, δώστε το διάγραμμα ροής, δώστε το πρόγραμμα όπως προκύπτει από τον προσομοιωτή που χρησιμοποιήσατε.

Παράδειγμα:
Παρακάτω φαίνονται αποθηκευμένοι με τον τρόπο που αναφέρθηκε οι αριθμοί 2309_{16}, $5A42_{16}$, $18FE_{16}$ και $E66A_{16}$ οι οποίοι προστίθενται και παράγονται τα αποτελέσματα 7D4B και FF68 τα οποία αποθηκεύονται, με τον ίδιο τρόπο, στην αντίστοιχη περιοχή μνήμης.

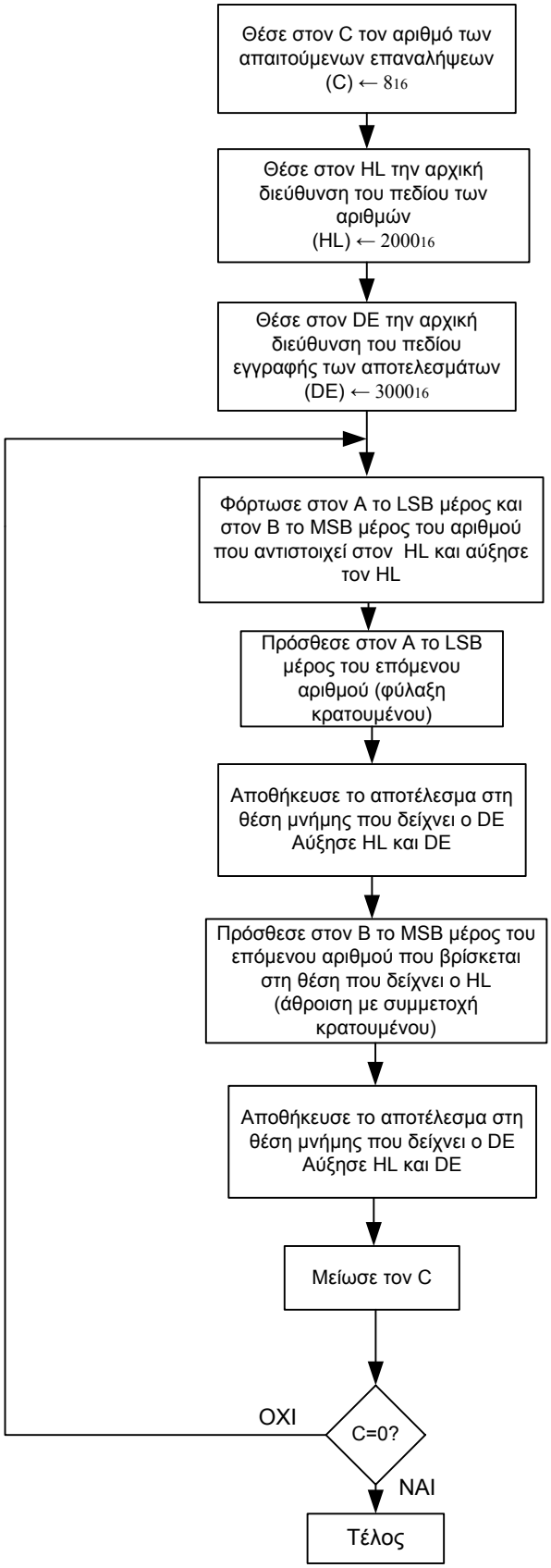


Λύση:

Για την υλοποίηση αντίστοιχου προγράμματος σε Assembly πρέπει να βρεθεί τρόπος άθροισης αριθμών δύο ψηφιολέξεων (2 bytes) σε επεξεργαστή που περιέχει μονάδες (π.χ. αθροιστή) 8-ψηφίων. Αυτό μπορεί να επιτευχθεί εύκολα με χρήση της εντολής ADC, δηλαδή άθροισης με συμμετοχή του κρατουμένου. Αθροίζουμε πρώτα τις λιγότερο σημαντικές ψηφιολέξεις των αριθμών που είναι να αθροιστούν και το κρατούμενο που πιθανώς παράγεται κατά την άθροιση αυτή αποθηκεύεται στον καταχωρητή που αντιστοιχεί στη σημαία CY.Κατόπιν αθροίζουμε τις περισσότερο σημαντικές ψηφιολέξεις με συμμετοχή του κρατουμένου από την προηγούμενη άθροιση. Έτσι εξασφαλίζεται η αντιμετώπιση αριθμών 2 ψηφιολέξεων ως ενός ενιαίου αριθμού μιας λέξης.

Η διαδικασία επαναλαμβάνεται μέχρι να εξαντληθούν οι αριθμοί, δηλαδή 8 φορές.

Παρακάτω δίνεται ένα διάγραμμα ροής για την ζητούμενη διαδικασία.



Ένα ενδεικτικό πρόγραμμα που υλοποιεί την απαιτούμενη διαδικασία δίνεται παρακάτω:

```
8085 Machine Code File
0000  MVI C 08    0E ;
0001               08 ;
0002  LXI H 2000 21 ;
0003               00 ;
0004               20 ;
0005  LXI D 3000 11 ;
0006               00 ;
0007               30 ;
0008  MOV A M  7E ;
0009  INX H      23 ;
000A  MOV B M  46 ;
000B  INX H      23 ;
000C  ADD M      86 ;
000D  STAX D     12 ;
000E  INX D      13 ;
000F  INX H      23 ;
0010  MOV A B  78 ;
0011  ADC M      8E ;
0012  STAX D    12 ;
0013  INX H      23 ;
0014  INX D      13 ;
0015  DCR C      0D ;
0016  JNZ 0008   C2 ;
0017               08 ;
0018               00 ;
0019  HLT         76 ;
```

ΑΣΚΗΣΗ 35

Γιαννη, νομίζω ότι σε ολο το κειμενο χρειαζεται μια μορφοποίηση οσον αφορά τα fonts & lines spacing. Υπάρχουν διαφορετικά fonts σε διαφορετικά σημεία. Επίσης, προτείνω να υπάρχουν διαφορετικά fonts στους κώδικες προγραμμάτων (courier new).Δεν τα εχω πειράξει αυτά.

Η ακολουθία αριθμών **Fibonacci** είναι μια γνωστή σειρά αριθμών που βρίσκει εφαρμογή σε διάφορα επιστημονικά πεδία. Οι αριθμοί της ακολουθίας παράγονται με τον παρακάτω αλγόριθμο: οι δύο πρώτοι αριθμοί είναι ίσοι με 1 ενώ ο κάθε επόμενος αριθμός ισούται με το άθροισμα των δύο προηγούμενων. Έτσι, οι αριθμοί της σειράς Fibonacci είναι οι εξής (σε δεκαδική αναπαράσταση): **1, 1, 2, 3, 5, 8, 13, 21, 34, 55**, κ.ο.κ.

Να γραφεί πρόγραμμα **assembly** για τον μικροεπεξεργαστή Intel **8085** που να **υπολογίζει** τους πρώτους **δέκα** αριθμούς της ακολουθίας Fibonacci και να τους αποθηκεύει στη μνήμη ξεκινώντας από τη διεύθυνση $\mathbf{3000_{16}}$.

Λύση

Αρχικά αποθηκεύουμε τους δύο πρώτους αριθμούς (01) και (01) στις δύο πρώτες θέσεις μνήμης (3000) και (3001). Για το σκοπό αυτό τα ζεύγη καταχωρητών (D, E) και (H, L) αρχικοποιούνται στις τιμές (3000) και (3001), αντίστοιχα. Επίσης, ο καταχωρητής B αρχικοποιείται στην τιμή 08. Στη συνέχεια μέσω ενός βρόγχου που επαναλαμβάνεται οκτώ φορές δημιουργούνται οι υπόλοιποι οκτώ αριθμοί της σειράς Fibonacci. Συγκεκριμένα, χρησιμοποιούνται τα ζεύγη καταχωρητών (D, E) και (H, L) ώστε

να δείχνουν στις θέσεις μνήμης που βρίσκονται ο προτελευταίος και τελευταίος αριθμός που δημιουργήθηκαν, αντίστοιχα. Στη συνέχεια οι αριθμοί αυτοί (ο προτελευταίος και τελευταίος αριθμός) προστίθενται και το αποτέλεσμα φυλάσσεται στο συσσωρευτή A, ενώ οι τιμές των ζευγών των καταχωρητών (D, E) και (H, L) αυξάνονται κατά ένα. Κατόπιν, αποθηκεύεται ο νέος αριθμός, μειώνεται η τιμή του καταχωρητή B και ελέγχεται αν αυτή έχει γίνει μηδέν. Αν δε μηδενιστεί η τιμή του καταχωρητή B ο βρόγχος επαναλαμβάνεται. Με βάση τα παραπάνω το πρόγραμμα είναι το ακόλουθο:

```
8085 Machine Code File
0000 LXI D 3000  11 ;
0001             00 ;
0002             30 ;
0003 LXI H 3001  21 ;
0004             01 ;
0005             30 ;
0006 MVI A 01    3E ;
0007             01 ;
0008 MVI B 08    06 ;
0009             08 ;
000A STAX D      12 ;
000B MOV M A     77 ;
000C LDAX D      1A ;
000D ADD M       86 ;
000E INX D       13 ;
000F INX H       23 ;
0010 MOV M A     77 ;
0011 DCR B       05 ;
0012 JNZ 000C    C2 ;
0013             0C ;
0014             00 ;
0015 HLT         76 ;
```

ΑΣΚΗΣΗ 36

Έστω δύο πίνακες X, Y τεσσάρων στοιχείων που είναι αποθηκευμένοι στις θέσεις μνήμης 2000_{16}-2003_{16} και 3000_{16}-3003_{16}, αντίστοιχα. Τα δεδομένα του κάθε πίνακα είναι τα ακόλουθα:

Πίνακας X	
Θέση μνήμης	Περιεχόμενο
2000	05
2001	0A
2002	06
2003	0E

Πίνακας Y	
Θέση μνήμης	Περιεχόμενο
3000	04
3001	10
3002	02
3003	17

Θεωρείστε το παρακάτω πρόγραμμα σε συμβολική γλώσσα (assembly) του 8085. Ποια είναι τα δεδομένα του πίνακα Χ μετά το πέρας του προγράμματος; Ποια συνάρτηση υλοποιεί το συγκεκριμένο πρόγραμμα;

```
      LXI   H 2000
      LXI   D 3000
      MVI   C 04
K:    LDAX  D
      ADI   05
      MOV   B, A
      MOV A, M
      RAL
      SUB   B
      JM    L
M:    MOV   M,A
      DCR   C
      JZ    EXIT
      INX   H
      INX   D
      JMP   K
L:    CMA
      ADI   01
      JMP   M
EXIT:HLT
```

Λύση

```
      LXI   H 2000
      LXI   D 3000
      MVI   C 04
K:    LDAX  D          // Μεταφορά στον A του y_i,  A= y_i //
      ADI   05         // A= y_i +5 //
      MOV   B, A       // B= A = y_i +5 //
      MOV A, M         // Μεταφορά στον A του x_i,  A= x_i //
      RAL              // Αριστερή ολίσθηση,  A= 2x_i //
      SUB   B          // A = A-B => A = 2x_i –( y_i +5 )//
      JM    L          // Έλεγχος προσήμου//
M:    MOV   M,A        // Αποτέλεσμα θετικό – μεταφορά στη θέση του x_i //
      DCR   C
      JZ    EXIT       // Έλεγχος τερματισμού //
      INX   H
      INXD             //Αύξηση (H, L) και (D, E) ώστε να δείχνουν στους επόμενους
                       αριθμούς//
      JMP   K          // Άλμα στην είσοδο του βρόχου //
L:    CMA
      ADI   01         //Υπολογισμός συμπληρώματος ως 2 //
      JMP   M          // Άλμα για αποθήκευση και έλεγχο τερματισμού//
EXIT: HLT
```

Με βάση τα παραπάνω ο πίνακας Χ περιέχει τις ακόλουθες τιμές με το πέρας του προγράμματος.

Πίνακας Χ

Θέση μνήμης	Περιεχόμενο
2000	01
2001	01
2002	05
2003	00

Μελετώντας το συγκεκριμένο πρόγραμμα παρατηρούμε ότι είναι ένας βρόχος που εκτελεί τις ακόλουθες ενέργειες.

Αρχικά φορτώνεται το δεδομένο y_i από τον πίνακα Υ, γίνεται πρόσθεση με το 05 και το αποτέλεσμα φυλάσσεται στον καταχωρητή Β. Επομένως, Β = y_i +5. Στη συνέχεια φορτώνεται το δεδομένο x_i από τον πίνακα Χ και ολισθαίνει αριστερά Επομένως, Α = $2x_i$. Κατόπιν γίνεται η αφαίρεση Α-Β = $2x_i$ – (y_i +5) και αν το αποτέλεσμα είναι θετικό τότε αυτό αποθηκεύεται στην θέση του x_i. Αν το αποτέλεσμα είναι αρνητικό τότε υπολογίζεται το συμπλήρωμα ως προς 2 και αυτό αποθηκεύεται στη θέση του x_i.

Η συνάρτηση που υλοποιείται είναι η $\left|2x_i - (y_i + 5)\right|$.

ΑΣΚΗΣΗ 37

Έστω δύο πίνακες Χ , Υ των πέντε στοιχείων που είναι αποθηκευμένοι στις θέσεις μνήμης 2000_{16}-2004_{16} και 3000_{16}-3004_{16}, αντίστοιχα. Τα δεδομένα του κάθε πίνακα είναι τα ακόλουθα:

Πίνακας Χ	
Θέση μνήμης	Περιεχόμενο
2000	04
2001	05
2002	03
2003	02
2004	0A

Πίνακας Υ	
Θέση μνήμης	Περιεχόμενο
3000	04
3001	03
3002	0A
3003	07
3004	09

Θεωρείστε το παρακάτω πρόγραμμα assembly για τον επεξεργαστή Intel 8085.
Α. Ποια είναι τα δεδομένα του πίνακα Χ μετά την εκτέλεση του προγράμματος;
Β. Ποια συνάρτηση υλοποιεί το συγκεκριμένο πρόγραμμα; Θεωρείστε ότι ο μέγιστος αριθμός του πίνακα Χ είναι το $0F_{16}$.
Υπόδειξη: Να χρησιμοποιήσετε τον προσομοιωτή για την εύρεση των δεδομένων του πίνακα Χ μετά την εκτέλεση του προγράμματος.

```
   LXI H 2000
   LXI D 3000
   MVI C 05
K: LDAX D
   CMP M
   JNC L
   RLC
```

```
   ADD M
   ADI 02
N: MOV M, A
   DCR C
   JZ EXIT
   INX H
   INX D
   JMP K
L: ADD M
   JMP N
EXIT: HLT
```

Λύση

```
   LXI H 2000      // αρχικοποίηση του (H, L)//
   LXI D 3000      // αρχικοποίηση του (D, E)//
   MVI C 05        // αρχικοποίηση του C//
K: LDAX D     // μετέφερε στο A το yi, A= yi //
   CMP M           // σύγκριση xi με yi //
   JNC L           // αν CY ≠ 1, τότε A ≥ xi , δηλαδή yi ≥ xi , άλμα στο L//
   RLC             //A=A*2, δηλαδή, A=2* yi //
   ADD M           // A=A+ xi=2* yi+ xi //
   ADI 02          // A=A+ 2=2* yi+ xi+2 //
N: MOV M, A        //αποθήκευση του A στη θέση του xi //
   DCR C           // μείωση του C //
   JZ EXIT         // έλεγχος τερματισμού, αν Z=0, τότε άλμα στον τερματισμό
                   προγράμματος//
   INX H           // αύξηση κατά 1 του (H, L)//
   INX D           // αύξηση κατά 1 του (D, E)//
   JMP K           // άλμα στην είσοδο του βρόγχου //
L: ADD M           // A=A+ xi = yi + xi //
   JMP N     // άλμα στο N για αποθήκευση και έλεγχο τερματισμού //
EXIT: HLT    // τερματισμός προγράμματος //
```

Μελετώντας το συγκεκριμένο πρόγραμμα παρατηρούμε ότι είναι ένας βρόχος που εξετάζει τα στοιχεία x_i και y_i των πινάκων X, Y και ο οποίος εκτελεί τα ακόλουθα.

Αρχικά μεταφέρεται το δεδομένο y_i στον A και συγκρίνεται με το x_i. Αν $y_i < x_i$ τότε υπολογίζεται το A=$2* y_i+ x_i+2$, το οποίο στη συνέχεια γράφεται στη θέση του x_i.

Στη συνέχεια μειώνεται ο καταχωρήτης C κατά 1 και ελέγχεται αν μηδενίστηκε (δηλαδή, αν υπάρχουν επιπλέον επαναλήψεις που πρέπει να εκτελεστούν). Στην περίπτωση που υπάρχουν επαναλήψεις αυξάνονται κατά 1 τα ζεύγη των καταχωρητών (H, L) και (D, E) ώστε να δείχνουν στις θέσεις μνήμης που βρίσκονται τα αμέσως επόμενα x_i και y_i και το πρόγραμμα μεταβαίνει στο K όπου αρχίζει μία νέα επανάληψη του βρόχου.

Αν $y_i \geq x_i$ τότε το πρόγραμμα μεταβαίνει στο L όπου υπολογίζεται το $A= y_i + x_i$ και μετά εκτελεί άλμα στο N για την αποθήκευση του $A= y_i + x_i$ στη θέση του x_i , ενώ κατόπιν ακολουθεί έλεγχος τερματισμού του προγράμματος.

Με βάση τα παραπάνω το πρόγραμμα υλοποιεί την ακόλουθη συνάρτηση

$$X = \begin{cases} X + Y & X \leq Y \\ 2 * Y + X + 2 & X > Y \end{cases}.$$

Μετά την εκτέλεση του προγράμματος τα δεδομένα είναι του πίνακα X είναι:

Πίνακας Χ	
Θέση μνήμης	Περιεχόμενο
2000	08
2001	0D
2002	0D
2003	09
2004	1E

ΑΣΚΗΣΗ 38

Έστω πίνακας Α που καταλαμβάνει τις θέσεις μνήμης 2000_{16}-$200A_{16}$. Να γραφεί πρόγραμμα assembly για τον μικροεπεξεργαστή Intel 8085 που να επιστρέφει στους καταχωρητές Β και C το πλήθος των άρτιων και περιττών αριθμών, αντίστοιχα, του πίνακα *A*, ενώ στον καταχωρητή Ε να γράφεται ο μεγαλύτερος άρτιος. Στην περίπτωση που δεν υπάρχει κανένας άρτιος τότε $E=00_{16}$. Ο αριθμός 00_{16} να αντιμετωπιστεί ως άρτιος. Να δώσετε επίσης το διάγραμμα ροής του προγράμματος. Να επισυνάψετε επίσης στην απάντηση της εργασίας το πρόγραμμα assembly που προέκυψε με τη χρήση του προσομοιωτή.

Λύση

Το πρόγραμμα αποτελείται από τις ακόλουθες ρουτίνες: α) ρουτίνα αρχικοποίησης, β) ρουτίνα ελέγχου αν ο τρέχον αριθμός είναι περιττός, γ) ρουτίνα χειρισμού των άρτιων αριθμών και δ) ρουτίνα ελέγχου τερματισμού του προγράμματος.

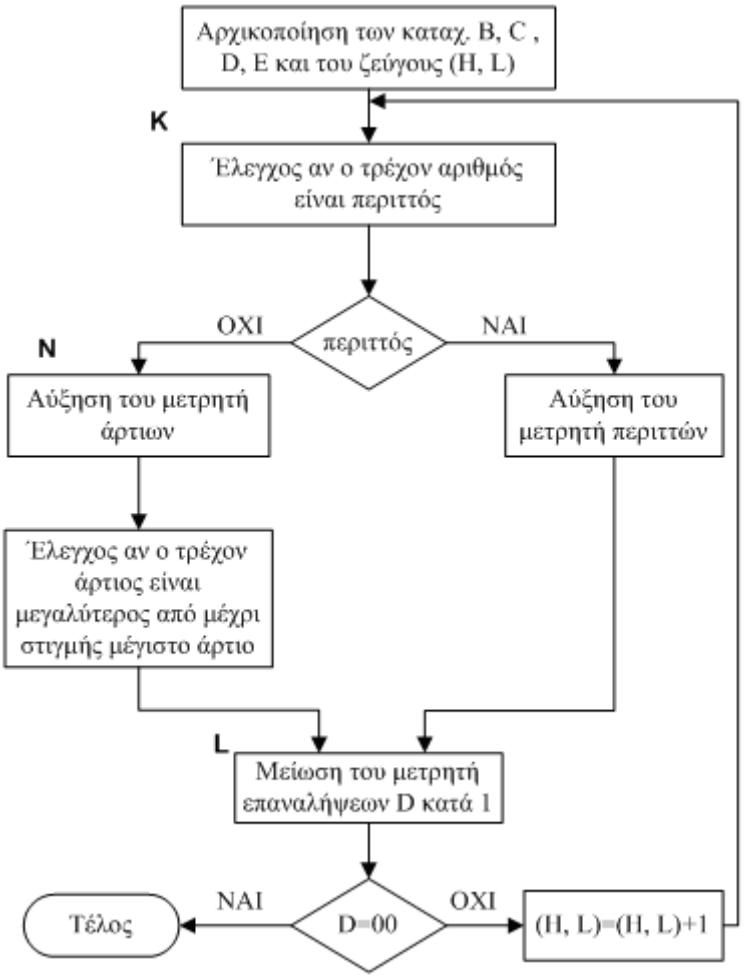
Στη ρουτίνα αρχικοποίησης οι καταχωρητές παίρνουν τις αρχικές τιμές σύμφωνα με την εκφώνηση. Το ζεύγος (H, L) χρησιμοποιείται για τη διευθυνσιοδότηση των θέσεων του πίνακα *A* και αρχικοποιείται στη διεύθυνση της πρώτης θέσης του πίνακα. Ο καταχωρητής D χρησιμοποιείται για τον έλεγχο τερματισμού του προγράμματος (πρέπει να εξεταστούν έντεκα θέσεις του πίνακα *A*) Επομένως, $B=C=E=00_{16}$, D=0B και $(H, L)=2000_{16}$.

Ο έλεγχος για τον αν ο τρέχον αριθμός είναι περιττός γίνεται εκτελώντας τη λογική πράξη AND μεταξύ του τρέχοντος αριθμού και του accumulator, ο οποίος έχει την τιμή 01_{16}. Σε κάθε περιττό αριθμό το LSB ισούται με 1. Επομένως, αν μετά τη λογική πράξη η σημαία Z (ZERO) είναι Z=0 (μη μηδενικό αποτέλεσμα) τότε συνεπάγεται ότι ο τρέχον αριθμός είναι περιττός. Τότε αυξάνει κατά ένα η τιμή του καταχωρητή C (μετρητής περιττών) και το πρόγραμμα μεταβαίνει στη ρουτίνα ελέγχου τερματισμού των επαναλήψεων. Αν Z=1 τότε ο αριθμός είναι άρτιος και το πρόγραμμα μεταβαίνει στη ρουτίνα χειρισμού των άρτιων αριθμών.

Στη ρουτίνα χειρισμού των άρτιων αριθμών αυξάνει κατά 1 ο καταχωρητής Β (μετρητής άρτιων) και γίνεται έλεγχος αν ο τρέχον άρτιος είναι μεγαλύτερος από το μέγιστο μέχρι στιγμής άρτιο που φυλάσσεται στον καταχωρητή Ε. Αν ο τρέχον άρτιος είναι μεγαλύτερος τότε αυτός αντιγράφεται στον καταχωρητή Ε. Στη συνέχεια το πρόγραμμα μεταβαίνει στη ρουτίνα ελέγχου τερματισμού.

Η ρουτίνα ελέγχου τερματισμού του προγράμματος μειώνει κάθε φορά την τιμή του καταχωρητή D κατά 1. Αν $D=00_{16}$ τότε ελέγχθηκαν και οι 11 θέσεις του πίνακα Α και περατώνεται η εκτέλεση του προγράμματος. Αν υπάρχουν ακόμη αριθμοί που πρέπει να εξεταστούν τότε αυξάνει κατά 1 η τιμή του ζεύγους (H, L), ώστε να δεικτοδοτεί στην επόμενη θέση του πίνακα Α, και το πρόγραμμα μεταβαίνει στη ρουτίνα ελέγχου για τον αν ο νέος αριθμός είναι περιττός.

Με βάση τα παραπάνω το διάγραμμα ροής του προγράμματος και το assembly πρόγραμμα είναι τα ακόλουθα.




```
// αρχικοποίηση//
    LXI H 2000
    LXI B 0000
    LXI D 0B00

// έλεγχος για τον αν ο τρέχον αριθμός είναι περιττός//
K: MVI A 01
    ANA M
    JZ N        // αν Z=1 ο αριθμός είναι άρτιος, άλμα στη ρουτίνα χειρισμού άρτιου//
    INR C       // ο αριθμός είναι περιττός, αύξηση του αντίστοιχου μετρητή//

// έλεγχος τερματισμού προγράμματος//
L: DRC D
    JZ EXIT  // αν Z=1, ελεγχθήκαν όλοι ο αριθμοί, άλμα στο τέλος προγράμματος//
    INXH
    JMP K       // άλμα στο K για έλεγχο του νέου αριθμού//

// ρουτίνα χειρισμού άρτιων αριθμών //
N: INR B        // αύξηση του αντίστοιχου μετρητή//
    MOV A, E
    CMP M  // έλεγχος αν ο τρέχον άρτιος είναι ο μέγιστος άρτιος//
```

```
    JNC L  //αν δεν είναι CY=1 τότε ο τρέχον άρτιος είναι ο μέγιστος άρτιος δηλαδή,
           E ≥ (M) . Ο καταχ. Ε δεν αλλάζει, άλμα στον έλεγχο τερματισμού προγράμματος
           //
    MOV E, M // αντιγραφή του τρέχοντος άρτιου στον καταχωρητή Ε //
    JMP L // άλμα στον έλεγχο τερματισμού προγράμματος//
EXIT: HLT
```

ΑΣΚΗΣΗ 39

Έστω οι πίνακες *A* και *B* που καταλαμβάνουν τις διευθύνσεις μνήμης 2000_{16} – 2009_{16} και 3000_{16}-3009_{16}, αντίστοιχα. Να γραφτεί πρόγραμμα assembly για τον μικροεπεξεργαστή Intel 8085 που θα ελέγχει αν ικανοποιείται η συνθήκη $04_{16} < X \le 16_{16}$, για κάθε αριθμό *X* του πίνακα *A* και με βάση τον παραπάνω έλεγχο θα εκτελεί τα ακόλουθα:

I. Αν η συνθήκη ικανοποιείται τότε ο αριθμός *X* αντιγράφεται στην αντίστοιχη θέση του πίνακα B. Δηλαδή, ο αριθμός *X* που αντιστοιχεί στη διεύθυνση 2000_{16} αντιγράφεται στη θέση μνήμης με διεύθυνση 3000_{16}, ο αριθμός *X* που αντιστοιχεί στη διεύθυνση 2001_{16} αντιγράφεται στη θέση μνήμης με διεύθυνση 3001_{16} κοκ.
II. Αν δεν ικανοποιείται η συνθήκη, τότε στην αντίστοιχη θέση του πίνακα *B* γράφεται το άθροισμα του αριθμού *X* με το δεδομένο της προηγούμενης θέσης του πίνακα *B*. Αν για το πρώτο στοιχείο του πίνακα B χρειάζεται υπολογισμός του αθροίσματος, τότε στη θέση μνήμης με διεύθυνση 3000_{16} γράφεται ο αριθμός *X* που αντιστοιχεί στη διεύθυνση 2000_{16}
III. Μετράει πόσες φορές ικανοποιείται η παραπάνω συνθήκη.

Να δοθεί επίσης το διάγραμμα ροής του προγράμματος. Να επισυνάψετε επίσης στην απάντηση της εργασίας το πρόγραμμα assembly που προέκυψε με τη χρήση του προσομοιωτή.

Σημείωση: Να αρχικοποιήσετε τη θέση μνήμης με διεύθυνση $2FFF_{16}$ με το δεδομένο 00_{16}.

Παράδειγμα:

Πίνακας Α		**Πίνακας Β**	
Θέση μνήμης	**Δεδομένο**	**Θέση μνήμης**	**Τελικό Δεδομένο**
2000	01	3000	01
2001	**05**	3001	**05**
2002	18	3002	1D
2003	22	3003	3F
2004	04	3004	43
2005	**12**	3005	**12**
2006	02	3006	14
2007	03	3007	17
2008	**16**	3008	**16**
2009	33	3009	49

Η συνθήκη ικανοποιείται 3 φορές.

Λύση

Το πρόγραμμα αποτελείται από 4 ρουτίνες, που είναι: α) η ρουτίνα αρχικοποίησης, β) η ρουτίνα όπου ελέγχεται η συνθήκη, αντιγράφεται ο αριθμός *X* στον πίνακα *B* και αυξάνει ο μετρητής ικανοποίησης της συνθήκης, όταν αυτή ικανοποιείται, γ) η ρουτίνα όπου υπολογίζεται το άθροισμα και αποθηκεύεται στον πίνακα *B*, όταν δεν ικανοποιείται η συνθήκη και δ) η ρουτίνα ελέγχου των επαναλήψεων.

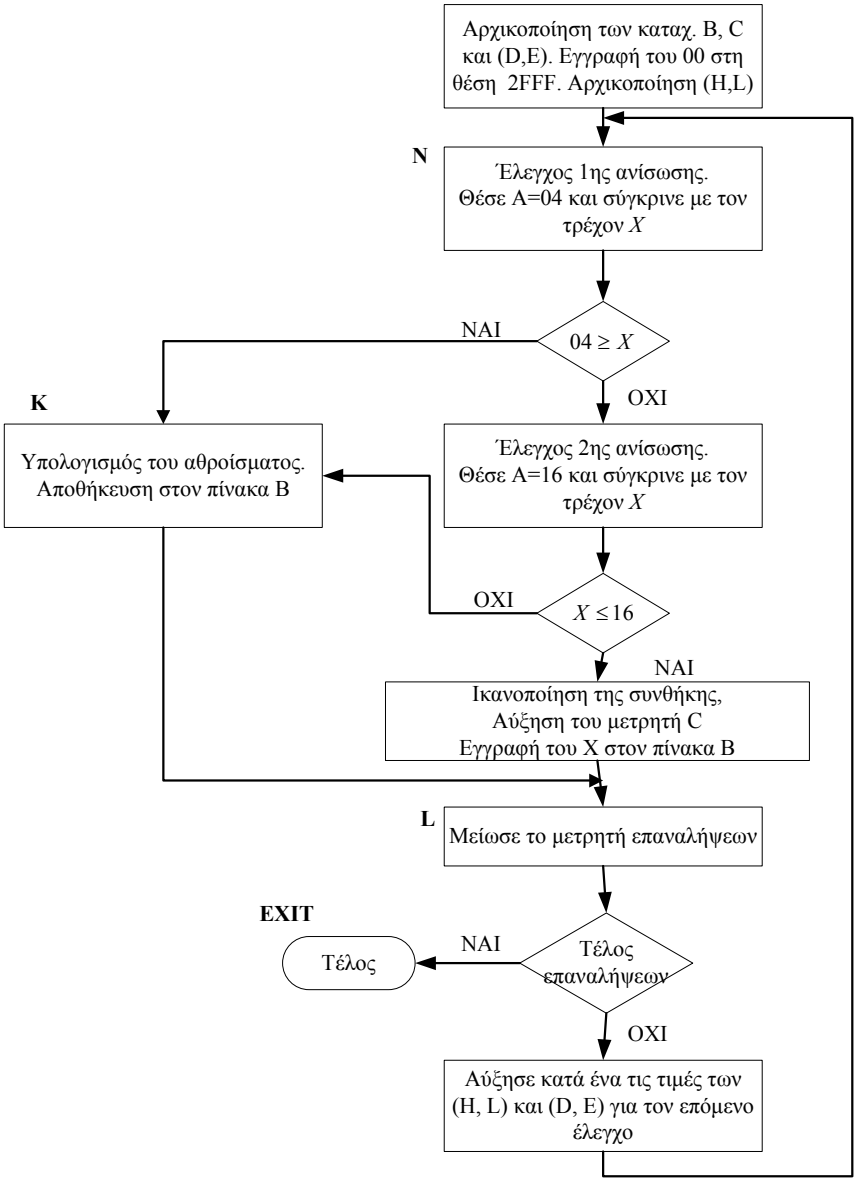
Η ρουτίνα αρχικοποίησης περιλαμβάνει την αρχικοποίηση των καταχωρητών B ($B=0A_{16}$) και C ($C=00_{16}$) και των ζευγών (D,E) και (H,L) με $(D,E)=3000_{16}$ και $(H,L)=2000_{16}$. Ο καταχωρητής B είναι ο μετρητής των επαναλήψεων, ο καταχωρητής C είναι ο μετρητής ικανοποίησης της συνθήκης, ενώ τα ζεύγη καταχωρητών (D,E) και (H,L) χρησιμοποιούνται για τη διευθυνσιοδότηση των θέσεων των πινάκων *B* και *A*, αντίστοιχα. Επίσης, σε αυτή τη ρουτίνα τίθεται το δεδομένο της θέσης μνήμης $2FFF_{16}$ ίσο με 00_{16}, έτσι ώστε όταν για τον πρώτο αριθμό *X* δεν ικανοποιείται η συνθήκη, τότε με την εκτέλεση της ρουτίνας υπολογισμού του αθροίσματος να υπολογιστεί ο σωστός αριθμός (δηλαδή, $X+00_{16}=X$) που θα αποθηκευτεί στην πρώτη θέση (3000_{16}) του πίνακα B.

Η ρουτίνα ελέγχου της συνθήκης ελέγχει τις δύο ανισώσεις δίνοντας κάθε φορά την κατάλληλη τιμή στον accumulator ($A=04_{16}$ και $A=16_{16}$) και συγκρίνοντας αυτόν με το αντίστοιχο δεδομένο του πίνακα *A*. Αν η συνθήκη ικανοποιείται τότε ο αριθμός *X* αντιγράφεται στην αντίστοιχη θέση του πίνακα *B* και αυξάνει ο μετρητής ικανοποίησης της συνθήκης. Ο έλεγχος ικανοποίησης της συνθήκης γίνεται εξετάζοντας την τιμή της σημαίας του κρατουμένου. Αν η συνθήκη δεν ικανοποιείται τότε γίνεται άλμα στη ρουτίνα μη ικανοποίησης της συνθήκης.

Η ρουτίνα μη ικανοποίησης της συνθήκης υπολογίζει το άθροισμα σύμφωνα με την εκφώνηση και γράφει το αποτέλεσμα στη θέση του πίνακα *B* που καθορίζει η τρέχουσα τιμή του ζεύγους (D,E).

Τέλος, η ρουτίνα ελέγχου των επαναλήψεων μειώνει κατά ένα την τιμή του καταχωρητή B (μετρητής επαναλήψεων) και ελέγχει αν αυτός έγινε ίσος με μηδέν όπου τότε μεταβαίνει στον τερματισμό του προγράμματος. Αλλιώς, αυξάνει κατά ένα τις τιμές των ζευγών (D,E) και (H,L), ώστε να δείχνουν στις επόμενες θέσεις των πινάκων *A* και *B*, και μεταβαίνει στη ρουτίνα ελέγχου της συνθήκης.

Με βάση τα παραπάνω το διάγραμμα ροής του προγράμματος είναι το ακόλουθο.



Κατά συνέπεια το πρόγραμμα είναι το ακόλουθο. Οι εντολές γράφονται με τη σειρά που φαίνεται παρακάτω.

// ρουτίνα αρχικοποίησης //

```
   LXI D 3000
   MVI C 00
   MVI B 0A
   LXI H 2FFF
   MVI M 00
   LXI H 2000
```

// ρουτίνα ελέγχου συνθήκης //

```
N: MVI A 04
   CMP M      // έλεγχος της πρώτης ανίσωσης//
   JNC K      //αν carry=0, τότε 04_16 ≥ X , άλμα στη ρουτίνα μη ικανοπ. συνθήκης//
   MVI A 16
   CMP M      // έλεγχος της δεύτερης ανίσωσης//
   JC K       //αν carry=1, τότε X > 16_16 , άλμα στη ρουτίνα μη ικανοπ. συνθήκης //
```

```
   MOV A, M //η συνθήκη ικανοποιείται, μεταφορά του X στον accumulator//
   STAX D   // εγγραφή του X στον πίνακα B//
   INR C    // αύξηση του μετρητή ικανοποίησης της συνθήκης//

// ρουτίνα τερματισμού των επαναλήψεων //
L: DCR B
   JZ EXIT  // έλεγχος τέλους των επαναλήψεων, άλμα στον τερματισμό //
   INX H
   INX D
   JMP N    // άλμα στη ρουτίνα ελέγχου της συνθήκης//

// ρουτίνα μη ικανοποίησης της συνθήκης //
K: DCX D
   LDAX D   // το προηγούμενο στοιχείο του πίνακα B μεταφέρεται στον accumulator //
   ADD M    // άθροιση του τρέχοντος X με το προηγούμενο στοιχείο του πίνακα B//
   INX D    // διόρθωση του (D, E)//
   STAX D   // αποθήκευση αθροίσματος στον πίνακα B//
   JMP L    // άλμα στη ρουτίνα ελέγχου τέλους των επαναλήψεων//
EXIT: HLT
```

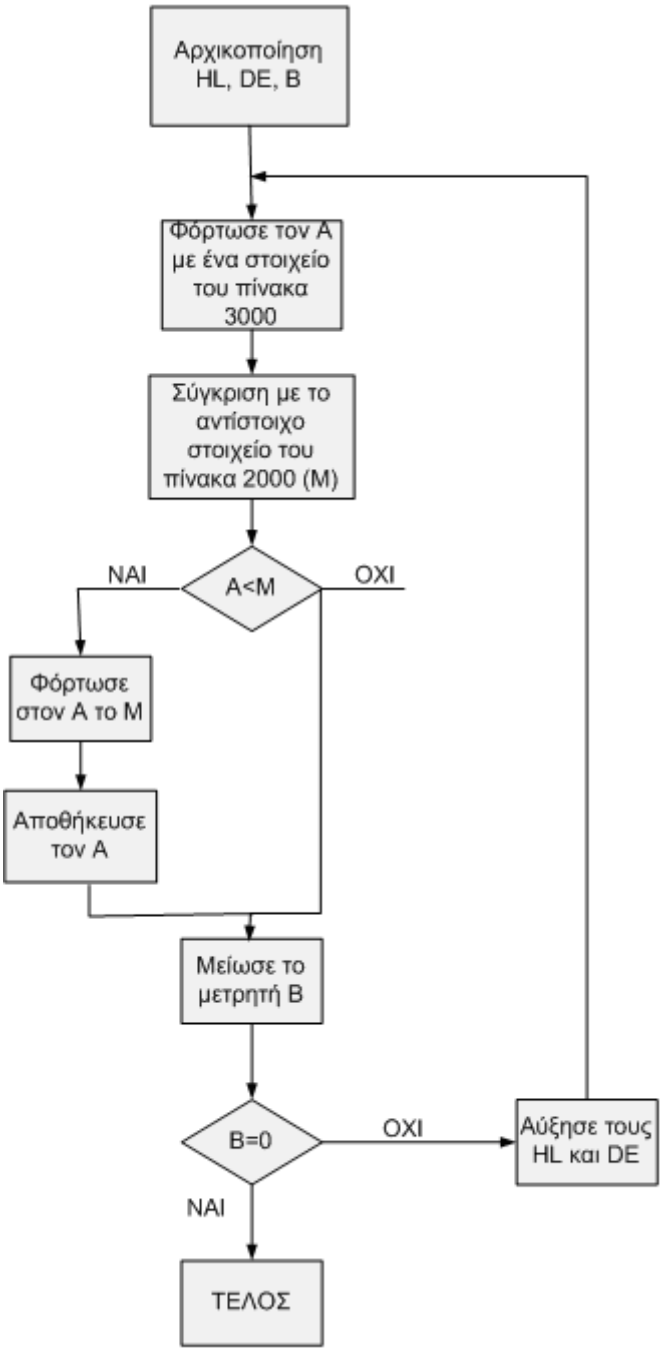
ΑΣΚΗΣΗ 40

Θεωρείστε δυο πίνακες από αριθμούς στις διευθύνσεις 2000-200F και 3000-300F. Να γράψετε πρόγραμμα assembly για τον μικροεπεξεργαστή Intel 8085 που να συγκρίνει θέση προς θέση τους αριθμούς στους δυο πίνακες και να γράφει τον μεγαλύτερο στον πίνακα με διευθύνσεις 3000-300F. Δηλαδή, να συγκρίνει τον αριθμό στη διεύθυνση 2000 με τον αριθμό στη διεύθυνση 3000 και να γράφει τον μεγαλύτερο στην διεύθυνση 3000, κατόπιν να συγκρίνει τον αριθμό στη διεύθυνση 2001 με τον αριθμό στη διεύθυνση 3001 και να γράφει τον μεγαλύτερο στην διεύθυνση 3001, κοκ. Θεωρείστε ότι έχετε θετικούς αριθμούς.

Λύση

Για την υλοποίηση του προγράμματος αρχικοποιούμε τους καταχωρητές HL και DE με τις τιμές 2000 και 3000, αντίστοιχα, για τη διευθυνσιοδότηση των πινάκων. Ο καταχωρητής B χρησιμοποιείται ως μετρητής για να μετρά τις επαναλήψεις της διαδικασίας. Σε κάθε επανάληψη φορτώνεται ο αριθμός του πίνακα με αρχή τη διεύθυνση 3000 στον καταχωρητή A και συγκρίνεται με τον αντίστοιχο αριθμό του πίνακα με αρχή το 2000. Ο μεγαλύτερος γράφεται στη θέση του πίνακα με αρχή το 3000. Ο μετρητής B μειώνεται κατά 1 και αν είναι ίσος με 0 το πρόγραμμα διακόπτεται ενώ διαφορετικά αυξάνονται κατά 1 οι καταχωρητές HL και DE για τον έλεγχο της επόμενης θέσης. Η διαδικασία επαναλαμβάνεται 16 φορές.

Το διάγραμμα ροής είναι:



Ένα ενδεικτικό πρόγραμμα είναι:

```
lxi h,2000
lxi d,3000          //αρχικοποίηση//
mvi b,10
n:
ldax d              // φόρτωσε στοιχείο πίνακα 3000  στον A //
cmp m               // σύγκριση με στοιχείο πίνακα 2000 (M) //
jnc k               // αν δεν ισχύει CY=1 τότε A ≥ (M) πήγαινε στο k //
mov a,m
stax d              // φόρτωσε τον M στον πίνακα 3000 //
k:
dcr b               // έλεγχος τερματισμού, B=0? //
```

```
jz exit          // αν Z=1 πήγαινε στο τέλος //
inx h
inx d            // αύξησε κατά 1 τους H και D //
jmp n            // πήγαινε στο n //
exit:
hlt              // τέλος//
```

ΑΣΚΗΣΗ 41

Θεωρείστε **δεκαέξι** αριθμούς που καταλαμβάνουν συνεχόμενες θέσεις μνήμης ξεκινώντας από τη θέση με διεύθυνση **2000_{16}**. Να γραφεί πρόγραμμα assembly για τον μικροεπεξεργαστή Intel 8085 που να επιστρέφει: (α) πόσοι αριθμοί είναι **μεγαλύτεροι ή ίσοι του 04_{16}** (το αποτέλεσμα να γράφεται στον καταχωρητή D) και (β) πόσοι είναι **ακέραια πολλαπλάσια του 04_{16}** (το αποτέλεσμα να γράφεται στον καταχωρητή E). Θεωρείστε ότι **όλοι οι αριθμοί είναι θετικοί**.

Λύση

Το παραπάνω πρόγραμμα αποτελείται από τα ακόλουθα τμήματα: α) τμήμα αρχικοποίησης, β) τμήμα ελέγχου για το αν ο τρέχων αριθμός είναι μεγαλύτερος ή ίσος του 04_{16} , γ) τμήμα ελέγχου για το αν ο αριθμός είναι ακέραιο πολλαπλάσιο του 04_{16} και γ) τμήμα ελέγχου επαναλήψεων ώστε να εξεταστούν και οι δεκαέξι αριθμοί.

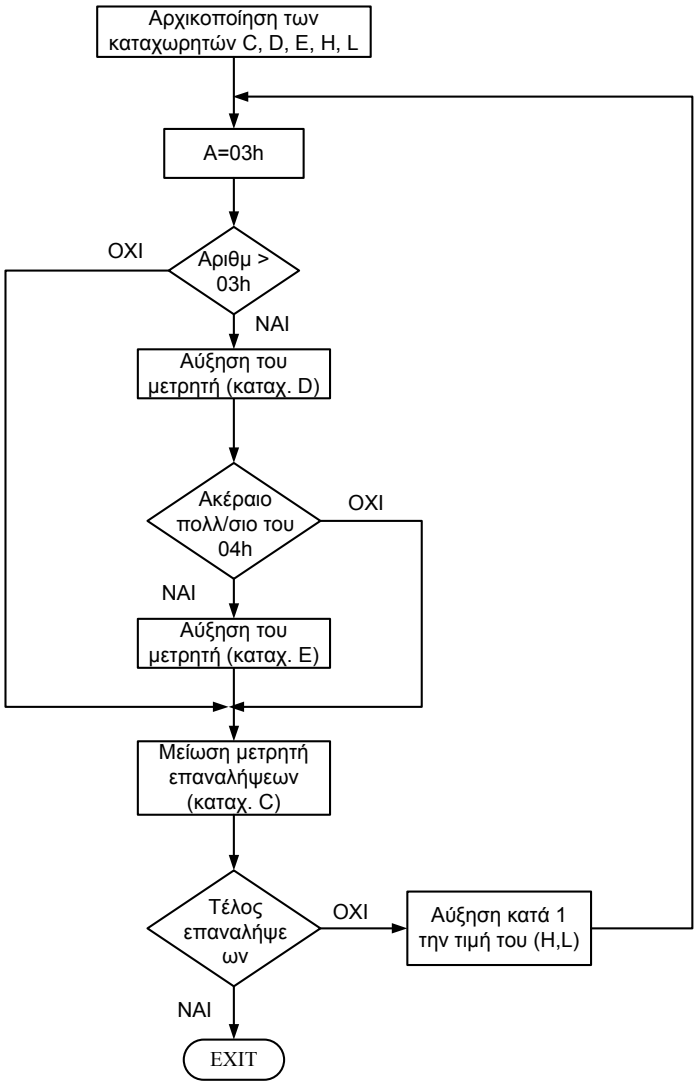
Στο τμήμα αρχικοποίησης οι καταχωρητές D, E παίρνουν τις τιμές 00_{16} (D=E=00_{16}), ο καταχωρητής C (μετρητής επαναλήψεων) παίρνει την τιμή 10_{16} (C=10_{16}) και το ζεύγος καταχωρητών (H,L) παίρνει την τιμή 2000_{16} ώστε να δείχνει στην πρώτη διεύθυνση της θέσης μνήμης που μας ενδιαφέρει.

Ο έλεγχος για το αν ο αριθμός είναι μεγαλύτερος ή ίσος του 04_{16} γίνεται θέτοντας το συσσωρευτή ίσο με 03_{16} και συγκρίνοντας αυτόν με το περιεχόμενο της θέσης μνήμης που δείχνει το ζεύγος (H,L). Αν μετά τη σύγκριση η σημαία του κρατούμενου (CY) είναι 0 τότε ο αριθμός είναι μικρότερος του 04_{16}, αλλιώς είναι μεγαλύτερος ή ίσος του 04_{16}.

Ο έλεγχος για το αν ο αριθμός είναι ακέραιο πολλαπλάσιο του 04_{16} βασίζεται στην ακόλουθη παρατήρηση. Αν ο αριθμός είναι ακέραιο πολλαπλάσιο του 04_{16} τότε είναι το αποτέλεσμα διαδοχικών προσθέσεων με το 04_{16}, όπου κάθε φορά προστίθεται στο αποτέλεσμα το 04_{16} (δηλαδή, 08=04+04, 12=04+04+04 κοκ.). Δεδομένου ότι κάθε φορά προστίθεται το 04_{16}, στο οποίο τα δύο λιγότερα σημαντικά ψηφία είναι 00, κάθε αριθμός που είναι ακέραιο πολλαπλάσιο του θα έχει τα δύο λιγότερο σημαντικά ψηφία ίσα με 00. Αυτό μπορεί να ελεγχθεί με αρκετούς τρόπους όπως α) θέτοντας στο συσσωρευτή την τιμή 03_{16} και κάνοντας AND με την υπό εξέταση λέξη (αριθμό), αν το αποτέλεσμα είναι 00_{16}, το οποίο ελέγχεται μέσω της σημαίας ZERO (Z), τότε ο αριθμός είναι ακέραιο πολλαπλάσιο του 04_{16}, β) ολισθαίνοντας δύο φορές δεξιά μέσω κρατούμενου την υπό εξέταση λέξη και ελέγχοντας αν το CY παίρνει δύο φορές την τιμή 0.

Η ρουτίνα ελέγχου επαναλήψεων μειώνει κάθε φορά το μετρητή επαναλήψεων (καταχωητής C) κατά 1. Όσο αυτός είναι μεγαλύτερος από μηδέν αυξάνεται κατά ένα η τιμή του ζεύγους (H,L) και επαναλαμβάνεται η παραπάνω διαδικασία. Το πρόγραμμα τερματίζει όταν η τιμή του καταχωρητή C γίνει ίση με 0.

Με βάση τα παραπάνω το διάγραμμα ροής του προγράμματος και το πρόγραμμα assembly είναι τα ακόλουθα:




```
    LXI H 2000      // αρχικοποίηση του ζεύγος (H, L) //
    LXI D 0000      // αρχικοποίηση των καταχωρητών D, E //
    MVI C 10        // αρχικοποίηση του μετρητή επαναλήψεων //

// έλεγχος αν ο αριθμός είναι μεγαλύτερος ή ίσος του 04(16) //
K   MVI A 03
    CMP M
    JNC L           // αν CY=0 τότε ο αριθμός είναι μικρότερος του 04(16) //
    INR D           //ο αριθμός είναι ≥04(16), αύξηση του αντίστοιχου μετρητή //

// έλεγχος αν ο αριθμός είναι ακέραιο πολλαπλάσιο του 04(16) //
    MVI A 03
    ANA M
    JNZ L           // Αν Z≠0 τότε κάποιο από τα δύο λιγότερο σημαντικά ψηφία είναι 1
                    και ο αριθμός δεν είναι ακέραιο πολλαπλάσιο του04(16) //
```

```
    INR E          // ο αριθμός είναι ακέραιο πολλαπλάσιο του ≥04₁₆, αύξηση του
                   αντίστοιχου μετρητή //

// έλεγχος επαναλήψεων //
L   DCR C
    JZ EXIT  // είναι X=Y, επομένως E=A=X //
    INX H
    JMP K
EXIT   HLT
```

ΑΣΚΗΣΗ 42

Διαθέτετε επεξεργαστή που έχει τις παρακάτω εντολές

Όνομα εντολής	Κωδικός λειτουργίας (1 byte)	Τρόπος διευθυνσιοδότησης
ADA	AA	Μηδενικής σελίδας
	AB	Απευθείας (κατ' ευθείαν)
	AC	Άμεσος
	AD	Σχετικός
SUA	BA	Μηδενικής σελίδας
	BB	Απευθείας (κατ' ευθείαν)
	BC	Άμεσος
	BD	Σχετικός
STA	CB	Απευθείας (κατ' ευθείαν)
	CD	Σχετικός
	CE	Δεικτοδοτημένος
LDA	DB	Απευθείας (κατ' ευθείαν)
	DD	Σχετικός
	DE	Δεικτοδοτημένος

Η εντολή ADA προσθέτει στο περιεχόμενο του καταχωρητή Α το δεδομένο που βρίσκεται στη θέση που προκύπτει από τον τρόπο διευθυνσιοδότησης. Το αποτέλεσμα αποθηκεύεται στον καταχωρητή Α. Η εντολή SUA αφαιρεί από το περιεχόμενο του καταχωρητή Α το δεδομένο που βρίσκεται στη θέση που προκύπτει από τον τρόπο διευθυνσιοδότησης. Το αποτέλεσμα αποθηκεύεται στον καταχωρητή Α. Η εντολή STA αποθηκεύει το περιεχόμενο του καταχωρητή Α στη θέση μνήμης που προκύπτει από τον τρόπο διευθυνσιοδότησης. Η εντολή LDA αποθηκεύει στον καταχωρητή Α το περιεχόμενο της θέσης μνήμης που προκύπτει από τον τρόπο διευθυνσιοδότησης. Επίσης υπάρχει και η εντολή STOP για σταμάτημα εκτέλεσης του προγράμματος.
Οι εντολές απευθείας διευθυνσιοδότησης καταλαμβάνουν 3 Bytes μνήμης ενώ όλες οι υπόλοιπες 2 Bytes. Εξαιρείται η εντολή STOP που καταλαμβάνει 1 Byte. Η μνήμη διευθυνσιοδοτείται κατά Bytes.

Θεωρείστε τα παρακάτω περιεχόμενα μνήμης (όλα σε δεκαεξαδική μορφή):

Διεύθυνση μνήμης	Περιεχόμενο
00B3	32
105A	06
10B8	10
2000	3A
3299	50

Ο καταχωρητής δείκτη X περιέχει την τιμή $326A_{16}$.

Για τα παρακάτω μέρη προγράμματος να συμπληρωθούν οι αντίστοιχοι πίνακες με τις διευθύνσεις μνήμης όπου αποθηκεύονται οι εντολές (η διεύθυνση του πρώτου Byte κάθε εντολής) και με τον αριθμό των Bytes που καταλαμβάνει κάθε εντολή.
Επίσης να δοθεί η διεύθυνση μνήμης όπου θα εγγραφεί το αποτέλεσμα καθώς επίσης και η τιμή του αποτελέσματος. Εξηγείστε το σκεπτικό σας.

A)
1000 LDA $2000 (κωδικός λειτουργίας DB)
ADA $B3 (κωδικός λειτουργίας AA)
SUA #45 (κωδικός λειτουργίας BC)
STA $9A, X (κωδικός λειτουργίας CE)
STOP

Πίνακας προς συμπλήρωση:

Εντολή	**Διεύθυνση πρώτου Byte εντολής**	**Πλήθος Bytes εντολής**
LDA $2000	1000	3
ADA $B3	…	…
SUA #45	…	…
STA $9A, X	…	…
STOP	…	…

B)
1000 LDA $2F, X (κωδικός λειτουργίας DE)
SUA $B4 (κωδικός λειτουργίας BD)
ADA $105A (κωδικός λειτουργίας AB)
STA $30 (κωδικός λειτουργίας CD)
STOP

Πίνακας προς συμπλήρωση:

Εντολή	**Διεύθυνση πρώτου Byte εντολής**	**Πλήθος Bytes εντολής**
LDA $2F, X	1000	2
SUA $B4	…	…
ADA $105A	…	…
STA $30	…	…
STOP	…	…

Λύση:
A)
Αφού οι εντολές απευθείας διευθυνσιοδότησης καταλαμβάνουν 3 Bytes και όλες οι άλλες από 2 Bytes
οι διευθύνσεις όπου θα αποθηκευτούν οι εντολές προκύπτουν ως:

Εντολή	**Διεύθυνση πρώτου Byte εντολής**	**Πλήθος Bytes εντολής**
LDA $2000	1000	3
ADA $B3	1003	2
SUA #45	1005	2
STA $9A, X	1007	2
STOP	1009	1

Η εντολή «LDA $2000 (κωδικός λειτουργίας DB)» είναι απευθείας διευθυνσιοδότησης και επομένως στον καταχωρητή Α αποθηκεύεται το περιεχόμενο της θέσης μνήμης με διεύθυνση 2000_{16}, δηλαδή $A = 3A_{16}$.
Η εντολή «ADA $B3 (κωδικός λειτουργίας AA)» αντιστοιχεί σε διευθυνσιοδότηση μηδενικής σελίδας και επομένως στην τιμή του καταχωρητή Α προστίθεται το περιεχόμενο της θέσης μνήμης με διεύθυνση $00B3_{16}$, δηλαδή $A = 3A_{16} + 32_{16} = 6C_{16}$.
Η εντολή «SUA #45 (κωδικός λειτουργίας BC)» είναι άμεσης διευθυνσιοδότησης και συνεπώς από την τιμή του καταχωρητή Α θα αφαιρεθεί το 45_{16}, άρα $A = 6C_{16} - 45_{16} = 27_{16}$.
Η εντολή «STA $9A, X (κωδικός λειτουργίας CE)» είναι δεικτοδοτούμενης διευθυνσιοδότησης και συνεπώς η ενεργός διεύθυνση της θέσης μνήμης όπου θα αποθηκευτεί το αποτέλεσμα προκύπτει από το άθροισμα του περιεχομένου του καταχωρητή δείκτη Χ με το $9A_{16}$. Άρα η ενεργός διεύθυνση είναι $326A_{16} + 9A_{16} = 3304_{16}$ όπου γράφεται η τιμή 27_{16}.

Β)
Οι διευθύνσεις όπου θα αποθηκευτούν οι εντολές προκύπτουν ως:

Εντολή	Διεύθυνση πρώτου Byte εντολής	Πλήθος Bytes εντολής
LDA $2F, X	1000	2
SUA $B4	1002	2
ADA $105A	1004	3
STA $30	1007	2
STOP	1009	1

Η εντολή «LDA $2F, X (κωδικός λειτουργίας DE)» είναι δεικτοδοτούμενης διευθυνσιοδότησης και συνεπώς η ενεργός διεύθυνση προκύπτει από το άθροισμα της τιμής του καταχωρητή δείκτη με την τιμή $2F_{16}$. Άρα, η ενεργός διεύθυνση είναι $326A_{16} + 2F_{16} = 3299_{16}$ και στον καταχωρητή Α αποθηκεύεται η τιμή 50_{16}.
Καθώς η εντολή «SUA $B4 (κωδικός λειτουργίας BD)» είναι σχετικής διευθυνσιοδότησης, η ενεργός διεύθυνση προκύπτει από το άθροισμα της τιμής του μετρητή προγράμματος στην τιμή $B4_{16}$. Όταν εκτελείται η εντολή ο μετρητής προγράμματος δείχνει την επόμενη εντολή και συνεπώς έχει τιμή 1004_{16}. Επομένως, η ενεργός διεύθυνση είναι $1004_{16} + B4_{16} = 10B8_{16}$ και από τον καταχωρητή Α αφαιρείται το 10_{16}, άρα $A = 50_{16} - 10_{16} = 40_{16}$.
Η εντολή «ADA $105A (κωδικός λειτουργίας AB)» είναι απευθείας διευθυνσιοδότησης και επομένως στον καταχωρητή Α προστίθεται το 06_{16}, άρα $A = 40_{16} + 06_{16} = 46_{16}$.
Καθώς η εντολή «STA $30 (κωδικός λειτουργίας CD)» είναι σχετικής διευθυνσιοδότησης η ενεργός διεύθυνση προκύπτει από το άθροισμα της τιμής του μετρητή προγράμματος στην τιμή 30_{16}. Όταν εκτελείται η εντολή ο μετρητής προγράμματος δείχνει την επόμενη εντολή και συνεπώς έχει τιμή 1009_{16}. Άρα, η ενεργός διεύθυνση είναι $1009_{16} + 30_{16} = 1039_{16}$ όπου γράφεται η τιμή 46_{16}.

ΑΣΚΗΣΗ 43

Γενικά στην βιβλιογραφία ορίζονται μεταξύ άλλων και οι παρακάτω τρόποι διευθυνσιοδότησης :

Τρόπος Διευθυνσιοδότησης	**Παράδειγμα**	**Πράξη**
Καταχωρητή	**Add R4, R3**	**R4← R4+R3**
Άμεση	**Add R4, #3**	**R4 ← R4+3**
Αντικατάστασης	**Add R4, 100(R1)**	**R4 ← R4+Mem[100+R1]**
Έμμεση Καταχωρητή	**Add R4, (R1)**	**R4 ← R4+Mem[R1]**
Δείκτη/Βάσης	**Add R3, (R1+R2)**	**R3 ← R3+Mem[R1+R2]**
Απευθείας ή απόλυτη	**Add R1, (1001)**	**R1 ← R1+Mem[1001]**

Έμμεση Μνήμης	**Add R1, @(R3)**	**R1 ← R1+Mem[Mem[R3]]**
Μετά-αύξησης	**Add R1, (R2)+**	**R1 ← R1+Mem[R2];** **R2 ← R2+d[1]**
Προ-μείωσης	**Add R1, –(R2)**	**R2 ← R2–d[1];** **R1 ← R1+Mem[R2]**
Κλιμακωτό	**Add R1, 100(R2)[R3]**	**R1 ← R1+Mem[100+R2+R3*d]**

Έστω ένας επεξεργαστής που υποστηρίζει μόνο τις παρακάτω εντολές:

- **Load R4, (R1)**, (R4 ← Mem[R1]) σύμφωνα με τον τρόπο διευθυνσιοδότησης «Έμμεση Καταχωρητή»,
- **Add R4, R3,#imm** (R4 ← R4 + R3 + imm) σύμφωνα με τον τρόπο διευθυνσιοδότησης «Άμεση» ενώ το imm μπορεί να είναι μόνο θετικός αριθμός και
- **Neg R4, R3** (αντιστρέφει όλα τα bits του R3 και βάζει το αποτέλεσμα στον R4) σύμφωνα με τον τρόπο διευθυνσιοδότησης «Καταχωρητή»

Έστω επίσης ότι σε αυτόν τον επεξεργαστή ο καταχωρητής R0 και η θέση μνήμης 0 έχουν πάντα την τιμή 0 (δεν αλλάζουν) ενώ υπάρχουν 8 καταχωρητές (R0-R7).

Μπορείτε σε αυτόν τον επεξεργαστή να υποστηρίξετε την λειτουργικότητα που παρέχουν όλοι οι παραπάνω τρόποι διευθυνσιοδότησης; Με άλλα λόγια μπορείτε να υλοποιήσετε σαν ψευδοεντολές όλες τις Add που αναφέρονται παραπάνω ; Για όλες αυτές που μπορείτε να τις υλοποιήσετε μέσω των 3 εντολών του επεξεργαστή αναφέρετε αναλυτικά την σειρά των εντολών που μπορούν να τις υλοποιήσουν. Αν δεν μπορείτε να υλοποιήσετε κάποιες από αυτές τις ψευδοεντολές αναφέρατε ποια/ες άλλη/ες βασική/ες εντολή/ες χρειάζεται να υποστηρίξει ο επεξεργαστής για να τις υλοποιήσετε.

Σημείωση : Για την υλοποίηση των ψευδοεντολών μπορείτε να χρησιμοποιήσετε οποιουσδήποτε καταχωρητές θέλετε φτάνει ο καταχωρητής προορισμού της εντολής να έχει την σωστή τιμή.

Λύση:

Οι ακολουθίες εντολών που υλοποιούν την κάθε μία εντολή φαίνονται παρακάτω.
Σημειώνεται ότι το Imm θεωρούμε ότι μπορεί να πάρει και την τιμή 0.

Add R4, R3	Add R4, R3, #0	R4← R4+R3
Add R4, #3	Add R4, R0, #3	R4 ← R4+3

Add R4, 100(R1)	Add R1, R0, #100
	Load R3, (R1)
	Add R4, R3, #0
		R1 ← R1 + 100
		R3 ← Mem[100+R1]
		R4 ← R4 + Mem[100+R1]

Add R4, (R1)	Load R3, (R1) Add R4, R3, #0	R3 ← Mem[R1] R4 ←R4 + Mem[R1]
Add R3, (R1+R2)	Add R1, R2, #0 Load R5, (R1)	R1 ← R1 + R2 R5 ← Mem[R1+R2]

[1] Το d είναι μία ακέραια σταθερά που είναι αποθηκευμένη στον επεξεργαστή.

	Add R3, R5, #0	R3 ← R3+Mem[R1+R2]
Add R1, (1001)	Load R2, (R0)	R2 ← 0
	Add R2, R0, #1001	R2 ← 1001
	Load R3, (R2)	R3 ← Mem[1001]
	Add R1, R3, #0	R1 ← R1+Mem[1001]
Add R1, @(R3)	Load R4, (R3)	R4 ← Mem[R3]
	Load R2, (R4)	R2 ← Mem[Mem[R3]
	Add R1, R2, #0	R1 ← R1+Mem[Mem[R3]]
Add R1, (R2)+	Load R3, (R2)	R3 ← Mem[R2]
	Add R1, R3, #0	R1 ← R1+Mem[R2]
	Add R2, R0, #d	R2 ← R2+d
Add R1, –(R2)	Load R3, (R0)	R3 ← 0
	Add R3, R0, #d	R3 ← d
	Neg R4, R3	R4 ← Not d
	Add R4, R0, #1	R4 ← -d (συμπλήρωμα ως προς 2)
	Add R2, R4, #0	R2 ← R2 - d
	Load R5, (R2)	R5 ← Mem[R2]
	Add R1, R5, #0	R1 ← R1 + Mem[R2]

Add R1, 100(R2)[R3] Δεν μπορεί να υλοποιηθεί γιατί απαιτείται πολλαπλασιασμός. Αν ο επεξεργαστής υποστήριζε είτε πολλαπλασιασμό είτε μια εντολή μεταφοράς ελέγχου υπό συνθήκη θα μπορούσε να υλοποιηθεί και αυτή η εντολή. Στην δεύτερη περίπτωση θα υλοποιούταν ο πολλαπλασιασμός μέσω διαδοχικών προσθέσεων.

ΑΣΚΗΣΗ 44

Για το υπολογισμό του Μέγιστου Κοινού Διαιρέτη (Μ.Κ.Δ) δύο θετικών ακεραίων αριθμών έχουν προταθεί διάφοροι αλγόριθμοι μεταξύ αυτών και ο αλγόριθμος του Ευκλείδη.

Ο αλγόριθμος του Ευκλείδη για δύο θετικούς ακέραιους αριθμούς *X, Y* είναι ο ακόλουθος:

Βήμα 1. Σύγκρινε τους X, Y. Αν X=Y τότε Μ.Κ.Δ=X=Y
Βήμα 2. Αν X>Y τότε X=X-Y, Y=Y, επιστροφή στο βήμα 1.
Βήμα 3. Αν X<Y τότε X=X, Y=Y-X, επιστροφή στο βήμα 1.

Ο αλγόριθμος επαναλαμβάνεται μέχρι να ικανοποιηθεί η συνθήκη του βήματος 1.

Παράδειγμα

	Συνθήκη ελέγχου	**X**	**Y**
Αρχικοί αριθμοί		**14**	**8**
1η επανάληψη	X>Y	X=X-Y=**6**	Y=Y=**8**
2η επανάληψη	X<Y	X=X=**6**	Y=Y-X=**2**
3η επανάληψη	X>Y	X=X-Y=**4**	Y=Y=**2**
4η επανάληψη	X>Y	X=X-Y=**2**	Y=Y=**2**
5η επανάληψη	X=Y	X=2	Y=2

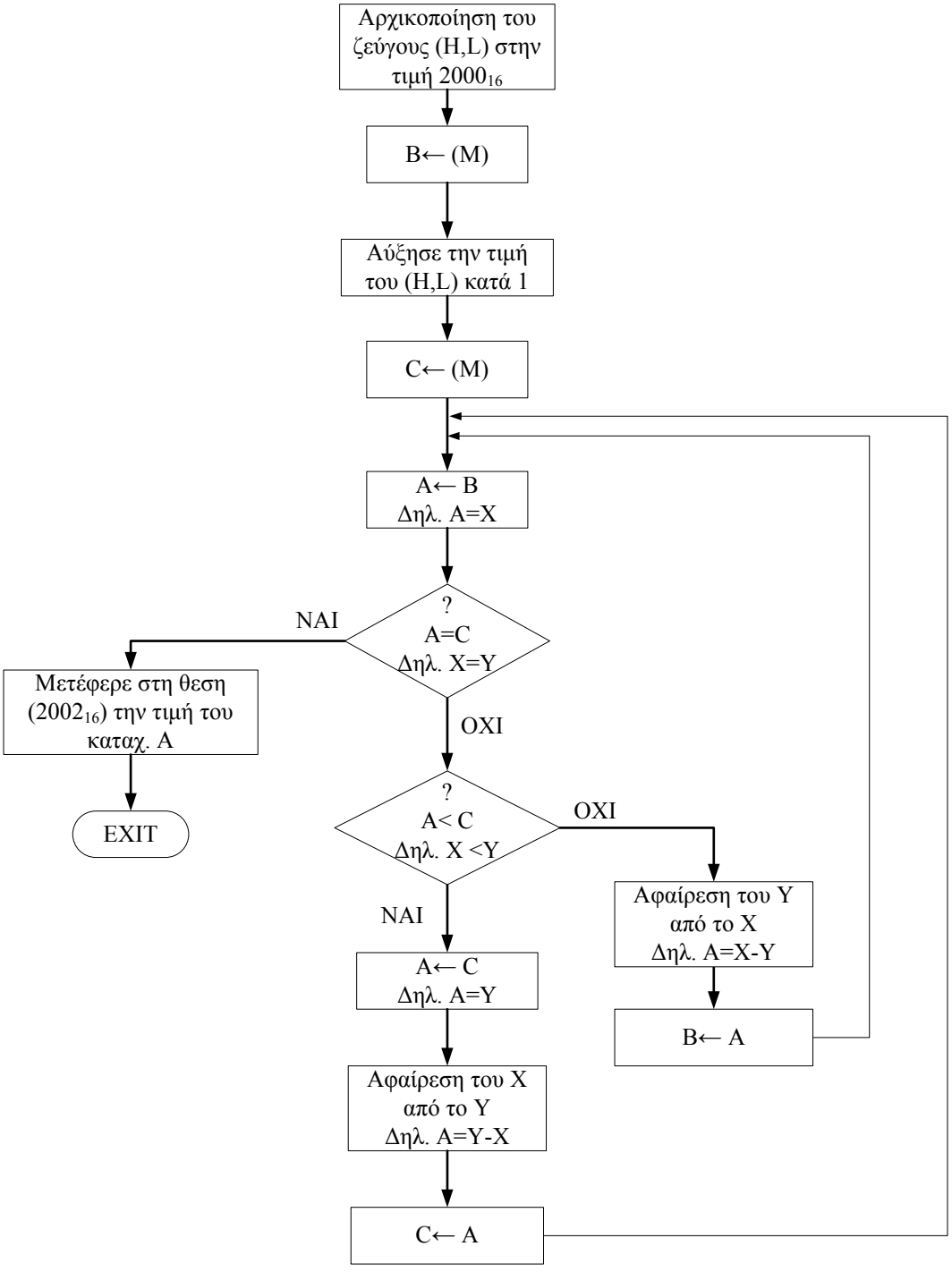
Άρα **Μ.Κ.Δ =2**

Να δώσετε το διάγραμμα ροής του προγράμματος για τον παραπάνω αλγόριθμο και να γράψετε κώδικα assembly για τον μικροεπεξεργαστή Intel 8085 όπου οι αριθμοί X, Y

βρίσκονται στις θέσεις μνήμης 2000_{16} και 2001_{16}, αντίστοιχα, ενώ ο Μ.Κ.Δ. να επιστρέφεται στη θέση μνήμης με διεύθυνση 2002_{16}.

Σημείωση: Να επισυνάψετε στις απαντήσεις σας και το αρχείο με τον κώδικα assembly

Λύση
Με βάση τον παραπάνω αλγόριθμο το διάγραμμα ροής του προγράμματος προκύπτει άμεσα:



Λαμβάνοντας υπόψη το διάγραμμα ροής του προγράμματος ο κώδικας assembly για τον μικροεπεξεργαστή Intel 8085 έχει ώς εξής:

```
   LXI H 2000     // αρχικοποίηση του ζεύγους (H,L) στην τιμή 2000_16//
   MOV B, M       // B=(M), δηλαδή, B=X //
   INX H          // αύξηση της τιμής του (H,L) κατά 1 //
   MOV C, M       // B=(M), δηλαδή, C=Y //
L: MOV A, B       // A=X//
   CMP C          // σύγκριση X, Y //
   JZ K           // αν X=Y τότε άλμα στο K //
   JC N           // έλεγχος αν X<Y, αν CY=1 τότε X<Y, άλμα στο N //
   SUB C          // A=A-C=X-Y //
   MOV B, A       // B=A=X-Y, δηλαδή, X=X-Y//
   JMP L
N: MOV A, C       // ισχύει X<Y, επομένως A=C=Y //
   SUB B          // A=A-B δηλαδή, A=Y-X //
   MOV C, A       // /C=A δηλαδή, Y=Y-X //
   JMP L
K: STA 2002       // είναι X=Y, αποθήκευση στη θέση μνήμης 2002_16 //
   HLT
```

ΑΣΚΗΣΗ 45

Σε ένα μικροϋπολογιστικό σύστημα βασισμένο στον 8085 έχουμε προσαρτήσει κύρια μνήμη οργανωμένη σε μπλοκ των 4 λέξεων, όπου κάθε λέξη είναι του ενός byte. Για την επιτάχυνση της εκτέλεσης του ακόλουθου προγράμματος

```
      MVI B 0A
AGAIN:LXI D, 3000
      LXI H, 2000
NEXT :LDAX D
      ADD M
      STAX D
      INX D
      INR L
      JNZ NEXT
      DCR B
      JNZ AGAIN
```

ένας σχεδιαστής πρότεινε τη χρήση κρυφής μνήμης δεδομένων μεγέθους 4KB, με άμεση οργάνωση και πλαίσια των 4 λέξεων. Ζητείται:

1. Να δώσετε μία σύντομη περιγραφή των λειτουργιών που επιτελεί το πρόγραμμα και να βρείτε το συνολικό αριθμό των προσπελάσεων της μνήμης δεδομένων που απαιτούνται.
2. Να εξηγήσετε γιατί η παραπάνω οργάνωση της κρυφή μνήμης δεν είναι η καλύτερη από άποψη επιτυχών προσπελάσεων στην κρυφή μνήμη και να προτείνετε κάποια καλύτερη με τους ακόλουθους περιορισμούς:
   a. Η πρότασή σας θα πρέπει να είναι το δυνατόν οικονομικότερη και
   b. Δεν πρέπει να αλλάξετε το μέγεθος της κρυφής μνήμης και το μέγεθος του πλαισίου.

3. Να υπολογίστε το ποσοστό επιτυχίας της κρυφής μνήμης στην αρχική και τη δική σας πρόταση.

Θεωρείστε ότι ακολουθείται η τακτική της τελικής ενημέρωσης της κύριας μνήμης και προσκόμιση κατά την εγγραφή.

Λύση:

Ερώτημα 1.

Το πρόγραμμα αποτελείται από 2 loops το ένα εντός του άλλου. Συγκεκριμένα, το εξωτερικό loop είναι αυτό που αντιστοιχεί στην ετικέτα AGAIN και το εσωτερικό είναι αυτό που αντιστοιχεί στην ετικέτα NEXT.

Η επεξεργασία των δεδομένων γίνεται κατά την εκτέλεση του εσωτερικού loop όπου προστίθεται το δεδομένο που βρίσκεται στη θέση μνήμης με διεύθυνση (3000+L) με αυτό της θέσης (2000+L) και το αποτέλεσμα αποθηκεύεται στη θέση μνήμης με διεύθυνση (3000+L).

Το εξωτερικό loop εκτελείται 10 φορές με μετρητή τον καταχωρητή B. Όσον αφορά το εσωτερικό loop, αυτό εκτελείται 256 φορές. Συγκεκριμένα, ο καταχωρητής L αρχικοποιείται στην τιμή $L=00_{16}$ και αυξάνει κατά ένα σε κάθε επανάληψη. Στο τέλος της εκτέλεσης του loop με $L=FF_{16}$ (δηλαδή, μετά από 256 επαναλήψεις) η τιμή του L γίνεται $L=L+1=00_{16}$ και ενεργοποιείται η σημαία μηδενικού αποτελέσματος (δηλαδή, Z=1), τότε το πρόγραμμα μεταβαίνει στην εκτέλεση της εντολής DCR B. Επομένως, το εσωτερικό loop εκτελείται συνολικά 2560 (256 * 10=2560) φορές.

Λόγω της ακολουθίας εντολών LDAX D, ADD M, STAX D γίνονται τρεις προσπελάσεις της μνήμης δεδομένων σε κάθε επανάληψη του εσωτερικού loop. Επομένως, ο συνολικός αριθμός προσπελάσεων του προγράμματος για ανάγνωση και αποθήκευση δεδομένων είναι ίσος με 7680 (3 * 2560=7680).

Ερώτημα 2.

Αφού στη κρυφή μνήμη υπάρχουν 4 λέξεις (bytes) σε κάθε πλαίσιο τα 2 τελευταία ψηφία της διεύθυνσης θα καθορίζουν τη λέξη εντός του πλαισίου. Επίσης, αφού υπάρχουν 4KBytes / 4(bytes/πλαίσιο) = 1024 πλαίσια, τα επόμενα 10 δυαδικά ψηφία της διεύθυνσης θα χρησιμοποιούνται για το καθορισμό πλαισίου. Τέλος, τα 4 σημαντικότερα ψηφία θα αντιστοιχούν στην ετικέτα.

Με την άμεση οργάνωση που προτείνεται τα δεδομένα των θέσεων μνήμης με διευθύνσεις (2000+L) και (3000+L) δε μπορεί να συνυπάρχουν στη κρυφή μνήμη. Συγκεκριμένα, οι διευθύνσεις (2000+L) και (3000+L) που στο δυαδικό έχουν μορφή 0010 0000 $L_7L_6L_5L_4$ $L_3L_2L_1L_0$ και 0011 0000 $L_7L_6L_5L_4$ $L_3L_2L_1L_0$, αντίστοιχα, ανταγωνίζονται για το ίδιο πλαίσιο της κρυφής μνήμης. Συγκεκριμένα ανταγωνίζονται για το πλαίσιο 0000 $L_7L_6L_5L_4$ L_3L_2, και συνεπώς δε μπορούν να συνυπάρχουν.

Μπορούμε να καταφύγουμε σε κρυφή μνήμη συνόλου συσχέτισης ή πλήρους συσχέτισης για να λύσουμε αυτές τις συγκρούσεις. Η πιο οικονομική λύση είναι η κρυφή μνήμη συνόλου συσχέτισης με δύο πλαίσια ανά σύνολο. Σε αυτήν πάλι χρειαζόμαστε τα 2 τελευταία ψηφία της διεύθυνσης για το καθορισμό λέξης αλλά καθώς υπάρχουν 4KBytes / 4(bytes/πλαίσιο) / 2(πλαίσια / σύνολο) = 512 σύνολα, θα χρειαστούμε τα 9 αμέσως επόμενα ψηφία της διεύθυνσης για τον καθορισμό συνόλου. Έτσι οι διευθύνσεις (2000+L) και (3000+L) παρότι απεικονίζονται στο ίδιο σύνολο (000 $L_7L_6L_5L_4$ L_3L_2) μπορούν να συνυπάρχουν στα δύο πλαίσια που αυτό διαθέτει.

Ερώτημα 3

Αρχική περίπτωση:

Τα δεδομένα προσκομίζονται και αποθηκεύονται στην κρυφή μνήμη κατά τετράδες (ένα μπλοκ/πλαίσιο), δηλαδή, όταν προσκομίζετε το δεδομένο της διεύθυνσης 3000_{16} προσκομίζονται και τα δεδομένα των διευθύνσεων 3001_{16}, 3002_{16} και 3003_{16}. Το ίδιο συμβαίνει και όταν γίνεται προσπέλαση στις θέσεις 2001_{16}, 2002_{16} και 2003_{16}.

Ας εξετάσουμε τι συμβαίνει στις τέσσερις πρώτες επαναλήψεις του εσωτερικού loop, όπου γίνεται επεξεργασία στα δεδομένα των θέσεων 3000_{16} -3003_{16} και 2000_{16} -2003_{16}. Λόγω του ότι οι εντολές ADD M, STAX D μάχονται για το ίδιο πλαίσιο της κρυφής μνήμης δε θα έχουν ποτέ επιτυχία (τα αντίστοιχα δεδομένα δε θα βρίσκονται ποτέ ταυτόχρονα στην κρυφή μνήμη). Όμως όσον αφορά την εντολή LDAX D μετά την πρώτη επανάληψη και δεδομένου ότι η προηγούμενη προσπέλαση μνήμης οφείλεται στην STAX D θα είναι επιτυχής. Το ίδιο ισχύει και για τις επόμενες τρεις επαναλήψεις. Επομένως, για τις τέσσερις πρώτες επαναλήψεις έχουμε 3 επιτυχίες.

Κάθε τέσσερις επαναλήψεις η επεξεργασία αναφέρεται σε διαφορετική τετράδα διευθύνσεων (μπλοκ) στη μνήμη και τα προηγούμενα ισχύουν. Επομένως, ανά τέσσερις επαναλήψεις έχουμε μόνο τρεις επιτυχίες. Επεκτείνοντας αυτό στις 256 επαναλήψεις του εσωτερικού loop έχουμε (256/4) * 3 = 64 * 3 = 192 επιτυχίες. Άρα, μετά την ολοκλήρωση της πρώτης επανάληψης του εξωτερικού loop ο αριθμός των επιτυχών προσπελάσεων είναι ίσος με 192.

Μετά την πρώτη εκτέλεση του εξωτερικού loop η κρυφή μνήμη περιέχει τα δεδομένα των θέσεων 3000-30FF. Επομένως, για κάθε επόμενη εκτέλεση του εσωτερικού loop έχουμε 1 hit (εντολή LDAX D) και 2 miss (ακολουθία εντολών ADD M, STAX D). Οι επιτυχείς προσπελάσεις της LDAX D οφείλονται στο γεγονός ότι σε κάθε επανάληψη του εξωτερικού loopη αρχική τιμή του ζεύγους καταχωρητών (D, E) είναι ίση με 3000_{16} και αυξάνει κατά 1 (μέχρι την τιμή 30FF) σε κάθε εσωτερική επανάληψη.

Δεδομένου ότι το εξωτερικό loop εκτελείται 9 επιπλέον φορές, ο αριθμός των επιτυχών προσπελάσεων για τις επόμενες 9 επαναλήψεις (εξαιρουμένης της πρώτης) του εξωτερικού loop είναι 9*(1*256)= 9*256=2304.

Τέλος για όλο το πρόγραμμα ο συνολικός αριθμός των επιτυχών προσπελάσεων είναι 2496 (192+2304=2496). Άρα το ποσοστό επιτυχίας, το οποίο δίνεται από το λόγο (*αριθμός επιτυχών προσπελάσεων*) / (*συνολικό αριθμό προσπελάσεων*), θα είναι 2496 / 7680 = 32,5%.

Στη περίπτωση κρυφής μνήμης συνόλου συσχέτισης με 2 πλαίσια ανά σύνολο έχουμε :

Στην πρώτη επανάληψη του εσωτερικού loop, οι προσπελάσεις μνήμης λόγω των εντολών LDAX D και ADD M είναι ανεπιτυχείς, ενώ αυτή της STAX D επιτυχής. Συγκεκριμένα, κατά την εκτέλεση της LDAX D προσκομίζονται και αποθηκεύονται στο ένα (έστω $1^{ο}$) πλαίσιο του συνόλου 0 τα δεδομένα με διευθύνσεις 3000_{16} -3003_{16}, ενώ κατά την εκτέλεσης της ADD M προσκομίζονται και αποθηκεύονται στο $2^{ο}$ πλαίσιο του συνόλου 0 τα δεδομένα με διευθύνσεις 2000_{16} -2003_{16}. Άρα για τις τρεις επόμενες επαναλήψεις οι προσπελάσεις μνήμης λόγω των εντολών LDAX D, ADD M, STAX D γίνονται πάντα από την κρυφή μνήμη. Επομένως, για τις τέσσερις πρώτες επαναλήψεις του εσωτερικού loop έχουμε 10 (1+ 3 * 3) επιτυχίες. Άρα για τις πρώτες 256 επαναλήψεις, που αντιστοιχούν στην πρώτη εκτέλεση του εξωτερικού loop, έχουμε (256 * 10)/4= 640 επιτυχίες.

Μετά την πρώτη εκτέλεση του εξωτερικού loop η κρυφή μνήμη είναι πλήρης και περιέχει τα δεδομένα των διευθύνσεων (3000-30FF) και (2000-20FF). Επομένως, για τις υπόλοιπες 9 εκτελέσεις του εξωτερικού loop οι προσπελάσεις είναι πάντα επιτυχείς. Δηλαδή, έχουμε 9*(3*256)=9*768=6912 hits.

Τέλος για όλο το πρόγραμμα ο συνολικός αριθμός των επιτυχών προσπελάσεων είναι 7552 (640+6912=7552). Άρα το ποσοστό επιτυχίας, το οποίο δίνεται από το λόγο (*αριθμός επιτυχών προσπελάσεων*) / (*συνολικό αριθμό προσπελάσεων*), θα είναι 7552 / 7680 = 98,3%.

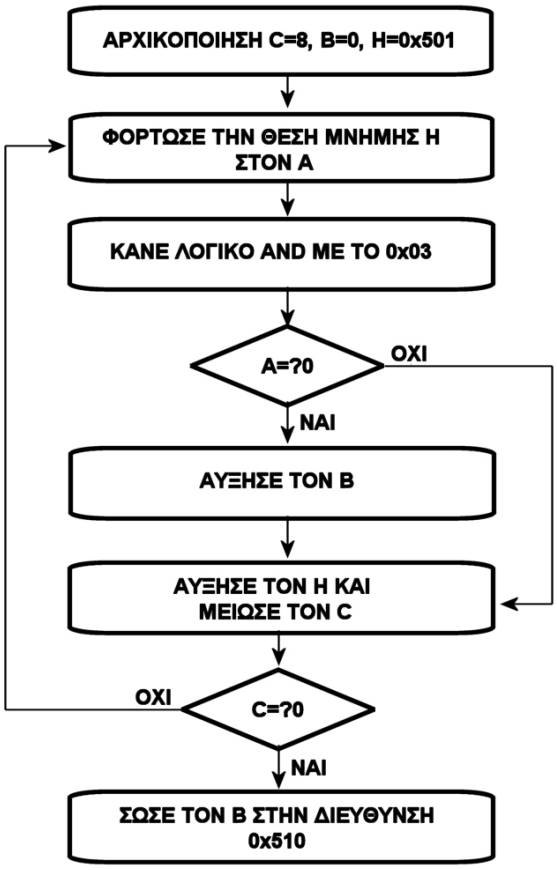
ΑΣΚΗΣΗ 46

Έστω μία λίστα οκτώ 16-bit αριθμών στις θέσεις 0x500-0x501, 0x502-0x503, 0x504-0x505, 0x506-0x507, 0x508-0x509, 0x50A-0x50B, 0x50C-0x50D, 0x50E-0x50F. Το σημαντικότερο byte κάθε αριθμού βρίσκεται στην μικρότερη διεύθυνση μνήμης (π.χ. στον πρώτο αριθμό είναι αυτό που έχει αποθηκευτεί στην διεύθυνση 0x500 και το δεύτερο σημαντικότερο στη θέση 0x501 κ.ο.κ.). Να γραφεί πρόγραμμα assembly για τον μικροεπεξεργαστή Intel 8085 που να αποθηκεύει στη μνήμη, στην θέση 0x510, πόσοι από αυτούς τους αριθμούς είναι ακέραια πολλαπλάσια του 4.

Σημείωση: Να επισυνάψετε στις απαντήσεις σας και το αρχείο με τον κώδικα assembly

Λύση:

Για την υλοποίηση του προγράμματος θα εκμεταλλευτούμε το γεγονός ότι αν τα δύο λιγότερα σημαντικά ψηφία ενός αριθμού είναι και τα δύο μηδέν τότε ο αριθμός είναι ακέραιο πολλαπλάσιο του 4. Έτσι το πρόγραμμα θα ελέγχει από κάθε αριθμό το λιγότερο σημαντικό byte και αν τα 2 λιγότερα σημαντικά ψηφία είναι μηδέν τότε θα αυξάνουμε έναν μετρητή, τον οποίο στο τέλος θα αποθηκεύσουμε στην θέση μνήμης 0x510. Έτσι το διάγραμμα ροής του προγράμματος είναι το παρακάτω.



Μια ενδεικτική υλοποίηση του διαγράμματος ροής δίνεται στον παρακάτω κώδικα assembly για τον μικροεπεξεργαστή Intel 8085. Σημειώνεται ότι ο καταχωρητής Α στον επεξεργαστή 8085 είναι ο συσσωρευτής(accumulator).

```
    LXI H 0501  21      ; Φορτώνουμε στον καταχωρητή HL την τιμή 0x501
                        ;(διεύθυνση λιγότερου σημαντικού Byte του 1ου αποθηκευμένου
                        ; αριθμού)
    MVI B 00    06      ; Μηδενίζουμε τον καταχωρητή B (στον οποίο θα αποθηκευτεί
                        ; το πλήθος πολλαπλάσιων του 4)
    MVI C 08    0E      ; Φορτώνουμε στον καταχωρητή C την τιμή 8
                        ; (πλήθος 16bit αριθμών που θέλουμε να ελέγξουμε)
L: MOV A M      7E      ; Φορτώνουμε τα δεδομένα της μνήμης από τη διεύθυνση που δείχνει
                        ; ο καταχωρητής H
    ANI 03      E6 ;    ; Κάνε AND με τον αριθμό 0x03 του λιγότερου σημαντικού byte
    JNZ M     C2 ; Εάν το αποτέλεσμα δεν είναι 0 τότε πήγαινε στην ετικέτα M
    INR B       04      ; Εάν το αποτέλεσμα είναι μηδέν, τότε βρήκαμε πολλαπλάσιο του 4
                        ; και άρα αύξησε κατά ένα τον μετρητή
M: INX H        23      ; Αύξησε κατά 1 την διεύθυνση μνήμης (πήγαινε στο περισσότερο
                        ; σημαντικό byte του επόμενου αριθμού)
    INX H       23      ; Αύξησε κατά 1 την διεύθυνση μνήμης (πήγαινε στο λιγότερο
                        ; σημαντικό byte του επόμενου αριθμού)
    DCR C       0D      ; Μείωσε κατά ένα τον καταχωρητή C
    JNZ L       C2      ; Εάν ο C είναι μηδέν, τότε τελειώσαμε με τους αριθμούς
    MOV A B  78         ; Μετακίνησε τον B στον A
    STA  0510 32        ; Σώσε την τιμή του A στην διεύθυνση 0x510
    HLT         76      ; Τέλος προγράμματος
```

Διασύνδεση περιφερειακών συσκευών

ΑΣΚΗΣΗ 47

Σε έναν επεξεργαστή Motorola 6800 να συνδέσετε ως περιφερειακές μονάδες δυο 8-ψηφίων καταχωρητές Α και Β. Οι καταχωρητές τοποθετούνται στην ίδια διεύθυνση μνήμης ΑΧΧΧ και οι δυο. Ο καταχωρητής Α θα χρησιμοποιηθεί για την εγγραφή σε αυτόν δεδομένων από τον επεξεργαστή και ο Β για την ανάγνωση από αυτόν δεδομένων. Οι καταχωρητές διαθέτουν σήμα ελέγχου επίτρεψης εγγραφής. Χρησιμοποιείστε στοιχεία τριών καταστάσεων για να πραγματοποιήσετε συνδέσεις όπου απαιτείται.
Ακολουθώντας το χρονισμό για τους κύκλους ανάγνωσης και εγγραφής που παρουσιάζονται στα σχήματα 4.5 και 4.6 (Τόμος Γ) να καθορίσετε κατάλληλα τα σήματα ελέγχου των μονάδων. Σχεδιάστε το σύστημα.

Λύση:
Όπως παρατηρούμε στο σχήμα 4.5 κατά τον κύκλο ανάγνωσης τα σήματα ελέγχου R/W' και VMA παίρνουν την τιμή '1'. Επίσης η τιμή της διεύθυνσης είναι σταθερή στο δίαυλο διευθύνσεων καθ' όλη τη διάρκεια του κύκλου.

Όπως παρατηρούμε στο σχήμα 4.6 κατά τον κύκλο εγγραφής το σήμα ελέγχου R/W' παίρνει την τιμή '0' ενώ το σήμα ελέγχου VMA παίρνει την τιμή '1'. Επίσης η τιμή της διεύθυνσης είναι σταθερή στο δίαυλο διευθύνσεων καθ' όλη τη διάρκεια του κύκλου ενώ τα δεδομένα εμφανίζονται στον δίαυλο δεδομένων προς το τέλος του κύκλου όταν το σήμα ελέγχου DBE είναι '1'. Το σήμα DBE είναι είσοδος στον επεξεργαστή και συνήθως οδηγείται από το σήμα ρολογιού Φ2.Η εγγραφή στους καταχωρητές γίνεται με την αρνητική ακμή του ρολογιού Φ_2.

Τα παραπάνω σήματα ελέγχου θα πρέπει να συνδυαστούν ώστε να δημιουργηθούν τα κατάλληλα σήματα ελέγχου για τους καταχωρητές.

Αρχικά πρέπει να υλοποιήσουμε τη διευθυνσιοδότηση των καταχωρητών. Με ΑΧΧΧ θα διευθυνσιοδοτούνται και οι δυο καταχωρητές. Επομένως στην παραγωγή του CS (σήμα επιλογής καταχωρητή) θα συμμετάσχουν τα ψηφία A_{15}, A_{14}, A_{13} και A_{12} και θα είναι:

$$CS = A_{15}\overline{A}_{14}A_{13}\overline{A}_{12}$$

ώστε οι καταχωρητές να επιλέγονται όταν τα τέσσερα πιο σημαντικά ψηφία του δίαυλου διεύθυνσης έχουν την τιμή Α.

Στον καταχωρητή Α θα εγγράφεται πληροφορία. Επομένως η είσοδος του καταχωρητή Α θα πρέπει να συνδέεται απευθείας στο δίαυλο δεδομένων. Από τον καταχωρητή Β θα διαβάζεται πληροφορία. Επομένως θα πρέπει η έξοδος του Β να συνδέεται στο δίαυλο δεδομένων. Αυτό όμως δεν μπορεί να συμβεί απευθείας. Αυτή θα πρέπει να συνδεθεί με τη μεσολάβηση στοιχείων τριών καταστάσεων. Τα στοιχεία αυτά έχουν ένα σήμα ελέγχου, Ε, και όταν αυτό είναι ενεργοποιημένο, π.χ. Ε=1, τότε μετάγουν την τιμή της εισόδου τους στην έξοδο, διαφορετικά η έξοδος είναι απομονωμένη από την είσοδο (οδηγείται από κάποιο άλλο στοιχείο). Κάθε ψηφίο εξόδου του καταχωρητή Β συνδέεται μέσω ενός στοιχείου τριών καταστάσεων σε ένα ψηφίο του δίαυλου δεδομένων. Και τα 8 στοιχεία που συνδέουν τις εξόδους του καταχωρητή Β στο δίαυλο δεδομένων έχουν κοινό σήμα ελέγχου. Επομένως απαιτείται ένα σήμα ελέγχου *EB* για τα στοιχεία τριών καταστάσεων που σχετίζονται με τον καταχωρητή Β.

Για την περίπτωση εγγραφής σε καταχωρητή θα πρέπει να επιλεγεί ο καταχωρητής όπου πρέπει να γραφτεί πληροφορία (CS=1), το σήμα VME να γίνει '1' και το σήμα R/W' να γίνει '0'. Επομένως, για να γράψω πληροφορία στον καταχωρητή Α, το σήμα ελέγχου επίτρεψης εγγραφής στον Α, *ENA (Enable_A)*, θα πρέπει να γίνει '1'. Συνεπώς

$$ENA = CS \cdot VMA \cdot \overline{R/W'}$$

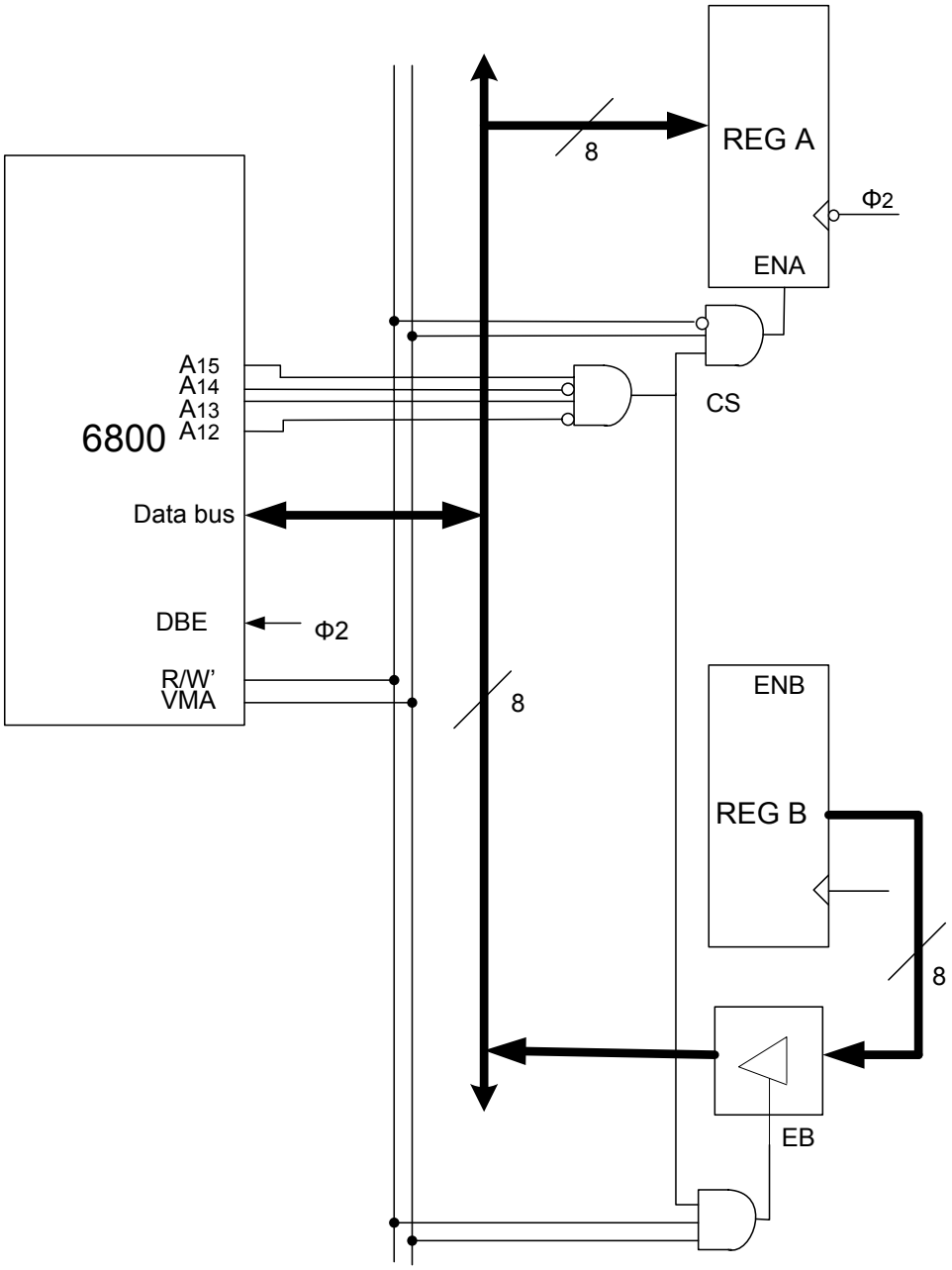
Ο καταχωρητής Α θα παίρνει ως σήμα ρολογιού το Φ_2. Η εγγραφή θα γίνεται στην αρνητική ακμή αυτού.

Για την ανάγνωση θα πρέπει να επιλεγεί η διάταξη των στοιχείων τριών καταστάσεων ώστε να συνδεθεί η έξοδος του καταχωρητή Β στο δίαυλο δεδομένων. Κατά την ανάγνωση έχουμε VMA=1 και R/W'=1. Επομένως

$$EB = CS \cdot VMA \cdot R/W'$$

Παρατηρούμε ότι αν και οι δυο καταχωρητές επιλέγονται με το ίδιο σήμα CS, το σήμα ελέγχου R/W' καθορίζει σε ποιον από τους δυό απευθύνεται η συγκεκριμένη λειτουργία. Καθώς η εγγραφή στον καταχωρητή Β δεν καθορίζεται από τον επεξεργαστή δεν μας ενδιαφέρει στην περίπτωση αυτή πιο σήμα ρολογιού απευθύνεται σε αυτόν ούτε το σήμα επίτρεψης φόρτωσης.

Το σχήμα της διάταξης δίνεται παρακάτω. Με παχιά γραμμή παριστάνεται ο 8-ψηφίων δίαυλος δεδομένων και οι διασυνδέσεις του.



ΑΣΚΗΣΗ 48

Ο μικροεπεξεργαστής 8085 περιλαμβάνει ενσωματωμένες τις λειτουργίες της γεννήτριας ρολογιού και του ελεγκτή συστήματος του μικροεπεξεργαστή 8080. Δώστε τα σήματα του 8085 (όπως το σχήμα 3.3 (Τόμος Γ))

Λύση:
Όλα τα σήματα του μικροεπεξεργαστή 8085 φαίνονται στο παρακάτω διάγραμμα.

X1	1	40	VCC
X2	2	39	HOLD
RESET OUT	3	38	HLDA
SOD	4	37	CLK (OUT)
SID	5	36	RESET IN
TRAP	6	35	READY
RST 7.5	7	34	IO/M
RST 6.5	8	33	S1
RST 5.5	9	32	RD
INTR	10	31	WR
INTA	11	30	ALE
AD0	12	29	S0
AD1	13	28	A15
AD2	14	27	A14
AD3	15	26	A13
AD4	16	25	A12
AD5	17	24	A11
AD6	18	23	A10
AD7	19	22	A9
VSS	20	21	A8

ΑΣΚΗΣΗ 49

Σε έναν επεξεργαστή Motorola 6800 συνδέστε ως περιφερειακές μονάδες 4 καταχωρητές (Κ1, Κ2, Κ3 και Κ4) των 8 ψηφίων. Οι καταχωρητές να συνδεθούν έτσι ώστε ο Κ1 να μπορεί να χρησιμοποιηθεί μόνο για εγγραφή δεδομένων, ο Κ2 μόνο για ανάγνωση δεδομένων και οι Κ3 και Κ4 και για εγγραφή και για ανάγνωση. Οι καταχωρητές διαθέτουν σήμα ελέγχου επίτρεψης εγγραφής. Για τη σύνδεση των καταχωρητών στο σύστημα διευθύνσεων μνήμης να χρησιμοποιηθεί αποκωδικοποιητής 2-σε-4 με είσοδο ενεργοποίησης. Οι καταχωρητές να τοποθετηθούν στις διευθύνσεις μνήμης Ε000Η – Ε003Η.
Σχεδιάστε το παραπάνω σύστημα.

Υπόδειξη: Μελετώντας το χρονισμό για τους κύκλους ανάγνωσης και εγγραφής που παρουσιάζονται στα σχήματα 4.5 και 4.6 (Τόμος Γ) να καθορίσετε κατάλληλα τα σήματα ελέγχου των μονάδων.

Λύση
Όπως παρατηρούμε στο σχήμα 4.5 κατά τον κύκλο ανάγνωσης τα σήματα ελέγχου R/W' και VMA παίρνουν την τιμή '1'. Επίσης η τιμή της διεύθυνσης είναι σταθερή στο δίαυλο διευθύνσεων καθ' όλη τη διάρκεια του κύκλου.

Όπως παρατηρούμε στο σχήμα 4.6 κατά τον κύκλο εγγραφής το σήμα ελέγχου R/W' παίρνει τη τιμή '0' ενώ το σήμα ελέγχου VMA παίρνει την τιμή '1'. Επίσης η τιμή της διεύθυνσης είναι σταθερή στο δίαυλο διευθύνσεων καθ' όλη τη διάρκεια του κύκλου ενώ τα δεδομένα εμφανίζονται στο δίαυλο δεδομένων προς το τέλος του κύκλου όταν το σήμα ελέγχου DBE είναι '1'. Το σήμα DBE είναι είσοδος στον επεξεργαστή και συνήθως οδηγείται από το σήμα ρολογιού Φ2. Η εγγραφή στους καταχωρητές γίνεται με την αρνητική ακμή του ρολογιού Φ_2.

Τα παραπάνω σήματα ελέγχου θα πρέπει να συνδυαστούν ώστε να δημιουργηθούν τα κατάλληλα σήματα ελέγχου για τους καταχωρητές.

Αρχικά θα πρέπει να υλοποιήσουμε τη διευθυνσιοδότηση των καταχωρητών. Οι καταχωρητές καταλαμβάνουν τις τέσσερις θέσεις Ε000Η – Ε0003 του συστήματος μνήμης. Όταν εμφανίζεται μία από αυτές τις διευθύνσεις στο δίαυλο διευθύνσεων τότε θα δημιουργείται

σήμα επιλογής καταχωρητή (CS). Παρατηρούμε επίσης ότι τα 14 πιο σημαντικά ψηφία των διευθύνσεων είναι κοινά και για τους τέσσερις καταχωρητές και ότι η διαφοροποίηση γίνεται στα δυο τελευταία ψηφία. Επομένως τα 14 πιο σημαντικά ψηφία θα χρησιμοποιηθούν για την επιλογή της τετράδας των καταχωρητών ενώ τα δυο τελευταία για την επιλογή κάθε ενός από αυτούς.

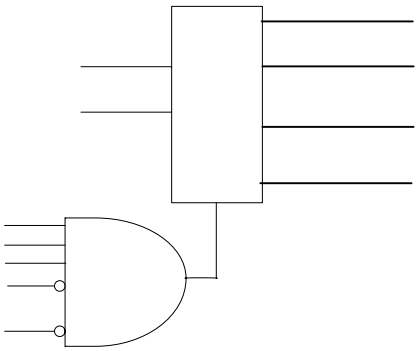
Άρα, τα ψηφία του διαύλου διευθύνσεων A_1 και A_0 θα πρέπει να αποτελούν τις εισόδους στον αποκωδικοποιητή για την επιλογή μεταξύ των καταχωρητών. Τα τέσσερα σήματα εξόδου του αποκωδικοποιητή αποτελούν τα σήματα CS των τεσσάρων καταχωρητών. Κάθε φορά ένα μόνο από αυτά μπορεί να είναι ενεργό.

Ο αποκωδικοποιητής πρέπει να λειτουργεί παρέχοντας σήματα CS μόνο όταν τα ψηφία του διαύλου διευθύνσεων $A_2 - A_{12}$ είναι 0 και τα A_{13}, A_{14} και A_{15} είναι 1, δηλαδή όταν το λογικό AND των

$$A_{15}, A_{14}, A_{13}, \overline{A}_{12}, \overline{A}_{11}, \overline{A}_{10}, \overline{A}_{9}, \overline{A}_{8}, \overline{A}_{7}, \overline{A}_{6}, \overline{A}_{5}, \overline{A}_{4}, \overline{A}_{3}, \overline{A}_{2}$$

είναι «1». Συνεπώς στην είσοδο επίτρεψης του καταχωρητή (En) θα συνδεθεί η έξοδος μιας AND με εισόδους τα παραπάνω αντίστοιχα ψηφία του δίαυλου διευθύνσεων σε κανονική ή αντιστραμμένη μορφή, όπως εμφανίζονται. (Για την υλοποίηση μιας πύλης τόσων εισόδων απαιτείται η χρήση δένδρου από πύλες με αριθμό εισόδων πύλης που συνήθως δεν ξεπερνά το 4).

Το κύκλωμα παραγωγής των σημάτων επιλογής (CS) των καταχωρητών φαίνεται στο παρακάτω σχήμα.



Οι είσοδοι των καταχωρητών στους οποίους γίνεται εγγραφή (K1, K3 και K4) συνδέονται απευθείας στο δίαυλο δεδομένων. Αυτό όμως δεν μπορεί να συμβεί με τις εξόδους των καταχωρητών. Αυτές θα πρέπει να συνδεθούν με τη μεσολάβηση στοιχείων τριών καταστάσεων. Τα στοιχεία αυτά έχουν ένα σήμα ελέγχου, E, και όταν αυτό είναι ενεργοποιημένο, π.χ. E=1, τότε μετάγουν την τιμή της εισόδου τους στην έξοδο, διαφορετικά η έξοδος είναι απομονωμένη από την είσοδο (οδηγείται από κάποιο άλλο στοιχείο). Κάθε ψηφίο εξόδου ενός καταχωρητή συνδέεται μέσω ενός στοιχείου τριών καταστάσεων σε ένα ψηφίο του δίαυλου δεδομένων. Και τα 8 στοιχεία που συνδέουν τις εξόδους ενός καταχωρητή στον δίαυλο δεδομένων έχουν κοινό σήμα ελέγχου. Επομένως απαιτούνται τρία σήματα ελέγχου EK2, EK3 και EK4 για τους καταχωρητές (K2, K3, K4) από τους οποίους διαβάζονται δεδομένα.

Για την περίπτωση εγγραφής σε καταχωρητή θα πρέπει να επιλεγεί ο καταχωρητής όπου πρέπει να γραφτεί πληροφορία, το σήμα VME να γίνει ‘1’ και το σήμα R/W’ να γίνει ‘0’. Επομένως, για να γράψουμε πληροφορία στον καταχωρητή K1, το σήμα ελέγχου επίτρεψης εγγραφής στον K1, *ENK1 (Enable_K1)*, θα πρέπει να γίνει ‘1’. Συνεπώς

$$ENK1 = CSK1 \cdot VMA \cdot \overline{R/W'}$$

Για το σήμα ελέγχου επίτρεψης εγγραφής στους Κ3 και Κ4, θα είναι:

$$ENK3 = CSK3 \cdot VMA \cdot \overline{R/W'}$$

$$ENK4 = CSK4 \cdot VMA \cdot \overline{R/W'}$$

Οι καταχωρητές θα παίρνουν ως σήμα ρολογιού το Φ_2. Η εγγραφή θα γίνεται στην αρνητική ακμή αυτού.

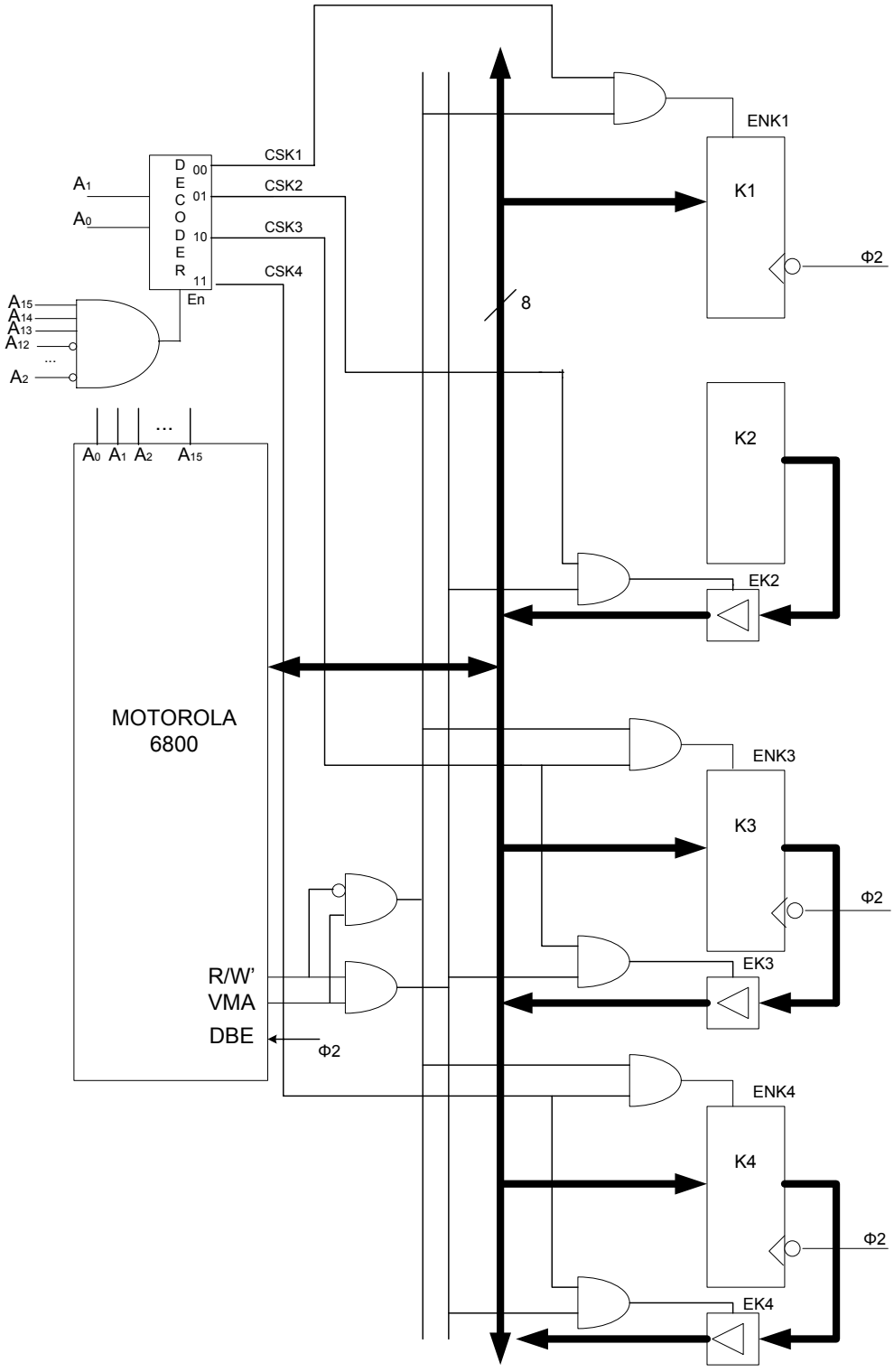
Για την ανάγνωση θα πρέπει να επιλεγεί η αντίστοιχη διάταξη στοιχείων τριών καταστάσεων ώστε να συνδεθεί η έξοδος του ζητούμενου καταχωρητή στο δίαυλο δεδομένων. Κατά την ανάγνωση έχουμε VMA=1 και R/W'=1. Επομένως

$$EK2 = CSK2 \cdot VMA \cdot R/W'$$

$$EK3 = CSK3 \cdot VMA \cdot R/W'$$

$$EK4 = CSK4 \cdot VMA \cdot R/W'$$

Το σχήμα της διάταξης δίνεται παρακάτω. Με έντονη γραμμή παριστάνεται ο 8-ψηφίων δίαυλος δεδομένων και οι συνδέσεις του.


A1
A0
DECODER
00
01
10
11
En
CSK1
CSK2
CSK3
CSK4
A15
A14
A13
A12
A2
A0 A1 A2 A15
MOTOROLA
6800
R/W'
VMA
DBE
Φ2
8
ENK1
K1
Φ2
K2
EK2
ENK3
K3
Φ2
EK3
ENK4
K4
Φ2
EK4

ΑΣΚΗΣΗ 50

Σε έναν επεξεργαστή Motorola 6800 συνδέστε ως περιφερειακές μονάδες 2 καταχωρητές (Κ1 και Κ2) των 8 δυαδικών ψηφίων. Οι καταχωρητές να συνδεθούν έτσι ώστε ο Κ1 να μπορεί να χρησιμοποιηθεί μόνο για εγγραφή δεδομένων και ο Κ2 και για εγγραφή και για ανάγνωση. Οι καταχωρητές διαθέτουν σήμα ελέγχου επίτρεψης εγγραφής (ενεργή στο λογικό 1). Έχετε στη διάθεση σας και στοιχεία τριών καταστάσεων των οποίων η έξοδος είναι σε κατάσταση υψηλής αντίστασης (εμπέδησης) όταν η είσοδος ελέγχου τους είναι στο λογικό 0. Οι καταχωρητές να τοποθετηθούν στις διευθύνσεις μνήμης `1FFEH – 1FFFH`. Σχεδιάστε το παραπάνω σύστημα.

Υπόδειξη: Μελετώντας το χρονισμό για τους κύκλους ανάγνωσης και εγγραφής που παρουσιάζονται στα σχήματα 4.5 και 4.6 (Τόμος Γ) να καθορίσετε κατάλληλα τα σήματα ελέγχου των μονάδων.

Λύση

Όπως παρατηρούμε στο σχήμα 4.5 του Τόμου Γ' κατά τον κύκλο ανάγνωσης τα σήματα ελέγχου R/W' και VMA παίρνουν την τιμή '1'. Επίσης η τιμή της διεύθυνσης είναι σταθερή στο δίαυλο διευθύνσεων καθ' όλη τη διάρκεια του κύκλου.

Όπως παρατηρούμε στο σχήμα 4.6 κατά τον κύκλο εγγραφής το σήμα ελέγχου R/W' παίρνει τη τιμή '0' ενώ το σήμα ελέγχου VMA παίρνει την τιμή '1'. Επίσης η τιμή της διεύθυνσης είναι σταθερή στο δίαυλο διευθύνσεων καθ' όλη τη διάρκεια του κύκλου ενώ τα δεδομένα εμφανίζονται στο δίαυλο δεδομένων προς το τέλος του κύκλου όταν το σήμα ελέγχου DBE είναι '1'. Το σήμα DBE είναι είσοδος στον επεξεργαστή και συνήθως οδηγείται από το σήμα ρολογιού Φ2. Η εγγραφή στους καταχωρητές γίνεται με την αρνητική ακμή του ρολογιού Φ_2.

Τα παραπάνω σήματα ελέγχου θα πρέπει να συνδυαστούν ώστε να δημιουργηθούν τα κατάλληλα σήματα ελέγχου για τους καταχωρητές.

Αρχικά θα πρέπει να υλοποιήσουμε τη διευθυνσιοδότηση των καταχωρητών. Οι καταχωρητές καταλαμβάνουν τις δύο θέσεις 1FFEH και 1FFFH του συστήματος μνήμης. Όταν εμφανίζεται μία από αυτές τις διευθύνσεις στο δίαυλο διευθύνσεων τότε θα δημιουργείται σήμα επιλογής καταχωρητή (CS). Παρατηρούμε επίσης ότι τα 15 πιο σημαντικά ψηφία των διευθύνσεων είναι κοινά και για τους δύο καταχωρητές και ότι η διαφοροποίηση γίνεται στο τελευταίο ψηφίο. Επομένως τα 15 πιο σημαντικά ψηφία θα χρησιμοποιηθούν για την επιλογή του ζεύγους των καταχωρητών (CS) ενώ το τελευταίο για την επιλογή κάθε ενός από αυτούς (CSK1 ή CSK2).

Άρα,

$$\mathrm{CS} = \overline{A}_{15}\overline{A}_{14}\overline{A}_{13}A_{12}A_{11}A_{10}A_9A_8A_7A_6A_5A_4A_3A_2A_1$$

και

$$\mathrm{CSK1} = \mathrm{CS}\,\overline{A}_0, \qquad CSK2 = CS\,A_0$$

Οι είσοδοι των καταχωρητών στους οποίους γίνεται εγγραφή (Κ1 και Κ2) συνδέονται απευθείας στο δίαυλο δεδομένων. Αυτό όμως δεν μπορεί να συμβεί με τις εξόδους των καταχωρητών. Αυτές θα πρέπει να συνδεθούν με τη μεσολάβηση στοιχείων τριών καταστάσεων. Τα στοιχεία αυτά έχουν ένα σήμα ελέγχου, Ε, και όταν αυτό είναι ενεργοποιημένο, π.χ. Ε=1, τότε μετάγουν την τιμή της εισόδου τους στην έξοδο, διαφορετικά η έξοδος είναι απομονωμένη από την είσοδο (οδηγείται από κάποιο άλλο στοιχείο). Κάθε ψηφίο εξόδου ενός καταχωρητή συνδέεται μέσω ενός στοιχείου τριών καταστάσεων σε ένα

ψηφίο του δίαυλου δεδομένων. Και τα 8 στοιχεία που συνδέουν τις εξόδους ενός καταχωρητή στον δίαυλο δεδομένων έχουν κοινό σήμα ελέγχου. Επομένως απαιτείται ένα επιπλέον σήμα ελέγχου ΕΚ2 για τον καταχωρητή από τον οποίο διαβάζονται δεδομένα.

Για την περίπτωση εγγραφής σε καταχωρητή θα πρέπει να επιλεγεί ο καταχωρητής όπου πρέπει να γραφτεί πληροφορία, το σήμα VME να γίνει '1' και το σήμα R/W' να γίνει '0'. Επομένως, για να γράψουμε πληροφορία στον καταχωρητή Κ1, το σήμα ελέγχου επίτρεψης εγγραφής στον Κ1, *ENK1 (Enable_K1)*, θα πρέπει να γίνει '1'. Συνεπώς

$$ENK1 = CSK1 \cdot VMA \cdot \overline{R/W}'$$

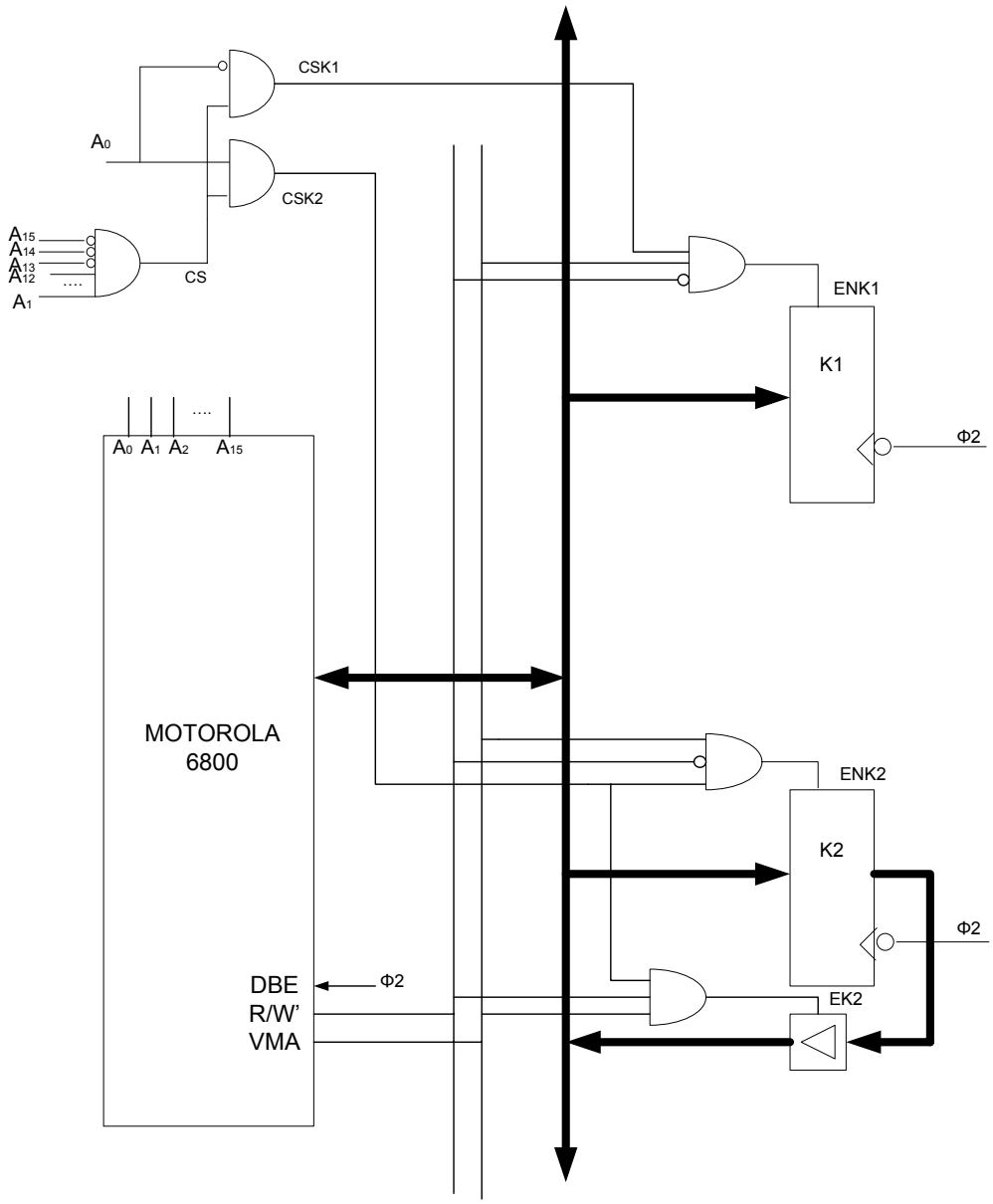
Για το σήμα ελέγχου επίτρεψης εγγραφής στον Κ2, θα είναι:

$$ENK2 = CSK2 \cdot VMA \cdot \overline{R/W}'$$

Οι καταχωρητές θα παίρνουν ως σήμα ρολογιού το Φ_2. Η εγγραφή θα γίνεται στην αρνητική ακμή αυτού.
Για την ανάγνωση θα πρέπει να επιλεγεί η αντίστοιχη διάταξη στοιχείων τριών καταστάσεων ώστε να συνδεθεί η έξοδος του καταχωρητή Κ2 στο δίαυλο δεδομένων. Κατά την ανάγνωση έχουμε VMA=1 και R/W'=1. Επομένως

$$EK2 = CSK2 \cdot VMA \cdot R/W'$$

Το σχήμα της διάταξης δίνεται παρακάτω. Με έντονη γραμμή παριστάνεται ο 8-ψηφίων δίαυλος δεδομένων και οι συνδέσεις του.



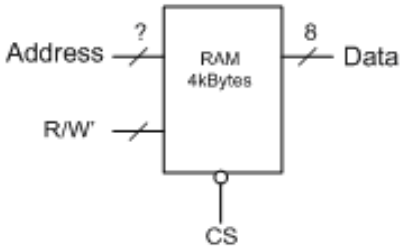
ΑΣΚΗΣΗ 51

Σ' ένα μικροϋπολογιστή βασισμένο στον 6800 της Motorola θέλουμε να συνδέσουμε ένα ολοκληρωμένο κύκλωμα μνήμης RAM χωρητικότητας 4kBytes από τη διεύθυνση `E000` hex και μετά. (Στο διάγραμμα του ολοκληρωμένου που δείχνεται στο σχήμα που ακολουθεί με CS (Chip Select) συμβολίζεται η είσοδος επιλογής ολοκληρωμένου).

α. Ποιο το πλήθος των bits που χρειάζονται για την προσπέλαση της μνήμης;

β. Ποια περιοχή της μνήμης θα καλύπτεται; (Δώστε την αρχική και τελική διεύθυνση σε hex).

γ. Σχεδιάστε το διάγραμμα το του κυκλώματος.



Υπόδειξη: Συμβουλευτείτε τα διαγράμματα χρονισμού για τους κύκλους ανάγνωσης και εγγραφής που παρουσιάζονται στα σχήματα 4.5 και 4.6 του Τόμου Γ.

Λύση

α. Εφόσον η μνήμη έχει μέγεθος 4K, απαιτούνται 12bits για την διευθυνσιοδότηση της, καθώς 2^{12}=4K.

β. Η περιοχή μνήμης που θα καλύπτει θα είναι η E000-EFFF.

γ. Όπως παρατηρούμε στο σχήμα 4.5 του Τόμου Γ' κατά τον κύκλο ανάγνωσης τα σήματα ελέγχου R/W' και VMA παίρνουν την τιμή '1'. Επίσης η τιμή της διεύθυνσης είναι σταθερή στο δίαυλο διευθύνσεων καθ' όλη τη διάρκεια του κύκλου.

Όπως παρατηρούμε στο σχήμα 4.6 κατά τον κύκλο εγγραφής το σήμα ελέγχου R/W' παίρνει τη τιμή '0' ενώ το σήμα ελέγχου VMA παίρνει την τιμή '1'. Επίσης η τιμή της διεύθυνσης είναι σταθερή στο δίαυλο διευθύνσεων καθ' όλη τη διάρκεια του κύκλου ενώ τα δεδομένα εμφανίζονται στο δίαυλο δεδομένων προς το τέλος του κύκλου.

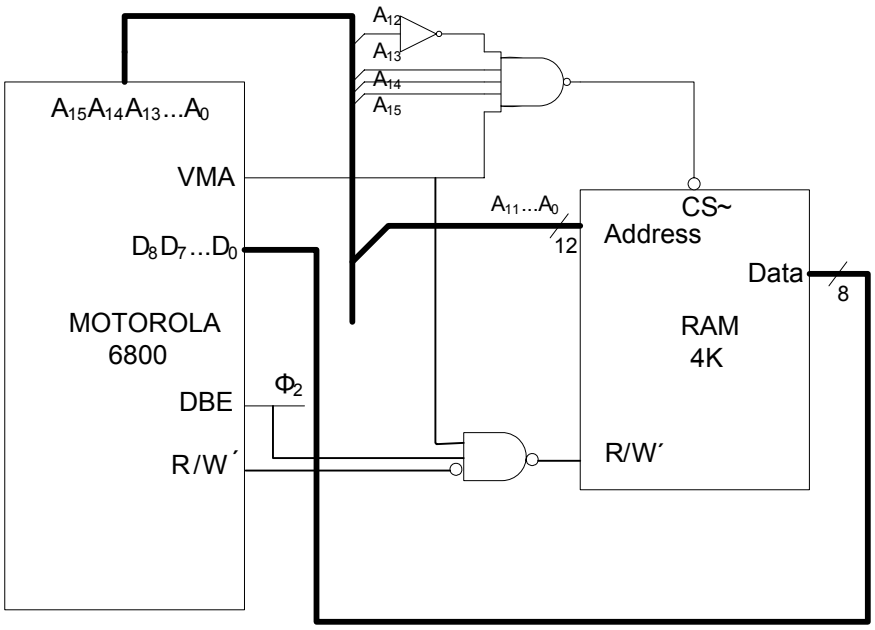
Τα παραπάνω σήματα ελέγχου θα πρέπει να συνδυαστούν ώστε να δημιουργηθούν τα κατάλληλα σήματα ελέγχου για την μνήμη.

Αρχικά θα πρέπει να υλοποιήσουμε τη διευθυνσιοδότηση της μνήμης. Η μνήμη καταλαμβάνει τις θέσεις E000H έως και EFFFH του συστήματος μνήμης. Όταν εμφανίζεται μία από αυτές τις διευθύνσεις στο δίαυλο διευθύνσεων τότε θα δημιουργείται σήμα επιλογής μνήμης (CS). Παρατηρούμε ότι τα 4 πιο σημαντικά ψηφία των διευθύνσεων θα χρησιμοποιηθούν για την αποκωδικοποίηση των διευθύνσεων που ανήκουν στην μνήμη. Τα 12 λιγότερο σημαντικά ψηφία θα χρησιμοποιηθούν για την εσωτερική διευθυνσιοδότηση της μνήμης. Επιπλέον πρέπει το σήμα VME να γίνει '1' ώστε να είναι έγκυρη η διεύθυνση.

Άρα,

$$\mathrm{CS} = \overline{VMA \cdot A_{15} \cdot A_{14} \cdot A_{13} \cdot \overline{A_{12}}}$$

Στη μνήμη RAM οδηγείται επίσης και το σήμα R/W' του επεξεργαστή. Παρατηρώντας το Σχήμα 4.6 (χρονισμός για εγγραφή) βλέπουμε ότι τα δεδομένα εμφανίζονται στο δίαυλο όταν Φ2=1. Επομένως με τη συμμετοχή του Φ2 στην παραγωγή του $R/W'_{(memory)}$ της μνήμης αποτρέπεται το να περάσουν αρχικά "λανθασμένα" δεδομένα στη μνήμη (και επίσης να ξεκινήσει εγγραφή πριν σταθεροποιηθεί απόλυτα η διεύθυνση μέσα σε αυτήν) αφού τα σωστά δεδομένα εμφανίζονται με την ανερχόμενη ακμή του Φ2. Επίσης για να αποκλείσουμε την περίπτωση εγγραφής λανθασμένης τιμής στη μνήμη (π.χ. όταν ο επεξεργαστής θέτει τους οδηγητές των διαύλων του σε κατάσταση υψηλής εμπέδησης, καταστάσεις WAIT, HALT, HOLD) πρέπει να συμμετάσχει και το VMA (που στις αντίστοιχες καταστάσεις είναι 0) στην παραγωγή του $R/W'_{(memory)}$ της μνήμης. Όταν R/W'=1, πρέπει να έχουμε $R/W'_{(memory)}$=1, ενώ $R/W'_{(memory)}$=0 πρέπει να έχουμε μόνο όταν R/W'=0 και VMA=Φ2=1. Αυτό επιτυγχάνεται με μια πύλη NAND με εισόδους $\overline{R/W'}, VMA, \Phi 2$.



ΑΣΚΗΣΗ 52

Σε έναν επεξεργαστή Motorola 6800 συνδέστε ως περιφερειακές μονάδες μία μνήμη ROM 16KBytes, μία μνήμη RAM 16KBytes και 2 καταχωρητές (K1, K2) των 8 ψηφίων. Οι καταχωρητές να συνδεθούν έτσι ώστε ο K1 να μπορεί να χρησιμοποιηθεί μόνο για εγγραφή δεδομένων ενώ ο K2 για εγγραφή και για ανάγνωση. Η μνήμη ROM να τοποθετηθεί στις αρχικές διευθύνσεις του πεδίου διευθύνσεων ενώ η μνήμη RAM στις αμέσως επόμενες. Οι καταχωρητές K1 και K2 να τοποθετηθούν στις διευθύνσεις μνήμης $C000_{16}$ και $C001_{16}$, αντίστοιχα. Η μνήμη ROM και η μνήμη RAM απαιτούν τα σήματα ελέγχου CS (chip select) και OE (output enable) τα οποία πρέπει να έχουν τιμή 1 για να είναι ενεργά, ενώ η μνήμη RAM απαιτεί επιπλέον και σήμα ελέγχου R/W' (read/write) που με 1 πραγματοποιεί ανάγνωση ενώ με 0 εγγραφή. Οι καταχωρητές διαθέτουν σήμα ελέγχου επίτρεψης εγγραφής. Σχεδιάστε το παραπάνω σύστημα. Εξηγείστε τον τρόπο δημιουργίας των σημάτων ελέγχου των μονάδων.

Υπόδειξη: Μελετώντας το χρονισμό για τους κύκλους ανάγνωσης και εγγραφής που παρουσιάζονται στα σχήματα 4.5 και 4.6 (Τόμος Γ) να καθορίσετε κατάλληλα τα σήματα ελέγχου των μονάδων.

Λύση:

Όπως παρατηρούμε στο σχήμα 4.5 κατά τον κύκλο ανάγνωσης τα σήματα ελέγχου R/W' και VMA παίρνουν την τιμή '1'. Επίσης η τιμή της διεύθυνσης είναι σταθερή στο δίαυλο διευθύνσεων καθ' όλη τη διάρκεια του κύκλου.

Όπως παρατηρούμε στο σχήμα 4.6 κατά τον κύκλο εγγραφής το σήμα ελέγχου R/W' παίρνει τη τιμή '0' ενώ το σήμα ελέγχου VMA παίρνει την τιμή '1'. Επίσης η τιμή της διεύθυνσης είναι σταθερή στο δίαυλο διευθύνσεων καθ' όλη τη διάρκεια του κύκλου ενώ τα δεδομένα εμφανίζονται στο δίαυλο δεδομένων προς το τέλος του κύκλου όταν το σήμα ελέγχου DBE είναι '1'. Το σήμα DBE είναι είσοδος στον επεξεργαστή και συνήθως οδηγείται από το σήμα ρολογιού Φ2. Η εγγραφή στους καταχωρητές γίνεται με την αρνητική ακμή του ρολογιού Φ_2.

Τα παραπάνω σήματα ελέγχου θα πρέπει να συνδυαστούν ώστε να δημιουργηθούν τα κατάλληλα σήματα ελέγχου για τις μονάδες.

Αρχικά θα πρέπει να υλοποιήσουμε τη διευθυνσιοδότηση των μνημών. Η μνήμη ROM έχει χωρητικότητα 16K λέξεων (bytes) και άρα απαιτούνται 14 ψηφία (2^{14}=16K) για τη διευθυνσιοδότησή της. Αφού θέλουμε να καταλαμβάνει τις πρώτες θέσεις του πεδίου διευθύνσεων μνήμης θα καταλαμβάνει τις διευθύνσεις 0000-3FFF, και θα χρησιμοποιηθούν τα ψηφία $A_{13}…A_0$ του διαύλου διευθύνσεων του επεξεργαστή για την εσωτερική διευθυνσιοδότηση της μνήμης. Τα ψηφία $A_{14,}A_{15}$ χρησιμοποιούνται για τη δημιουργία του σήματος επιλογής (CS_{ROM}) και επειδή η μνήμη επιλέγεται όταν αυτά είναι ίσα με 0 έχουμε $CS_{\text{ROM}} = \overline{A_{15}} \cdot \overline{A_{14}}$.

Η μνήμη RAM έχει χωρητικότητα 16K λέξεων (bytes) και άρα απαιτούνται 14 ψηφία για τη διευθυνσιοδότησή της, για τα οποία θα χρησιμοποιήσουμε τα $A_{13}…A_0$ του διαύλου διευθύνσεων του επεξεργαστή για την εσωτερική διευθυνσιοδότηση της μνήμης.. Αφού θέλουμε να καταλαμβάνει τις θέσεις του πεδίου διευθύνσεων μνήμης που είναι αμέσως μετά από αυτές που καταλαμβάνει η ROM, θα καταλαμβάνει τις διευθύνσεις 4000-7FFF και τα ψηφία $A_{14,}A_{15}$ που χρησιμοποιούνται για τη δημιουργία του σήματος επιλογής (CS_{RAM}) θα έχουν τιμές A_{14} = 1 και A_{15}=0, δηλαδή $CS_{\text{RAM}} = \overline{A_{15}} \cdot A_{14}$.

Οι καταχωρητές θέλουμε να καταλαμβάνουν τις διευθύνσεις $C000_{16}$ και $C0001_{16}$ του συστήματος μνήμης. Όταν εμφανίζεται μία από αυτές τις διευθύνσεις στο δίαυλο διευθύνσεων τότε θα δημιουργείται σήμα επιλογής καταχωρητή (CSK). Παρατηρούμε επίσης ότι τα 15 πιο σημαντικά ψηφία των διευθύνσεων είναι κοινά και για τους δύο καταχωρητές και ότι η διαφοροποίηση γίνεται στο τελευταίο ψηφίο. Επομένως τα 15 πιο σημαντικά ψηφία θα χρησιμοποιηθούν για την επιλογή των καταχωρητών ενώ το τελευταίο για την επιλογή κάθε ενός από αυτούς.

Παράγεται δηλαδή CSK μόνο όταν τα ψηφία του διαύλου διευθύνσεων A_1 – A_{13} είναι 0 και τα A_{14} και A_{15} είναι 1, δηλαδή έχουμε

$CSK = A_{15}A_{14}\overline{A_{13}}\overline{A_{12}}\overline{A_{11}}\overline{A_{10}}\overline{A_9}\overline{A_8}\overline{A_7}\overline{A_6}\overline{A_5}\overline{A_4}\overline{A_3}\overline{A_2}\overline{A_1}$.

Το ψηφίο του διαύλου διευθύνσεων A_0 θα διαχωρίσει τους δυο καταχωρητές. Με A_0=0 θα ενεργοποιείται ο K1 ενώ με A_0=1 θα ενεργοποιείται ο K2. Επομένως: $CSK1 = CSK \cdot \overline{A_0}$ και $CSK2 = CSK \cdot A_0$.

Οι δίαυλοι δεδομένων των μνημών συνδέονται στο δίαυλο δεδομένων του επεξεργαστή και οι δίαυλοι διευθύνσεων συνδέονται, αντίστοιχα, στα πρώτα 14 ψηφία του δίαυλου διευθύνσεων του επεξεργαστή, όπως αναφέρθηκε παραπάνω. Τα σήματα CS_{ROM} και CS_{RAM} συνδέονται στα αντίστοιχα σήματα επιλογής των μνημών. Όταν επιλέγεται μια μνήμη για να εξάγει αποτελέσματα στην έξοδό της (ανάγνωση μνήμης) πρέπει το σήμα ελέγχου OE=1. Σε περίπτωση εγγραφής δεδομένων στη μνήμη πρέπει το σήμα OE να μένει στο 0 ώστε ο τρισταθής οδηγητής της εξόδου της μνήμης να είναι απενεργοποιημένος διασφαλίζοντας την αποφυγή σύγκρουσης δεδομένων. Οι παραπάνω λειτουργίες σχετικά με το OE επιτυγχάνονται αν οδηγήσουμε στην είσοδο αυτή τον AND συνδυασμό του VMA και του R/W', όπως προκύπτει από τα Σχήματα 4.5 και 4.6. Επομένως σήμα OE=1 θα παραχθεί μόνο κατά την ανάγνωση όπου VMA= R/W'=1 και όχι κατά την εγγραφή όπου R/W'=0.

Στη μνήμη RAM οδηγείται επίσης και το σήμα R/W' του επεξεργαστή. Παρατηρώντας το Σχήμα 4.6 (χρονισμός για εγγραφή) βλέπουμε ότι τα δεδομένα εμφανίζονται στο δίαυλο όταν Φ2=1. Επομένως με τη συμμετοχή του Φ2 στην παραγωγή του $R/W'_{(memory)}$ της μνήμης αποτρέπεται το να περάσουν αρχικά "λανθασμένα" δεδομένα στη μνήμη (και επίσης να ξεκινήσει εγγραφή πριν σταθεροποιηθεί απόλυτα η διεύθυνση μέσα σε αυτήν) αφού τα

σωστά δεδομένα εμφανίζονται με την ανερχόμενη ακμή του Φ2. Επίσης για να αποκλείσουμε την περίπτωση εγγραφής λανθασμένης τιμής στη μνήμη (π.χ. όταν ο επεξεργαστής θέτει τους οδηγητές των διαύλων του σε κατάσταση υψηλής εμπέδησης, καταστάσεις WAIT, HALT, HOLD) πρέπει να συμμετάσχει και το VMA (που στις αντίστοιχες καταστάσεις είναι 0) στην παραγωγή του $R/W'_{(memory)}$ της μνήμης. Όταν R/W'=1, πρέπει να έχουμε $R/W'_{(memory)}=1$, ενώ $R/W'_{(memory)}=0$ πρέπει να έχουμε μόνο όταν R/W'=0 και VMA=Φ2=1. Αυτό επιτυγχάνεται με μια πύλη NAND με εισόδους $\overline{R/W'}, VMA, \Phi 2$.

Οι είσοδοι των καταχωρητών συνδέονται απευθείας στο δίαυλο δεδομένων. Αυτό όμως δεν μπορεί να συμβεί με την έξοδο του καταχωρητή από τον οποίο διαβάζονται δεδομένα. Αυτή θα πρέπει να συνδεθεί με τη μεσολάβηση στοιχείων τριών καταστάσεων. Τα στοιχεία αυτά έχουν ένα σήμα ελέγχου, Ε, και όταν αυτό είναι ενεργοποιημένο, π.χ. Ε=1, τότε μετάγουν την τιμή της εισόδου τους στην έξοδο, διαφορετικά (όταν Ε=0) η έξοδος είναι απομονωμένη από την είσοδο (οδηγείται από κάποιο άλλο στοιχείο). Κάθε ψηφίο εξόδου ενός καταχωρητή συνδέεται μέσω ενός στοιχείου τριών καταστάσεων σε ένα ψηφίο του δίαυλου δεδομένων. Και τα 8 στοιχεία που συνδέουν τις εξόδους ενός καταχωρητή στον δίαυλο δεδομένων έχουν κοινό σήμα ελέγχου. Επομένως απαιτείται ένα σήμα ελέγχου ΕΚ2, για τον καταχωρητή Κ2 από τον οποίο διαβάζονται δεδομένα.

Για την περίπτωση εγγραφής σε καταχωρητή θα πρέπει να επιλεγεί ο καταχωρητής όπου πρέπει να γραφτεί πληροφορία, το σήμα VME να γίνει '1' και το σήμα R/W' να γίνει '0'. Επομένως, για να γράψουμε πληροφορία στον καταχωρητή Κ1, το σήμα ελέγχου επίτρεψης εγγραφής στον Κ1, *ENK1 (Enable_K1)*, θα πρέπει να γίνει '1'. Συνεπώς

$$ENK1 = CSK1 \cdot VMA \cdot \overline{R/W'}$$

Για το σήμα ελέγχου επίτρεψης εγγραφής στον Κ2 θα είναι:

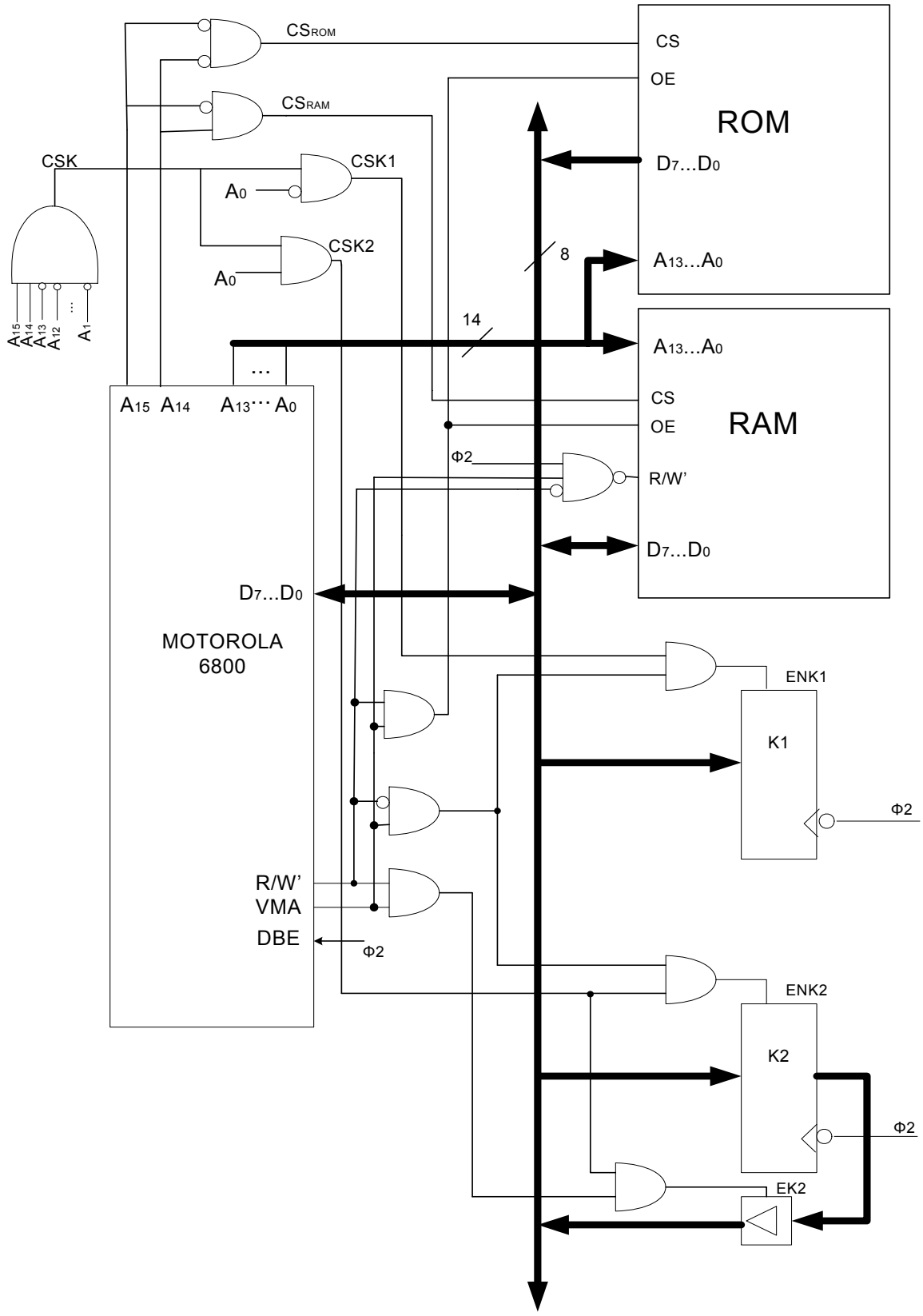
$$ENK2 = CSK2 \cdot VMA \cdot \overline{R/W'}$$

Οι καταχωρητές θα παίρνουν ως σήμα ρολογιού το Φ_2. Η εγγραφή θα γίνεται στην αρνητική ακμή αυτού.

Για την ανάγνωση θα πρέπει να επιλεγεί η αντίστοιχη διάταξη στοιχείων τριών καταστάσεων ώστε να συνδεθεί η έξοδος του ζητούμενου καταχωρητή στο δίαυλο δεδομένων. Κατά την ανάγνωση έχουμε VMA=1 και R/W'=1. Επομένως

$$EK2 = CSK2 \cdot VMA \cdot R/W'$$

Το σχήμα της διάταξης δίνεται παρακάτω.


CSROM
CSRAM
CSK
CSK1
A0
CSK2
A0
A15
A14
A13
A12
...
A1
CS
OE
ROM
D7...D0
A13...A0
8
14
A13...A0
CS
OE
RAM
R/W'
D7...D0
...
A15 A14 A13··· A0
Φ2
D7...D0
MOTOROLA
6800
R/W'
VMA
DBE
Φ2
ENK1
K1
Φ2
ENK2
K2
Φ2
EK2

ΑΣΚΗΣΗ 53

Σε έναν επεξεργαστή Motorola 6800 συνδέστε ως περιφερειακές μονάδες μία μνήμη ROM 8KBytes, μία μνήμη RAM 4KBytes και ένα καταχωρητή των 8 ψηφίων. Ο καταχωρητής να συνδεθεί έτσι ώστε να μπορεί να χρησιμοποιηθεί για εγγραφή και ανάγνωση δεδομένων. Η μνήμη ROM να τοποθετηθεί στις αρχικές διευθύνσεις του πεδίου διευθύνσεων ενώ η μνήμη RAM στις αμέσως επόμενες. Ο καταχωρητής να τοποθετηθεί στη διεύθυνση μνήμης $D000_{16}$. Η μνήμη ROM και η μνήμη RAM απαιτούν τα σήματα ελέγχου CS (chip select) και OE (output enable) τα οποία πρέπει να έχουν τιμή 1 για να είναι ενεργά, ενώ η μνήμη RAM απαιτεί επιπλέον και σήμα ελέγχου R/W' (read/write) που με 1 πραγματοποιεί ανάγνωση ενώ με 0 εγγραφή. Ο καταχωρητής διαθέτει σήμα ελέγχου επίτρεψης εγγραφής.
Σχεδιάστε το παραπάνω σύστημα. Εξηγείστε τον τρόπο δημιουργίας των σημάτων ελέγχου των μονάδων.

Υπόδειξη: Μελετώντας το χρονισμό για τους κύκλους ανάγνωσης και εγγραφής που παρουσιάζονται στα σχήματα 4.5 και 4.6 (Τόμος Γ) να καθορίσετε κατάλληλα τα σήματα ελέγχου των μονάδων.

Λύση
Όπως παρατηρούμε στο σχήμα 4.5 κατά τον κύκλο ανάγνωσης τα σήματα ελέγχου R/W' και VMA παίρνουν την τιμή '1'. Επίσης η τιμή της διεύθυνσης είναι σταθερή στο δίαυλο διευθύνσεων καθ' όλη τη διάρκεια του κύκλου.

Όπως παρατηρούμε στο σχήμα 4.6 κατά τον κύκλο εγγραφής το σήμα ελέγχου R/W' παίρνει τη τιμή '0' ενώ το σήμα ελέγχου VMA παίρνει την τιμή '1'. Επίσης η τιμή της διεύθυνσης είναι σταθερή στο δίαυλο διευθύνσεων καθ' όλη τη διάρκεια του κύκλου ενώ τα δεδομένα εμφανίζονται στο δίαυλο δεδομένων προς το τέλος του κύκλου όταν το σήμα ελέγχου DBE είναι '1'. Το σήμα DBE είναι είσοδος στον επεξεργαστή και συνήθως οδηγείται από το σήμα ρολογιού Φ2. Η εγγραφή στους καταχωρητές γίνεται με την αρνητική ακμή του ρολογιού Φ_2.

Τα παραπάνω σήματα ελέγχου θα πρέπει να συνδυαστούν ώστε να δημιουργηθούν τα κατάλληλα σήματα ελέγχου για τις μονάδες.

Αρχικά θα πρέπει να υλοποιήσουμε τη διευθυνσιοδότηση των μνημών. Η μνήμη ROM έχει χωρητικότητα 8K λέξεων (bytes) και άρα απαιτούνται 13 ψηφία (2^{13}=8K) για τη διευθυνσιοδότησή της. Αφού θέλουμε να καταλαμβάνει τις πρώτες θέσεις του πεδίου διευθύνσεων μνήμης, θα καταλαμβάνει τις διευθύνσεις 0000-1FFF, και θα χρησιμοποιηθούν τα ψηφία $A_{12}\ldots A_0$ του διαύλου διευθύνσεων του επεξεργαστή για την εσωτερική διευθυνσιοδότηση της μνήμης. Τα ψηφία A_{13},A_{14},A_{15} χρησιμοποιούνται για τη δημιουργία του σήματος επιλογής (CS_{ROM}) και επειδή η μνήμη επιλέγεται όταν αυτά είναι ίσα με 0 έχουμε $CS_{\text{ROM}} = \overline{A_{15}} \cdot \overline{A_{14}} \cdot \overline{A_{13}}$.

Η μνήμη RAM έχει χωρητικότητα 4K λέξεων (bytes) και άρα απαιτούνται 12 ψηφία για τη διευθυνσιοδότησή της, για τα οποία θα χρησιμοποιήσουμε τα $A_{11}\ldots A_0$ του διαύλου διευθύνσεων του επεξεργαστή. Αφού θέλουμε να καταλαμβάνει τις θέσεις του πεδίου διευθύνσεων μνήμης που είναι αμέσως μετά από αυτές που καταλαμβάνει η ROM θα καταλαμβάνει τις διευθύνσεις 2000-2FFF. Για τη δημιουργία του σήματος επιλογής (CS_{RAM}) για τις διευθύνσεις αυτές χρησιμοποιούνται τα ψηφία A_{15},A_{14},A_{13},A_{12} τα οποία πρέπει να έχουν τις τιμές A_{15}=0, A_{14}=0, A_{13}=1, και A_{12}=0, δηλαδή $CS_{\text{RAM}} = \overline{A_{15}} \cdot \overline{A_{14}} \cdot A_{13} \cdot \overline{A_{12}}$.

Ο καταχωρητής θέλουμε να καταλαμβάνει τη διεύθυνση $D000_{16}$ του συστήματος μνήμης. Όταν εμφανίζεται η διεύθυνση αυτή στο δίαυλο διευθύνσεων τότε θα δημιουργείται σήμα

επιλογής καταχωρητή (CSK). Παράγεται δηλαδή CSK μόνο όταν τα ψηφία του διαύλου διευθύνσεων $A_0 - A_{11}$ και A_{13} είναι 0 και τα A_{12} , A_{14} και A_{15} είναι 1, δηλαδή έχουμε
$CSK = A_{15}A_{14}\overline{A_{13}}A_{12}\overline{A}_{11}\overline{A}_{10}\overline{A}_9\overline{A}_8\overline{A}_7\overline{A}_6\overline{A}_5\overline{A}_4\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$.

Οι δίαυλοι δεδομένων των μνημών συνδέονται στο δίαυλο δεδομένων του επεξεργαστή και οι δίαυλοι διευθύνσεων συνδέονται στα πρώτα 13 ψηφία του δίαυλου διευθύνσεων του επεξεργαστή για τη ROM, και στα πρώτα 12 ψηφία του δίαυλου διευθύνσεων για τη RAM, όπως αναφέρθηκε παραπάνω. Τα σήματα CS_{ROM} και CS_{RAM} συνδέονται στα αντίστοιχα σήματα επιλογής των μνημών. Όταν επιλέγεται μια μνήμη για να εξάγει αποτελέσματα στην έξοδό της (ανάγνωση μνήμης) πρέπει το σήμα ελέγχου OE=1. Σε περίπτωση εγγραφής δεδομένων στη μνήμη πρέπει το σήμα OE να μένει στο 0 ώστε ο τρισταθής οδηγητής της εξόδου της μνήμης να είναι απενεργοποιημένος διασφαλίζοντας την αποφυγή σύγκρουσης δεδομένων. Οι παραπάνω λειτουργίες σχετικά με το OE επιτυγχάνονται αν οδηγήσουμε στην είσοδο αυτή τον AND συνδυασμό του VMA και του R/W', όπως προκύπτει από τα Σχήματα 4.5 και 4.6. Επομένως σήμα OE=1 θα παραχθεί μόνο κατά την ανάγνωση όπου VMA= R/W'=1 και όχι κατά την εγγραφή όπου R/W'=0.

Στη μνήμη RAM οδηγείται επίσης και το σήμα R/W' του επεξεργαστή. Παρατηρώντας το Σχήμα 4.6 (χρονισμός για εγγραφή) βλέπουμε ότι τα δεδομένα εμφανίζονται στο δίαυλο όταν Φ2=1. Επομένως με τη συμμετοχή του Φ2 στην παραγωγή του $R/W'_{(memory)}$ της μνήμης αποτρέπεται το να περάσουν αρχικά "λανθασμένα" δεδομένα στη μνήμη (και επίσης να ξεκινήσει εγγραφή πριν σταθεροποιηθεί απόλυτα η διεύθυνση μέσα σε αυτήν) αφού τα σωστά δεδομένα εμφανίζονται με την ανερχόμενη ακμή του Φ2. Επίσης για να αποκλείσουμε την περίπτωση εγγραφής λανθασμένης τιμής στη μνήμη (π.χ. όταν ο επεξεργαστής θέτει τους οδηγητές των διαύλων του σε κατάσταση υψηλής εμπέδησης, καταστάσεις WAIT, HALT, HOLD) πρέπει να συμμετάσχει και το VMA (που στις αντίστοιχες καταστάσεις είναι 0) στην παραγωγή του $R/W'_{(memory)}$ της μνήμης. Όταν R/W'=1, πρέπει να έχουμε $R/W'_{(memory)}$=1, ενώ $R/W'_{(memory)}$=0 πρέπει να έχουμε μόνο όταν R/W'=0 και VMA=Φ2=1. Αυτό επιτυγχάνεται με μια πύλη NAND με εισόδους $\overline{R/W'}, VMA, \Phi 2$.

Η είσοδος του καταχωρητή συνδέεται απευθείας στο δίαυλο δεδομένων. Αυτό όμως δεν μπορεί να συμβεί με την έξοδο του καταχωρητή από όπου διαβάζονται δεδομένα. Αυτή θα πρέπει να συνδεθεί με τη μεσολάβηση στοιχείων τριών καταστάσεων. Τα στοιχεία αυτά έχουν ένα σήμα ελέγχου, ΕΚ, και όταν αυτό είναι ενεργοποιημένο, π.χ. ΕΚ=1, τότε μετάγουν την τιμή της εισόδου τους στην έξοδο, διαφορετικά (όταν ΕΚ=0) η έξοδος είναι απομονωμένη από την είσοδο (οδηγείται από κάποιο άλλο στοιχείο). Κάθε ψηφίο εξόδου ενός καταχωρητή συνδέεται μέσω ενός στοιχείου τριών καταστάσεων σε ένα ψηφίο του δίαυλου δεδομένων. Και τα 8 στοιχεία που συνδέουν τις εξόδους ενός καταχωρητή στον δίαυλο δεδομένων έχουν κοινό σήμα ελέγχου. Επομένως απαιτείται ένα σήμα ελέγχου ΕΚ για τον καταχωρητή.

Για την περίπτωση εγγραφής σε καταχωρητή θα πρέπει να επιλεγεί ο καταχωρητής όπου πρέπει να γραφτεί πληροφορία, το σήμα VME να γίνει '1' και το σήμα R/W' να γίνει '0'. Επομένως, για να γράψουμε πληροφορία στον καταχωρητή, το σήμα ελέγχου επίτρεψης εγγραφής *ENK (Enable_K)*, θα πρέπει να γίνει '1'. Συνεπώς

$$ENK = CSK \cdot VMA \cdot \overline{R/W'}$$

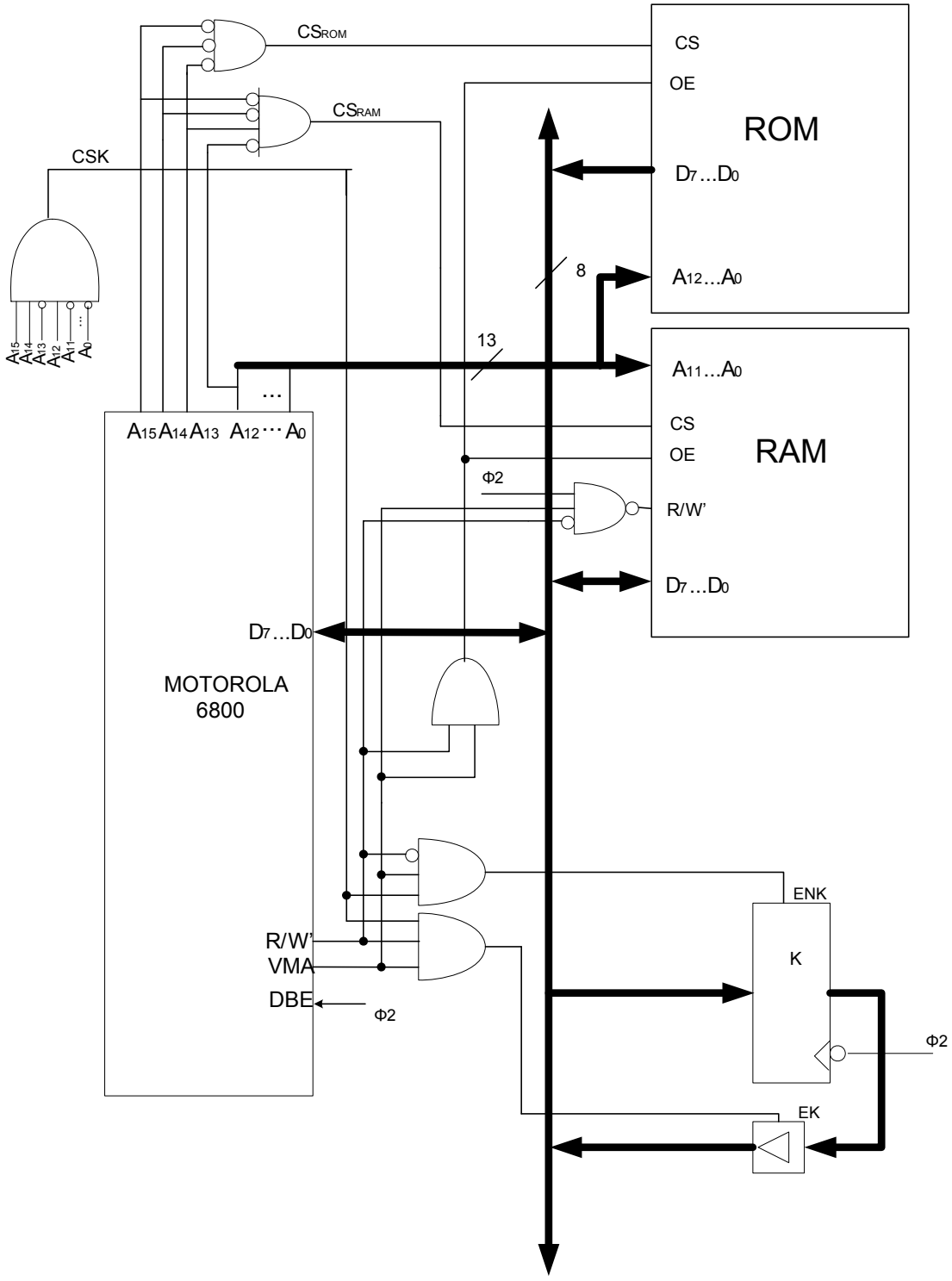
Ο καταχωρητής θα παίρνει ως σήμα ρολογιού το Φ_2. Η εγγραφή θα γίνεται στην αρνητική ακμή αυτού.

Για την ανάγνωση θα πρέπει να επιλεγεί η αντίστοιχη διάταξη στοιχείων τριών καταστάσεων ώστε να συνδεθεί η έξοδος του καταχωρητή στο δίαυλο δεδομένων. Κατά την ανάγνωση έχουμε VMA=1 και R/W'=1. Επομένως

$$EK = CSK \cdot VMA \cdot R/W'$$

Το σχήμα της διάταξης δίνεται παρακάτω.

○



ΑΣΚΗΣΗ 54

Σε έναν επεξεργαστή Motorola 6800 συνδέστε ως περιφερειακές μονάδες μία μνήμη **ROM των 4Kbytes** και μία μνήμη **RAM των 8KBytes**. Και οι δυο μνήμες έχουν οργάνωση **8 δυαδικών ψηφίων ανά λέξη**. Η μνήμη ROM να τοποθετηθεί στις **αρχικές διευθύνσεις του πεδίου διευθύνσεων** ενώ η μνήμη RAM στο πεδίο διευθύνσεων **με αρχή το 2000_{16}**. Η μνήμη ROM και η μνήμη RAM απαιτούν τα σήματα ελέγχου CS (chip select) και OE (output enable) τα οποία πρέπει να έχουν τιμή 1 για να είναι ενεργά, ενώ η μνήμη RAM απαιτεί επιπλέον και σήμα ελέγχου R/W' (read/write) που με 1 πραγματοποιεί ανάγνωση ενώ με 0 εγγραφή.

Σχεδιάστε το παραπάνω σύστημα. Προσδιορίστε τις διευθύνσεις του συστήματος μνήμης που καλύπτει κάθε μονάδα μνήμης. Εξηγείστε τον τρόπο δημιουργίας των σημάτων ελέγχου των μονάδων.

Υπόδειξη: Συμβουλευτείτε τα διαγράμματα χρονισμού για τους κύκλους ανάγνωσης και εγγραφής που παρουσιάζονται στα σχήματα 4.5 και 4.6 του Τόμου Γ.

Λύση

Η μνήμη ROM είναι των 4KBytes και τοποθετείται στις αρχικές διευθύνσεις του συστήματος μνήμης, άρα καταλαμβάνει τις διευθύνσεις 0000-0FFF.

Η μνήμη RAM είναι των 8KBytes και τοποθετείται στο πεδίο διευθύνσεων με αρχή τη διεύθυνση 2000_{16}. Άρα καταλαμβάνει τις διευθύνσεις 2000-3FFF.

Η μνήμη ROM έχει χωρητικότητα 4KBytes και άρα απαιτούνται 12 ψηφία (2^{12}=4K) για την εσωτερική διευθυνσιοδότησή της και συνεπώς θα χρησιμοποιηθούν τα ψηφία $A_{11}\ldots A_0$ του διαύλου διευθύνσεων του επεξεργαστή. Τα ψηφία $A_{12},A_{13},A_{14},A_{15}$ χρησιμοποιούνται για τη δημιουργία του σήματος επιλογής (CS_{ROM}) και επειδή η μνήμη επιλέγεται όταν αυτά είναι ίσα με 0, αφού καταλαμβάνει τις διευθύνσεις 0000-0FFF, έχουμε $CS_{\text{ROM}} = \overline{A_{15}} \cdot \overline{A_{14}} \cdot \overline{A_{13}} \cdot \overline{A_{12}}$.

Η μνήμη RAM έχει χωρητικότητα 8KBytes και άρα απαιτούνται 13 ψηφία για τη διευθυνσιοδότησή της, για τα οποία θα χρησιμοποιήσουμε τα $A_{12}\ldots A_0$ του διαύλου διευθύνσεων του επεξεργαστή. Αφού καταλαμβάνει τις θέσεις του πεδίου μνήμης 2000-3FFF για τη δημιουργία του σήματος επιλογής (CS_{RAM}) χρησιμοποιούνται τα ψηφία A_{15},A_{14},A_{13} τα οποία πρέπει να έχουν τις τιμές A_{15}=0, A_{14}=0, A_{13}=1, δηλαδή $CS_{\text{RAM}} = \overline{A_{15}} \cdot \overline{A_{14}} \cdot A_{13}$.

Οι δίαυλοι δεδομένων των μνημών συνδέονται στο δίαυλο δεδομένων του επεξεργαστή και οι δίαυλοι διευθύνσεων συνδέονται στα πρώτα 12 ψηφία του δίαυλου διευθύνσεων του επεξεργαστή για τη ROM και στα πρώτα 13 ψηφία του δίαυλου διευθύνσεων για τη RAM όπως αναφέρθηκε παραπάνω. Τα σήματα CS_{ROM} και CS_{RAM} συνδέονται στα αντίστοιχα σήματα επιλογής των μνημών.

Όπως παρατηρούμε στο σχήμα 4.5 κατά τον κύκλο ανάγνωσης τα σήματα ελέγχου R/W' και VMA παίρνουν την τιμή '1'. Επίσης η τιμή της διεύθυνσης είναι σταθερή στο δίαυλο διευθύνσεων καθ' όλη τη διάρκεια του κύκλου.

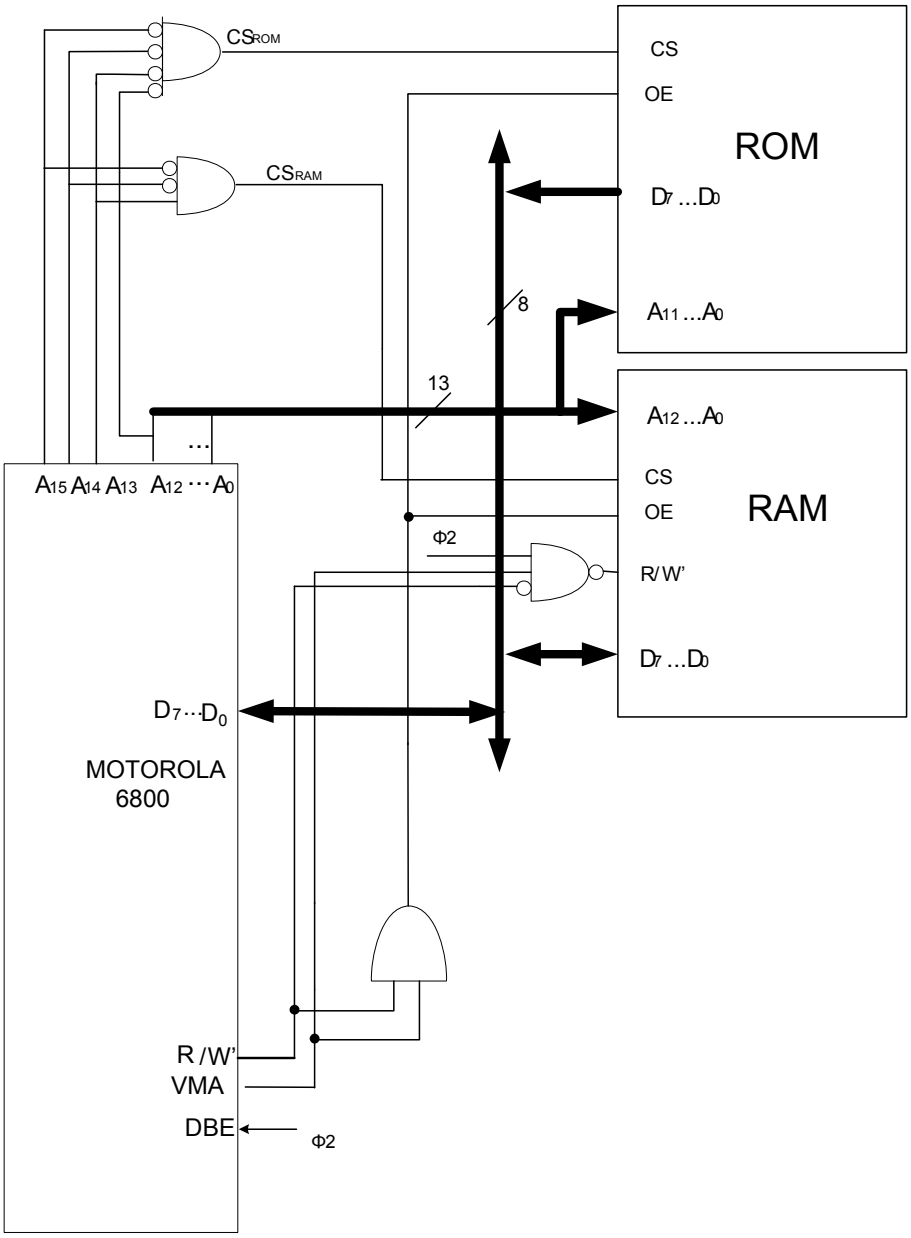
Όπως παρατηρούμε στο σχήμα 4.6 κατά τον κύκλο εγγραφής το σήμα ελέγχου R/W' παίρνει τη τιμή '0' ενώ το σήμα ελέγχου VMA παίρνει την τιμή '1'. Επίσης η τιμή της διεύθυνσης είναι σταθερή στο δίαυλο διευθύνσεων καθ' όλη τη διάρκεια του κύκλου ενώ τα δεδομένα εμφανίζονται στο δίαυλο δεδομένων προς το τέλος του κύκλου όταν το σήμα ελέγχου DBE

είναι ‘1’. Το σήμα DBE είναι είσοδος στον επεξεργαστή και συνήθως οδηγείται από το σήμα ρολογιού Φ2.

Όταν επιλέγεται μια μνήμη για να εξάγει αποτελέσματα στην έξοδό της (ανάγνωση μνήμης) πρέπει το σήμα ελέγχου OE=1. Σε περίπτωση εγγραφής δεδομένων στη μνήμη πρέπει το σήμα OE να μένει στο 0 ώστε ο τρισταθής οδηγητής της εξόδου της μνήμης να είναι απενεργοποιημένος διασφαλίζοντας την αποφυγή σύγκρουσης δεδομένων. Οι παραπάνω λειτουργίες σχετικά με το OE επιτυγχάνονται αν οδηγήσουμε στην είσοδο αυτή τον AND συνδυασμό του VMA και του R/W’. Επομένως σήμα OE=1 θα παραχθεί μόνο κατά την ανάγνωση όπου VMA= R/W’=1 και όχι κατά την εγγραφή όπου R/W’=0.

Στη μνήμη RAM οδηγείται επίσης και το σήμα R/W’ του επεξεργαστή. Παρατηρώντας το σχήμα χρονισμού για εγγραφή, βλέπουμε ότι τα δεδομένα εμφανίζονται στο δίαυλο όταν Φ2=1. Επομένως με τη συμμετοχή του Φ2 στην παραγωγή του $R/W'_{(memory)}$ της μνήμης αποτρέπεται το να περάσουν αρχικά "λανθασμένα" δεδομένα στη μνήμη (και επίσης να ξεκινήσει εγγραφή πριν σταθεροποιηθεί απόλυτα η διεύθυνση μέσα σε αυτήν) αφού τα σωστά δεδομένα εμφανίζονται με την ανερχόμενη ακμή του Φ2. Επίσης για να αποκλείσουμε την περίπτωση εγγραφής λανθασμένης τιμής στη μνήμη (π.χ. όταν ο επεξεργαστής θέτει τους οδηγητές των διαύλων του σε κατάσταση υψηλής εμπέδησης, καταστάσεις WAIT, HALT, HOLD) πρέπει να συμμετάσχει και το VMA (που στις αντίστοιχες καταστάσεις είναι 0) στην παραγωγή του $R/W'_{(memory)}$ της μνήμης. Όταν R/W’=1, πρέπει να έχουμε $R/W'_{(memory)}$=1, ενώ $R/W'_{(memory)}$=0 πρέπει να έχουμε μόνο όταν R/W’=0 και VMA=Φ2=1. Αυτό επιτυγχάνεται με μια πύλη NAND με εισόδους $\overline{R/W'}, VMA, \Phi 2$.

Το σχήμα της διάταξης δίνεται παρακάτω.



ΑΣΚΗΣΗ 55

Σε έναν επεξεργαστή Motorola 6800 συνδέστε ως περιφερειακές μονάδες μία μνήμη ROM 16KBytes, και δύο μνήμες RAM, RAM1 και RAM2, των 4KBytes και 2KBytes, αντίστοιχα. Η μνήμη ROM να τοποθετηθεί στις αρχικές διευθύνσεις του πεδίου διευθύνσεων ενώ οι μνήμες RAM στις αμέσως επόμενες. Η μνήμη ROM και οι μνήμες RAM απαιτούν τα σήματα ελέγχου CS (chip select) και OE (output enable) τα οποία πρέπει να έχουν τιμή 1 για να είναι ενεργά, ενώ οι μνήμες RAM απαιτούν επιπλέον και σήμα ελέγχου R/W' (read/write) που με 1 πραγματοποιεί ανάγνωση ενώ με 0 εγγραφή. Σχεδιάστε το παραπάνω σύστημα. Εξηγείστε τον τρόπο δημιουργίας των σημάτων ελέγχου των μονάδων.

Υπόδειξη: Μελετώντας το χρονισμό για τους κύκλους ανάγνωσης και εγγραφής που παρουσιάζονται στα σχήματα 4.5 και 4.6 (Τόμος Γ) να καθορίσετε κατάλληλα τα σήματα ελέγχου των μονάδων.

Λύση:

Όπως παρατηρούμε στο σχήμα 4.5 κατά τον κύκλο ανάγνωσης τα σήματα ελέγχου R/W' και VMA παίρνουν την τιμή '1'. Επίσης η τιμή της διεύθυνσης είναι σταθερή στο δίαυλο διευθύνσεων καθ' όλη τη διάρκεια του κύκλου.

Όπως παρατηρούμε στο σχήμα 4.6 κατά τον κύκλο εγγραφής το σήμα ελέγχου R/W' παίρνει τη τιμή '0' ενώ το σήμα ελέγχου VMA παίρνει την τιμή '1'. Επίσης η τιμή της διεύθυνσης είναι σταθερή στο δίαυλο διευθύνσεων καθ' όλη τη διάρκεια του κύκλου ενώ τα δεδομένα εμφανίζονται στο δίαυλο δεδομένων προς το τέλος του κύκλου όταν το σήμα ελέγχου DBE είναι '1'. Το σήμα DBE είναι είσοδος στον επεξεργαστή και συνήθως οδηγείται από το σήμα ρολογιού Φ2.

Τα παραπάνω σήματα ελέγχου θα πρέπει να συνδυαστούν ώστε να δημιουργηθούν τα κατάλληλα σήματα ελέγχου για τις μονάδες μνήμης.

Αρχικά θα πρέπει να υλοποιήσουμε τη διευθυνσιοδότηση των μνημών. Τοποθετώντας σε διαδοχικές διευθύνσεις τις μονάδες μνήμης που διαθέτουμε προκύπτει η χαρτογράφηση μνήμης που δίνεται στον παρακάτω πίνακα:

Μνήμη	**Μέγεθος (bytes)**	**Ψηφία Διεύθυνσης**	**Πεδίο Διευθύνσεων**	**Διευθύνσεις σε δυαδική μορφή** $A_{15}\ A_{14}\ \ldots\ A_2\ A_1\ A_0$
ROM1	16K	14	0000 – 3FFF	0000 0000 0000 0000 0011 1111 1111 1111
RAM1	4K	12	4000 – 4FFF	0100 0000 0000 0000 0100 1111 1111 1111
RAM2	2K	11	5000 – 57FF	0101 0000 0000 0000 0101 0111 1111 1111

Η μνήμη ROM καταλαμβάνει τα πρώτα 16KBytes του πεδίου διευθύνσεων και συγκεκριμένα τις διευθύνσεις 0000-3FFF. Παρατηρούμε ότι τα 14 λιγότερο σημαντικά ψηφία ($A_0 – A_{13}$) της λέξης διεύθυνσής της παίρνουν όλες τις δυνατές τιμές. Συνεπώς τα ψηφία αυτά χρησιμοποιούνται για την εσωτερική διευθυνσιοδότηση της μνήμης και εφαρμόζονται στις εισόδους διεύθυνσης της μνήμης, Τα ψηφία A_{14},A_{15} έχουν σταθερή τιμή 0 για όλο το πεδίο διευθύνσεων της ROM και άρα θα χρησιμοποιηθούν για τη δημιουργία του σήματος επιλογής (CS_{ROM}) και επειδή η μνήμη επιλέγεται όταν αυτά είναι ίσα με 0 έχουμε $CS_{\text{ROM}} = \overline{A_{15}} \cdot \overline{A_{14}}$.

Η μνήμη RAM1 έχει χωρητικότητα 4KBytes και καταλαμβάνει τις διευθύνσεις 4000-4FFF. Παρατηρούμε ότι τα 12 λιγότερο σημαντικά ψηφία ($A_0 – A_{11}$) της λέξης διεύθυνσής της παίρνουν όλες τις δυνατές τιμές. Συνεπώς τα ψηφία αυτά χρησιμοποιούνται για την εσωτερική διευθυνσιοδότηση της μνήμης και εφαρμόζονται στις εισόδους διεύθυνσης της μνήμης. Τα ψηφία $A_{15},A_{14},A_{13},A_{12}$ έχουν σταθερή τιμή για όλο το πεδίο διευθύνσεων της RAM1 (A_{15}=0, A_{14}=1, A_{13}=0, A_{12}=0) και άρα θα χρησιμοποιηθούν για τη δημιουργία του σήματος επιλογής (CS_{RAM1}). Συνεπώς $CS_{\text{RAM1}} = \overline{A_{15}} \cdot A_{14} \cdot \overline{A_{13}} \cdot \overline{A_{12}}$.

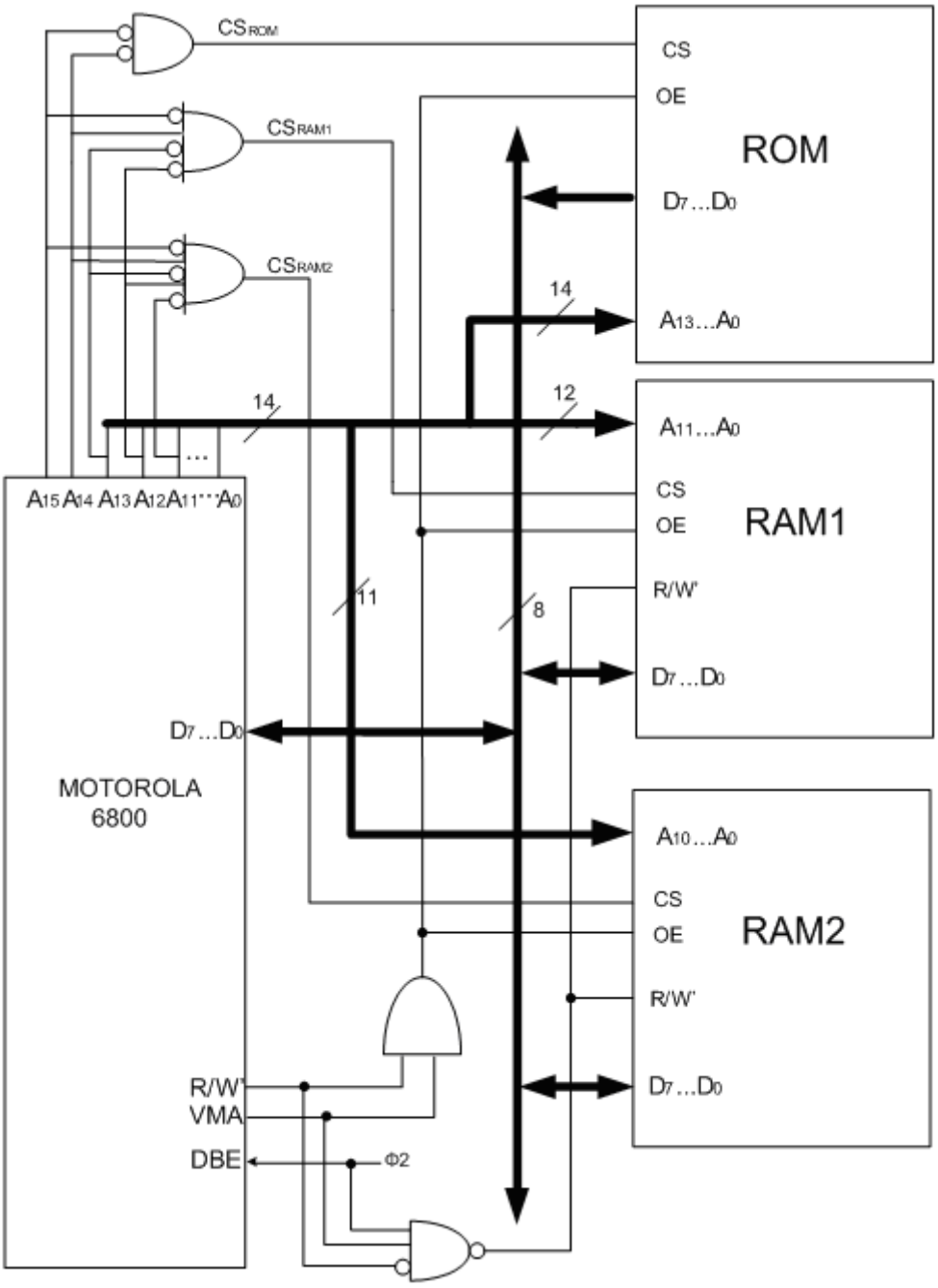
Η μνήμη RAM2 έχει χωρητικότητα 2KBytes και καταλαμβάνει τις διευθύνσεις 5000-57FF. Παρατηρούμε ότι τα 11 λιγότερο σημαντικά ψηφία ($A_0 – A_{10}$) της λέξης διεύθυνσής της παίρνουν όλες τις δυνατές τιμές. Συνεπώς τα ψηφία αυτά χρησιμοποιούνται για την εσωτερική διευθυνσιοδότηση της μνήμης και εφαρμόζονται στις εισόδους διεύθυνσης της μνήμης. Τα ψηφία A_{15}, $A_{14},A_{13},A_{12},A_{11}$ έχουν σταθερή τιμή για όλο το πεδίο διευθύνσεων της

RAM2 (A_{15}=0, A_{14}=1, A_{13}=0, A_{12}=1, A_{11}=0) και άρα θα χρησιμοποιηθούν για τη δημιουργία του σήματος επιλογής (CS_{RAM2}). Συνεπώς, $CS_{\text{RAM2}} = \overline{A_{15}} \cdot A_{14} \cdot \overline{A_{13}} \cdot A_{12} \cdot \overline{A_{11}}$.

Οι δίαυλοι δεδομένων των μονάδων μνήμης συνδέονται στο δίαυλο δεδομένων του επεξεργαστή και οι δίαυλοι διευθύνσεων συνδέονται στα αντίστοιχα ψηφία του δίαυλου διευθύνσεων του επεξεργαστή, όπως αναφέρθηκε παραπάνω. Τα σήματα CS_{ROM}, CS_{RAM1} και CS_{RAM2} συνδέονται στα αντίστοιχα σήματα επιλογής των μονάδων μνήμης. Όταν επιλέγεται μια μονάδα μνήμης για να εξάγει αποτελέσματα στην έξοδό της (ανάγνωση μνήμης) πρέπει το σήμα ελέγχου OE=1. Σε περίπτωση εγγραφής δεδομένων στη μνήμη πρέπει το σήμα OE να μένει στο 0 ώστε ο τρισταθής οδηγητής της εξόδου της μονάδας μνήμης να είναι απενεργοποιημένος διασφαλίζοντας την αποφυγή σύγκρουσης δεδομένων. Οι παραπάνω λειτουργίες σχετικά με το OE επιτυγχάνονται αν οδηγήσουμε στην είσοδο αυτή τον AND συνδυασμό του VMA και του R/W', όπως προκύπτει από τα Σχήματα 4.5 και 4.6. Επομένως σήμα OE=1 θα παραχθεί μόνο κατά την ανάγνωση όπου VMA= R/W'=1 και όχι κατά την εγγραφή όπου R/W'=0.

Στις μνήμες RAM οδηγείται επίσης και το σήμα R/W' του επεξεργαστή. Παρατηρώντας το Σχ. 4.6 (χρονισμός για εγγραφή) βλέπουμε ότι τα δεδομένα εμφανίζονται στο δίαυλο όταν Φ2=1. Επομένως με τη συμμετοχή του Φ2 στην παραγωγή του $R/W'_{(memory)}$ της μνήμης αποτρέπεται το να περάσουν αρχικά "λανθασμένα" δεδομένα στη μνήμη (και επίσης να ξεκινήσει εγγραφή πριν σταθεροποιηθεί απόλυτα η διεύθυνση μέσα σε αυτήν) αφού τα σωστά δεδομένα εμφανίζονται με την ανερχόμενη ακμή του Φ2. Επίσης για να αποκλείσουμε την περίπτωση εγγραφής λανθασμένης τιμής στη μνήμη (π.χ. όταν ο επεξεργαστής θέτει τους οδηγητές των διαύλων του σε κατάσταση υψηλής εμπέδησης, καταστάσεις WAIT, HALT, HOLD) πρέπει να συμμετάσχει και το VMA (που στις αντίστοιχες καταστάσεις είναι 0) στην παραγωγή του $R/W'_{(memory)}$ της μνήμης. Όταν R/W'=1, πρέπει να έχουμε $R/W'_{(memory)}$=1, ενώ $R/W'_{(memory)}$=0 πρέπει να έχουμε μόνο όταν R/W'=0 και VMA=Φ2=1. Αυτό επιτυγχάνεται με μια πύλη NAND με εισόδους $\overline{R/W'}, VMA, \Phi 2$.

Το σχήμα της διάταξης δίνεται παρακάτω.


CSROM
CSRAM1
CSRAM2
14
14
12
11
8
A15 A14 A13 A12 A11 ... A0
D7 ...D0
MOTOROLA
6800
R/W'
VMA
DBE
Φ2
CS
OE
ROM
D7 ...D0
A13 ...A0
A11 ...A0
RAM1
R/W'
A10 ...A0
RAM2

Γενικές Ερωτήσεις

ΑΣΚΗΣΗ 56

Εάν το σήμα INT του σχήματος 3.13 (Τόμος Γ) στον κύκλο ρολογιού T_1 του κύκλου μηχανής N+1 ήταν 0 και γινόταν 1 στον κύκλο ρολογιού T_2 του κύκλου μηχανής N+1 θα γινόταν εξυπηρέτηση της διακοπής στο κύκλο μηχανής N+1 και γιατί; Πότε θα γινόταν η εξυπηρέτηση της διακοπής;

Λύση:

Όχι δεν θα γινόταν εξυπηρέτηση της διακοπής στο κύκλο μηχανής N+1 διότι ο 8080 αναγνωρίζει αίτηση εξυπηρέτησης διακοπής κατά τον επόμενο κύκλο ρολογιού T_1 αφότου γίνει η αίτηση διακοπής. Η αίτηση διακοπής γίνεται με την οδήγηση του σήματος INT στην στάθμη 1. Άρα σε αυτή την περίπτωση ο κύκλος μηχανής N+1 δεν θα είναι κύκλος εξυπηρέτησης διακοπής αλλά θα είναι ο κύκλος μηχανής που πραγματοποιείται (δηλαδή κανονικός κύκλος μηχανής του προγράμματος που εκτελείται). Ο κύκλος μηχανής N+2 θα ήταν ο πρώτος κύκλος εξυπηρέτησης της διακοπής.

ΑΣΚΗΣΗ 57

Ποια είναι η τιμή των καταχωρητών γενικού σκοπού στους επεξεργαστές 8080 και 6800 μόλις αυτοί συνδεθούν με την τροφοδοσία.

Λύση:
Η τιμή των καταχωρητών γενικού σκοπού στους επεξεργαστές 8080 και 6800 μόλις αυτοί συνδεθούν με την τροφοδοσία είναι τυχαία (αυθαίρετη). Για αυτό το λόγο όταν γίνεται η αρχικοποίηση του συστήματος δεν έχει χρησιμότητα να διαβάσουμε το περιεχόμενο των καταχωρητών αυτών. Απαιτείται να προηγηθεί απαραίτητα μια εντολή εγγραφής στο καταχωρητή για να έχει κάποιο χρήσιμο περιεχόμενο για ανάγνωση.

ΑΣΚΗΣΗ 58

Σε τι πλεονεκτεί και σε τι μειονεκτεί ο μικροεπεξεργαστής 6800 εν συγκρίσει με το μικροεπεξεργαστή 8085;

Λύση:
Ο 6800 έχει πιο απλό χρονισμό σε σχέση με τον 8085. Ο κύκλος ρολογιού στην περίπτωση του 6800 ταυτίζεται με τον κύκλο μηχανής, ενώ στον 8085 ο κάθε κύκλος μηχανής αποτελείται από τρεις έως πέντε κύκλους ρολογιού.

Ο 8085 διαθέτει διαφορετικές εντολές για ανάγνωση ή εγγραφή σε συσκευές εισόδου/εξόδου και διαφορετικές για ανάγνωση ή εγγραφή της μνήμης. Στον 6800 όλες οι συσκευές εισόδου/εξόδου είναι προσπελάσιμες σαν θέσεις μνήμης με αποτέλεσμα να μην χρειάζονται ειδικές εντολές για την προσπέλαση τους.

Ο 6800 έχει πιο απλό σύνολο σημάτων ελέγχου και δεν πολυπλέκει το δίαυλο δεδομένων με το δίαυλο διευθύνσεων.

Το σύνολο εντολών του 6800 είναι πιο κατανοητό από αυτό του 8085. Ο 6800 έχει λιγότερους βασικούς τύπους εντολών, ενώ έχει περισσότερες μεθόδους διευθυνσιοδότησης.

Ο 6800 δεν έχει ενσωματωμένο το κύκλωμα χρονισμού στο ολοκληρωμένο κύκλωμα εν αντιθέσει με τον 8085 που διαθέτει ενσωματωμένο κύκλωμα χρονισμού.

ΑΣΚΗΣΗ 59

Δώστε σύντομες περιγραφές για τις παρακάτω ερωτήσεις:

Α)Τι είναι μικροεπεξεργαστής (microprocessor);
Β) Τι είναι μικροελεγκτής (microcontroler) ;
Γ) Τι είναι CPU (ΚΜΕ);
Δ) Τι είναι κύκλος μηχανής ;
Ε) Τι είναι κύκλος εντολής ;

Λύση

Α) Μικροεπεξεργαστής είναι ένα προγραμματιζόμενο ακολουθιακό λογικό σύστημα. Ως τέτοιο, έχει τη δυνατότητα να εκτελεί προγράμματα, δηλ. σύνολα εντολών που αποσκοπούν στην επίτευξη συγκεκριμένων λειτουργιών.

Β) Μικροελεκτής είναι ένα σύστημα που αποτελείται από έναν επεξεργαστή, μνήμη και συσκευές εισόδου/εξόδου.

Γ) Η ΚΜΕ ενός επεξεργαστή είναι η μονάδα που είναι υπεύθυνη για την εκτέλεση των λειτουργιών που καθορίζονται από τις εντολές του προγράμματος. Αποτελείται από τη μονάδα επεξεργασίας δεδομένων και τη μονάδα ελέγχου. Σε περιπτώσεις απλών μικροεπεξεργαστών αντιστοιχεί σε αυτόν τον ίδιο τον επεξεργαστή.

Δ) Για την εκτέλεση μιας εντολής σε έναν μικροεπεξεργαστή απαιτείται η εκτέλεση επιμέρους ολοκληρωμένων λειτουργιών όπως προσπέλαση σε μνήμη (ανάγνωση/εγγραφή), υπολογισμός πράξεων, αναγνώριση διακοπής κ.α. Κάθε μια από αυτές τις ολοκληρωμένες λειτουργίες εκτελείται σε έναν αριθμό κύκλων ρολογιού και αποτελεί έναν κύκλο μηχανής (βλέπε κεφάλαιο 3.2 και σχήμα 3.5 από Τόμο Γ).

Ε) Κύκλος εντολής είναι ο χρόνος ή αντίστοιχα ο αριθμός των κύκλων ρολογιού που απαιτούνται για την εκτέλεση μιας εντολής. Ένας κύκλος εντολής αποτελείται από μερικούς κύκλους μηχανής αφού για την εκτέλεση μιας εντολής απαιτείται εκτέλεση ενός αριθμού από επιμέρους λειτουργίες (βλέπε κεφάλαιο 3.2 από Τόμο Γ).

ΑΣΚΗΣΗ 60

Θεωρείστε δυο μικροεπεξεργαστές (μΕ1 και μΕ2) που ο κάθε ένας εκτελεί το ίδιο πρόγραμμα 1000 εντολών. Το ποσοστό συμμετοχής κάθε τύπου εντολών στο πρόγραμμα καθώς επίσης και οι κύκλοι μηχανής των αντίστοιχων εντολών δίνονται στον παρακάτω πίνακα.

Εντολές	**Ποσοστό τύπου εντολών (%)**	**Κύκλοι μηχανής**	
		μΕ1	**μΕ2**
Αριθμητικές/λογικές	50	5	4
Μετακίνησης δεδομένων	30	4	3
Άλλες εντολές	20	3	2

Αν ο μΕ1 λειτουργεί σε συχνότητα 800MHz και ο μΕ2 σε συχνότητα 650MHz και θεωρώντας ότι και για τους δυο επεξεργαστές ένας κύκλος μηχανής αντιστοιχεί σε ένα κύκλο ρολογιού να βρεθεί ποιος επεξεργαστής εκτελεί ταχύτερα το πρόγραμμα.
Θεωρείστε, επίσης, ότι το σύστημα μνήμης δεν προσθέτει επιπλέον καθυστέρηση.

Λύση
Ο αριθμός κύκλων μηχανής για την εκτέλεση εντολών του προγράμματος που ανήκουν σε συγκεκριμένο τύπο δίνεται ως

κύκλοι_μηχανής_εντολής x *ποσοστό_τύπου_εντολών* x *αριθμός_εντολών_προγράμματος*

Συνεπώς, ο αριθμός κύκλων μηχανής και άρα κύκλων ρολογιού που χρειάζεται ο μΕ1 για την εκτέλεση του προγράμματος δίνεται ως 5x0,5x1000+4x0,3x1000+3x0,2x1000=4300 κύκλοι ρολογιού.

Για τον μΕ2 απαιτούνται 4x0,5x1000+3x0,3x1000+2x0,2x1000=3300 κύκλοι ρολογιού.

Ο μΕ1 λειτουργεί στα 800MHz και συνεπώς ο κύκλος ρολογιού του διαρκεί 1,25ns. Ο μΕ2 λειτουργεί στα 650MHz και συνεπώς ο κύκλος ρολογιού του διαρκεί 1,538ns.

Άρα ο μΕ1 απαιτεί 5375ns ενώ ο μΕ2 απαιτεί 5076ns για την εκτέλεση του προγράμματος.

Ο μΕ2 αν και λειτουργεί σε μικρότερη συχνότητα εκτελεί ταχύτερα το πρόγραμμα.

ΠΑΡΑΡΤΗΜΑ

The 8085 Instruction Set

Instructions can be categorized according to their method of addressing the hardware registers and/or memory.

Implied Addressing:
The addressing mode of certain instructions is implied by the instruction's function. For example, the STC (set carry flag) instruction deals only with the carry flag, the DAA (decimal adjust accumulator) instruction deals with the accumulator.

Register Addressing:
Quite a large set of instructions call for register addressing. With these instructions, you must specify one of the registers A through E, H or L as well as the operation code. With these instructions, the accumulator is implied as a second operand. For example, the instruction CMP E may be interpreted as 'compare the contents of the E register with the contents of the accumulator.

Most of the instructions that use register addressing deal with
8-bit values. However, a few of these instructions deal with 16-bit register pairs. For example, the PCHL instruction exchanges the contents of the program counter with the contents of the H and L registers.

Immediate Addressing:
Instructions that use immediate addressing have data assembled as a part of the instruction itself. For example, the instruction CPI 'C' may be interpreted as 'compare the contents of the accumulator with the letter C. When assembled, this instruction has the hexadecimal value FE43. Hexadecimal 43 is the internal representation for the letter C. When this instruction is executed, the processor fetches the first instruction byte and determines that it must fetch one more byte. The processor fetches the next byte into one of its internal registers and then performs the compare operation.

Notice that the names of the immediate instructions indicate that they use immediate data. Thus, the name of an add instruction is ADD; the name of an add immediate instruction is ADI.

All but two of the immediate instructions uses the accumulator as an implied operand, as in the CPI instruction shown previously. The MVI (move immediate)

instruction can move its immediate data to any of the working registers including the accumulator or to memory. Thus, the instruction MVI D, OFFH moves the hexadecimal value FF to the D register.

The LXI instruction (load register pair immediate) is even more unusual in that its immediate data is a 16-bit value. This instruction is commonly used to load addresses into a register pair. As mentioned previously, your program must initialize the stack pointer; LXI is the instruction most commonly used for this purpose. For example, the instruction LXI SP,3OFFH loads the stack pointer with the hexadecimal value 30FF.

Direct Addressing:
Jump instructions include a 16-bit address as part of the instruction. For example, the instruction JMP 1000H causes a jump to the hexadecimal address 1000 by replacing the current contents of the program counter with the new value 1000H.

Instructions that include a direct address require three bytes of storage: one for the instruction code, and two for the 16-bit address

Register Indirect Addressing:
Register indirect instructions reference memory via a register pair. Thus, the instruction MOV M,C moves the contents of the C register into the memory address stored in the H and L register pair. The instruction LDAX B loads the accumulator with the byte of data specified by the address in the B and C register pair.

Combined Addressing Modes:
Some instructions use a combination of addressing modes. A CALL instruction, for example, combines direct addressing and register indirect addressing. The direct address in a CALL instruction specifies the address of the desired subroutine; the register indirect address is the stack pointer. The CALL instruction pushes the current contents of the program counter into the memory location specified by the stack pointer.

Timing Effects of Addressing Modes:
Addressing modes affect both the amount of time required for executing an instruction and the amount of memory required for its storage. For example, instructions that use implied or register addressing, execute very quickly since they deal directly with the processor’s hardware or with data already present in hardware registers. Most

important, however is that the entire instruction can be fetched with a single memory access. The number of memory accesses required is the single greatest factor in determining execution timing. More memory accesses therefore require more execution time. A CALL instruction for example, requires five memory accesses: three to access the entire instruction and two more to push the contents of the program counter onto the stack.

The processor can access memory once during each processor cycle. Each cycle comprises a variable number of states. (See below and the appendix of "USING THE SDK-85 MICROPROCESSOR TRAINER"). The length of a state depends on the clock frequency specified for your system, and may range from 480 nanoseconds to 2 microseconds. Thus, the timing for a four state instruction may range from 1.920 microseconds through 8 microseconds. (The 8085 have a maximum clock frequency of 5 MHz and therefore a minimum state length of 200 nanoseconds.)

Instruction Naming Conventions:
The mnemonics assigned to the instructions are designed to indicate the function of the instruction. The instructions fall into the following functional categories:

Data Transfer Croup:
The data transfer instructions move data between registers or between memory and registers.

MOV	Move
MVI	Move Immediate
LDA	Load Accumulator Directly from Memory
STA	Store Accumulator Directly in Memory
LHLD	Load H & L Registers Directly from Memory
SHLD	Store H & L Registers Directly in Memory

An 'X' in the name of a data transfer instruction implies that it deals with a register pair (16-bits);

LXI	Load Register Pair with Immediate data
LDAX	Load Accumulator from Address in Register Pair
STAX	Store Accumulator in Address in Register Pair
XCHG	Exchange H & L with D & E
XTHL	Exchange Top of Stack with H & L

Arithmetic Group:
The arithmetic instructions add, subtract, increment, or decrement data in registers or memory.

ADD	Add to Accumulator
ADI	Add Immediate Data to Accumulator
ADC	Add to Accumulator Using Carry Flag
ACI	Add Immediate data to Accumulator Using Carry
SUB	Subtract from Accumulator
SUI	Subtract Immediate Data from Accumulator
SBB	Subtract from Accumulator Using Borrow (Carry) Flag
SBI	Subtract Immediate from Accumulator Using Borrow (Carry) Flag
INR	Increment Specified Byte by One
DCR	Decrement Specified Byte by One
INX	Increment Register Pair by One
DCX	Decrement Register Pair by One
DAD	Double Register Add; Add Content of Register Pair to H & L Register Pair

Logical Group:
This group performs logical (Boolean) operations on data in registers and memory and on condition flags.

The logical AND, OR, and Exclusive OR instructions enable you to set specific bits in the accumulator ON or OFF.

ANA	Logical AND with Accumulator
ANI	Logical AND with Accumulator Using Immediate Data
ORA	Logical OR with Accumulator
OR	Logical OR with Accumulator Using Immediate Data
XRA	Exclusive Logical OR with Accumulator
XRI	Exclusive OR Using Immediate Data

The Compare instructions compare the content of an 8-bit value with the contents of the accumulator;

CMP	Compare
CPI	Compare Using Immediate Data

The rotate instructions shift the contents of the accumulator one bit position to the left or right:

RLC	Rotate Accumulator Left
RRC	Rotate Accumulator Right
RAL	Rotate Left Through Carry
RAR	Rotate Right Through Carry

Complement and carry flag instructions:

```
CMA             Complement Accumulator
CMC             Complement Carry Flag
STC             Set Carry Flag
```

Branch Group:
The branching instructions alter normal sequential program flow, either unconditionally or conditionally. The unconditional branching instructions are as follows:

```
JMP             Jump
CALL            Call
RET             Return
```

Conditional branching instructions examine the status of one of four condition flags to determine whether the specified branch is to be executed. The conditions that may be specified are as follows:

```
NZ              Not Zero (Z = 0)
Z               Zero (Z = 1)
NC              No Carry (C = 0)
C               Carry (C = 1)
PO              Parity Odd (P = 0)
PE              Parity Even (P= 1)
P               Plus (S = 0)
M               Minus (S = 1)
```

Thus, the conditional branching instructions are specified as follows:

Jumps	Calls	Returns	
C	CC	RC	(Carry)
INC	CNC	RNC	(No Carry)
JZ	CZ	RZ	(Zero)
JNZ	CNZ	RNZ	(Not Zero)
JP	CP	RP	(Plus)
JM	CM	RM	(Minus)
JPE	CPE	RPE	(Parity Even)
JP0	CPO	RPO	(Parity Odd)

Two other instructions can affect a branch by replacing the contents or the program counter:

```
PCHL            Move H & L to Program Counter
RST             Special Restart Instruction Used
                with Interrupts
```

Stack I/O, and Machine Control Instructions:
The following instructions affect the Stack and/or Stack Pointer:

PUSH	Push Two bytes of Data onto the Stack
POP	Pop Two Bytes of Data off the Stack
XTHL	Exchange Top of Stack with H & L
SPHL	Move content of H & L to Stack Pointer

The I/0 instructions are as follows:

IN	Initiate Input Operation
OUT	Initiate Output Operation

The Machine Control instructions are as follows:

EI	Enable Interrupt System
DI	Disable Interrupt System
HLT	Halt
NOP	No Operation

Mnemonic	Op	SZAPC	~s	Description	Notes
ACI n	CE	*****	7	Add with Carry Immediate	A=A+n+CY
ADC r	8F	*****	4	Add with Carry	A=A+r+CY(21X)
ADC M	8E	*****	7	Add with Carry to Memory	A=A+[HL]+CY
ADD r	87	*****	4	Add	A=A+r (20X)
ADD M	86	*****	7	Add to Memory	A=A+[HL]
ADI n	C6	*****	7	Add Immediate	A=A+n
ANA r	A7	**0*0	4	AND Accumulator	A=A&r (24X)
ANA M	A6	**0*0	7	AND Accumulator and Memory	A=A&[HL]
ANI n	E6	**0*0	7	AND Immediate	A=A&n
CALL a	CD	-----	18	Call unconditional	-[SP]=PC,PC=a
CC a	DC	-----	9	Call on Carry	If CY=1(18~s)
CM a	FC	-----	9	Call on Minus	If S=1 (18~s)
CMA	2F	-----	4	Complement Accumulator	A=~A
CMC	3F	----*	4	Complement Carry	CY=~CY
CMP r	BF	*****	4	Compare	A-r (27X)
CMP M	BF	*****	7	Compare with Memory	A-[HL]
CNC a	D4	-----	9	Call on No Carry	If CY=0(18~s)
CNZ a	C4	-----	9	Call on No Zero	If Z=0 (18~s)
CP a	F4	-----	9	Call on Plus	If S=0 (18~s)
CPE a	EC	-----	9	Call on Parity Even	If P=1 (18~s)
CPI n	FE	*****	7	Compare Immediate	A-n
CPO a	E4	-----	9	Call on Parity Odd	If P=0 (18~s)
CZ a	CC	-----	9	Call on Zero	If Z=1 (18~s)
DAA	27	*****	4	Decimal Adjust Accumulator	A=BCD format
DAD B	09	----*	10	Double Add BC to HL	HL=HL+BC
DAD D	19	----*	10	Double Add DE to HL	HL=HL+DE
DAD H	29	----*	10	Double Add HL to HL	HL=HL+HL
DAD SP	39	----*	10	Double Add SP to HL	HL=HL+SP
DCR r	3D	****-	4	Decrement	r=r-1 (0X5)
DCR M	35	****-	10	Decrement Memory	[HL]=[HL]-1
DCX B	0B	-----	6	Decrement BC	BC=BC-1
DCX D	1B	-----	6	Decrement DE	DE=DE-1
DCX H	2B	-----	6	Decrement HL	HL=HL-1
DCX SP	3B	-----	6	Decrement Stack Pointer	SP=SP-1
DI	F3	-----	4	Disable Interrupts	
EI	FB	-----	4	Enable Interrupts	
HLT	76	-----	5	Halt	
IN p	DB	-----	10	Input	A=[p]
INR r	3C	****-	4	Increment	r=r+1 (0X4)

INR M	3C	****-	10	Increment Memory	[HL]=[HL]+1
INX B	03	-----	6	Increment BC	BC=BC+1
INX D	13	-----	6	Increment DE	DE=DE+1
INX H	23	-----	6	Increment HL	HL=HL+1
INX SP	33	-----	6	Increment Stack Pointer	SP=SP+1
JMP a	C3	-----	7	Jump unconditional	PC=a
JC a	DA	-----	7	Jump on Carry	If CY=1(10~s)
JM a	FA	-----	7	Jump on Minus	If S=1 (10~s)
JNC a	D2	-----	7	Jump on No Carry	If CY=0(10~s)
JNZ a	C2	-----	7	Jump on No Zero	If Z=0 (10~s)
JP a	F2	-----	7	Jump on Plus	If S=0 (10~s)
JPE a	EA	-----	7	Jump on Parity Even	If P=1 (10~s)
JPO a	E2	-----	7	Jump on Parity Odd	If P=0 (10~s)
JZ a	CA	-----	7	Jump on Zero	If Z=1 (10~s)
LDA a	3A	-----	13	Load Accumulator direct	A=[a]
LDAX B	0A	-----	7	Load Accumulator indirect	A=[BC]
LDAX D	1A	-----	7	Load Accumulator indirect	A=[DE]
LHLD a	2A	-----	16	Load HL Direct	HL=[a]
LXI B,nn	01	-----	10	Load Immediate BC	BC=nn
LXI D,nn	11	-----	10	Load Immediate DE	DE=nn
LXI H,nn	21	-----	10	Load Immediate HL	HL=nn
LXI SP,nn	31	-----	10	Load Immediate Stack Ptr	SP=nn
MOV r1,r2	7F	-----	4	Move register to register	r1=r2 (1XX)
MOV M,r	77	-----	7	Move register to Memory	[HL]=r (16X)
MOV r,M	7E	-----	7	Move Memory to register	r=[HL] (1X6)
MVI r,n	3E	-----	7	Move Immediate	r=n (0X6)
MVI M,n	36	-----	10	Move Immediate to Memory	[HL]=n
NOP	00	-----	4	No Operation	
ORA r	B7	**0*0	4	Inclusive OR Accumulator	A=Avr (26X)
ORA M	B6	**0*0	7	Inclusive OR Accumulator	A=Av[HL]
ORI n	F6	**0*0	7	Inclusive OR Immediate	A=Avn
OUT p	D3	-----	10	Output	[p]=A
PCHL	E9	-----	6	Jump HL indirect	PC=[HL]
POP B	C1	-----	10	Pop BC	BC=[SP]+
POP D	D1	-----	10	Pop DE	DE=[SP]+
POP H	E1	-----	10	Pop HL	HL=[SP]+
POP PSW	F1	-----	10	Pop Processor Status Word	{PSW,A}=[SP]+

Mnemonic	Op	SZAPC	~s	Description	Notes
PUSH B	C5	-----	12	Push BC	-[SP]=BC
PUSH D	D5	-----	12	Push DE	-[SP]=DE
PUSH H	E5	-----	12	Push HL	-[SP]=HL
PUSH PSW	F5	-----	12	Push Processor Status Word	-[SP]={PSW,A}
RAL	17	----*	4	Rotate Accumulator Left	A={CY,A}<-
RAR	1F	----*	4	Rotate Accumulator Righ	A=->{CY,A}
RET	C9	-----	10	Return	PC=[SP]+
RC	D8	-----	6	Return on Carry	If CY=1(12~s)
RIM	20	-----	4	Read Interrupt Mask	A=mask
RM	F8	-----	6	Return on Minus	If S=1 (12~s)
RNC	D0	-----	6	Return on No Carry	If CY=0(12~s)
RNZ	C0	-----	6	Return on No Zero	If Z=0 (12~s)
RP	F0	-----	6	Return on Plus	If S=0 (12~s)
RPE	E8	-----	6	Return on Parity Even	If P=1 (12~s)
RPO	E0	-----	6	Return on Parity Odd	If P=0 (12~s)
RZ	C8	-----	6	Return on Zero	If Z=1 (12~s)
RLC	07	----*	4	Rotate Left Circular	A=A<-
RRC	0F	----*	4	Rotate Right Circular	A=->A
RST z	C7	-----	12	Restart (3X7)	-[SP]=PC,PC=z
SBB r	9F	*****	4	Subtract with Borrow	A=A-r-CY

SBB M	9E	*****	7	Subtract with Borrow	A=A-[HL]-CY
SBI n	DE	*****	7	Subtract with Borrow Immed	A=A-n-CY
SHLD a	22	-----	16	Store HL Direct	[a]=HL
SIM	30	-----	4	Set Interrupt Mask	mask=A
SPHL	F9	-----	6	Move HL to SP	SP=HL
STA a	32	-----	13	Store Accumulator	[a]=A
STAX B	02	-----	7	Store Accumulator indirect	[BC]=A
STAX D	12	-----	7	Store Accumulator indirect	[DE]=A
STC	37	----1	4	Set Carry	CY=1
SUB r	97	*****	4	Subtract	A=A-r (22X)
SUB M	96	*****	7	Subtract Memory	A=A-[HL]
SUI n	D6	*****	7	Subtract Immediate	A=A-n
XCHG	EB	-----	4	Exchange HL with DE	HL<->DE
XRA r	AF	**0*0	4	Exclusive OR Accumulator	A=Axr (25X)
XRA M	AE	**0*0	7	Exclusive OR Accumulator	A=Ax[HL]
XRI n	EE	**0*0	7	Exclusive OR Immediate	A=Axn
XTHL	E3	-----	16	Exchange stack Top with HL	[SP]<->HL

PSW	-*01	Flag unaffected/affected/reset/set
S	S	Sign (Bit 7)
Z	Z	Zero (Bit 6)
AC	A	Auxilary Carry (Bit 4)
P	P	Parity (Bit 2)
CY	C	Carry (Bit 0)

a p	Direct addressing
M z	Register indirect addressing
n nn	Immediate addressing
r	Register addressing

DB n(,n)	Define Byte(s)
DB 'string'	Define Byte ASCII character string
DS nn	Define Storage Block
DW nn(,nn)	Define Word(s)

A B C D E H L	Registers (8-bit)
BC DE HL	Register pairs (16-bit)
PC	Program Counter register (16-bit)
PSW	Processor Status Word (8-bit)
SP	Stack Pointer register (16-bit)

a nn	16-bit address/data (0 to 65535)
n p	8-bit data/port (0 to 255)
r	Register (X=B,C,D,E,H,L,M,A)
z	Vector (X=0H,8H,10H,18H,20H,28H,30H,38H)

+ -	Arithmetic addition/subtraction
& ~	Logical AND/NOT
v x	Logical inclusive/exclusive OR
<- ->	Rotate left/right
<->	Exchange
[ ]	Indirect addressing
[ ]+ -[ ]	Indirect address auto-inc/decrement
{ }	Combination operands
(X)	Octal op code where X is a 3-bit code
If (~s)	Number of cycles if condition true