ΣΗΜΕΙΩΣΕΙΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ Ι

(Βασικά Θέματα Αρχιτεκτονικής Υπολογιστών)

Computer – Ανάλυση

Περιεχόμενα

Τυπολόγιο 6
Κεφάλαιο 1 – Απόδοση Η/Υ 1
Θέμα Σεπτεμβρίου 2021 - Υπολογισμός μέσου πλήθους κύκλων ρολογιού 1
Θέμα Προόδου 2021 - Υπολογισμός χρόνου εκτέλεσης 2
Παράδειγμα 1.1 - Υπολογισμός χρόνου εκτέλεσης - ΜΧΠΜ - μέσου χρόνου εκτέλεσης εντολής 3
Θέμα Φεβρουαρίου 2014 με ΜΚΡΕ και ΜΧΠΜ 4
Θέμα Ιουνίου 2021 5
Θέμα Ιουνίου 2021 7
Θέμα 2016 με υπολογισμό ΜΧΠΜ και ξεχωριστή κρυφή μνήμη εντολών/δεδομένων 8
Θέμα 2012 με υπολογισμό χρόνου εκτέλεσης και δύο επίπεδα κρυφής μνήμης 9
Παράδειγμα 1.2 - Χρήση μέτρου MIPS για την αξιολόγηση της απόδοσης Η/Υ 10
Άσκηση υπολογισμού χρόνου εκτέλεσης 11
Άσκηση με μονάδα πολλαπλασιασμού και διαίρεσης 12
Κεφάλαιο 2 – Αρχιτεκτονικές Η/Υ – Μορφή εντολής – Είδη αριθμών 13
Επεξηγήσεις Κεφαλαίου 2 13
Άσκηση 2.12 με μηχανισμό στοίβας – συσσωρευτή – καταχωρητή – καταχωρητή και μνήμης 17
Άσκηση 2.13 με μηχανισμό στοίβας – συσσωρευτή – καταχωρητή – καταχωρητή και μνήμης 19
Λύση θέματος 3 – Ιανουάριος 2020 21
Θέμα με αρχιτεκτονική καταχωρητή – καταχωρητή με 3 τελούμενα - Σεπτέμβριος 2015 23
Πρόοδος Νοεμβρίου 2016 24
Θέμα με μεταβλητή μορφή εντολών – χώρο και προσπελάσεις προγράμματος – Ιανουάριος 2015 25
Θέμα με μεταβλητή μορφή εντολών – χώρο και προσπελάσεις προγράμματος – Ιούνιος 2016 26
Θέμα με σταθερή μορφή εντολών και πλήθος θ.μ. που προσπελαύνονται - Νοέμβριος 2016 27
Θέμα με σταθερή μορφή εντολών και πλήθος θ.μ. που προσπελαύνονται - Ιούνιος 2016 28
Θέμα Φεβρουαρίου 2021 28
Θέμα Ιουνίου 2020 30
Θέμα Σεπτεμβρίου 2020 31
Θέμα Σεπτεμβρίου 2020 (άλλη ομάδα) 33
Θέμα Σεπτεμβρίου 2020 34
Άσκηση 2.10 με ευθυγράμμιση μνήμης 35
Άσκηση με ευθυγράμμιση μνήμης 36
Άσκηση με μετατροπή αριθμού από κινητή σε σταθερή υποδιαστολή 37
Θέμα Ιανουαρίου 2017 με μορφή εντολών 38
Θέματα Σεπτεμβρίου 2020 - Φεβρουαρίου - Ιουνίου 2021 με εντολές αρχιτεκτονικής υπολογιστών 42
Κεφάλαιο 3 – Κατηγορίες αριθμών και εντολές μεταφοράς από/προς μνήμη 45
Κατηγορίες αριθμών Η/Υ 45
Άθροισμα - Διαφορά προσημασμένων αριθμών 46
Τεχνική μικροπρογραμματισμού – Τεχνικές μείωσης χωρητικότητας μνήμης ελέγχου 46

Άσκηση 3.1 για άθροιση προσημασμένων αριθμών 48
Άσκηση 3.2 για άθροιση προσημασμένων αριθμών 49
Θέμα 2013 με πράξεις προσημασμένων/μη προσημασμένων αριθμών 51
Επεξηγήσεις για εντολές LOAD/STORE 52
Θέμα Φεβρουαρίου 2016 με πράξεις προσημασμένων/μη προσημασμένων αριθμών 54
Θέμα 1 με θύρες ανάγνωσης/εγγραφής 54
Θέμα 2 με υπερχείλιση/κρατούμενο 54
Θέμα 3 με καταχωρητές γενικού σκοπού 54
Θέματα Ιουνίου 2021 με πράξεις προσημασμένων/μη προσημασμένων αριθμών 55
Θέματα Ιουνίου 2021 με εντολές LOAD/STORE 55
Θέματα Σεπτεμβρίου 2020 με θύρες καταχωρητών και σταθερή/κινητή υποδιαστολή 57
Θέματα Ιουνίου 2020 με πράξεις προσημασμένων/μη προσημασμένων αριθμών 59
Άσκηση 3.3 βιβλίου για εντολές LOAD/STORE 62
Θέματα Φεβρουαρίου 2014 με εντολή LOAD και πράξεις αριθμών 63
Θέματα Σεπτεμβρίου 2016 με εντολές LOAD/STORE 64
Διαφορά little – endian και big – endian αποθήκευσης 67
Άσκηση με big – endian και little – endian 69
Θέμα Φεβρουαρίου 2021 με little – endian και big – endian 70
Θέμα Σεπτεμβρίου 2021 με little – endian και big – endian 71
Θέμα Σεπτεμβρίου 2020 με little – endian και big – endian 71
Θέμα προόδου 2021 με little – endian και big – endian 72
Θέμα Ιουνίου 2020 73
Θέμα Σεπτεμβρίου 2020 74
Θέμα Σεπτεμβρίου 2020 75
Άσκηση με μετατροπή από σταθερή σε κινητή υποδιαστολή 75
Θέμα προόδου με άθροισμα αριθμών σε μορφή IEEE 754 – Ιανουάριος 2022 76
1η Άσκηση με τεχνική μικροπρογραμματισμού δύο επιπέδων 77
2η Άσκηση με τεχνική μικροπρογραμματισμού δύο επιπέδων – Θέμα Σεπτέμβριος 2021 78
3η Άσκηση με τεχνική μικροπρογραμματισμού δύο επιπέδων από ΕΑΠ 79
4η Άσκηση με τεχνική μικροπρογραμματισμού δύο επιπέδων από το μάθημα 80
Θέμα προόδου με τεχνική μικροπρογραμματισμού δύο επιπέδων – Ιανουάριος 2022 81
Κεφάλαιο 5 – Μνήμες 82
Επεξηγήσεις κεφαλαίου 5 82
1ο Θέμα με ιεραρχία μνήμης 86
2ο Θέμα με ιεραρχία μνήμης 86
Παράδειγμα 5.6 βιβλίου – Σχεδιασμός υψηλής τάξης διαφύλλωση μνήμης 87
Παράδειγμα 5.7 βιβλίου – Σχεδιασμός υψηλής τάξης διαφύλλωση μνήμης 89
Θέμα Φεβρουαρίου 2016 92
Θέμα Ιουνίου 2016 93

Άσκηση 5.4 με τμήματα μνήμης....95
Θέμα Σεπτεμβρίου 2016....97
Λύση θέματος Ιανουαρίου 2018....99
Θέμα Ιουνίου 2014....102
Θέμα Σεπτεμβρίου 2015....103
Θέμα Φεβρουαρίου 2021....105
Θέμα Ιουνίου 2021....106
Θέμα Ιουνίου 2020....107
Θέμα Σεπτεμβρίου 2020....108
Θέμα Φεβρουαρίου 2021....109
Θέμα Ιουνίου 2021....110
Θέμα Ιουνίου 2020....110
Θέμα Σεπτεμβρίου 2021....111
Θέμα προόδου Ιανουαρίου 2022....112
Θέμα προόδου Ιανουαρίου 2022....113
Θέμα προόδου Ιανουαρίου 2022....114
Παράδειγμα 5.8 με χαμηλής τάξης διαφύλλωση....115
Θέμα Φεβρουαρίου 2016 με χαμηλής τάξης διαφύλλωση....117
Θέμα Ιουνίου 2016 με σχεδιασμό κύριας μνήμης χαμηλής τάξης διαφύλλωσης....118
Θέμα Φεβρουαρίου 2012 με χαμηλής τάξης διαφύλλωση....118
Κεφάλαιο 6 – Αρτηρίες....122
Βασικά σημεία θεωρίας....122
Άσκηση 6.1 βιβλίου με σύγχρονη/ασύγχρονη αρτηρία....124
Άσκηση 6.2 βιβλίου με σύγχρονη/ασύγχρονη αρτηρία....125
Θέμα προόδου Ιανουαρίου 2022....126
Θέμα προόδου Ιανουαρίου 2022....127
Θέμα προόδου Ιανουαρίου 2022....127
Άσκηση 6.3 βιβλίου με polling/interrupts....127
Άσκηση 6.4 βιβλίου με ΑΠΜ (DMA)....129
Άσκηση 6.5 βιβλίου με DMA....130
Άσκηση με προγραμματισμένη είσοδο – έξοδο με χρήση διακοπών....130
Άσκηση με διαιτησία αρτηρίας....131
Θέμα με σύγχρονη/ασύγχρονη αρτηρία - Ιανουάριος 2020....132
Θέμα με σύγχρονη και ασύγχρονη αρτηρία....133
Θέμα Φεβρουαρίου 2021....134
Θέμα Ιουνίου 2020....136
Θέμα Σεπτεμβρίου 2020....137
Θέμα Ιουνίου 2016 με χρονοπρογραμματισμένη διαδικασία (polling)....138

Τυπολόγιο

Μετατροπή από δεκαεξαεδικό σε δυαδικό/δεκαδικό σύστημα

Hex	Binary				Δεκαδικό
	2^3	2^2	2^1	2^0	
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	1	1	1	7
8	1	0	0	0	8
9	1	0	0	0	9
A	1	0	1	0	10
B	1	0	1	1	11
C	1	1	0	0	12
D	1	1	0	1	13
E	1	1	1	0	14
F	1	1	1	1	15

Μετατροπή από οκταδικό σε δυαδικό σύστημα

Octal	Binary		
	2^2	2^1	2^0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Τυπολόγιο δυνάμεων του 2

$2^0 = 1$

$2^1 = 2$

$2^2 = 4$

$2^3 = 8$

$2^4 = 16$

$2^5 = 32$

$2^6 = 64$

$2^7 = 128$

$2^8 = 256$

$2^9 = 512$

$2^{10} = 1024 = 1K$

$2^{11} = 2048 = 2K$

$2^{12} = 4096 = 4K$

$2^{13} = 8192 = 8K$

$2^{14} = 16384 = 16K$

$2^{15} = 32768 = 32K$

$2^{16} = 65536 = 64K$

Μονάδες μέτρησης χωρητικότητας μνήμης

1 bit = 0 ή 1 (δυαδικό ψηφίο)

1 byte = 8 bits (ψηφιολέξη) = 1B

1KB = 2^{10} bytes (Kilobyte)

1MB = 2^{20} bytes (Megabyte)

1GB = 2^{30} bytes (Gigabyte)

1TB = 2^{40} bytes (Terabyte)

Πολλαπλάσια του byte

1 byte (ψηφιολέξη) = 8 bits

1 halfword (μισή λέξη) = 16 bits = 2 bytes

1 word (λέξη) = 32 bits = 4 bytes

1 double word (διπλή λέξη) = 64 bits = 8 bytes

1 quad word (τετραπλή λέξη) = 128 bits = 16 bytes

Κεφάλαιο 1 – Απόδοση Η/Υ

Στην επίλυση ασκήσεων αυτού του κεφαλαίου, όταν υπάρχει κρυφή μνήμη (είτε ενοποιημένη, είτε ξεχωριστή) και κύρια μνήμη, οι ασκήσεις επιλύονται κάνοντας μια από τις παρακάτω **δύο υποθέσεις. Θα πρέπει όμως να επισημανθεί ότι** η πλειοψηφία των ασκήσεων του Κεφαλαίου αυτού επιλύεται με βάση την Α′ υπόθεση.

Α′ Υπόθεση: Γίνεται **ταυτόχρονη προσπέλαση της κρυφής μνήμης και της κύριας μνήμης** και στην περίπτωση που η ζητούμενη πληροφορία βρεθεί στην κρυφή μνήμη **σταματά** η προσπέλαση της κύριας μνήμης. Τότε αν υπάρχει **ενοποιημένη** (κοινή) κρυφή μνήμη εντολών και δεδομένων, ο $\text{ΧΕ}_{\text{εντολών προγρ.}} = \left\{\sum_{i=1}^{v}[\text{ΚΡΕ}_i(\text{ΚΜΕ}) + \text{Ε} * \text{ΚΡΕ}_{i(\text{κρυφή μνήμη})} + (1-\text{Ε}) * \text{ΚΡΕ}_{i(\text{κύρια μνήμη})}] * \Pi_i\right\}$ * tcycle. Αν υπάρχει **ξεχωριστή** κρυφή μνήμη εντολών και δεδομένων, ο $\text{ΧΕ}_{\text{εντολών προγρ.}} = \left\{\sum_{i=1}^{v}[\text{ΚΡΕ}_i(\text{ΚΜΕ}) + \text{Ε}_1 * \text{ΚΡΕ}_{i(\text{κρυφή μνήμη εντολών})} + (1-\text{Ε}_1) * \text{ΚΡΕ}_{i(\text{κύρια μνήμη−προσπέλαση εντολών})} + \text{Ε}_2 * \text{ΚΡΕ}_{i\ (\text{κρυφή μνήμη δεδομένων})} + (1-\text{Ε}_2) * \text{ΚΡΕ}_{i(\text{κύρια μνήμη−προσπέλαση δεδομένων})}] * \Pi_i\right\}$ * tcycle

Β′ Υπόθεση: Γίνεται προσπέλαση της κρυφής μνήμης και **εάν η ζητούμενη πληροφορία δεν βρεθεί στην κρυφή μνήμη τότε γίνεται προσπέλαση της κύριας μνήμης**. Αν υπάρχει **ενοποιημένη** κρυφή μνήμη εντολών και δεδομένων, ο $\text{ΧΕ}_{\text{εντολών προγρ.}} = \left\{\sum_{i=1}^{v}[\text{ΚΡΕ}_i(\text{ΚΜΕ}) + \text{ΚΡΕ}_{i(\text{κρυφή μνήμη})} + (1-\text{Ε}) * \text{ΚΡΕ}_{i(\text{κύρια μνήμη})}] * \Pi_i\right\}$ * tcycle. Αν υπάρχει **ξεχωριστή** κρυφή μνήμη εντολών και δεδομένων, ο $\text{ΧΕ}_{\text{εντολών προγρ.}} = \left\{\sum_{i=1}^{v}[\text{ΚΡΕ}_i(\text{ΚΜΕ}) + \text{ΚΡΕ}_{i(\text{κρυφή μνήμη εντολών})} + (1-\text{Ε}_1) * \text{ΚΡΕ}_{i(\text{κύρια μνήμη−προσπέλαση εντολών})} + \text{ΚΡΕ}_{i\ (\text{κρυφή μνήμη δεδομένων})} + (1-\text{Ε}_2) * \text{ΚΡΕ}_{i(\text{κύρια μνήμη−προσπέλαση δεδομένων})}] * \Pi_i\right\}$ * tcycle. Δηλαδή στην περίπτωση αυτή **δεν** εμφανίζονται καθόλου οι λόγοι επιτυχίας στην κρυφή μνήμη, ενώ οι λόγοι αποτυχίας εμφανίζονται κανονικά.

Θέμα Σεπτεμβρίου 2021 - Υπολογισμός μέσου πλήθους κύκλων ρολογιού

Θεωρήστε επεξεργαστή με ένα επίπεδο κρυφής μνήμης εντολών με λόγο επιτυχίας h = 0.9 (δεν διαθέτει κρυφή μνήμη δεδομένων). Ο χρόνος προσπέλασης της κρυφής μνήμης εντολών είναι 2 κύκλοι ρολογιού ενώ ο χρόνος προσπέλασης της κύριας μνήμης είναι 17 κύκλοι ρολογιού. Ο συνολικός χρόνος προσπέλασης και μεταφοράς ενός μπλοκ πληροφορίας από την κύρια μνήμη στην κρυφή μνήμη εντολών ισούται με 35 κύκλους ρολογιού. Θεωρήστε ότι όλες οι εντολές έχουν το ίδιο μήκος και όλα τα δεδομένα έχουν το ίδιο μήκος και ότι η προσπέλαση εντολών και δεδομένων αντίστοιχα γίνεται ανά εντολή και ανά δεδομένο (μία εντολή ανά προσπέλαση ή ένα δεδομένο ανά προσπέλαση). Επίσης θεωρήστε ότι η ΚΜΕ προσπελάζει πληροφορία μόνο από το κοντινότερο επίπεδο της ιεραρχικής μνήμης.

Α. Θεωρήστε ότι η **προσπέλαση της κύριας μνήμης γίνεται παράλληλα με την προσπέλαση της κρυφής μνήμης**.

1. Να υπολογίσετε το μέσο πλήθος κύκλων ρολογιού που απαιτούνται για την προσπέλαση μιας εντολής.

Δώστε την τιμή που υπολογίσατε χωρίζοντας το ακέραιο από το δεκαδικό μέρος (εάν υπάρχει) με κόμμα [ ].

2. Να υπολογίσετε το μέσο πλήθος κύκλων ρολογιού που απαιτούνται για την προσπέλαση ενός δεδομένου.

Δώστε την τιμή που υπολογίσατε χωρίζοντας το ακέραιο από το δεκαδικό μέρος (εάν υπάρχει) με κόμμα [ ].

Β. Θεωρήστε ότι η **προσπέλαση της κύριας μνήμης γίνεται μόνο σε περίπτωση αποτυχίας εύρεσης του ζητούμενου στην κρυφή μνήμη**.

3. Να υπολογίσετε το μέσο πλήθος κύκλων ρολογιού που απαιτούνται για την προσπέλαση μιας εντολής.

Δώστε την τιμή που υπολογίσατε χωρίζοντας το ακέραιο από το δεκαδικό μέρος (εάν υπάρχει) με κόμμα [ ].

4. Να υπολογίσετε το μέσο πλήθος κύκλων ρολογιού που απαιτούνται για την προσπέλαση ενός δεδομένου.

Δώστε την τιμή που υπολογίσατε χωρίζοντας το ακέραιο από το δεκαδικό μέρος (εάν υπάρχει) με κόμμα [ ].

Λύση

Δεν δίνονται οι κύκλοι ρολογιού που δαπανώνται στην ΚΜΕ, και εστιάζουμε μόνο στο σύστημα μνήμης (κύρια – κρυφή).

Α.1. Μέσο πλήθος κ.ρ. για την προσπέλαση μιας εντολής: $ΧΕ_{εντολών\ προγρ.}$ = 0.9 * 2 κ.ρ. + (1 - 0.9) * 37 κ.ρ. = 5.5 κ.ρ.

Α.2. Μέσο πλήθος κ.ρ. για την προσπέλαση ενός δεδομένου: $ΧΕ_{εντολών\ προγρ.}$ = 17 κ.ρ.

Β.3. Μέσο πλήθος κ.ρ. για την προσπέλαση μιας εντολής: $ΧΕ_{εντολών\ προγρ.}$ = 2 κ.ρ. + (1 - 0.9) * 37 κ.ρ. = 5.7 κ.ρ.

Β.4. Μέσο πλήθος κ.ρ. για την προσπέλαση ενός δεδομένου: $ΧΕ_{εντολών\ προγρ.}$ = 17 κ.ρ. (δεν αλλάζει)

Θέμα Προόδου 2021 - Υπολογισμός χρόνου εκτέλεσης

Θεωρείστε επεξεργαστή με συχνότητα ρολογιού 0.25 GHz και **κύρια μνήμη** με χρόνο προσπέλασης 228 nsec. Οι εντολές του επεξεργαστή σε επίπεδο γλώσσας μηχανής ταξινομούνται σε τρία είδη Α, Β και Γ, τα χαρακτηριστικά των οποίων δίνονται στον επόμενο πίνακα. Στον πίνακα αυτό δίνεται επίσης και το πλήθος των εκτελούμενων εντολών κάθε είδους ενός προγράμματος. Σε κάθε προσπέλαση της κύριας μνήμης προσπελαύνεται μία ψηφιολέξη.

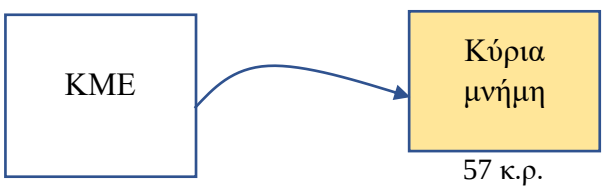
είδος εντολής	κύκλοι ρολογιού ανά εντολή	πλήθος ψηφιολέξεων που καταλαμβάνει η εντολή	πλήθος ψηφιολέξεων μνήμης που προσπελάζει η εντολή κατά την εκτέλεσή της	Πλήθος εκτελούμενων εντολών
Α	1	1	0	276
Β	4	4	2	523
Γ	6	4	3	616

1. Να υπολογίσετε το χρόνο εκτέλεσης του προγράμματος. Να δώσετε το αποτέλεσμα χωρίς χρήση τελειών π.χ. 67402
2. Ο σχεδιαστής του επόμενου μοντέλου αποφάσισε να προσθέσει στον υπολογιστή μία **ενοποιημένη** κρυφή μνήμη εντολών και δεδομένων με λόγο επιτυχίας 0.90 και χρόνο προσπέλασης 8 nsec. Ο χρόνος προσπέλασης και μεταφοράς ενός μπλοκ εντολών ή δεδομένων από την κύρια μνήμη στην κρυφή μνήμη είναι 424 nsec. Σημειώστε ότι: 1. Η κεντρική μονάδα επεξεργασίας διαβάζει πληροφορία μόνο από την κρυφή μνήμη. 2. Η ΚΜΕ αναζητεί πληροφορία **παράλληλα** στην κρυφή μνήμη και στην κύρια μνήμη. Να υπολογίσετε **πόσες φορές μικρότερος** έγινε ο χρόνος εκτέλεσης του προγράμματος στο νέο μοντέλο σε σύγκριση με το παλιό. Να δώσετε το αποτέλεσμα ως ακέραιο μέρος και τα δύο πιο σημαντικά ψηφία του δεκαδικού μέρους, απορρίπτοντας τα υπόλοιπα δεκαδικά ψηφία (περικοπή) π.χ. 3.41.

Λύση

Αρχικά θα υπολογίσουμε τη διάρκεια του κύκλου ρολογιού $t_{cycle} = \frac{1}{\text{συχνότητα λειτουργίας επεξεργαστή}} = \frac{1}{0.25\ \text{GHz}}$ = 4 nsec. Στη συνέχεια πρέπει να διαιρέσουμε το χρόνο προσπέλασης της κύριας μνήμης που δίνεται σε nsec με το t_{cycle} που υπολογίσαμε, για να τον μετατρέψουμε σε κύκλους ρολογιού. Πιο συγκεκριμένα, για την κύρια μνήμη που έχει χρόνο προσπέλασης 228 nsec, θα έχουμε 228 nsec/4 nsec = 57 κ.ρ.

1. Υπολογίζουμε το χρόνο εκτέλεσης του προγράμματος, όταν στο σύστημα υπάρχει **μόνο** κύρια μνήμη. Η κύρια μνήμη **περιέχει πάντα εντολές και δεδομένα**.



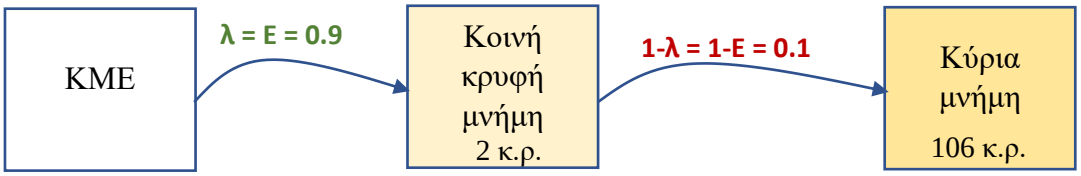
$ΧΕΕ_Α$ = 1 κ.ρ. + (1 + 0) * 57 κ.ρ. = 58 κ.ρ.

$ΧΕΕ_Β$ = 4 κ.ρ. + (4 + 2) * 57 κ.ρ. = 346 κ.ρ.

$ΧΕΕ_Γ$ = 6 κ.ρ. + (4 + 3) * 57 κ.ρ. = 405 κ.ρ.

$XE_{εντολών προγρ.}$ = (XEE_A * $Π_A$ + XEE_B * $Π_B$ + $XEE_Γ$ * $Π_Γ$) * t_{cycle} = (58 κ.ρ. * 276 εντολές + 346 κ.ρ. * 523 εντολές + 405 κ.ρ. * 616 εντολές) * 4 nsec = 1785784 nsec.

2. Υπολογίζουμε το χρόνο εκτέλεσης του προγράμματος, όταν στο σύστημα υπάρχει **ενοποιημένη κρυφή μνήμη** και κύρια μνήμη. Ο χρόνος προσπέλασης της ενοποιημένης κρυφής μνήμης που είναι της τάξης των 8 nsec, θα γίνει 8 nsec/4 nsec = 2 κ.ρ. και ο χρόνος προσπέλασης και μεταφοράς ενός μπλοκ εντολών/δεδομένων από την κύρια στην κρυφή της τάξης των 424 nsec, θα γίνει 424 nsec/4 nsec = 106 κ.ρ.



Επειδή η ΚΜΕ αναζητά πληροφορία **παράλληλα** στην κρυφή και την κύρια μνήμη, θα χρησιμοποιήσουμε τους τύπους που χρησιμοποιούν λόγο επιτυχίας για την κρυφή μνήμη (είτε ενοποιημένη όπως εδώ, είτε ξεχωριστή), δηλαδή θα έχουμε:

$$XE_{εντολών προγρ.} = \left\{\sum_{i=1}^{v}[KPE_i(ΚΜΕ) + E * KPE_{i(κρυφή μνήμη)} + (1 - E) * KPE_{i(κύρια μνήμη)}] * Π_i\right\} * tcycle.$$

XEE_A = 1 κ.ρ. + 0.9 * (1 + 0) * 2 κ.ρ. + 0.1 * (1 + 0) * 108 κ.ρ. = 13.6 κ.ρ.

XEE_B = 4 κ.ρ. + 0.9 * (4 + 2) * 2 κ.ρ. + 0.1 * (4 + 2) * 108 κ.ρ. = 79.6 κ.ρ.

$XEE_Γ$ = 6 κ.ρ. + 0.9 * (4 + 3) * 2 κ.ρ. + 0.1 * (4 + 3) * 108 κ.ρ = 94.2 κ.ρ.

$XE_{εντολών προογρ.}$ = (XEE_A * $Π_A$ + XEE_B * $Π_B$ + $XEE_Γ$ * $Π_Γ$) * t_{cycle} = (13.6 κ.ρ. * 276 εντολές + 79.6 κ.ρ. * 523 εντολές + 94.2 κ.ρ. * 616 εντολές) * 4 nsec = 413646.4 nsec. Αυτός είναι ο νέος χρόνος εκτέλεσης. Για να υπολογίσουμε πόσες φορές μικρότερος έγινε ο χρόνος εκτέλεσης του νέου προγράμματος στο νέο μοντέλο σε σχέση με το παλιό, θα διαιρέσουμε το νέο χρόνο εκτέλεσης προς τον παλιό χρόνο εκτέλεσης και θα έχουμε: $\frac{413646.4\ \text{nsec}}{1785784\ \text{nsec}}$ = 0.231632941 → 23.1%. Ο χρόνος εκτέλεσης ενός προγράμματος ονομάζεται ισοδύναμα απόδοση.

Παράδειγμα 1.1 - Υπολογισμός χρόνου εκτέλεσης - ΜΧΠΜ - μέσου χρόνου εκτέλεσης εντολής

Θεωρούμε έναν υπολογιστή που αποτελείται από την ΚΜΕ, μια ενοποιημένη κρυφή μνήμη για εντολές και δεδομένα και κύρια μνήμη. Η κρυφή μνήμη έχει λόγο επιτυχίας 0.9 και η προσπέλασή της βάζει καθυστέρηση **ενός** κύκλου ρολογιού. Η κύρια μνήμη σε περίπτωση αποτυχίας να βρεθεί η πληροφορία στην κρυφή μνήμη, βάζει καθυστέρηση 20 κύκλων ρολογιού για τη μεταφορά ενός μπλοκ πληροφορίας από την κύρια μνήμη στην κρυφή. Ο χρόνος κύκλου ρολογιού είναι 4 ns. Υποθέστε ότι το σύνολο των εντολών του προγράμματος σε επίπεδο γλώσσας μηχανής αποτελείται από τρία είδη εντολών, όπως φαίνεται στον Πίνακα 1. Υποθέστε ότι τόσο το εκτελούμενο πρόγραμμα όσο και τα απαιτούμενα δεδομένα βρίσκονται στην κύρια μνήμη. Η ΚΜΕ μπορεί να προσπελάσει μόνο πληροφορία που βρίσκεται στην κρυφή μνήμη.

α. Υπολογίστε το χρόνο εκτέλεσης ενός προγράμματος του οποίου τα χαρακτηριστικά δίνονται στον Πίνακα 2.

β. Υπολογίστε το μέσο χρόνο προσπέλασης του συστήματος μνήμης.

γ. Υπολογίστε το μέσο χρόνο εκτέλεσης εντολής.

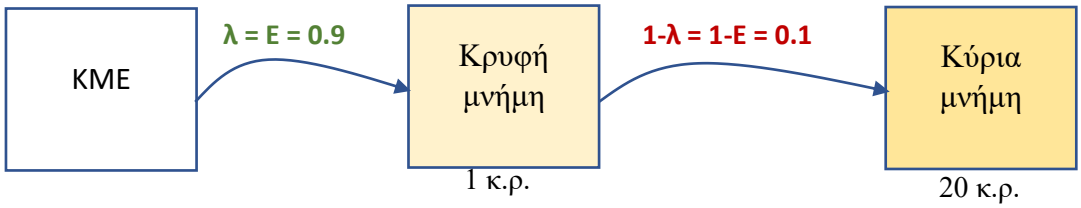
Πίνακας 1

Είδος εντολής	Κύκλοι ρολογιού ανά εντολή	Ψηφιολέξεις που καταλαμβάνει η εντολή στη μνήμη	Πλήθος ψηφιολέξεων που προσπελαύνει η εντολή κατά την εκτέλεσή της
A	4	1	0
Γ	7	2	1
E	10	3	2

Πίνακας 2

Είδος εντολής	Πλήθος εκτελούμενων εντολών
Α	400
Γ	10
Ε	30

Λύση



α. Θα υπολογίσουμε το χρόνο εκτέλεσης για το κάθε είδος εντολής ξεχωριστά. Επειδή έχουμε ενοποιημένη (κοινή) κρυφή μνήμη εντολών και δεδομένων, θα πρέπει σε κάθε περίπτωση (σε κάθε είδος εντολής) να αθροίσουμε ψηφιολέξεις εντολών και δεδομένων μαζί. Σε περίπτωση αποτυχίας να βρεθεί η ζητούμενη πληροφορία (εντολές – δεδομένα) στην κρυφή μνήμη, έχουμε καθυστέρηση 21 κ.ρ. και αυτό γιατί πρώτα προσπελαύνεται η κρυφή μνήμη και μετά η κύρια μνήμη.

ΧΕΕ$_{Α}$ = 4 κ.ρ. + **0.9 * (1 + 0) * 1 κ.ρ.** + **0.1 * (1 + 0) * 21 κ.ρ.** = 7 κ.ρ.

ΧΕΕ$_{Γ}$ = 7 κ.ρ. + **0.9 * (2 + 1) * 1 κ.ρ.** + **0.1 * (2 + 1) * 21 κ.ρ.** = 16 κ.ρ.

ΧΕΕ$_{Ε}$ = 10 κ.ρ. + **0.9 * (3 + 2) * 1 κ.ρ.** + **0.1 * (3 + 2) * 21 κ.ρ.** = 25 κ.ρ.

ΧΕ$_{εντ.\ πρ.}$ = **(ΧΕΕ$_{Α}$ * 400 + ΧΕΕ$_{Γ}$ * 10 + ΧΕΕ$_{Ε}$ * 30) * 4 nsec = 14840 nsec**

β. Ζητάμε το **μέσο χρόνο προσπέλασης του συστήματος μνήμης (κρυφή – κύρια), δηλ. το μέγεθος ΜΧΠΜ**. Για να το βρούμε θα πρέπει να υπολογίσουμε αρχικά το συνολικό χρόνο προσπέλασης του συστήματος μνήμης, που είναι ο αριθμητής του κλάσματος και στη συνέχεια θα πρέπει να υπολογίσουμε το πλήθος προσπελάσεων μνήμης που είναι ο παρονομαστής του κλάσματος, δηλαδή: ΜΧΠΜ $= \frac{\text{Συνολικός χρόνος προσπέλασης συστήματος μνήμης}}{\text{Πλήθος προσπελάσεων μνήμης}}$.

Αναφορικά με το συνολικό χρόνο προσπέλασης του συστήματος μνήμης, θα πρέπει να αφαιρέσουμε από κάθε επιμέρους χρόνο που υπολογίσαμε πριν, τους κ.ρ. που δαπανώνται στην ΚΜΕ και στη συνέχεια να πολλαπλασιάσουμε αυτή τη διαφορά με το πλήθος εντολών του αντίστοιχου είδους. Αναφορικά με το πλήθος προσπελάσεων μνήμης, αυτό προκύπτει αν αθροίσουμε για κάθε είδους εντολή, τις ψηφιολέξεις εντολών και δεδομένων και πολλαπλασιάσουμε αυτό το άθροισμα με το αντίστοιχο πλήθος εντολών. Επομένως, το μέγεθος ΜΧΠΜ $= \frac{\text{Συνολικός χρόνος προσπέλασης συστήματος μνήμης}}{\text{Πλήθος προσπελάσεων μνήμης}} = \frac{[(7\ \text{κ.ρ.} - 4\ \text{κ.ρ.}) * 300 + (18\ \text{κ.ρ.} - 7\ \text{κ.ρ.}) * 20 + (29\ \text{κ.ρ.} - 10\ \text{κ.ρ.}) * 50] * 5\ \text{nsec}}{(1+0) * 300 + (2+1) * 20 + (3+2) * 50} = 3.39$ nsec.

γ. **Μέσος χρόνος εκτέλεσης εντολής** $= \frac{\text{συνολικός χρόνος εκτέλεσης προγράμματος}}{\text{πλήθος εντολών}} = \frac{14840\ \text{nsec}}{440\ \text{εντολές}} = 33.73$ nsec/εντολή.

Παρατήρηση: Ο **μέσος χρόνος εκτέλεσης εντολής** είναι διαφορετικός από το **μέσο αριθμό κύκλων ρολογιού ανά εντολή (ΜΚΡΕ), καθώς το** ΜΚΡΕ $= \frac{7\ \text{κ.ρ.x}\ 400 + 16\ \text{κ.ρ.}\ \text{x}\ 10 + 25\ \text{κ.ρ.x}\ 30}{440\ \text{εντολές}} =$ ……κ.ρ./εντολή. Ουσιαστικά, η διαφορά των δύο μεγεθών, που είναι πολύ μικρή, έγκειται στον αριθμητή του κλάσματος, καθώς στη δεύτερη περίπτωση (ΜΚΡΕ) δεν πολλαπλασιάζεται με τη διάρκεια του κύκλου ρολογιού.

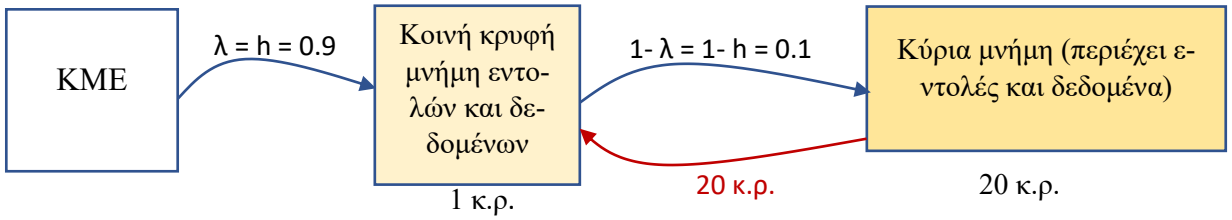
Θέμα Φεβρουαρίου 2014 με ΜΚΡΕ και ΜΧΠΜ

Να βρείτε το πλήθος των κύκλων ρολογιού που απαιτούνται για την εκτέλεση 1.000.000 εντολών σε ένα υπολογιστή που

ο μέσος αριθμός κύκλων ρολογιού εκτέλεσης στην ΚΜΕ ανά εντολή είναι 4 και ο μέσος αριθμός προσπελάσεων του συστήματος μνήμης ανά εντολή είναι 6. Θεωρείστε ότι ο υπολογιστής διαθέτει ενοποιημένη κρυφή μνήμη με ρυθμό επιτυχίας 0.9, ο χρόνος προσπέλασης της κρυφής μνήμης και της κύριας μνήμης σε κύκλους ρολογιού είναι 1 και 20 κύκλοι ρολογιού αντίστοιχα, η κύρια μνήμη περιέχει όλες τις εντολές και τα απαιτούμενα δεδομένα και η ΚΜΕ επικοινωνεί με το σύστημα μνήμης μέσω της κρυφής μνήμης.

Λύση

Από την εκφώνηση της άσκησης δίνονται ως μεγέθη ο μέσος αριθμός κύκλων ρολογιού ανά εντολή, δηλ. το **ΜΚΡΕ** και ο μέσος αριθμός προσπελάσεων του συστήματος μνήμης, δηλ. το **ΜΧΠΜ**.



$$\text{ΜΚΡΕ} = \frac{\text{Συνολικός αριθμός κύκλων ρολογιού}}{\text{πλήθος εντολών που εκτελέστηκαν}} = \frac{\text{Συνολικός αριθμός κύκλων ρολογιού}}{1000000} \Rightarrow \text{Συνολικός αριθμός κύκλων ρολογιού}$$

= 4 * 1.000.000 = 4.000.000 κ.ρ. Αυτό το μέγεθος **αφορά τους κύκλους ρολογιού που δαπανώνται στην ΚΜΕ**. Θα πρέπει στη συνέχεια να υπολογίσουμε και τους αντίστοιχους κύκλους ρολογιού που δαπανώνται στο σύστημα μνήμης (κύρια και κρυφή μνήμη) και να αθροίσουμε τα δύο αποτελέσματα.

Θεωρώντας ότι ο **μέσος αριθμός προσπελάσεων του συστήματος μνήμης ανά εντολή είναι 6**, και με δεδομένο ότι έχουμε 1.000.000 εντολές και θεωρώντας ότι κάθε εντολή καταλαμβάνει μια θέση μνήμης, ο τύπος για το ΜΧΠΜ είναι:

$$\text{ΜΧΠΜ} = \frac{\text{Συνολικός χρόνος προσπέλασης του συστήματος μνήμης}}{\text{πλήθος προσπελάσεων μνήμης}} = \frac{\text{Συνολικός χρόνος προσπέλασης του συστήματος μνήμης}}{1.000.000}$$

$\Rightarrow$ **Συνολικός χρόνος προσπελάσεων του συστήματος μνήμης για αυτές τις εντολές είναι 6.000.000**. Αυτό το **μέγεθος θα πρέπει να μετατραπεί σε κύκλους ρολογιού** και αυτό γίνεται θεωρώντας ότι οι προσπελάσεις αυτές μπορούν να γίνουν τόσο στην κρυφή μνήμη με πιθανότητα 0.9, όσο και στην κύρια μνήμη με πιθανότητα 0.1, δηλ. 6.000.000 x 0.9 x 1 κ.ρ. + 6.000.000 x 0.1 x 21 κ.ρ. = 5.400.000 κ.ρ. + 12.600.000 κ.ρ. Επομένως το **συνολικό πλήθος των κ.ρ.** που απαιτούνται είναι 4.000.000 κ.ρ. + 5.400.000 κ.ρ. + 12.600.000 κ.ρ. = 22.000.000 κ.ρ.

Παρατήρηση: Για τον υπολογισμό αυτού του μεγέθους, έχουμε κάνει την παραδοχή ότι κάθε εντολή καταλαμβάνει 1 θ.μ., οπότε για την προσκόμιση κάθε εντολής απαιτείται αντίστοιχα 1 προσπέλαση μνήμης, και επειδή έχουμε 1.000.000 εντολές, έχουμε και 1.000.000 θέσεις μνήμης, άρα απαιτούνται και 1.000.000 προσπελάσεις, για την προσκόμιση αυτών των εντολών.

Θέμα Ιουνίου 2021

Ένας υπολογιστής ειδικού σκοπού διαθέτει **μόνο κρυφή μνήμη εντολών με λόγο επιτυχίας hε = 90**. Η προσπέλαση του συστήματος μνήμης (κρυφή μνήμη ή κύρια μνήμη) από την ΚΜΕ γίνεται ανά ψηφιολέξη (byte). Ο χρόνος προσπέλασης της κρυφής μνήμης ισούται με 3 κύκλους ρολογιού, ενώ ο χρόνος προσπέλασης της κύριας μνήμης ισούται με 24 κύκλους ρολογιού. Ο συνολικός χρόνος προσπέλασης και μεταφοράς ενός μπλοκ πληροφοριών από την κύρια στην κρυφή μνήμη ισούται με 80 κύκλους ρολογιού. Η ΚΜΕ μπορεί να προσπελάσει εντολές μόνο από την κρυφή μνήμη. Ο χρόνος κύκλου ρολογιού είναι 3 ns. Υποθέστε ότι το σύνολο εντολών του προγράμματος σε επίπεδο γλώσσας μηχανής αποτελείται από τρία είδη εντολών που φαίνονται παρακάτω. Υποθέστε ότι τόσο το εκτελούμενο πρόγραμμα όσο και τα απαιτούμενα δεδομένα

βρίσκονται στην κύρια μνήμη. Θεωρείστε ότι εκτελείται ένα πρόγραμμα του οποίου το πλήθος των εκτελούμενων εντολών κάθε είδους Α, Β, Γ δίνεται πιο κάτω στα Δεδομένα.

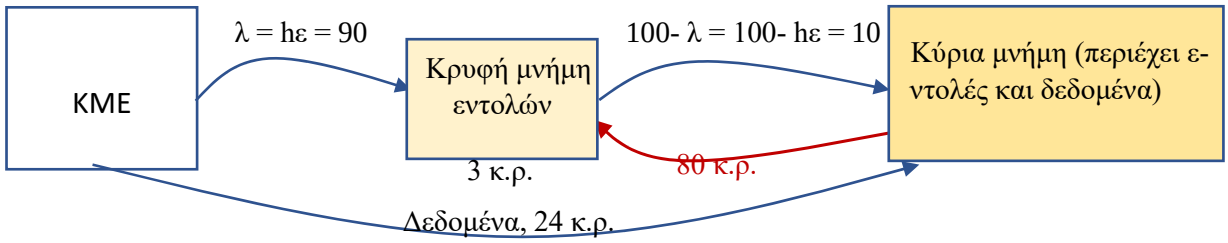
Δεδομένα: Εντολές τύπου Α, κύκλοι ρολογιού για την εκτέλεση μιας εντολής στην ΚΜΕ: 7, ψηφιολέξεις που καταλαμβάνει μια εντολή στη μνήμη: 3, πλήθος ψηφιολέξεων δεδομένων που προσπελαύνει μια εντολή κατά την εκτέλεσή της: 0, πλήθος εκτελούμενων εντολών: 121. Εντολές τύπου Β, κύκλοι ρολογιού για την εκτέλεση μιας εντολής στην ΚΜΕ: 5, ψηφιολέξεις που καταλαμβάνει μια εντολή στη μνήμη: 1, πλήθος ψηφιολέξεων δεδομένων που προσπελαύνει την εκτέλεσή της:2, πλήθος εκτελούμενων εντολών: 213. Εντολές τύπου Γ, κύκλοι ρολογιού για την εκτέλεση μιας εντολής στην ΚΜΕ 6, ψηφιολέξεις που καταλαμβάνει μια εντολή στη μνήμη: 2, πλήθος ψηφιολέξεων δεδομένων που προσπελαύνει την εκτέλεσή της:0, πλήθος εκτελούμενων εντολών: 169.

Υπολογίστε το **μέσο χρόνο εκτέλεσης του συστήματος μνήμης κατά την εκτέλεση του προγράμματος σε nsec**. Να δώσετε μόνο το ακέραιο μέρος του αποτελέσματος χωρίς να χρησιμοποιήσετε τελεία, π.χ. εάν το αποτέλεσμα είναι 123456.56, να δώσετε τον αριθμό 123456

Είδος εντολής	**Κύκλοι ρολογιού ανά εντολή (ΚΜΕ)**	**Ψηφιολέξεις που καταλαμβάνει μια εντολή στη μνήμη (εντολές)**	**Πλήθος ψηφιολέξεων που προσπελαύνει η εντολή κατά την εκτέλεσή της (δεδομένα)**	**Πλήθος εκτελούμενων εντολών**
Α	7	3	0	121
Β	5	1	2	213
Γ	6	2	0	169

Λύση

Παρότι από την εκφώνηση ζητείται **μόνο** το μέγεθος ΜΧΠΜ, θα υπολογίσουμε **επιπλέον** και το συνολικό χρόνο εκτέλεσης αυτού του προγράμματος. Επειδή υπάρχει **μόνο κρυφή μνήμη εντολών**, τα δεδομένα τα παίρνει η ΚΜΕ **απευθείας** από την κύρια μνήμη σε χρόνο ίσο με 24 κ.ρ. Για να υπολογίσουμε το **συνολικό χρόνο εκτέλεσης του προγράμματος**, θα πρέπει να υπολογίσουμε πρώτα το χρόνο εκτέλεσης κάθε είδους εντολήςχωριστά.



ΧΕΕ$_A$ = 7 κ.ρ. + **0.9 * 3 ψηφ. εντολών * 3 κ.ρ.** + 0.1 * 3 ψηφ. εντολών* 83 κ.ρ. + 0 ψηφ. δεδομένων * 24 κ.ρ. = 40 κ.ρ.

ΧΕΕ$_B$ = 5 κ.ρ. + **0.9 * 1 ψηφ. εντολών * 3 κ.ρ.** + 0.1 * 1 ψηφ. εντολών * 83 κ.ρ. + 2 ψηφ. δεδομένων* 24 κ.ρ. = 64 κ.ρ.

ΧΕΕ$_Γ$ = 6 κ.ρ. + **0.9 * 2 ψηφ. εντολών * 3 κ.ρ.** + 0.1 * 2 ψηφ. εντολών * 83 κ.ρ. + 0 ψηφ. δεδομένων* 24 κ.ρ. = 28 κ.ρ.

$ΧΕ_{εντ.\ πρ.}$ = (ΧΕΕ$_A$ * Π$_A$ + ΧΕΕ$_B$ * Π$_B$ + ΧΕΕ$_Γ$ * Π$_Γ$) * 3 nsec = (40 κ.ρ.* 121 + 64 κ.ρ.* 213 + 28 κ.ρ.* 169) * 3 nsec = 69612 nsec.

Στη συνέχεια θα υπολογίσουμε το μέσο χρόνο προσπέλασης του συστήματος μνήμης. Για να γίνει αυτό, θα πρέπει να λάβουμε υπόψη μας **μόνο το σύστημα μνήμης**, δηλ. κύρια μνήμη και κρυφή μνήμη και όχι την ΚΜΕ. Το μέγεθος ΜΧΠΜ δίνεται από τον εξής τύπο:

$$\text{ΜΧΠΜ} = \frac{\textbf{συνολικός χρόνος προσπέλασης συστήματος μνήμης}}{\textbf{πλήθος προσπελάσεων μνήμης}} = \frac{[(40-7)*121+(64-5)*213+(28-6)*169]*3\ \textbf{nsec}}{(3+0)*121+(1+2)*213+(2+0)*169} = \frac{\textbf{60834 nsec}}{\textbf{1340 πρ}} =$$

45.3985 nsec/πρ. → 45

Παρατήρηση 1: Για τον υπολογισμό του μεγέθους ΜΧΠΜ λαμβάνουμε υπόψη μόνο το σύστημα μνήμης (κρυφή και κύρια

μνήμη) με αποτέλεσμα να αφαιρούμε στον αριθμητή τους κύκλους ρολογιού που δαπανώνται στην ΚΜΕ. Στον παρανομαστή του ίδιου κλάσματος αθροίζουμε ψηφιολέξεις εντολών και δεδομένων και πολλαπλασιάζουμε το κάθε άθροισμα με το πλήθος εντολών του αντίστοιχου είδους.

Παρατήρηση 2: Υπάρχει περίπτωση οι χρόνοι προσπέλασης της κρυφής μνήμης ή/και της κύριας μνήμης να δοθούν όχι σε κ.ρ. όπως στη συγκεκριμένη περίπτωση, αλλά να δοθούν σε nsec. Θα πρέπει **τότε να διαιρέσουμε κάθε τέτοιο χρόνο με τη διάρκεια (περίοδο) του κύκλου ρολογιού**, προκειμένου να μετατρέψουμε τους χρόνους αυτούς σε κ.ρ.

Θέμα Ιουνίου 2021

Ένας υπολογιστής ειδικού σκοπού **διαθέτει μόνο κρυφή μνήμη εντολών** με λόγο επιτυχίας hε = 95. Η προσπέλαση του συστήματος μνήμης (κρυφή μνήμη ή κύρια μνήμη) από την ΚΜΕ γίνεται ανά ψηφιολέξη (byte). Ο χρόνος προσπέλασης της κρυφής μνήμης ισούται με 3 κύκλους ρολογιού, ενώ ο χρόνος προσπέλασης της κύριας μνήμης ισούται με 26 κύκλους ρολογιού. Ο συνολικός χρόνος προσπέλασης και μεταφοράς ενός μπλοκ πληροφοριών από την κύρια στην κρυφή μνήμη ισούται με 70 κύκλους ρολογιού. Η ΚΜΕ μπορεί να προσπελάσει εντολές μόνο από την κρυφή μνήμη. Ο χρόνος κύκλου ρολογιού είναι 4 ns. Υποθέστε ότι το σύνολο εντολών του προγράμματος σε επίπεδο γλώσσας μηχανής αποτελείται από τρία είδη εντολών που φαίνονται παρακάτω. Υποθέστε ότι τόσο το εκτελούμενο πρόγραμμα όσο και τα απαιτούμενα δεδομένα βρίσκονται στην κύρια μνήμη. Θεωρείστε ότι εκτελείται ένα πρόγραμμα του οποίου το πλήθος των εκτελούμενων εντολών κάθε είδους Α, Β, Γ δίνεται πιο κάτω στα δεδομένα.

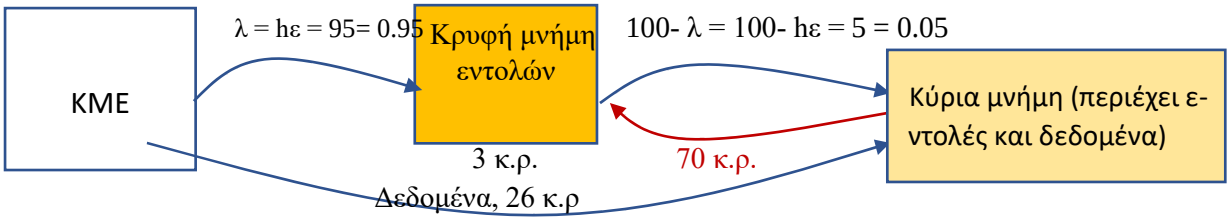
Δεδομένα: Εντολές τύπου Α, κύκλοι ρολογιού για την εκτέλεση μιας εντολής στην ΚΜΕ: 7, ψηφιολέξεις που καταλαμβάνει μια εντολή στη μνήμη: 1 πλήθος ψηφιολέξεων δεδομένων που προσπελαύνει μια εντολή κατά την εκτέλεσή της: 0, πλήθος εκτελούμενων εντολών: 468. Εντολές τύπου Β, κύκλοι ρολογιού για την εκτέλεση μιας εντολής στην ΚΜΕ: 9, ψηφιολέξεις που καταλαμβάνει μια εντολή στη μνήμη: 1, πλήθος ψηφιολέξεων δεδομένων που προσπελαύνει την εκτέλεσή της: 1, πλήθος εκτελούμενων εντολών: 56. Εντολές τύπου Γ, κύκλοι ρολογιού για την εκτέλεση μιας εντολής στην ΚΜΕ 5, ψηφιολέξεις που καταλαμβάνει μια εντολή στη μνήμη: 3, πλήθος ψηφιολέξεων δεδομένων που προσπελαύνει την εκτέλεσή της:3, πλήθος εκτελούμενων εντολών: 150.

Υπολογίστε το **μέσο χρόνο εκτέλεσης του συστήματος μνήμης κατά την εκτέλεση του προγράμματος σε nsec**. Να δώσετε μόνο το ακέραιο μέρος του αποτελέσματος χωρίς να χρησιμοποιήσετε τελεία, π.χ. εάν το αποτέλεσμα είναι 123456.56, να δώσετε τον αριθμό 123456

Είδος εντολής	Κύκλοι ρολογιού ανά εντολή (ΚΜΕ)	Ψηφιολέξεις που καταλαμβάνει μια εντολή στη μνήμη (εντολές)	Πλήθος ψηφιολέξεων που προσπελαύνει η εντολή κατά την εκτέλεσή της (δεδομένα)	Πλήθος εκτελούμενων εντολών
Α	7	1	0	**468**
Β	9	1	1	**56**
Γ	5	3	3	**150**

Λύση

Από τη στιγμή που δεν υπάρχει κρυφή μνήμη δεδομένων, τα δεδομένα τα παίρνει η ΚΜΕ απευθείας από την κύρια μνήμη.



Ζητάμε το ΜΧΠΜ, το οποίο δίνεται από τον τύπο: ΜΧΠΜ = $\frac{\text{Συνολικός χρόνος προσπέλασης συστήματος μνήμης}}{\text{Πλήθος προσπελάσεων μνήμης}}$ =

$$\frac{[(0.95*1*3\ \kappa.\rho.+0.05*1*73\ \kappa.\rho.+0*26\ \kappa.\rho.)*468+(0.95*1*3\ \kappa.\rho.+0.05*1*73\ \kappa.\rho.+1*26\ \kappa.\rho.)*56+(0.95*3*3\ \kappa.\rho.+0.05*3*73\ \kappa.\rho.+3*26\ \kappa.\rho.)*150]*4nsec}{(1+0)*468 + (1+1)*56 + (3+3)*150}$$

$= \frac{77948\text{nsec}}{1480\ \text{εντολές}}$ = 52.66 nsec/εντολή → 52.

Παρατήρηση: αν η εκφώνηση έδινε **ενιαία (κοινή) κρυφή μνήμη εντολών και δεδομένων**, θα είχαμε ένα ποσοστό στην κρυφή μνήμη, έστω λ = h = 0.95, τότε: ΜΧΠΜ = $\frac{\text{Συνολικός χρόνος προσπέλασης συστήματος μνήμης}}{\text{Πλήθος προσπελάσεων μνήμης}}$ =

$$\frac{[(0.95*(1+0)*3\ \kappa.\rho.+0.05*(1+0)*73\ \kappa.\rho.)*468+(0.95*(1+1)*3\ \kappa.\rho.+0.05*(1+1)*73\ \kappa.\rho.)*56+(0.95*(3+3)*3\ \kappa.\rho.+0.05*(3+3)*73\ \kappa.\rho.)*150]*4nsec}{(1+0)*468+(1+1)*56+(3+3)*150}$$

= …

Θέμα 2016 με υπολογισμό ΜΧΠΜ και ξεχωριστή κρυφή μνήμη εντολών/δεδομένων

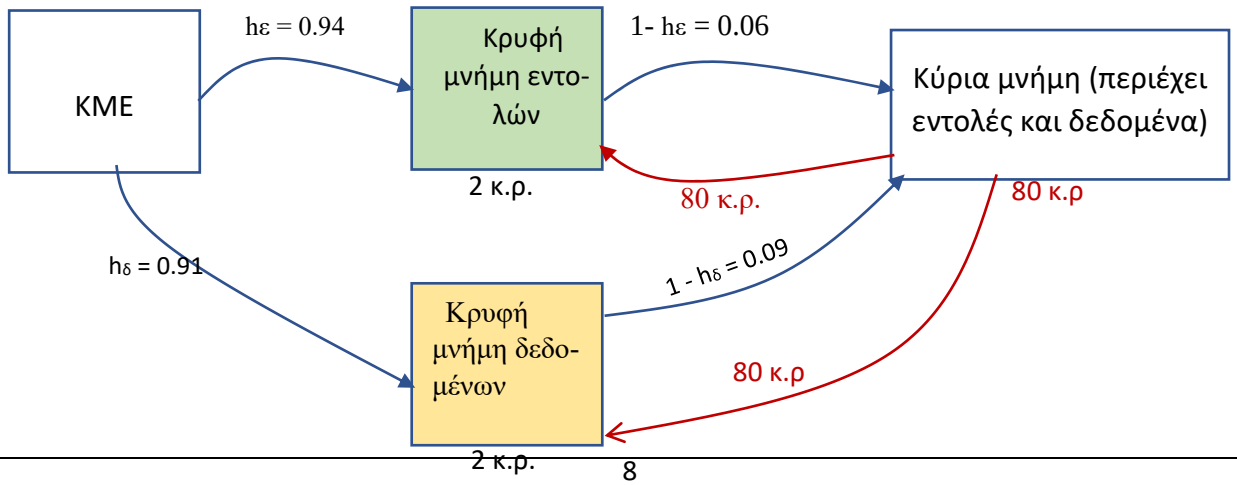
Σ' ένα υπολογιστή υπάρχει **ξεχωριστή κρυφή μνήμη εντολών και δεδομένων** με λόγο επιτυχίας αντίστοιχα $h_ε$ = 0.94 και $h_δ$ = 0.91. Η προσπέλαση οποιασδήποτε από τις δύο κρυφές μνήμες βάζει καθυστέρηση δύο κύκλων ρολογιού. Σε κάθε περίπτωση αποτυχίας να βρεθεί η εντολή ή τα δεδομένα στην κρυφή μνήμη υπάρχει καθυστέρηση 80 κύκλων ρολογιού για τη μεταφορά ενός μπλοκ εντολών ή δεδομένων από την κύρια μνήμη στην αντίστοιχη κρυφή μνήμη. Ο χρόνος κύκλου ρολογιού είναι 5 ns. Υποθέστε ότι το σύνολο των εντολών του προγράμματος σε επίπεδο γλώσσας μηχανής αποτελείται από τρία είδη εντολών, όπως φαίνεται στον Πίνακα που ακολουθεί. Υποθέστε ότι τόσο το εκτελούμενο πρόγραμμα όσο και τα απαιτούμενα δεδομένα βρίσκονται στην κύρια μνήμη. Η ΚΜΕ μπορεί να προσπελάσει εντολές και δεδομένα μόνο από τις αντίστοιχες κρυφές μνήμες. Θεωρήστε ότι η κύρια μνήμη έχει οργάνωση μίας ψηφιολέξης ανά θέση και η προσπέλαση γίνεται ανά ψηφιολέξη. Υπολογίστε το μέσο χρόνο προσπέλασης του συστήματος μνήμης κατά την εκτέλεση του προγράμματος τα χαρακτηριστικά του οποίου δίνονται στον πίνακα.

Πίνακας

Είδος εντολής	Κύκλοι ρολογιού ανά εντολή	Ψηφιολέξεις που καταλαμβάνει μια εντολή στη μνήμη (εντολές)	Πλήθος ψηφιολέξεων που προσπελαύνει η εντολή κατά την εκτέλεσή της (δεδομένα)	Πλήθος εκτελούμενων εντολών
A	4	1	0	**300**
B	6	2	1	**500**
Γ	9	3	2	**100**

Λύση

Θα αγνοήσουμε, παρότι δίνονται, τους κύκλους ρολογιού που δαπανώνται στην ΚΜΕ.



Ζητάμε το ΜΧΠΜ, το οποίο δίνεται από τον τύπο: ΜΧΠΜ = $\frac{\text{Συνολικός χρόνος προσπέλασης συστήματος μνήμης}}{\text{Πλήθος προσπελάσεων μνήμης}}$ =

$\frac{[(0.94*1*2\ \text{κ.ρ.}+0.06*1*82\ \text{κ.ρ.}+0.91*0*2\ \text{κ.ρ.}+0.09*0*82\ \text{κ.ρ.})*300+(0.94*2*2\ \text{κ.ρ.}+0.06*2*82\ \text{κ.ρ.}+0.91*1*2\ \text{κ.ρ.}+0.09*1*82\ \text{κ.ρ.})*500+]}{(1+0)*300+(2+1)*500+(3+2)*100}$ =

$\frac{(0.94*3*2\ \text{κ.ρ.}+0.06*3*82\ \text{κ.ρ.}+0.91*2*2\ \text{κ.ρ.}+0.09*2*82\ \text{κ.ρ.})*100]*5\ \text{ns}}{(1+0)*300+(2+1)*500+(3+2)*100}$ nsec = ….. nsec

Θέμα 2012 με υπολογισμό χρόνου εκτέλεσης και δύο επίπεδα κρυφής μνήμης

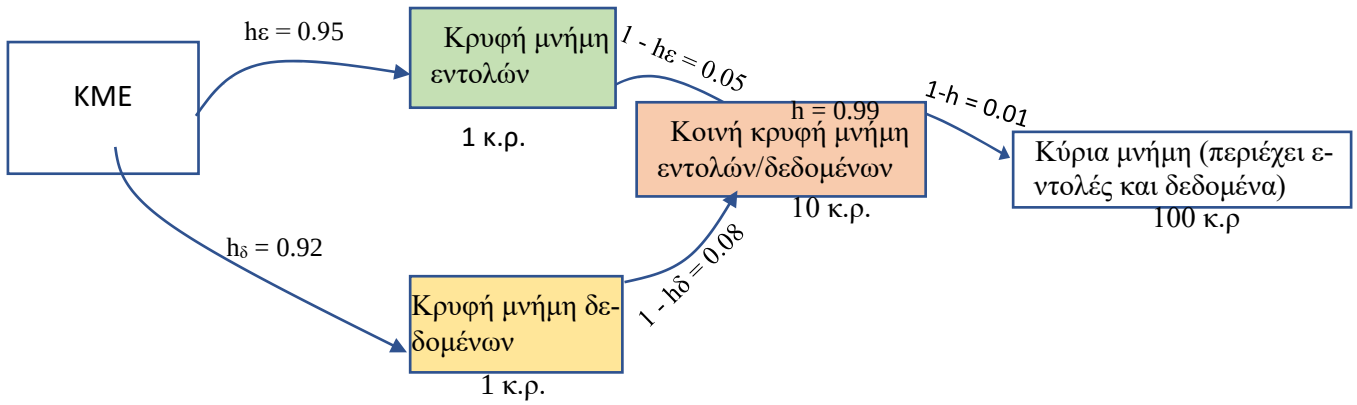
Σ' ένα υπολογιστή υπάρχει **ξεχωριστή κρυφή μνήμη εντολών και δεδομένων** με λόγο επιτυχίας αντίστοιχα $h_ε = 0.95$ και $h_δ = 0.92$. Η προσπέλαση οποιασδήποτε από τις δύο κρυφές μνήμες βάζει καθυστέρηση ενός κύκλου ρολογιού. Στη συνέχεια υπάρχει μια **ενοποιημένη** κρυφή μνήμη με λόγο επιτυχίας h = 0.99. Σε κάθε περίπτωση αποτυχίας να βρεθεί η εντολή ή τα δεδομένα στην κρυφή μνήμη υπάρχει καθυστέρηση 10 κύκλων ρολογιού για τη μεταφορά ενός μπλοκ εντολών ή δεδομένων από την ενοποιημένη κρυφή μνήμη στην αντίστοιχη κρυφή μνήμη. Σε κάθε περίπτωση αποτυχίας να βρεθεί η εντολή ή τα δεδομένα στην ενοποιημένη κρυφή μνήμη, υπάρχει καθυστέρηση 100 κύκλων ρολογιού για τη μεταφορά ενός μπλοκ εντολών ή δεδομένων από την κύρια μνήμη στην ενοποιημένη κρυφή μνήμη. Ο χρόνος κύκλου ρολογιού είναι 1 ns. Υποθέστε ότι το σύνολο των εντολών του προγράμματος σε επίπεδο γλώσσας μηχανής αποτελείται από τρία είδη εντολών, όπως φαίνεται στον Πίνακα που ακολουθεί. Υποθέστε ότι τόσο το εκτελούμενο πρόγραμμα όσο και τα απαιτούμενα δεδομένα βρίσκονται στην κύρια μνήμη. Η ΚΜΕ μπορεί να προσπελάσει εντολές και δεδομένα μόνο από τις αντίστοιχες κρυφές μνήμες. Θεωρήστε ότι η κύρια μνήμη έχει οργάνωση μίας ψηφιολέξης ανά θέση και η προσπέλαση γίνεται ανά ψηφιολέξη. Υπολογίστε το μέσο χρόνο προσπέλασης του συστήματος μνήμης κατά την εκτέλεση του προγράμματος τα χαρακτηριστικά του οποίου δίνονται στον πίνακα.

Πίνακας

Είδος εντολής	Κύκλοι ρολογιού ανά εντολή	Ψηφιολέξεις που καταλαμβάνει μια εντολή στη μνήμη (εντολές)	Πλήθος ψηφιολέξεων που προσπελαύνει η εντολή κατά την εκτέλεσή της (δεδομένα)	Πλήθος εκτελούμενων εντολών
A	4	1	0	**1000**
B	10	3	2	**2000**

Λύση

Στην περίπτωση αυτή υπάρχουν δύο ξεχωριστά επίπεδα κρυφής μνήμης. Το πρώτο περιέχει δύο ξεχωριστές κρυφές μνήμες και το δεύτερο περιέχει μια ενιαία κρυφή μνήμη.



$ΧΕΕ_Α$ = 4 κ.ρ. + **0.95 * 1 ψηφιολέξη εντολών * 1 κ.ρ.** + **0.05** * **[0.99 * 1 ψηφιολέξη εντολών * (1 + 10) κ.ρ. + (1 – 0.99) * 1 ψηφιολέξη εντολών * (1 + 10 + 100) κ.ρ.]** + **0.92 * 0 ψηφιολέξεις δεδομένων * 1 κ.ρ.** + **0.08** * **[0.99 * 0 ψηφιολέξεις δεδομένων * (1 + 10) κ.ρ. + (1 – 0.99) * 0 ψηφιολέξεις δεδομένων * (1 + 10 + 100) κ.ρ.]** = κ.ρ.

XEE_B = 10 κ.ρ. + **0.95 * 3 ψηφιολέξεις εντολών * 1 κ.ρ.** + **0.05 * [0.99 * 3 ψηφιολέξεις εντολών * (1 + 10) κ.ρ. + (1 – 0.99) * 3 ψηφιολέξεις εντολών * (1 + 10 + 100) κ.ρ.]** + **0.92 * 2 ψηφιολέξεις δεδομένων * 1 κ.ρ.** + **0.08 * [0.99 * 2 ψηφιολέξεις δεδομένων * (1 + 10) κ.ρ. + (1 – 0.99) * 2 ψηφιολέξεις δεδομένων * (1 + 10 + 100) κ.ρ.]** = κ.ρ.

$\mathbf{XEE_{εντ.\ προγρ.}}$= ($XEE_A$ * $Π_A$ + XEE_B * $Π_B$) * t_{cycle} = (XEE_A * 1000 + XEE_B * 2000) * 1 ns = …. ns

Παράδειγμα 1.2 - Χρήση μέτρου MIPS για την αξιολόγηση της απόδοσης Η/Υ

Θεωρούμε δύο υπολογιστές Y_1 και Y_2 με διαφορετική αρχιτεκτονική και την ίδια συχνότητα λειτουργίας 2 GHz. Το σύνολο εντολών του υπολογιστή Y_1 σε επίπεδο γλώσσας μηχανής μπορεί να ταξινομηθεί σε τρεις ομάδες Α, Β και Γ ανάλογα με το πλήθος των κύκλων ρολογιού που απαιτούνται για την εκτέλεση κάθε εντολής. Το σύνολο των εντολών του Y_2 μπορεί να ταξινομηθεί σε δύο ομάδες Α και Β.

Πίνακας 1.4

	Κύκλοι ρολογιού για την εκτέλεση μιας εντολής της ομάδας		
clock cycles	**Α**	**Β**	**Γ**
Y_1	1	2	3
Y_2	1	2	-

Πίνακας 1.5

	εκατομμύρια εντολών για κάθε ομάδα		
Πi	**Α**	**Β**	**Γ**
Y_1	5	1	1
Y_2	7	2	0

Λύση

Το ζητούμενο στη συγκεκριμένη άσκηση είναι να **εκτιμήσουμε την απόδοση δύο Η/Υ** του Y_1 και του Y_2 με βάση (κριτήριο) το **χρόνο εκτέλεσης** και με βάση το **μέτρο MIPS**.

a) Επειδή στα δύο υπολογιστικά συστήματα που δίνονται, **δεν** υπάρχει ούτε κρυφή μνήμη αλλά ούτε και κύρια μνήμη, για να υπολογίσουμε το χρόνο εκτέλεσης ($XE_{εντολών\ προγράμματος}$) θα χρησιμοποιήσουμε τον τύπο (2) της θεωρίας, που αναφέρει ότι $XE_{εντολών\ προγράμματος} = [\sum_{i=1}^{n}(KPE_i * Π_i)] * t_{cycle}$ όπου το $t_{cycle} = \frac{1}{συχνότητα\ λειτουργίας} = \frac{1}{2\ GHz} = \frac{1}{2\ x\ 10^9 Hz}$ = 0.5 x 10^{-9} sec = 0.5 nsec. Επομένως, $XE_{εντολών\ προγράμματος} = [\sum_{i=1}^{n}(KPE_i * Π_i)] * 0.5$ nsec.

Ο τύπος αυτός εφαρμόζεται για κάθε μια περίπτωση, δηλαδή για τον Η/Υ Y_1 έχουμε: $XE_{εντολών\ προγράμματος\ Y1} = [\sum_{i=1}^{n=3}(KPE_i * Π_i)] * 0.5$ nsec = $(KPE_1 * Π_1 + KPE_2 * Π_2 + KPE_3 * Π_3) * 0.5$ nsec = $(1 * 5 * 10^6 + 2 * 1 * 10^6 + 3 * 1 * 10^6) * 0.5$ nsec = $10 * 10^6 * 0.5$ nsec = $10^7 * 0.5 * 10^{-9}$ sec = $0.5 * 10^{-2}$ sec = $5 * 10^{-1} * 10^{-2}$ sec = $5 * 10^{-3}$ sec = **5 msec**.

Για τον Η/Υ Y_2 έχουμε: $XE_{εντολών\ προγράμματος\ Y2} = [\sum_{i=1}^{n=2}(KPE_i * Π_i)] * 0.5$ nsec = $(KPE_1 * Π_1 + KPE_2 * Π_2) * 0.5$ nsec = $(1 * 7 * 10^6 + 2 * 2 * 10^6) * 0.5$ nsec = $11 * 10^6 * 0.5$ nsec = $11 * 10^6 * 0.5 * 10^{-9}$ sec = $5.5 * 10^{-3}$ sec = **5.5 msec**. Από τη σύγκριση των δύο χρόνων εκτέλεσης, παρατηρούμε ότι $XE_{εντολών\ προγράμματος\ Y1} < XE_{εντολών\ προγράμματος\ Y2}$ κάτι που σημαίνει ότι ο Η/Υ Y_1 έχει μικρότερο χρόνο εκτέλεσης από τον Y_2, με αποτέλεσμα να έχει καλύτερη απόδοση.

b) Για να συγκρίνουμε την απόδοση των δύο Η/Υ με βάση το **μέτρο MIPS** πρέπει αυτοί να διαθέτουν το **ίδιο σύνολο εντολών** (instruction set), δηλαδή **την ίδια αρχιτεκτονική**. Αυτό προσδιορίζει το πλήθος των μικροεντολών στα οποία υποδιαιρείται κάθε κανονική εντολή. Στην προκειμένη περίπτωση η εκφώνηση αναφέρει ότι οι δύο υπολογιστές έχουν διαφορετική αρχιτεκτονική, κάτι που σημαίνει ότι έχουν **διαφορετικό σύνολο εντολών, κάτι που σημαίνει ότι δεν μπορεί να**

χρησιμοποιηθεί το μέτρο MIPS για τη σύγκριση της απόδοσής τους. Αν παρόλα αυτά το χρησιμοποιήσουμε, τότε ενδέχεται να καταλήξουμε σε λάθος αποτελέσματα. Το μέτρο MIPS = $\frac{\text{πλήθος εντολών}}{\text{ΧΕεντολών προγράμματος}}$. Για τον υπολογιστή Y_1 έχουμε $MIPS_{Y1}= \frac{7*10^6 \text{εντολές}}{5 \text{ msec}}$ = 1.4 * 10^6 *10^3 εντολές/sec = 1400 * 10^6 εντολές/sec. Για τον υπολογιστή Y_2 έχουμε $MIPS_{Y2}= \frac{9*10^6 \text{εντολές}}{5.5 \text{ msec}}$ = 1.63 * 10^6 *10^3 εντολές/sec = 1630 * 10^6 εντολές/sec. Επομένως, καταλήγουμε στο λανθασμένο συμπέρασμα ότι ο Y_2 έχει καλύτερη απόδοση από τον Y_1.

Άσκηση υπολογισμού χρόνου εκτέλεσης

Θεωρήστε δύο υπολογιστές W και Z που **δεν** διαθέτουν κρυφή μνήμη, η κύρια μνήμη έχει χρόνο προσπέλασης 20 κύκλων ρολογιού, ο χρόνος κύκλου ρολογιού είναι 5 ns και η κύρια μνήμη έχει οργάνωση μίας ψηφιολέξης ανά θέση. Υποθέστε ότι το σύνολο των εντολών του προγράμματος σε επίπεδο γλώσσας μηχανής αποτελείται από τρία είδη εντολών, όπως φαίνεται στον Πίνακα που ακολουθεί. Υποθέστε ότι τόσο το εκτελούμενο πρόγραμμα όσο και τα απαιτούμενα δεδομένα βρίσκονται στην κύρια μνήμη.

Θεωρήστε ότι **στον υπολογιστή W η προσπέλαση γίνεται ανά ψηφιολέξη**, ενώ **στον υπολογιστή Z ανά δύο ψηφιολέξεις**.

α. Υπολογίστε το χρόνο εκτέλεσης ενός προγράμματος του οποίου τα χαρακτηριστικά δίνονται στον Πίνακα που ακολουθεί.

β. Υπολογίστε το μέσο αριθμό κύκλων ανά εντολή.

γ. Υπολογίστε το μέσο χρόνο προσπέλασης του συστήματος μνήμης

Είδος εντολής	Κύκλοι ρολογιού ανά εντολή	Ψηφιολέξεις που καταλαμβάνει η εντολή στη μνήμη	Πλήθος ψηφιολέξεων μνήμης που προσπελαύνει η εντολή κατά την εκτέλεσή της	Πλήθος εκτελούμενων εντολών
A	4	1	0	300
B	7	2	1	20
Γ	10	3	2	50

Λύση

α. Στον υπολογιστή **W η προσπέλαση (διάρκειας 20 κ.ρ.) γίνεται ανά ψηφιολέξη**, άρα ανά θέση μνήμης και διαρκεί 20 κ.ρ. Αντίθετα, στον υπολογιστή Z, η προσπέλαση **(διάρκειας 20 κ.ρ.) γίνεται** ανά δύο ψηφιολέξεις. Αυτό σημαίνει ότι στο συγκεκριμένο υπολογιστή κάθε θέση μνήμης προσπελαύνεται σε χρόνο ίσο με 10 κ.ρ.

Υπολογιστής W

$ΧΕΕ_A$ = 4 κ.ρ. + (1 + 0) * 20 κ.ρ. = 24 κ.ρ.
$ΧΕΕ_B$ = 7 κ.ρ. + (2 + 1) * 20 κ.ρ. = 67 κ.ρ.
$ΧΕΕ_Γ$ = 10 κ.ρ. + (3 + 2) * 20 κ.ρ. = 110 κ.ρ.
$ΧΕ_{εντ.προγράμματος}$ = ($ΧΕΕ_A$ * $Π_A$ + $ΧΕΕ_B$ * $Π_B$ + $ΧΕΕ_Γ$ * $Π_Γ$) * t_{cycle} = (24 * 300 + 67 * 20 + 110 * 50) * 5 nsec = 70200 nsec

Υπολογιστής Z

$ΧΕΕ_A$ = 4 κ.ρ. + (1 + 0) * 10 κ.ρ. = 14 κ.ρ.
$ΧΕΕ_A$ = 7 κ.ρ. + (2 + 1) * 10 κ.ρ. = 37 κ.ρ.
$ΧΕΕ_A$ = 10 κ.ρ. + (3 + 2) * 10 κ.ρ. = 60 κ.ρ.
$ΧΕ_{εντ.προγράμματος}$ = ($ΧΕΕ_A$ * $Π_A$ + $ΧΕΕ_B$ * $Π_B$ + $ΧΕΕ_Γ$ * $Π_Γ$) * t_{cycle} = (14 * 300 + 37 * 20 + 60 * 50) * 5 nsec = 39700 nsec

β. Επειδή ζητείται ο **μέσος αριθμός κύκλων ανά εντολή**, θα λάβουμε υπόψη μας από τα δύο προηγούμενα μεγέθη, το άθροισμα μέσα στις παρενθέσεις, **πριν** το πολλαπλασιάσουμε με τη διάρκεια του κύκλου ρολογιού. Ο μέσος αριθμός κύκλων ανά εντολή = $\frac{\text{Συνολικός αριθμός κύκλων ρολογιού}}{\text{πλήθος εντολών}} = \frac{24 \text{ κ.ρ.}*300+67 \text{ κ.ρ.}*20+110 \text{ κ.ρ.}*50}{370} = \frac{14040 \text{ κ.ρ}}{370}$, ενώ στην περίπτωση του υπολογιστή Z είναι: $\frac{\text{Συνολικός αριθμός κύκλων ρολογιού}}{\text{πλήθος εντολών}} = \frac{14 \text{ κ.ρ.}*300+37 \text{ κ.ρ.}*20+60 \text{ κ.ρ.}*50}{370} = \frac{7940 \text{ κ.ρ}}{370}$.

γ. **W:** $\frac{\text{Συνολικός χρόνος προσπέλασης συστήματος μνήμης}}{\text{Πλήθος προσπελάσεων μνήμης}} = \frac{[(24\ \text{κ.ρ.} - 4\ \text{κ.ρ}) * 300 + (67\ \text{κ.ρ.} - 7\ \text{κ.ρ.}) * 20 + (110\ \text{κ.ρ.} - 10\ \text{κ.ρ.}) * 50] * 5\ \text{nsec}}{(1+0) * 300 + (2+1) * 20 + (3+2) * 50} =$

100 nsec. Ομοίως και για τον Z.

Άσκηση με μονάδα πολλαπλασιασμού και διαίρεσης

Ένας επεξεργαστής χρησιμοποιείται σε εφαρμογές στις οποίες το 30% του χρόνου δαπανάται για την εκτέλεση πράξεων πολλαπλασιασμού και το 10% για την εκτέλεση πράξεων διαίρεσης. Κατά τη σχεδίαση ενός νέου μοντέλου οι σχεδιαστές έχουν τη δυνατότητα λόγω κόστους να υλοποιήσουν είτε ένα νέο κύκλωμα πολλαπλασιασμού, το οποίο θα **διπλασιάσει** την ταχύτητα των πολλαπλασιασμών, είτε ένα νέο κύκλωμα διαίρεσης, το οποίο θα **τετραπλασιάσει** την ταχύτητα των διαιρέσεων. Ποια επιλογή θα οδηγήσει στη μέγιστη απόδοση

Λύση

Έστω ότι T_x είναι ο χρόνος εκτέλεσης των εντολών τύπου x. Τότε ο συνολικός χρόνος εκτέλεσης του προγράμματος θα είναι: $T = T_{\text{πολλαπλασιασμών}} + T_{\text{διαιρέσεων}} + T_{\text{άλλων εντολών}}$ με

$T_{\text{πολλαπλασιασμών}} = 0.3 \text{ x } T$

$T_{\text{διαιρέσεων}} = 0.1 \text{ x } T$ (1.3.1)

$T_{\text{άλλων εντολών}} = 0.6 \text{ x } T$

Εάν αντικατασταθεί η μονάδα του πολλαπλασιασμού, τότε ο χρόνος εκτέλεσης του προγράμματος θα είναι:

$T_\pi = (T_{\text{πολλαπλασιασμών}} /2) + T_{\text{διαιρέσεων}} + T_{\text{άλλων εντολών}}$ (1.3.2)

ενώ αν αντικατασταθεί η μονάδα της διαίρεσης τότε ο χρόνος εκτέλεσης του προγράμματος θα είναι:

$T_\delta = T_{\text{πολλαπλασιασμών}} + (T_{\text{διαιρέσεων}}/4) + T_{\text{άλλων εντολών}}$ (1.3.3)

Λαμβάνοντας υπόψη τις σχέσεις (1.3.1), από τις σχέσεις (1.3.2) και (1.3.3) παίρνουμε:

$T_\pi = (0.3 \text{ x } T/2) + 0.1 \text{ x } T + 0.6 \text{ x } T = 0.15 \text{ x } T + 0.1 \text{ x } T + 0.6 \text{ x } T = 0.85 \text{ x } T$

και

$T_\delta = 0.3 \text{ x } T + (0.1 \text{ x } T/4) + 0.6 \text{ x } T = 0.3 \text{ x } T + 0.025 \text{ x } T + 0.6 \text{ x } T = 0.925 \text{ x } T$

Από τις παραπάνω σχέσεις παρατηρούμε ότι η αντικατάσταση του πολλαπλασιαστή οδηγεί στη μεγαλύτερη μείωση του χρόνου εκτέλεσης ενός προγράμματος, επομένως στη μεγαλύτερη αύξηση της απόδοσης.

Κεφάλαιο 2 – Αρχιτεκτονικές Η/Υ – Μορφή εντολής – Είδη αριθμών

Επεξηγήσεις Κεφαλαίου 2

1. Υπάρχουν **δύο τρόποι αναπαράστασης αριθμών σε ένα Η/Υ**: η **παράσταση σταθερής υποδιαστολής** απεικονίζει τους αριθμούς σε μια περιορισμένη περιοχή (μειονέκτημα), αλλά το κόστος των κυκλωμάτων που χρησιμοποιούνται είναι σχετικά χαμηλό (πλεονέκτημα). Όταν θέλουμε να αναπαραστήσουμε αριθμούς σε ένα μεγαλύτερο εύρος, χρησιμοποιείται η **παράσταση κινητής υποδιαστολής** (πλεονέκτημα), όμως το κόστος των χρησιμοποιούμενων κυκλωμάτων είναι υψηλό ή ο χρόνος επεξεργασίας μπορεί να είναι υψηλός (μειονεκτήματα). Επιπλέον, στην **αναπαράσταση σταθερής υποδιαστολής οι αριθμοί είναι ομοιόμορφα κατανεμημένοι στην περιοχή αριθμών που μπορούν να παρασταθούν, ενώ στην αναπαράσταση κινητής υποδιαστολής οι αριθμοί δεν είναι ομοιόμορφα κατανεμημένοι.**

2. Ειδικά για τους αριθμούς **σταθερής υποδιαστολής** υπάρχουν τρεις επιμέρους επιλογές (τρόποι) με τις οποίες μπορούν να παρασταθούν. Πιο συγκεκριμένα, με δεδομένο ότι ένας αριθμός σε παράσταση **σταθερής υποδιαστολής** είναι της μορφής $\alpha_{v-1}\alpha_{v-2}.....\alpha_1\alpha_0$, οι **τρεις** τρόποι αναπαράστασής τους σχετίζονται με τη θέση της υποδιαστολής, η οποία μπορεί να είναι δεξιότερα του ψηφίου α_0 με αποτέλεσμα όλοι οι αριθμοί να είναι **ακέραιοι** ≥ 1, αριστερότερα του ψηφίου α_{v-1} με αποτέλεσμα όλοι οι αριθμοί να είναι **δεκαδικοί** $\geq \beta^{-v}$ ή σε μια ενδιάμεση θέση, οπότε ο αριθμός στην περίπτωση αυτή περιέχει και ακέραια και δεκαδικά ψηφία. Στην τελευταία περίπτωση η μετακίνηση της υποδιαστολής προς τα αριστερά/δεξιά μειώνει/αυξάνει το πλήθος των ακέραιων ψηφίων αντίστοιχα. Πιο συγκεκριμένα:

$\alpha_{v-1}\alpha_{v-2}.....\alpha_1\alpha_0$ → μετακίνηση της υποδιαστολής προς τα **αριστερά μειώνει την περιοχή αναπαράστασης των αριθμών (μέσω της μείωσης των ακέραιων ψηφίων)**, ενώ **αυξάνει την ακρίβεια (μέσω της αύξησης των κλασματικών ψηφίων) και το αντίστροφο.**

Παράδειγμα (θέμα Φεβρουαρίου 2021)

Έχετε αριθμούς σε αναπαράσταση **σταθερής υποδιαστολής των κ δυαδικών ψηφίων**. Εάν **αυξήσουμε** το πλήθος των ακέραιων δυαδικών ψηφίων διατηρώντας σταθερό το συνολικό πλήθος των δυαδικών ψηφίων κ τότε: Το πλήθος των αριθμών με τιμή μεταξύ 1 και 0 που μπορούν να αναπαρασταθούν παραμένει σταθερό.

α) Σωστό

β) Λάθος

Λύση

Λάθος, επειδή αυξάνεται το πλήθος των ακέραιων δυαδικών ψηφίων σημαίνει ότι η υποδιαστολή μετακινείται προς τα δεξιά και επομένως μεγαλώνει το εύρος της περιοχής των αριθμών που μπορούν να παρασταθούν. Από τη στιγμή που μεγαλώνει το εύρος, μεγαλώνει και το πλήθος των αριθμών με τιμή μεταξύ 1 και 0 που μπορούν να αναπαρασταθούν.

Παράδειγμα (θέμα Σεπτεμβρίου 2020)

Έχετε αριθμούς σε αναπαράσταση σταθερής υποδιαστολής των κ δυαδικών ψηφίων. Εάν **αυξήσουμε το πλήθος των ακέραιων δυαδικών ψηφίων** διατηρώντας σταθερό το συνολικό πλήθος των δυαδικών ψηφίων κ τότε: Το πλήθος των ακέραιων αριθμών που μπορούν να αναπαρασταθούν αυξάνεται.

α) Σωστό

β) Λάθος

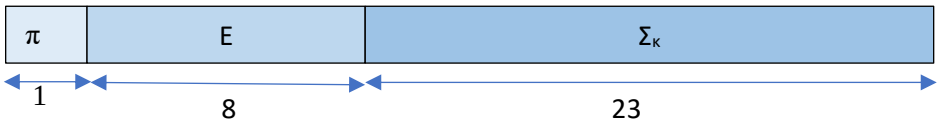
Λύση

Σωστό, αφού αυξάνεται το πλήθος των ακέραιων δυαδικών ψηφίων με αποτέλεσμα να αυξάνεται το εύρος της περιοχής αριθμών που μπορούν να παρασταθούν και συνεπώς και συνεπώς και το πλήθος των ακέραιων αριθμών που μπορούν να αναπαρασταθούν.

3. Αναφορικά με την **αναπαράσταση προσημασμένων αριθμών σταθερής υποδιαστολής**, όταν αυτοί είναι θετικοί παριστάνονται με έναν τρόπο. Όταν όμως είναι **αρνητικοί**, τότε μπορούν να παρασταθούν με τρεις διαφορετικούς τρόπους: **συμπλήρωμα ως προς βάση (συμπλήρωμα ως προς 2, δεδομένου ότι μιλάμε για αριθμούς δυαδικούς, β = 2)**, **συμπλήρωμα ως προς μειωμένη βάση (συμπλήρωμα ως προς 1, δεδομένου ότι μιλάμε για αριθμούς δυαδικούς, β = 2)** και **παράσταση προσημασμένου μεγέθους**. Στην περίπτωση **αναπαράστασης προσημασμένου μεγέθου**ς έχουμε πολύπλοκες πράξεις μεταξύ αριθμών, διότι οι πράξεις είναι ανεξάρτητες από τα μεγέθη και τα πρόσημα των αριθμών. Στην περίπτωση αναπαράστασης **ως προς μειωμένη βάση**, όταν προκύπτει **κρατούμενο** από μια πράξη, το προσθέτουμε στο δεξιότερο ψηφίο (λιγότερο σημαντικό) του αποτελέσματος, ενώ στην περίπτωση αναπαράστασης **ως προς βάση**, όταν προκύπτει **κρατούμενο** από μια πράξη, αγνοείται. Από τους τρεις τρόπους αναπαράστασης προσημασμένων αριθμών σταθερής υποδιαστολής, αυτός που έχει καθιερωθεί είναι ο τρόπος **συμπληρώματος ως προς βάση (2)**, για τους εξής λόγους:

- Οι αθροιστές απαιτούν λιγότερο υλικό και έχουν μικρότερη καθυστέρηση (σε σχέση με το συμπλήρωμα ως προς 1)
- Το «0» αναπαρίσταται μόνο με ένα τρόπο, ενώ στις άλλες δύο αναπαραστάσεις με δύο τρόπους.

4. Ένας αριθμός σε παράσταση **κινητής υποδιαστολής** έχει τη μορφή: $Z = (-1)^{\pi} \times \Sigma_{\kappa} \times B^{E}$, όπου π = πρόσημο (0 ή 1), το Σ_{κ} = συντελεστής (κλασματικό μέρος αριθμού), το B = βάση του αριθμητικού και το E = εκθέτης. Για την αναπαράσταση αριθμών κινητής υποδιαστολής χρησιμοποιείται το πρότυπο IEEE – 754 και σύμφωνα με αυτό, ένας αριθμός κινητής υποδιαστολής αποτελείται από τρία μέρη:



Μετατροπή από κινητή υποδιαστολή (IEEE – 754) σε σταθερή υποδιαστολή (N)

$N = NaN$, αν $E = 255$ και $\Sigma_{\kappa} \neq 0$

$N = (-1)^{\pi} \times \infty$, αν $E = 255$ και $\Sigma_{\kappa} = 0$

$N = (-1)^{\pi} \times 2^{E-127} \times 1.\Sigma_{\kappa}$, αν $0 < E < 255$ – κανονικοποιημένοι αριθμοί

$N = (-1)^{\pi} \times 2^{126} \times 0.\Sigma_{\kappa}$, αν $E = 0$ και $\Sigma_{\kappa} \neq 0$

$N = (-1)^{\pi} \times 0$, αν $E = 0$ και $\Sigma_{\kappa} = 0$

Πρόσημο (π) 1 - bit	Εκθέτης (E) 8 - bit	συντελεστής (Σ_{κ}) 23 - bit

5. Όταν μετατρέπουμε έναν αριθμό από παράσταση σταθερής υποδιαστολής σε παράσταση κινητής υποδιαστολής, πρέπει πρώτα να **κανονικοποιήσουμε** τον αριθμό αυτό (να τον φέρουμε στη μορφή **1.xxxx…x**, με μετακίνηση της υποδιαστολής προς τα αριστερά/δεξιά, αναλόγως την περίπτωση), οπότε τότε ο εκθέτης αυξάνεται/μειώνεται αντίστοιχα. Στη συνέχεια, θα πρέπει να γράψουμε τον αριθμό αυτό με τρία πεδία, που είναι το πρόσημο, ο εκθέτης και ο συντελεστής. Στο πρόσημο θέτουμε «0»/»1», ανάλογα με τον αν ο αριθμός είναι θετικός/αρνητικός αντίστοιχα και στον εκθέτη πρέπει τώρα να προσθέσουμε την πόλωση (127 για απλή ακρίβεια ή 1023 για διπλή ακρίβεια).

6. Τα πλεονεκτήματα των αριθμών κινητής υποδιαστολής είναι: **μεγαλύτερο εύρος περιοχής** που μπορεί να παρασταθεί (μεγαλύτερη δυναμική περιοχή) και **μικρότερο μέσο σχετικό σφάλμα αναπαράστασης** σε σχέση με την παράσταση σταθερής υποδιαστολής.

7. Κάθε εντολή του υπολογιστή περιγράφει μια συγκεκριμένη λειτουργία και τα τελούμενα ή δεδομένα που θα χρησιμοποιηθούν. Αποτελείται από τον **κωδικό λειτουργίας (opcode)** και τα **έντελα ή τελούμενα (operands)**. Τα τελούμενα μπορεί να είναι αριθμοί, διευθύνσεις θέσεων κύριας μνήμης ή καταχωρητές. Είναι πιθανό επίσης μια εντολή να μην έχει **κανένα** τελούμενο. Ορισμένα παραδείγματα εντολών με διαφορετικό αριθμό τελούμενων, φαίνεται στη συνέχεια:

```
ADD          // παράδειγμα εντολής χωρίς τελούμενα (για την περίπτωση στοίβας)
LOAD X       // παράδειγμα εντολής με ένα τελούμενο (για την περίπτωση συσσωρευτή)
```

ADD R1, #10 //R1 ← R1 + 10 (παράδειγμα με δύο τελούμενα, καταχωρητή και αριθμό)

ADD R1, $200 //R1 ← R1 + $200 (παράδειγμα με δύο τελούμενα, καταχωρητή και διεύθυνση μνήμης)

ADD R1, R2 //R1 ← R1 + R2 (παράδειγμα με δύο τελούμενα, που είναι καταχωρητές)

ADD R1, R2, R3 //R1 ← R2 + R3 (παράδειγμα με τρία τελούμενα, που είναι καταχωρητές)

ADD R1, R2, $200 //R1 ← R2 + $200 (παράδειγμα με τρία τελούμενα, τα δύο είναι καταχωρητές και το άλλο διεύθυνση μνήμης)

8. Υπάρχουν **δύο είδη ασκήσεων** στο κεφάλαιο 2, αναφορικά με τη **μορφή εντολής σε γλώσσα μηχανής**: Αυτές που έχουν **μεταβλητό μήκος εντολής** (διατυπώνεται με την πρόταση: **κάθε εντολή καταλαμβάνει ακέραιο αριθμό ψηφιολέξεων**) και αυτές που έχουν **σταθερό μήκος εντολής** (διατυπώνεται με την πρόταση: **εντολές με σταθερό μήκος**). Όταν ζητείται να βρεθούν η μορφή των εντολών ενός προγράμματος, τα bytes που καταλαμβάνουν (χωρητικότητα) καθώς και οι προσπελάσεις μνήμης, διακρίνουμε δύο περιπτώσεις:

i) Όταν το μήκος μιας εντολής είναι **ακέραιο πολλαπλάσιο της ψηφιολέξης**, δηλαδή μεταβλητό, τότε θα πρέπει να **ελέγξουμε** αν το συνολικό μήκος της εντολής (συμφέρει να είναι εκφρασμένο σε bits) είναι πολλαπλάσιο του «8» και αν **δεν** είναι, τότε να επεκτείνουμε την εντολή στο τέλος της με τα δυαδικά ψηφία που απαιτούνται, προκειμένου να γίνει πολλαπλάσια του «8». Συνίσταται το μέγεθος (χωρητικότητα) κάθε εντολής να υπολογίζεται σε δυαδικά ψηφία (bits) και στο **τέλος**, αν απαιτείται, να γίνεται στρογγυλοποίηση του αποτελέσματος σε πολλαπλάσιο του «8». Στο μεταβλητό μήκος εντολής, για να υπολογίσουμε τις προσπελάσεις μνήμης, θα πρέπει να εφαρμόσουμε τον εξής κανόνα: Όσες **εντολές έχουν έντελο (τελούμενο) μνήμη**, έχουν περισσότερες προσπελάσεις από bytes και μάλιστα **τόσες περισσότερες,** όσο είναι το **μέγεθος των δεδομένων**, λαμβάνοντας όμως υπόψη και το **εύρος της αρτηρίας δεδομένων**. Για παράδειγμα:

Δεδομένα: 8 bits Αρτηρία δεδομένων (data bus): 8 bits → 8/8 = 1 **επιπλέον** προσπέλαση

Δεδομένα: 16 bits Αρτηρία δεδομένων (data bus): 8 bits → 16/8 = 2 **επιπλέον** προσπελάσεις

Δεδομένα: 32 bits Αρτηρία δεδομένων (data bus): 8 bits → 32/8 = 4 επιπλέον προσπελάσεις

Δεδομένα: 16 bits Αρτηρία δεδομένων (data bus): 16 bits → 16/16 = 1 επιπλέον προσπέλαση

Δεδομένα: 32 bits Αρτηρία δεδομένων (data bus): 16 bits → 32/16 = 2 επιπλέον προσπελάσεις

Όσες **εντολές δεν έχουν ως έντελο (τελούμενο) μνήμη**, **τότε αυτές έχουν τόσες προσπελάσεις όσα και τα bytes.**

ii) Όταν το μήκος μιας εντολής είναι **σταθερό**, θα πρέπει αφού υπολογίσουμε το μήκος του κωδικού λειτουργίας της (opcode) και των εντελών της (π.χ. καταχωρητές), στη συνέχεια για να υπολογίσουμε το μήκος εκείνων των εντέλων που **δεν** δίνονται από την εκφώνηση, θα πρέπει να αφαιρέσουμε όλα τα γνωστά πεδία της εντολής από το συνολικό μήκος της.

9. Όταν πρέπει **να υπολογίσουμε προσπελάσεις μνήμης**, πρέπει να έχουμε υπόψη:

i) Στον υπολογισμό αυτό, **όσες εντολές εμφανίζουν το σύμβολο της #**, αναφέρονται σε δεδομένα και δεν έχουν ποτέ επιπλέον προσπελάσεις μνήμης από ότι χωρητικότητα, αφού τότε δεν προσπελαύνεται μνήμη. Δηλαδή, σε αυτή την περίπτωση οι ζητούμενες προσπελάσεις μνήμης = πλήθος των bytes (ψηφιολέξεων) των εντολών. Με άλλα λόγια, το σύμβολο # μέσα σε μια εντολή υποδηλώνει **δεδομένα** και το μέγεθός του προκύπτει από την αρτηρία δεδομένων. Αυτό ισχύει **και** για μεταβλητό αλλά **και** για σταθερό μήκος εντολής.

ii) Στον υπολογισμό αυτό, **όσες εντολές εμφανίζουν το σύμβολο του $**, υποδηλώνει διεύθυνση μνήμης και το μέγεθός της προκύπτει από την αρτηρία διευθύνσεων. Σε αυτή την περίπτωση υπάρχουν πάντα **επιπλέον** προσπελάσεις μνήμης από ότι χωρητικότητα και έχουμε ότι: προσπελάσεις μνήμης = πλήθος των bytes (ψηφιολέξεων) των εντολών + τόσες **επιπλέον** προσπελάσεις μνήμης όσο είναι το: $\frac{\text{μέγεθος δεδομένων}}{\text{μέγεθος αρτηρίας δεδομένων}}$ Για παράδειγμα, δεδομένα μεγέθους 32 bits θέλουν 4 **επιπλέον** προσπελάσεις μνήμης, όταν το data bus = 8 γραμμών (bits), ενώ θέλουν 2 **επιπλέον** προσπελάσεις μνήμης όταν το

data bus = 16 γραμμών και μόνο μια **επιπλέον** προσπέλαση μνήμης όταν το data bus = 32 γραμμών κ.ο.κ. Αν δεν αναφέρεται σχετικά κάτι από την εκφώνηση για την αρτηρία δεδομένων, αυτή λαμβάνεται ως 8 bits.

iii) Για **προσπέλαση μνήμης**, εκτός της ένδειξης του $, μπορεί να υπάρχει καταχωρητής μέσα σε ( ), όπως π.χ. (Rx) που υποδηλώνει **έμμεσο τρόπο διευθυνσιοδότησης με χρήση καταχωρητή**. Σε μια τέτοια περίπτωση, στον υπολογισμό του μεγέθους (χωρητικότητας) της εντολής, ο καταχωρητής λαμβάνεται υπόψη ως κανονικός καταχωρητής, όμως στον υπολογισμό των προσπελάσεων μνήμης, λαμβάνεται υπόψη ως μνήμη, **με αποτέλεσμα να έχουμε περισσότερες προσπελάσεις από ότι bytes, διότι τότε προσπελαύνεται μνήμη.**

10. Όταν δίνεται πλήθος καταχωρητών με συγκεκριμένη χωρητικότητα (π.χ. 32 καταχωρητές των 16 δυαδικών ψηφίων), πρέπει να λαμβάνονται υπόψη, **μόνο** σε περιπτώσεις που **ζητείται το πλήθος των θ.μ. που μπορεί να διευθυνσιοδοτηθεί από μια εντολή.** Διευκρινίζεται ότι είναι **διαφορετικές έννοιες το πλήθος των προσπελάσεων μνήμης κατά την εκτέλεση μιας εντολής** και το **πλήθος των θέσεων μνήμης (θ.μ.) που μπορεί να διευθυνσιοδοτηθεί από μια εντολή.**

11. Από το **πλήθος των εντολών** υπολογίζουμε το μήκος του κωδικού λειτουργίας κάθε εντολής, χρησιμοποιώντας τον τύπο $\mathbf{2^x >=}$ **πλήθος εντολών**. Αυτό χρησιμοποιείται σε περίπτωση που δεν δίνεται το μέγεθος του κωδικού λειτουργίας (opcode) κάθε εντολής. Από το **πλήθος των καταχωρητών** βρίσκουμε το πλήθος των bits που χρειαζόμαστε για την κωδικοποίηση κάθε καταχωρητή, δηλ. $\mathbf{2^x >=}$ **πλήθος καταχωρητών** π.χ. για 32 καταχωρητές: $2^x >= 32 \Rightarrow x = 5$ bits.

12. Στο **σταθερό μήκος εντολής**, για τον υπολογισμό του πλήθους των θέσεων μνήμης που μπορεί να διευθυνσιοδοτηθεί από μια εντολή, όταν υπάρχει καταχωρητής μέσα σε ( ), για να υπολογίσουμε το πλήθος των θέσεων μνήμης που προσπελαύνονται, θα πρέπει να λάβουμε υπόψη μας το μήκος του κάθε καταχωρητή, δηλαδή, πλήθος των θέσεων μνήμης που μπορεί να διευθυνσιοδοτηθεί από την εντολή = $2^{\text{μήκος καταχωρητή}}$, ενώ αν ένα έντελο είναι **διεύθυνση μνήμης**, τότε πλήθος των θέσεων μνήμης που μπορεί να διευθυνσιοδοτηθεί από την εντολή = $2^{\text{μήκος εντέλου}}$.

13. Στην **ευθυγράμμιση μνήμης**, η αποθήκευση των δεδομένων στη μνήμη γίνεται σε συγκεκριμένες διευθύνσεις και όσο μεγαλύτερο είναι το μέγεθος τους, τόσο περισσότεροι περιορισμοί ισχύουν αναφορικά με την αποθήκευση. Για παράδειγμα, για τετραπλή λέξη (16 bytes) αποθηκεύεται σε θ.μ. με τα τέσσερα τελευταία ψηφία της διεύθυνσης να είναι 0000. Πιο συγκεκριμένα, ισχύει το εξής τυπολόγιο:

Τελούμενο	**Τιμή 4 λιγότερο σημαντικών ψηφίων διεύθυνσης**
Byte (ψηφιολέξη)	xxxx (οποιαδήποτε διεύθυνση, δεν υπάρχει περιορισμός)
Halfword (μισή λέξη)	xxx0 (τέσσερα τελευταία δυαδικά ψηφία διεύθυνσης)
Word (λέξη)	xx00 (τέσσερα τελευταία δυαδικά ψηφία διεύθυνσης)
Double word (Διπλή λέξη)	x000 (τέσσερα τελευταία δυαδικά ψηφία διεύθυνσης)
Quad word (Τετραπλή λέξη)	0000 (τέσσερα τελευταία δυαδικά ψηφία διεύθυνσης)

- **Σε ασκήσεις με ευθυγράμμιση μνήμης**, έχει σημασία αν τα δεδομένα/εντολές που αποθηκεύονται στη μνήμη είναι ευθυγραμμισμένα ή όχι. Αν **δεν** είναι, τότε οι προσπελάσεις θα είναι κατά μια περισσότερες από την περίπτωση της ευθυγράμμισης.
- Επιπλέον, πρέπει το πλήθος των θέσεων μνήμης που καταλαμβάνει μια δομή να είναι πολλαπλάσιο του «4». Αν δεν είναι, προστίθενται επιπλέον θέσεις μνήμης στο τέλος της δομής.

14. Υπάρχουν τέσσερις αρχιτεκτονικές για γλώσσα μηχανής: αρχιτεκτονική στοίβας (σωρού), συσσωρευτή, καταχωρητή - καταχωρητή με δύο ή τρία τελούμενα και καταχωρητή – μνήμης με δύο ή τρία τελούμενα. Αν θέλουμε να υλοποιήσουμε μια έκφραση σε **αρχιτεκτονική στοίβας (σωρού), πρέπει να τη μετατρέψουμε από μορφή infix σε μορφή postfix**

(πρώτα τελούμενα και μετά τελεστές). Αν θέλουμε να υλοποιήσουμε μια έκφραση σε **αρχιτεκτονική συσσωρευτή**, θα πρέπει αν στην έκφραση αυτή υπάρχει μια ποσότητα που **αφαιρείται** ή **διαιρείται**, να ξεκινήσουμε με τον υπολογισμό αυτής της ποσότητας.

Άσκηση 2.12 με μηχανισμό στοίβας – συσσωρευτή – καταχωρητή – καταχωρητή και μνήμης

α) Για τον υπολογισμό της έκφρασης Y = A + B x C + B x E να γραφεί πρόγραμμα για Η/Υ που να βασίζεται στη χρήση:

i. του μηχανισμού στοίβας

ii. του συσσωρευτή

iii. 16 καταχωρητών γενικού σκοπού με εντολές μόνο καταχωρητή-καταχωρητή (ή φόρτωσης-αποθήκευσης) με δύο τελούμενα

iv. με εντολές καταχωρητή-καταχωρητή και καταχωρητή-μνήμης με δύο τελούμενα.

β) Να υπολογιστεί ο χώρος που καταλαμβάνει κάθε πρόγραμμα, εάν ο κωδικός λειτουργίας κάθε εντολής καταλαμβάνει μία ψηφιολέξη (byte), ο τρόπος διευθυνσιοδότησης του συστήματος μνήμης που χρησιμοποιείται στις εντολές είναι ο κατ' ευθείαν τρόπος (direct addressing mode), οι διευθύνσεις είναι των 32 δυαδικών ψηφίων και οι μηχανές που βασίζονται στη χρήση καταχωρητών γενικού σκοπού, έχουν 16 καταχωρητές γενικού σκοπού. Κάθε εντολή **καταλαμβάνει ακέραιο αριθμό ψηφιολέξεων**.

γ) Θεωρήστε ότι τα δεδομένα είναι μιας ψηφιολέξης, η αρτηρία δεδομένων μεταξύ της Κεντρικής Μονάδας Επεξεργασίας (ΚΜΕ) και του συστήματος μνήμης έχει εύρος μιας ψηφιολέξης (byte) και σε κάθε διεύθυνση του συστήματος μνήμης αντιστοιχεί μια ψηφιολέξη. Στην περίπτωση μηχανής που βασίζεται, στο μηχανισμό στοίβας, η στοίβα υλοποιείται στην ΚΜΕ. Πόσες **προσπελάσεις στη μνήμη** απαιτεί το κάθε πρόγραμμα.

Λύση

α) Μηχανισμός στοίβας (σωρού)

Όταν υλοποιούμε μια έκφραση σε μηχανισμό στοίβας, πρέπει να μετατρέψουμε αυτήν την έκφραση από μορφή **infix** σε μορφή **postfix (πρώτα τα τελούμενα και μετά οι τελεστές)**. Στη μετατροπή αυτή ισχύουν οι εξής κανόνες:

A + B → AB+ A – B → BA– A x B → ABx A/B → BA/

Y = A + B x C + B x E = A + BCx + BEx = ABCx+ + BEx = ABCx+BEx+

Σημείωση: είναι απαραίτητη η χρήση σχολίων σε όλους τους κώδικες που ακολουθούν

8 bits 32 bits

```
PUSH A      //A (βάζουμε στη στοίβα το περιεχόμενο της θέσης μνήμης με διεύθυνση Α)
PUSH B      //B, A (ότι είναι πιο αριστερά, σημαίνει ότι βρίσκεται στην κορυφή της στοίβας)
PUSH C      //C, B, A (ότι είναι πιο αριστερά, σημαίνει ότι βρίσκεται στην κορυφή της στοίβας)
MUL         //CxB, A (κάθε φορά η οποιαδήποτε πράξη γίνεται μεταξύ του κορυφαίου στοιχείου της στοίβας και του
            // στοιχείου που είναι από κάτω).
ADD         // CxB + A
PUSH B      //B, CxB + A
PUSH E      //E, B, CxB + A
MUL         //ExB, CxB + A
ADD         //ExB + CxB + A
POP Y       //Y ← ExB + CxB + A
```

Κανόνας: όταν υπολογίζουμε το μέγεθος κάθε εντολής, συμφέρει να το υπολογίζουμε **αρχικά σε bits** (δυαδικά ψηφία) και στη συνέχεια να μετατρέπουμε το μέγεθος αυτό σε bytes (αν χρειάζεται πρέπει να κάνουμε στρογγυλοποίηση του μεγέθους αυτού προς τα **πάνω**, έτσι ώστε το μέγεθος αυτό να είναι ακέραιο πολλαπλάσιο του «8», δηλ. της ψηφιολέξης).

Για να υπολογίσουμε το **μέγεθος του παραπάνω προγράμματος** θα πρέπει να ομαδοποιήσουμε **πανομοιότυπες** εντολές, οπότε έχουμε: Μέγεθος = 6 εντολές x 5 bytes + 4 εντολές x 1 byte = 34 bytes.

Για να υπολογίσουμε τις προσπελάσεις μνήμης **του παραπάνω προγράμματος** θα πρέπει να εφαρμόσουμε τον εξής κανόνα: Όσες **εντολές έχουν ως έντελο (τελούμενο) μνήμη**, έχουν περισσότερες προσπελάσεις από bytes και μάλιστα **τόσες περισσότερες,** όσο είναι το **μέγεθος των δεδομένων**, λαμβάνοντας όμως υπόψη και το **εύρος της αρτηρίας δεδομένων**.

Δεδομένα: 8 bits Αρτηρία δεδομένων (data bus): 8 bits → 8/8 = 1 **επιπλέον** προσπέλαση

Δεδομένα: 16 bits Αρτηρία δεδομένων (data bus): 8 bits → 16/8 = 2 **επιπλέον** προσπελάσεις

Δεδομένα: 32 bits Αρτηρία δεδομένων (data bus): 8 bits → 32/8 = 4 επιπλέον προσπελάσεις

Δεδομένα: 16 bits Αρτηρία δεδομένων (data bus): 16 bits → 16/16 = 1 επιπλέον προσπέλαση

Δεδομένα: 32 bits Αρτηρία δεδομένων (data bus): 16 bits → 32/16 = 2 επιπλέον προσπελάσεις

Όσες **εντολές δεν έχουν ως έντελο (τελούμενο) μνήμη**, **έχουν τόσες προσπελάσεις όσα και τα bytes.** Επομένως για προσπελάσεις μνήμης έχουμε: Προσπελάσεις = 6 εντολές x 6 προσπελάσεις + 4 εντολές x 1 προσπέλαση = 40 προσπελάσεις.

β) Μηχανισμός συσσωρευτή: Y = A + B x C + B x E

Ο συσσωρευτής είναι ο βασικός καταχωρητής ενός Η/Υ και όλες οι λειτουργίες γίνονται μέσω αυτού. Ζητάμε την υλοποίηση αυτής της έκφρασης μέσω συσσωρευτή. Λαμβάνουμε υπόψη μας την προτεραιότητα πράξεων και αν **στην έκφραση που υλοποιούμε με το συσσωρευτή υπάρχει κάποια ποσότητα που αφαιρείται/διαιρείται**, πρέπει να ξεκινήσουμε με την ποσότητα αυτήν, να την υπολογίσουμε και να την αποθηκεύσουμε.

8 bits 32 bits

```
LOAD B      //Σ ← B
MUL C       //Σ ← B x C
ADD A       //Σ ← B x C + A
STORE Y     //Y ← B x C + A
LOAD B      //Σ ← B
MUL E       //Σ ← B x E
ADD Y       //Σ ← B x E + B x C + A
STORE Y     //Y ← B x E + B x C + A
```

Για να υπολογίσουμε το μέγεθος του παραπάνω προγράμματος θα πρέπει να ομαδοποιήσουμε πανομοιότυπες εντολές, οπότε έχουμε: **Μέγεθος** = 8 εντολές x 5 bytes = 40 bytes.

Για να υπολογίσουμε τις προσπελάσεις μνήμης του παραπάνω προγράμματος θα πρέπει να εφαρμόσουμε τον προηγούμενο κανόνα. Επειδή και οι 8 εντολές του παραπάνω προγράμματος έχουν έντελο μνήμη, οι προσπελάσεις είναι: **Προσπελάσεις** = 8 εντολές x 6 προσπελάσεις = 48 προσπελάσεις.

γ) Μηχανισμός καταχωρητή – καταχωρητή με δύο τελούμενα: Y = A + B x C + B x E

Στην περίπτωση αυτή θα πρέπει να λάβουμε υπόψη τον αριθμό των καταχωρητών που έχουμε στη διάθεσή μας και οι οποίοι είναι 16 και από το πλήθος τους **συμπεραίνουμε τον αριθμό των bits που χρειαζόμαστε για να τους προσπελάσουμε**. Ισχύει ότι $2^x \geq 16 \Rightarrow x = 4$ bits.

8 bits 4 bits 32 bits

```
LOAD R1, B    //R1 ← B      44 bits → 48 bits → 6 bytes
LOAD R2, C    // R2 ←C
```

8 bits 4 bits 4 bits

```
MUL R2, R1    // R2 ← B x C      16 bits → 2 bytes
LOAD R3, E    // R3 ←E
MUL R1, R3    // R1 ← B x E
ADD R1, R2    // R1 ← B x E + B x C
LOAD R2, A    // R2 ←A
ADD R1, R2    // R1 ← B x E + B x C + A
STORE Y, R1   //Y ← B x E + B x C + A
```

Ο μηχανισμός καταχωρητή – καταχωρητή με δύο τελούμενα αναφέρεται στις πράξεις, οι οποίες γίνονται αποκλειστικά μεταξύ καταχωρητών και το πλήθος τους είναι δύο. Σε αυτήν την περίπτωση το εκάστοτε αποτέλεσμα καταχωρείται στον **πρώτο** από τους δύο καταχωρητές.

Για να υπολογίσουμε το μέγεθος του παραπάνω προγράμματος θα πρέπει να ομαδοποιήσουμε πανομοιότυπες εντολές, οπότε έχουμε: **Μέγεθος** = 5 εντολές x 6 bytes + 4 εντολές x 2 bytes = 38 bytes.

Για να υπολογίσουμε τις προσπελάσεις μνήμης του παραπάνω προγράμματος θα πρέπει να εφαρμόσουμε τον προηγούμενο κανόνα: **Προσπελάσεις** = 5 εντολές x 7 προσπελάσεις + 4 εντολές x 2 προσπελάσεις = 43 προσπελάσεις.

δ) Μηχανισμός καταχωρητή – καταχωρητή και καταχωρητή – μνήμης με δύο τελούμενα: Y = A + B x C + B x E

Στην περίπτωση θα πρέπει να λάβουμε και πάλι υπόψη τον αριθμό των καταχωρητών που έχουμε στη διάθεσή μας και οι οποίοι είναι 16 και από το πλήθος τους **συμπεραίνουμε τον αριθμό των bits που χρειαζόμαστε για να τους προσπελάσουμε**. Ισχύει ότι $2^x \geq 16 \Rightarrow x = 4$ bits. Επιπλέον, τώρα μπορούμε να κάνουμε πράξεις τόσο μεταξύ καταχωρητών όσο και μεταξύ καταχωρητών και μνήμης.

8 bits 4 bits 32 bits

```
LOAD R1, B    //R1 ← B    44 bits → 48 bits → 6 bytes
MUL R1, C     //R1 ← B x C
ADD R1, A     //R1 ← B x C + A
LOAD R2, B    // R2 ←B
MUL R2, E     // R2 ← B x E
```

8 bits 4 bits 4 bits

```
ADD R1, R2    //R1 ← B x C + A + B x E    16 bits → 2 bytes
STORE Y, R1
```

Ο μηχανισμός καταχωρητή – καταχωρητή και καταχωρητή – μνήμης με δύο τελούμενα αναφέρεται στις πράξεις, οι οποίες τώρα γίνονται **τόσο** μεταξύ καταχωρητών **όσο** και μεταξύ καταχωρητών και θέσεων μνήμης. Σε αυτήν την περίπτωση το εκάστοτε αποτέλεσμα καταχωρείται στον **πρώτο** από τους δύο καταχωρητές.

Για να υπολογίσουμε το μέγεθος του παραπάνω προγράμματος θα πρέπει να ομαδοποιήσουμε πανομοιότυπες εντολές, οπότε έχουμε: **Μέγεθος** = 6 εντολές x 6 bytes + 1 εντολή x 2 bytes = 38 bytes.

Για να υπολογίσουμε τις προσπελάσεις μνήμης του παραπάνω προγράμματος θα πρέπει να εφαρμόσουμε τον προηγούμενο κανόνα: **Προσπελάσεις** = 6 εντολές x 7 προσπελάσεις + 1 εντολή x 2 προσπελάσεις = 44 προσπελάσεις.

Παρατήρηση 1: Η ένδειξη «καταχωρητή – καταχωρητή με δύο τελούμενα» αναφέρεται στο γεγονός ότι όταν κάνουμε πράξεις μεταξύ καταχωρητών, αυτοί θα είναι πάντα δύο. **Δεν** αναφέρεται στις εντολές LOAD, STORE, οι οποίες θα έχουν πάντα δύο τελούμενα.

Παρατήρηση 2: Η ένδειξη «καταχωρητή – καταχωρητή με τρία τελούμενα» αναφέρεται στο γεγονός ότι όταν κάνουμε πράξεις μεταξύ καταχωρητών, αυτοί θα είναι πάντα τρεις. Δεν αναφέρεται στις εντολές LOAD, STORE, οι οποίες θα έχουν πάντα δύο τελούμενα.

Παρατήρηση 3: Η ένδειξη «καταχωρητή – καταχωρητή και καταχωρητή – μνήμης με δύο τελούμενα» αναφέρεται στο γεγονός ότι όταν κάνουμε πράξεις μεταξύ καταχωρητών, αλλά και μεταξύ καταχωρητών και θέσεων μνήμης, αυτοί θα είναι πάντα δύο. Δεν αναφέρεται στις εντολές LOAD, STORE, **οι οποίες θα έχουν πάντα δύο τελούμενα**.

Άσκηση 2.13 με μηχανισμό στοίβας – συσσωρευτή – καταχωρητή – καταχωρητή και μνήμης

Δίνεται η έκφραση **E = A x (B – C x D)/(C+G)**. Να υλοποιηθεί με πρόγραμμα για Η/Υ που να βασίζεται στη χρήση μηχανισμού στοίβας, συσσωρευτή, καταχωρητή – καταχωρητή με δύο τελούμενα και τέλος καταχωρητή – καταχωρητή και καταχωρητή - μνήμης με δύο τελούμενα.

Λύση

α) Μηχανισμός στοίβας (σωρού)

Όταν υλοποιούμε μια έκφραση σε μηχανισμό στοίβας, πρέπει να μετατρέψουμε αυτήν την έκφραση από μορφή **infix** σε μορφή **postfix (πρώτα τα τελούμενα και μετά οι τελεστές).** Στη μετατροπή αυτή ισχύουν οι εξής κανόνες:

A + B → AB+ A – B → BA– A x B → ABx A/B → BA/

E = A x (B – C x D)/(C + G) = A x (B – CDx)/CG+ = A x CDxB– / CG+ = ACDxB–x / CG+ = CG+ACDxB–x/

PUSH C	//C
PUSH G	//G, C
ADD	//G + C
PUSH A	//A, G + C
PUSH C	//C, A, G + C
PUSH D	//D, C, A, G + C
MUL	//DxC, A, G + C
PUSH B	//B, DxC, A, G+C
SUB	//B – DxC, A, G + C
MUL	//(B – DxC)xA, G + C
DIV	//(B – DxC) x A /(G+C)
POP E	//E ← (B –Dx C) x A / (G + C)

β) Μηχανισμός συσσωρευτή: E = A x (B – C x D)/(C+G)

Α′ τρόπος (κακός):

LOAD C	//Σ ← C
MUL D	//Σ ← C x D
STORE X	//X ← C x D
LOAD B	//Σ ← B
SUB X	//Σ ← B – X = B – C x D
MUL A	//Σ ← (B – C x D) x A
STORE X	//X ← (B – C x D) x A
LOAD C	//Σ ← C
ADD G	// Σ ← C + G
STORE Y	//Y ← C + G
LOAD X	// Σ ← X = (B – C x D) x A
DIV Y	//Σ ← (B – C x D) x A / (C+G)
STORE E	//E ← (B – C x D) x A / (C+G)

13 εντολές

Β′ τρόπος (καλός): E = A x (B – CxD)/(C+G)

LOAD C	//Σ ← C
ADD G	//Σ ← C + G
STORE X	//X ← C + G
LOAD C	//Σ ← C
MUL D	//Σ ← C x D
STORE Y	//Y ← C x D
LOAD B	//Σ ← B
SUB Y	//Σ ← B – Y = B – C x D
MUL A	//Σ ← (B – C x D) x A
DIV X	//Σ ← (B – C x D) x A / (C+G)
STORE E	//E ← (B – C x D) x A / (C+G)

11 εντολές

γ_1) Μηχανισμός καταχωρητή – καταχωρητή με δύο τελούμενα: E = A x (B – C x D) / (C + G)

LOAD R1, C	//R1 ← C
LOAD R2, G	//R2 ← G
ADD R2, R1	//R2 ← C + G
LOAD R3, D	//R3 ← D
MUL R1, R3	//R1 ← C x D

```
LOAD R3, B          //R3 ← B
SUB R3, R1          // R3 ← B - C x D
LOAD R1, A          // R1 ← A
MUL R1, R3          // R1 ← A x (B - C x D)
DIV R1, R2          // R1 ← A x (B - C x D)/ (C + G)
STORE E, R1         // E ← A x (B - C x D)/ (C + G)
```

γ_2) Μηχανισμός καταχωρητή – καταχωρητή και καταχωρητή – μνήμης με δύο τελούμενα: E = Ax(B–C x D)/(C+ G)

```
LOAD R1, C          //R1 ← C
ADD R1, G           //R1 ← C + G
LOAD R2, C          //R2 ← C
MUL R2, D           //R2 ← C x D
LOAD R3, B          //R3 ← B
SUB R3, R2          // R3 ← B - C x D
MUL R3, A           // R3 ← A x (B - C x D)
DIV R3, R1          // R3 ← A x (B - C x D)/ (C + G)
STORE E, R3         // E ← A x (B - C x D)/ (C + G)
```

Λύση θέματος 3 – Ιανουάριος 2020

Για τον υπολογισμό της έκφρασης **Y = A + B x C + D x E**, να γραφτεί πρόγραμμα για υπολογιστή που βασίζεται σε (i) συσσωρευτή, (ii) μηχανισμού στοίβας και (iii) 16 καταχωρητών γενικού σκοπού των 32 δυαδικών ψηφίων, με εντολές μόνο καταχωρητή – καταχωρητή (ή φόρτωσης – αποθήκευσης) με δύο τελούμενα.

α. Το σύνολο των εντολών σε επίπεδο γλώσσας μηχανής αποτελείται από **200 εντολές**. Το μήκος κάθε εντολής σε δυαδικά ψηφία είναι ακέραιο πολλαπλάσιο του 8. Όπου απαιτείται διευθυνσιοδότηση της μνήμης να χρησιμοποιήσετε τον κατευθείαν τρόπο διευθυνσιοδότησης (direct addressing mode). Το **εύρος της αρτηρίας διευθύνσεων είναι 24 δυαδικά ψηφία**. Για κάθε διαφορετική εντολή, από τις εντολές σε συμβολική γλώσσα που θα χρησιμοποιήσετε, να σχεδιάσετε τη μορφή της εντολής σε επίπεδο γλώσσας μηχανής (δηλαδή τα πεδία από τα οποία αποτελείται κάθε εντολή σε επίπεδο γλώσσας μηχανής και το εύρος του κάθε πεδίου).

β. Θεωρείστε ότι τα **δεδομένα είναι μίας λέξης** (32 δυαδικά ψηφία), η **αρτηρία δεδομένων** μεταξύ της Κεντρικής Μονάδας Επεξεργασίας (ΚΜΕ) και του συστήματος μνήμης **έχει εύρος 4 ψηφιολέξεις (byte)** και σε κάθε διεύθυνση του συστήματος μνήμης αντιστοιχεί μια ψηφιολέξη και τόσο οι διευθύνσεις των εντολών όσο και των δεδομένων είναι ευθυγραμμισμένες (aligned). Στην περίπτωση μηχανής που βασίζεται στο μηχανισμό στοίβας, η στοίβα υλοποιείται στην ΚΜΕ. Πόσες προσπελάσεις στη μνήμη απαιτεί η εκτέλεση κάθε εντολής.

Λύση

(i) Συσσωρευτής

```
LOAD B      //Σ ← B
MUL C       //Σ ← B x C
ADD A       //Σ ← A + B x C
STORE Y     //Y ← Σ
LOAD D      //Σ ← D
MUL E       //Σ ← D x E
ADD Y       //Σ ← A + B x C + D x E
STORE Y     //Y ← A + B x C + D x E
```

Δεν δίνεται άμεσα το μήκος (μέγεθος) του opcode, αλλά δίνεται **έμμεσα** μέσω του πλήθους των εντολών, που είναι 200. Ισχύει από τη θεωρία ότι από το **πλήθος των εντολών** υπολογίζουμε το μήκος του κωδικού λειτουργίας κάθε εντολής,

χρησιμοποιώντας τον τύπο 2^x >= **πλήθους εντολών = 200 ⇒ x = 8 bits**. Επομένως, επειδή όλες οι εντολές του συσσωρευτή είναι πανομοιότυπες, κάθε εντολή έχει μήκος 8 + 24 = 32 bits = 4 bytes, αφού δίνεται ότι το μέγεθος της αρτηρίας διευθύνσεων είναι των 24 bits.

Μέγεθος: 8 εντολές x 4 bytes = 32 bytes.

Προσπελάσεις: 8 εντολές x 5 προσπελάσεις = 40 προσπελάσεις, διότι σε εντολές που έχουν έντελο μνήμη, οι προσπελάσεις είναι τόσες περισσότερες από τα bytes, όσο είναι το μέγεθος των δεδομένων, λαμβάνοντας υπόψη και την αρτηρία δεδομένων (data bus). Εδώ τα δεδομένα είναι των 32 bits = 4 bytes και η **α**ρτηρία δεδομένων είναι και αυτή των 32 bits = 4 bytes, οπότε έχουμε $\frac{\text{μέγεθος δεδομένων}}{\text{μέγεθος αρτηρίας δεδομένων}} = \frac{32 \text{ bits}}{32 \text{ bits}} = 1$, άρα χρειαζόμαστε μια **επιπλέον** προσπέλαση.

Σημείωση: Αν η αρτηρία δεδομένων ήταν του ενός byte (8 bits), θα χρειαζόμασταν 4 επιπλέον προσπελάσεις

(ii) Στοίβα (infix → postfix)

Y = A + B x C + D x E = A+ BCx + DEx = ABCx+ + DEx = ABCx+ DEx+

```
PUSH A      //A
PUSH B      //B, A
PUSH C      //C, B, A
MUL         //C x B, A
ADD         //C x B + A
PUSH D      //D, C x B + A
PUSH E      //E, D, C x B + A
MUL         //E x D, C x B + A
ADD         //E x D + C x B + A
POP Y       //Y ← E x D + C x B + A
```

Μέγεθος: 6 εντολές x 4 bytes + 4 εντολές x 1 byte = 28 bytes.

Προσπελάσεις: 6 εντολές x 5 προσπελάσεις + 4 εντολές x 1 προσπέλαση = 34 προσπελάσεις

(iii) 16 καταχωρητές γενικού σκοπού με εντολές μόνο καταχωρητή – καταχωρητή με δύο τελούμενα.

Y = A + B x C + D x E

```
LOAD R1, B      //R1 ← B
LOAD R2, C      //R2 ← C
MUL R1, R2      //R1 ← R1 x R2 = B x C
LOAD R2, D      //R2 ← D
LOAD R3, E      //R3 ← E
MUL R2, R3      //R2 ← R2 x R3 = D x E
LOAD R3, A      //R3 ← A
ADD R1, R3      //R1 ← R1 + R3
ADD R1, R2      //R1 ← R1 + R2
STORE Y, R1
```

Μέγεθος: 6 εντολές x 5 bytes + 4 εντολές x 2 bytes = 38 bytes

Προσπελάσεις: 6 εντολές x 6 προσπελάσεις + 4 εντολές x 2 προσπελάσεις = 44 προσπελάσεις

Θέμα με αρχιτεκτονική καταχωρητή – καταχωρητή με 3 τελούμενα - Σεπτέμβριος 2015

Για κάθε μια των κάτωθι περιπτώσεων να γράψετε πρόγραμμα για τον υπολογισμό της έκφρασης F = (A-B)xC – A/D + E

α. Αρχιτεκτονική που βασίζεται στη χρήση μηχανισμού στοίβας.

β. Αρχιτεκτονική που βασίζεται στη χρήση συσσωρευτή.

γ. Αρχιτεκτονική καταχωρητή – καταχωρητή με δύο τελούμενα.

δ. Αρχιτεκτονική καταχωρητή – καταχωρητή με τρία τελούμενα.

Λύση

α) Μηχανισμός στοίβας (σωρού)

Όταν υλοποιούμε μια έκφραση σε μηχανισμό στοίβας, πρέπει να μετατρέψουμε αυτήν την έκφραση από infix σε postfix:

F = (A - B) x C – A/D + E = BA- x C – DA/ + E = BA–Cx – DA/ + E = DA/BA–Cx–+ E = DA/BA–Cx–E+

```
PUSH D      //D (βάζουμε στη στοίβα το περιεχόμενο της θέσης μνήμης με διεύθυνση D)
PUSH A      //A, D (ότι είναι πιο αριστερά, σημαίνει ότι βρίσκεται στην κορυφή της στοίβας)
DIV         //A/D (διαιρούμε το στοιχείο που είναι στην κορυφή της στοίβας με το από κάτω στοιχείο)
PUSH B      //B, A/D (ότι είναι πιο αριστερά, σημαίνει ότι βρίσκεται στην κορυφή της στοίβας)
PUSH A      //A, B, A/D (ότι είναι πιο αριστερά, σημαίνει ότι βρίσκεται στην κορυφή της στοίβας)
SUB         //A - B, A/D
PUSH C      //C, A - B, A/D
MUL         // C x (A – B), A/D
SUB         // C x (A – B) - A/D
PUSH E      //E, C x (A – B) - A/D
ADD         // E + C x (A – B) - A/D
POP F       //F = E + C x (A – B) - A/D
```

b) Μηχανισμός συσσωρευτή: F = (A – B) x C – A/D + E

Επειδή στην έκφραση υπάρχει ποσότητα που **διαιρείται**, πρέπει να ξεκινήσουμε με αυτήν.

```
LOAD A      //Σ ← καταχωρούμε στο συσσωρευτή το περιεχόμενο της θέσης μνήμης με διεύθυνση A
DIV D       //Σ ← A/D
STORE X1    //X1 ← A/D
LOAD A      //Σ ← A
SUB B       //Σ ← A – B
MUL C       //Σ ← (A – B) x C
SUB X1      //Σ ← (A – B) x C - A/D
ADD E       //Σ ← (A – B) x C - A/D + E
STORE F     //F ← (A – B) x C - A/D + E
```

c) Μηχανισμός καταχωρητή – καταχωρητή με δύο τελούμενα: F = (A – B) x C – A/D + E

```
LOAD R1, A  //R1 ← A
LOAD R2, D  //R2 ← D
DIV R1, R2  //R1 ← A/D
LOAD R2, A  //R2 ← A
LOAD R3, B  //R3 ← B
SUB R2, R3  //R2 ← A – B
LOAD R3, C  //R3 ← C
MUL R2, R3  //R2 ← (A – B) x C
SUB R2, R1  // R2 ← (A – B) x C - A/D
LOAD R3, E  //R3 ← E
ADD R2, R3  // R2 ← (A – B) x C - A/D + E
STORE F, R2 //F ← R2 =(A – B) x C - A/D + E
```

d) Μηχανισμός καταχωρητή – καταχωρητή με τρία τελούμενα: F = (A – B) x C – A/D + E

Σημείωση: στην περίπτωση **των τριών εντέλων** (τελούμενων) μπορούμε να θεωρήσουμε ότι η κατεύθυνση των πράξεων είναι από αριστερά προς τα δεξιά ή αντίστροφα (εκτός και αν καθορίζεται από την εκφώνηση της άσκησης, συγκεκριμένη κατεύθυνση για την εκτέλεση των πράξεων). Για παράδειγμα, μπορούμε να θεωρήσουμε ότι η πράξη ADD R1, R2, R3 θα έχει ως αποτέλεσμα R1 + R2 → R3 ή μπορούμε να θεωρήσουμε ότι η ίδια πράξη θα έχει ως αποτέλεσμα R2 + R3 → R1). **Για τον κώδικα που ακολουθεί, κάνουμε τη θεώρηση ότι οι πράξεις εκτελούνται από δεξιά προς τα αριστερά**.

LOAD R1, A	//R1 ← A
LOAD R2, D	//R2 ← D
DIV R2, R1, R2	//**R2 ← R1/R2 = A/D**. Εναλλακτικά, αν η κατεύθυνση είναι προς τα δεξιά: DIV R1, R2, R2
LOAD R3, B	//R3 ← B
SUB R1, R1, R3	// R1 ← A – B
LOAD R3, C	//R3 ← C
MUL R1, R1, R3	// R1 ← (A – B) x C
SUB R1, R1, R2	// R1 ← (A – B) x C - A/D
LOAD R3, E	//R3 ← E
ADD R1, R1, R3	// R1 ← (A – B) x C- A/D + E
STORE F, R1	//F ← (A – B) x C- A/D + E

Πρόοδος Νοεμβρίου 2016

Για κάθε μία εντολή συμβολικής γλώσσας του ακόλουθου πίνακα να δηλώσετε (χωρίς αιτιολόγηση) την αρχιτεκτονική του υπολογιστή στο σύνολο εντολών του οποίου μπορεί να ανήκει. Δηλαδή στο πίνακα δίπλα σε κάθε εντολή να δηλώσετε α ή/και β ή/και γ ή/και δ ή/και ε ή/και ζ, όπου τα α, β, γ, δ, ε και ζ έχουν τη σημασία που δίνεται στη συνέχεια. Τα R1, R2 και R3 δηλώνουν καταχωρητές, ενώ τα A και B δηλώνουν διευθύνσεις θέσεων μνήμης.

Αρχιτεκτονική που βασίζεται σε:

α. χρήση μηχανισμού στοίβας, β. συσσωρευτή, γ. καταχωρητές γενικού σκοπού με εντολές καταχωρητή- καταχωρητή με δύο τελούμενα, δ. καταχωρητές γενικού σκοπού με εντολές καταχωρητή-καταχωρητή με τρία τελούμενα, ε. καταχωρητές γενικού σκοπού με εντολές καταχωρητή-μνήμης με δύο τελούμενα, ζ. καταχωρητές γενικού σκοπού με εντολές καταχωρητή-μνήμης με τρία τελούμενα. Για κάθε λάθος δήλωση θα ακυρώνεται μια σωστή.

Λύση

εντολή συμβολικής γλώσσας	**αρχιτεκτονική του υπολογιστή στο σύνολο εντολών του οποίου μπορεί να ανήκει**
LOAD R1, A	γ, δ, ε, ζ
LOAD R1, (R2)	γ, δ, ε, ζ
ADD	α
PUSH A	α
LOAD A	β
ADD R1, R2, R3	δ
ADD R1, A, B	ζ
SUB A	β
ADD R1, R2	γ
ADD R1, A	ε
STORE Y, R1	γ, δ, ε, ζ

Συμπέρασμα: οι εντολές LOAD, STORE είναι **ανεξάρτητες** από το **πλήθος** και από το **είδος** των εντέλων των αριθμητικών εντολών.

Θέμα με μεταβλητή μορφή εντολών – χώρο και προσπελάσεις προγράμματος – Ιανουάριος 2015

Ο επόμενος πίνακας συνοψίζει τους τρόπους αναγραφής της επιθυμητής διευθυνσιοδότησης σε ένα επεξεργαστή που βασίζεται στη χρήση καταχωρητών γενικού σκοπού (32 καταχωρητές γενικού σκοπού των 8 δυαδικών ψηφίων ο κάθε ένας), σύνολο εντολών **καταχωρητή – καταχωρητή και καταχωρητή – μνήμης με δύο τελούμενα, ο κωδικός λειτουργίας κάθε εντολής καταλαμβάνει μια ψηφιολέξη (byte)** και κάθε εντολή **καταλαμβάνει ακέραιο αριθμό ψηφιολέξεων, αρτηρία διευθύνσεων των 16 γραμμών και αρτηρία δεδομένων των 8 γραμμών**. Η κύρια μνήμη έχει οργάνωση μιας ψηφιολέξης (byte) ανά θέση μνήμης.

Μορφή	Διευθυνσιοδότηση
#X	Άμεση (immediate)
$X	Κατ' ευθείαν θέσης μνήμης
RX	Κατ' ευθείαν καταχωρητή X
(RX)	Έμμεσος τρόπος με χρήση του καταχωρητή X

α. Δώστε τη μορφή κάθε εντολής του διπλανού προγράμματος
β. Να υπολογιστεί ο χώρος που καταλαμβάνει το διπλανό πρόγραμμα
γ. Θεωρήστε ότι τα **δεδομένα είναι μιας ψηφιολέξης**.
Να υπολογιστούν οι προσπελάσεις κατά την εκτέλεση του προγράμματος

```
LDA R1, #5
LDA R2, $90
ADD R1, R2     //R1← R1 + R2
LDA R2, (R3)
SUB R1, R2     //R1← R1 - R2
STORE R1, $200
END
```

Λύση

Υπάρχουν **δύο είδη ασκήσεων**: Αυτές με **μεταβλητό μήκος εντολής** (διατύπωση: **κάθε εντολή καταλαμβάνει ακέραιο αριθμό ψηφιολέξεων**) και αυτές με **σταθερό μήκος εντολής** (διατύπωση: **εντολές με σταθερό μήκος εντολών**).

- Η ένδειξη # δηλώνει δεδομένα, οπότε **δεν** προσπελαύνεται μνήμη και συνεπώς **προσπελάσεις μνήμης = bytes**.

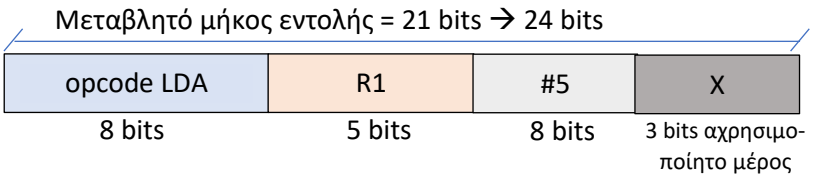
- Η ένδειξη **$** δηλώνει διεύθυνση και το μέγεθός της προκύπτει (προσδιορίζεται) από την αρτηρία διευθύνσεων. Για **προσπέλαση μνήμης**, υπάρχουν δύο τρόποι: είτε με χρήση του συμβόλου $, είτε με χρήση καταχωρητή μέσα σε ( ), όπως π.χ. (Rx) που υποδηλώνει **έμμεσο τρόπο διευθυνσιοδότησης με χρήση καταχωρητή. Σε αυτή την περίπτωση, στην αποθήκευση τον λαμβάνουμε υπόψη ως κανονικό καταχωρητή, όμως στις προσπελάσεις μνήμης λαμβάνουμε υπόψη μας περισσότερες προσπελάσεις από bytes.**

- Από **πλήθος των εντολών** υπολογίζουμε **μήκος του κωδικού λειτουργίας κάθε εντολής**, από τύπο **2^x >= πλήθος εντολών**. Αυτό χρησιμοποιείται όταν δεν δίνεται το μέγεθος του κωδικού λειτουργίας (opcode) κάθε εντολής.

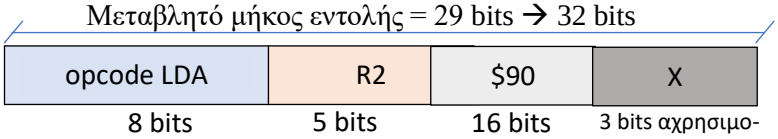
- Από **πλήθος των καταχωρητών** υπολογίζουμε πλήθος των bits για την κωδικοποίησή τους, από **2^x >= πλήθος καταχωρητών** $\Rightarrow 2^x >= 32 \Rightarrow x = 5$ bits.

- Η πληροφορία του μεγέθους των καταχωρητών δεν ενδιαφέρει (χρησιμοποιείται), από τη στιγμή που έχουμε μεταβλητό μήκος εντολής. Θα ενδιέφερε, αν είχαμε σταθερό μήκος εντολής.

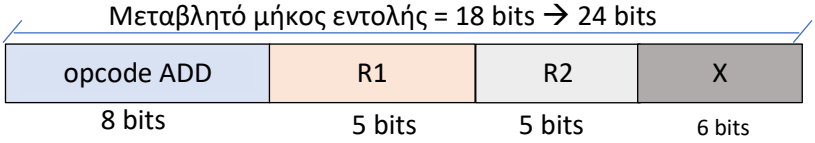
Στη συνέχεια αναλύουμε **την κάθε εντολή χωριστά** και υπολογίζουμε για κάθε μία χώρο/προσπελάσεις μνήμης



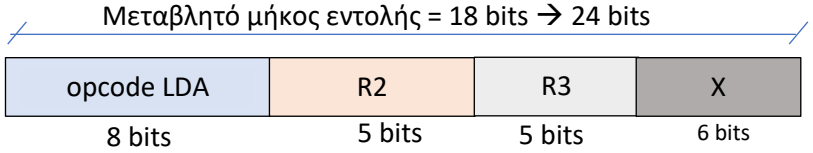
Χωρητικότητα: 24 bits = 3 bytes
Προσπελάσεις μνήμης: 3 προσπελάσεις



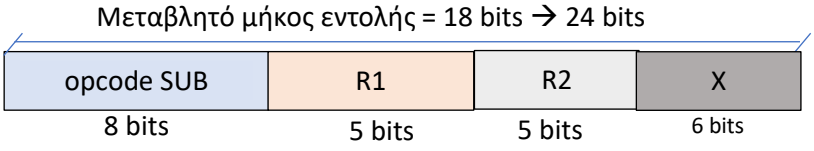
Χωρητικότητα: 32 bits = 4 bytes
Προσπελάσεις μνήμης: 5 προσπελάσεις



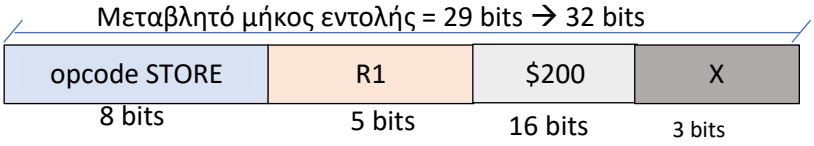
24 bits = 3 bytes 3 προσπελάσεις



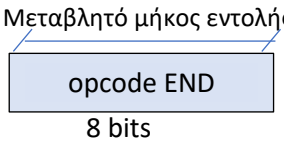
24 bits = 3 bytes 4 προσπελάσεις



24 bits = 3 bytes 3 προσπελάσεις



32 bits = 4 bytes 5 προσπελάσεις



8 bits = 1 byte 1 προσπέλαση

Σύνολο: 21 bytes **24 προσπελάσεις**

Θέμα με μεταβλητή μορφή εντολών – χώρο και προσπελάσεις προγράμματος – Ιούνιος 2016

Δίνεται το κάτωθι τμήμα προγράμματος Α το οποίο εκτελείται σε υπολογιστή με 16 καταχωρητές γενικού σκοπού καθένας των 32 δυαδικών ψηφίων. Θεωρήστε ότι ο κωδικός λειτουργίας κάθε εντολής καταλαμβάνει μία ψηφιολέξη (byte), οι διευθύνσεις είναι των 16 δυαδικών ψηφίων και κάθε εντολή καταλαμβάνει ακέραιο αριθμό ψηφιολέξεων, τα δεδομένα είναι των 32 δυαδικών ψηφίων, εκτός από την περίπτωση του άμεσου τρόπου διευθυνσιοδότησης που είναι των 8 δυαδικών ψηφίων, η αρτηρία δεδομένων μεταξύ της Κεντρικής Μονάδας Επεξεργασίας (ΚΜΕ) και του συστήματος μνήμης έχει εύρος μιας ψηφιολέξης (byte) και σε κάθε διεύθυνση του συστήματος μνήμης αντιστοιχεί μια ψηφιολέξη (δηλαδή σε κάθε θέση μνήμης αποθηκεύεται μια ψηφιολέξη). α. Να δώσετε τη μορφή κάθε είδους εντολής σε επίπεδο γλώσσας μηχανής. β) Να υπολογιστεί ο χώρος που καταλαμβάνει το πρόγραμμα στην κύρια μνήμη του υπολογιστή. γ. Πόσες προσπελάσεις στη μνήμη γίνονται κατά την εκτέλεση του προγράμματος.

Τμήμα Προγράμματος Α	μορφή εντολής σε επίπεδο γλώσσας μηχανής	Χώρος που καταλαμβάνει η εντολή στην κύρια μνήμη σε bytes	Πλήθος προσπελάσεων κατά την εκτέλεση της εντολής
LOAD R1, (R2) /έμμεσος (indirect) τρόπος διευθυνσιοδότησης με χρήση καταχωρητή			
LOAD R3, (R4)			
SUB R2, R1 /R2 – R1 → R2			
LOAD R5, #4D$_{(16)}$ /άμεσος (immediate) τρόπος διευθυνσιοδότησης			
SUB R5, R4			
ADD R3, R1			
STORE R3, $0800 κατ' ευθείαν (direct) τρόπος διευθυνσιοδότησης			

Λύση

Η πρώτη, η δεύτερη και η τελευταία εντολή που δίνονται, προσπελαύνουν μνήμη, οπότε σε αυτές τις τρεις περιπτώσεις οι προσπελάσεις μνήμης είναι περισσότερες από τα bytes και μάλιστα τόσες περισσότερες όσες δηλώνει το κλάσμα $\frac{\text{μέγεθος δεδομένων}}{\text{μέγεθος αρτηρίας δεδομένων}} = \frac{32 \text{ bits}}{8 \text{ bits}} = 4$. Από **πλήθος των καταχωρητών** υπολογίζουμε πλήθος των bits για την κωδικοποίησή τους, δηλαδή **2^x >= πλήθος καταχωρητών** $\Rightarrow 2^x >= 16 \Rightarrow x = 4$ bits. Οι εντολές SUB, ADD και ADD έχουν παρόμοια μορφή, μέγεθος και προσπελάσεις και καμία από αυτές δεν προσπελαύνει μνήμη.

					Χώρος (bytes)	**Προσπελάσεις**
a)	opcode LOAD	R1	R2		2	6
b)	opcode LOAD	R3	R4		2	6
c)	opcode SUB	R2	R1		2	2
d)	opcode LOAD	R5	4D	**X**	3	3
e)	opcode SUB	R5	R4		2	2
f)	opcode ADD	R3	R1		2	2
g)	opcode STORE	R3	0800	**X**	4	8

Θέμα με σταθερή μορφή εντολών και πλήθος θ.μ. που προσπελαύνονται - Νοέμβριος 2016

Θεωρήστε ένα υπολογιστή που βασίζεται στη χρήση καταχωρητών γενικού σκοπού και του οποίου το σύνολο των εντολών σε επίπεδο γλώσσας μηχανής αποτελείται από 200 εντολές, με **σταθερό μήκος εντολών των 24 δυαδικών ψηφίων** και 16 καταχωρητές **με μήκος 32 δυαδικών ψηφίων ο καθένας**.

α. Για κάθε μία από τις ακόλουθες εντολές σε συμβολική γλώσσα να σχεδιάσετε τη μορφή των εντολών σε επίπεδο γλώσσας μηχανής (δηλαδή τα πεδία από τα οποία αποτελείται κάθε εντολή σε επίπεδο γλώσσας μηχανής και το εύρος κάθε πεδίου).

β. Ποιο είναι το πλήθος των θέσεων μνήμης που μπορεί να διευθυνσιοδοτηθεί από κάθε μία των κάτωθι εντολών;

Λύση

Τμήμα Προγράμματος Α	**μορφή εντολής σε επίπεδο γλώσσας μηχανής**	**πλήθος θέσεων μνήμης που μπορεί να διευθυνσιοδοτηθεί από την εντολή**
LOAD R, A // R ← A, όπου A διεύθυνση θέσης μνήμης (κατ' ευθείαν τρόπος διευθυνσιοδότησης)	8 bits: κωδικός λειτουργίας; 4 bits: R1; 12 bits: διεύθυνση θέσης μνήμης	2^{12} θέσεις
LOAD R1, (R2) // R1 ← M(R2), έμμεσος (indirect) τρόπος διευθυνσιοδότησης με χρήση καταχωρητή	8 bits: κωδικός λειτουργίας; 4 bits: R1; 4 bits: R2; 8 bits: αχρησιμοποίητα	2^{32} θέσεις
LOAD R1, d(R2) // R1 ← M[d+(R2)], έμμεσος (indirect) τρόπος διευθυνσιοδότησης με χρήση καταχωρητή	8 bits: κωδικός λειτουργίας; 4 bits: R1; 4 bits: R2; 8 bits: d	2^{32} θέσεις
LOAD R, #0AB$_{(16)}$ // άμεσος (immediate) τρόπος διευθυνσιοδότησης	8 bits: κωδικός λειτουργίας; 4 bits: R5; 12 bits: δεδομένο	0
BRE d // σχετικός τρόπος διευθυνσιοδότησης με χρήση του μετρητή προγράμματος, εάν η σημαία της ισότητας είναι 1, αλλαγή της ροής του προγράμματος	8 bits: κωδικός λειτουργίας; 16 bits: d	2^{16} θέσεις (με την έννοια ότι μπορεί να μεταφέρει τον έλεγχο σε 2^{16} διαφορετικές διευθύνσεις)

Θέμα με σταθερή μορφή εντολών και πλήθος θ.μ. που προσπελαύνονται - Ιούνιος 2016

Θεωρήστε ένα υπολογιστή που βασίζεται στη χρήση καταχωρητών γενικού σκοπού και του οποίου το σύνολο των εντολών σε επίπεδο γλώσσας μηχανής αποτελείται από 200 εντολές, με **σταθερό μήκος εντολών των 24 δυαδικών ψηφίων** και 64 καταχωρητές **με μήκος 32 δυαδικών ψηφίων ο καθένας**.

α. Για κάθε μία από τις ακόλουθες εντολές σε συμβολική γλώσσα να σχεδιάσετε τη μορφή των εντολών σε επίπεδο γλώσσας μηχανής (δηλαδή τα πεδία από τα οποία αποτελείται κάθε εντολή σε επίπεδο γλώσσας μηχανής και το εύρος κάθε πεδίου).

β. Ποιο είναι το πλήθος των θέσεων μνήμης που μπορεί να διευθυνσιοδοτηθεί από κάθε μία των κάτωθι εντολών;

Τμήμα Προγράμματος Α	μορφή εντολής σε επίπεδο γλώσσας μηχανής	Πλήθος των θέσεων μνήμης
LOAD R, A // R ← A, όπου Α διεύθυνση θέσης μνήμης (κατ' ευθείαν τρόπος διευθυνσιοδότησης)		
LOAD R1, (R2) // R1 ← M(R2), έμμεσος (indirect) τρόπος διευθυνσιοδότησης με χρήση καταχωρητή		
LOAD R1, d(R2) // R1 ← M[d+(R2)], έμμεσος (indirect) τρόπος διευθυνσιοδότησης με χρήση καταχωρητή		
LOAD R, #4DF7$_{(16)}$ // άμεσος (immediate) τρόπος διευθυνσιοδότησης		

Λύση

Τμήμα Προγράμματος Α	μορφή εντολής σε επίπεδο γλώσσας μηχανής				Πλήθος των θέσεων μνήμης
LOAD R, A // R ← A, όπου Α διεύθυνση θέσης μνήμης (κατ' ευθείαν τρόπος διευθυνσιοδότησης)	8 bits	6 bits		18 bits	2^{18}
	opcode	καταχωρητής R		A	
LOAD R1, (R2) // R1 ← M(R2), έμμεσος (indirect) τρόπος διευθυνσιοδότησης με χρήση καταχωρητή	8 bits	6 bits	6 bits	12 bits	2^{32}
	opcode	R1	(R2)	X.	
LOAD R1, d(R2) // R1 ← M[d+(R2)], έμμεσος (indirect) τρόπος διευθυνσιοδότησης με χρήση καταχωρητή	8 bits	6 bits	6 bits	12 bits	2^{32}
	opcode	R1	(R2)	d	
LOAD R, #4DF7$_{(16)}$ //άμεσος (immediate) τρόπος διευθυνσιοδότησης	8 bits	6 bits	16 bits	2 bits	0
	opcode	R	αριθμός	X	

Θέμα Φεβρουαρίου 2021

Θεωρήστε υπολογιστή ο οποίος διαθέτει 253 εντολές σε επίπεδο γλώσσας μηχανής, 52 καταχωρητές των 36 δυαδικών ψηφίων ο καθένας, κύρια μνήμη με **18 δυαδικά ψηφία ανά θέση μνήμης** και **εύρος αρτηρίας δεδομένων 18 δυαδικών ψηφίων**. Θεωρείστε ότι το σύστημα της κύριας μνήμης έχει σχεδιαστεί έτσι ώστε να είναι δυνατή η ταυτόχρονη προσπέλαση (για ανάγνωση ή εγγραφή) τόσων διαδοχικών θέσεων μνήμης όσων επιτρέπει το εύρος της αρτηρίας δεδομένων. Για κάθε μία από τις ακόλουθες εντολές **να δώσετε το πλήθος των θέσεων μνήμης που καταλαμβάνει η εντολή** (ένα αριθμό) και το **πλήθος των προσπελάσεων (ένα αριθμό) που απαιτούνται για την προσκόμιση και εκτέλεση κάθε εντολής**.

1. **LOAD1m R1, (R2)** //έμμεσος (indirect) τρόπος διευθυνσιοδότησης με χρήση καταχωρητή και προσκόμιση του περιεχομένου 1 θέσης μνήμης.

Πλήθος θέσεων μνήμης που καταλαμβάνει η εντολή [ ]. Πλήθος προσπελάσεων που απαιτούνται για την προσκόμιση και εκτέλεση της εντολής [ ].

2. **LOAD3m R1, (R2)** //έμμεσος (indirect) τρόπος διευθυνσιοδότησης με χρήση καταχωρητή και προσκόμιση του περιεχομένου 3 θέσεων μνήμης.

Πλήθος θέσεων μνήμης που καταλαμβάνει η εντολή [ ]. Πλήθος προσπελάσεων που απαιτούνται για την προσκόμιση και εκτέλεση της εντολής [ ].

3. **ADD R3, R1** //R3 ← R3 + R1

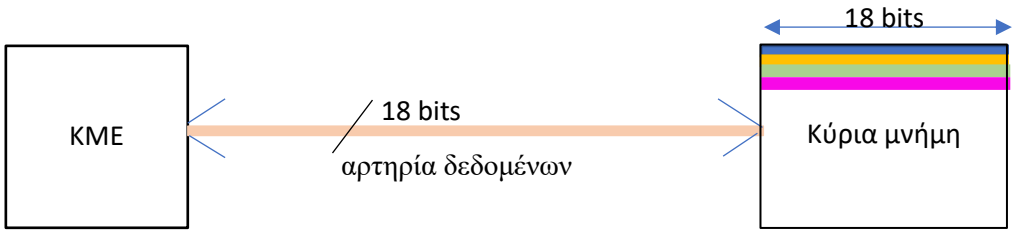
Πλήθος θέσεων μνήμης που καταλαμβάνει η εντολή [ ]. Πλήθος προσπελάσεων που απαιτούνται για την προσκόμιση και εκτέλεση της εντολής [ ].

4. **LOAD1m R1, $X** //κατευθείαν (direct) τρόπος διευθυνσιοδότησης, το X είναι αριθμός των 16 δυαδικών ψηφίων και προσκόμιση περιεχομένου 1 θέσης μνήμης.

Πλήθος θέσεων μνήμης που καταλαμβάνει η εντολή [ ]. Πλήθος προσπελάσεων που απαιτούνται για την προσκόμιση και εκτέλεση της εντολής [ ].

Λύση

Μια εντολή είναι αποθηκευμένη στην κύρια μνήμη και για να εκτελεστεί θα πρέπει να προσκομιστεί από την κύρια μνήμη στην ΚΜΕ. Μετά το στάδιο της προσκόμισης, ακολουθεί το στάδιο της εκτέλεσης της εντολής. Στην προκειμένη περίπτωση το **εύρος της αρτηρίας δεδομένων ταυτίζεται με το εύρος της κάθε μιας θέσης μνήμης**. Αυτό σημαίνει ότι με κάθε προσπέλαση μνήμης μεταφέρεται μέσω της αρτηρίας δεδομένων, το περιεχόμενο μιας θέσης μνήμης (εντολή) στην ΚΜΕ. Αν το μέγεθος της αρτηρίας δεδομένων ήταν 36 bits, αυτό σημαίνει ότι με κάθε προσπέλαση μνήμης θα μεταφερόντουσαν στην ΚΜΕ τα περιεχόμενα δύο θέσεων μνήμης, αν το μέγεθος της αρτηρίας δεδομένων ήταν 54 bits, αυτό σημαίνει ότι με κάθε προσπέλαση μνήμης θα μεταφερόντουσαν στην ΚΜΕ τα περιεχόμενα τριών θέσεων μνήμης κ.ο.κ.



Από πλήθος εντολών, υπολογίζουμε το μέγεθος του κωδικού λειτουργίας και πιο συγκεκριμένα: $2^x \geq 253 \Rightarrow x = 8$ **bits** (μέγεθος opcode). Από πλήθος καταχωρητών, υπολογίζουμε μέγεθος κάθε καταχωρητή: $2^x \geq 52 \Rightarrow x = 6$ **bits**.

1) LOAD1m R1, (R2)

opcode LOAD1m	R1	R2
8	6	6

Μήκος εντολής = 20 bits → με δεδομένο ότι κάθε θ.μ. έχει χωρητικότητα 18 bits, **η εντολή καταλαμβάνει 2 θ.μ.**

Επειδή το εύρος του data bus = 18 bits μπορούμε κάθε φορά να προσπελάσουμε **μια θέση μνήμης, αφού η κάθε θέση μνήμης έχει και αυτή μέγεθος 18 bits**. Η **προσκόμιση** της εντολής των 20 bits στην ΚΜΕ θα γίνει με **δύο προσπελάσεις μνήμης** (σε κάθε προσπέλαση μεταφέρονται έως 18 bits). Η **εκτέλεση** της εντολής θα προσκομίσει το περιεχόμενο μιας θ.μ. (18 bits), αφού στην εντολή υπάρχει η ένδειξη 1m και για αυτό θα απαιτηθεί άλλη μια προσπέλαση. Άρα συνολικά για την προσκόμιση (fetch) και για την εκτέλεση της εντολής **απαιτούνται 3 προσπελάσεις**.

2) LOAD3m R1, (R2)

opcode LOAD3m	R1	R2

8	6	6

Μήκος εντολής = 20 bits → με δεδομένο ότι κάθε θ.μ. έχει 18 bits, **η εντολή καταλαμβάνει 2 θ.μ**.

Επειδή το εύρος του data bus = 18 bits μπορούμε κάθε φορά να προσπελάσουμε **μια θέση μνήμης, αφού η κάθε θέση μνήμης έχει μέγεθος 18 bits**. Η **προσκόμιση** της εντολής των 20 bits θα γίνει με δύο προσπελάσεις μνήμης (σε κάθε προσπέλαση μεταφέρονται έως 18 bits). Η **εκτέλεση** της εντολής **θα προσκομίσει το περιεχόμενο τριών θ.μ. (54 bits)**, αφού στην εντολή υπάρχει η ένδειξη 3m, οπότε θα χρειαστούν 3 προσπελάσεις (καθεμία μεταφέρει 18 bits). Άρα συνολικά για την προσκόμιση (fetch) και για την εκτέλεση της εντολής **απαιτούνται 5 προσπελάσεις**.

3) ADD R3, R1

opcode ADD	R3	R1
8	6	6

Μήκος εντολής = 20 bits → με δεδομένο ότι κάθε θ.μ. έχει 18 bits, **η εντολή καταλαμβάνει 2 θ.μ**.

Επειδή το εύρος του data bus = 18 bits μπορούμε κάθε φορά να προσπελάσουμε **μια θέση μνήμης, αφού η κάθε θέση μνήμης έχει μέγεθος 18 bits**. Η **προσκόμιση** της εντολής των 20 bits θα γίνει με **δύο** προσπελάσεις μνήμης (σε κάθε προσπέλαση μεταφέρονται έως 18 bits). Η εντολή αυτή **δεν** προσπελαύνει μνήμη. Άρα συνολικά για την προσκόμιση (fetch) και για την εκτέλεση της εντολής **απαιτούνται 2 προσπελάσεις**

4) LOAD1m R1, $X

opcode LOAD1m	R1	X
8	6	16

Μήκος εντολής = 30 bits → με δεδομένο ότι κάθε θ.μ. έχει 18 bits, **η εντολή καταλαμβάνει 2 θ.μ**.

Επειδή το εύρος του data bus = 18 bits μπορούμε κάθε φορά να προσπελάσουμε **μια θέση μνήμης, αφού η κάθε θέση μνήμης έχει μέγεθος 18 bits**. Η **προσκόμιση** της εντολής των 30 bits θα γίνει με δύο προσπελάσεις μνήμης (σε κάθε προσπέλαση μεταφέρονται έως 18 bits). Η **εκτέλεση** της εντολής θα προσκομίσει το περιεχόμενο μιας θ.μ. (18 bits), αφού στην εντολή υπάρχει η ένδειξη 1m, οπότε θα χρειαστεί μια προσπέλαση. Άρα συνολικά για την προσκόμιση (fetch) και για την εκτέλεση της εντολής **απαιτούνται 3 προσπελάσεις**.

Θέμα Ιουνίου 2020

Θεωρήστε υπολογιστή ο οποίος διαθέτει 205 εντολές σε επίπεδο γλώσσας μηχανής, 20 καταχωρητές των 42 δυαδικών ψηφίων ο καθένας, κύρια μνήμη με **14 δυαδικά ψηφία ανά θέση μνήμης** και εύρος **αρτηρίας δεδομένων 14 δυαδικών ψηφίων**. Θεωρείστε ότι το σύστημα της κύριας μνήμης έχει σχεδιαστεί έτσι ώστε να είναι δυνατή η ταυτόχρονη προσπέλαση (για ανάγνωση ή εγγραφή) τόσων διαδοχικών θέσεων μνήμης όσων επιτρέπει το εύρος της αρτηρίας δεδομένων. Για κάθε μία από τις ακόλουθες εντολές να δώσετε το πλήθος των θέσεων μνήμης που καταλαμβάνει η εντολή (ένα αριθμό) και το πλήθος των προσπελάσεων (ένα αριθμό) που απαιτούνται για την προσκόμιση και εκτέλεση κάθε εντολής.

1. LOAD1m R1, (R2) //έμμεσος (indirect) τρόπος διευθυνσιοδότησης με χρήση καταχωρητή και προσκόμιση του περιεχομένου 1 θέσης μνήμης.

Πλήθος θέσεων μνήμης που καταλαμβάνει η εντολή ____. Πλήθος προσπελάσεων που απαιτούνται για την προσκόμιση και εκτέλεση της εντολής ____.

2. LOAD3m R1, (R2) //έμμεσος (indirect) τρόπος διευθυνσιοδότησης με χρήση καταχωρητή και προσκόμιση του περιεχομένου 3 θέσεων μνήμης.

Πλήθος θέσεων μνήμης που καταλαμβάνει η εντολή [ ]. Πλήθος προσπελάσεων που απαιτούνται για την προσκόμιση και εκτέλεση της εντολής [ ].

3. ADD R3, R1 //R3 ← R3 + R1

Πλήθος θέσεων μνήμης που καταλαμβάνει η εντολή [ ]. Πλήθος προσπελάσεων που απαιτούνται για την προσκόμιση και εκτέλεση της εντολής [ ].

4. LOAD1m R1, $X //κατευθείαν (direct) τρόπος διευθυνσιοδότησης, το X είναι αριθμός των 16 δυαδικών ψηφίων και προσκόμιση περιεχομένου 1 θέσης μνήμης.

Πλήθος θέσεων μνήμης που καταλαμβάνει η εντολή [ ]. Πλήθος προσπελάσεων που απαιτούνται για την προσκόμιση και εκτέλεση της εντολής [ ].

Λύση

Από το πλήθος των εντολών που είναι **205** θα υπολογίσουμε το opcode για κάθε εντολή που είναι 8 bit. Από το πλήθος των καταχωρητών που είναι **20** θα υπολογίσουμε το πεδίο R για κάθε εντολή που είναι 5 bit

1. LOAD1m R1, (R2) //φέρε περιεχόμενο 1 θέσης μνήμης

Opcode	R1	R2	Total
8	5	5	18 bits

14 bits ανά θέση μνήμης άρα **2 θέσεις μνήμης**

Προσπελάσεις = 3 = **2 για την προσκόμιση και 1 για την προσπέλαση 1 θέσης μνήμης κατά την εκτέλεση**.

2. LOAD3m R1, (R2) //φέρε περιεχόμενο 3 θέσεων μνήμης

Opcode	R1	R2	Total
8	5	5	18 bits

14 bits ανά θέση μνήμης άρα **2 θέσεις μνήμης**

Προσπελάσεις = 5 = 2 για την προσκόμιση και 3 για την εκτέλεση, όπου τότε γίνεται προσπέλαση 3 θέσεων μνήμης.

3. ADD R3, R1

Opcode	R1	R3	Total
8	5	5	18 bits

14bit ανά θέση μνήμης, άρα **2 θέσεις μνήμης**

Προσπελάσεις = 2 για την προσκόμιση, ενώ για την εκτέλεσή της δεν χρειάζεται καμία προσπέλαση μνήμης.

4. LOAD1m R1, $X //φέρε περιεχόμενο 1 θέσης μνήμης

Opcode	R1	X	Total
8	5	16	29 bits

14bit ανά θέση μνήμης, **άρα 3 θέσεις μνήμης**

Προσπελάσεις = 4 = εκ' των οποίων οι 3 για την προσκόμιση εντολής και 1 για την προσπέλαση μιας θέσης μνήμης.

Θέμα Σεπτεμβρίου 2020

Θεωρήστε υπολογιστή ο οποίος διαθέτει 94 εντολές σε επίπεδο γλώσσας μηχανής, 52 καταχωρητές των 90 δυαδικών ψηφίων ο καθένας, κύρια μνήμη με **10 δυαδικά ψηφία ανά θέση μνήμης** και εύρος **αρτηρίας δεδομένων 30 δυαδικών**

ψηφίων. Θεωρείστε ότι το σύστημα της κύριας μνήμης έχει σχεδιαστεί έτσι ώστε να είναι δυνατή η ταυτόχρονη προσπέλαση (για ανάγνωση ή εγγραφή) τόσων διαδοχικών θέσεων μνήμης όσων επιτρέπει το εύρος της αρτηρίας δεδομένων. Για κάθε μία από τις ακόλουθες εντολές να δώσετε το πλήθος των θέσεων μνήμης που καταλαμβάνει η εντολή (ένα αριθμό) και το πλήθος των προσπελάσεων (ένα αριθμό) που απαιτούνται για την προσκόμιση και εκτέλεση κάθε εντολής.

1. LOAD1m R1, (R2) //έμμεσος (indirect) τρόπος διευθυνσιοδότησης με χρήση καταχωρητή και προσκόμιση του περιεχομένου 1 θέσης μνήμης.

Πλήθος θέσεων μνήμης που καταλαμβάνει η εντολή [ ]. Πλήθος προσπελάσεων που απαιτούνται για την προσκόμιση και εκτέλεση της εντολής [ ].

2. LOAD3m R1, (R2) //έμμεσος (indirect) τρόπος διευθυνσιοδότησης με χρήση καταχωρητή και προσκόμιση του περιεχομένου 3 θέσεων μνήμης.

Πλήθος θέσεων μνήμης που καταλαμβάνει η εντολή [ ]. Πλήθος προσπελάσεων που απαιτούνται για την προσκόμιση και εκτέλεση της εντολής [ ].

3. ADD R3, R1 //R3 ← R3 + R1

Πλήθος θέσεων μνήμης που καταλαμβάνει η εντολή [ ]. Πλήθος προσπελάσεων που απαιτούνται για την προσκόμιση και εκτέλεση της εντολής [ ].

4. LOAD1m R1, $X //κατευθείαν (direct) τρόπος διευθυνσιοδότησης, το X είναι αριθμός των 16 δυαδικών ψηφίων και προσκόμιση περιεχομένου 1 θέσης μνήμης.

Πλήθος θέσεων μνήμης που καταλαμβάνει η εντολή [ ]. Πλήθος προσπελάσεων που απαιτούνται για την προσκόμιση και εκτέλεση της εντολής [ ].

Λύση

Από το πλήθος των εντολών που είναι 94 θα υπολογίσουμε το opcode για κάθε εντολή που είναι 7 bit. Από το πλήθος των καταχωρητών που είναι 52, θα υπολογίσουμε το πεδίο R για κάθε εντολή που είναι 6 bit. Επίσης, επειδή το εύρος της αρτηρίας δεδομένων είναι τριπλάσιο του μεγέθους κάθε θέσης μνήμης, σημαίνει ότι **με κάθε προσπέλαση μνήμης μεταφέρεται το περιεχόμενο 3 θέσεων μνήμης**.

1. LOAD1m R1, (R2) //φέρε περιεχόμενο 1 θέσης μνήμης

Opcode	R1	R2	Total
7	6	6	19 bits

19 bits ανά θέση μνήμης, άρα **2 θέσεις μνήμης** (αφού κάθε θέση μνήμης έχει χωρητικότητα 10 bits)

Προσπελάσεις = 2 (1 για την προσκόμιση και 1 για την εκτέλεση)

2. LOAD3m R1, (R2) //φέρε περιεχόμενο 3 θέσεων μνήμης

Opcode	R1	R2	Total
7	6	6	19 bits

19 bits ανά θέση μνήμης άρα **2 θέσεις μνήμης** (αφού κάθε θέση μνήμης έχει χωρητικότητα 10 bits)

Προσπελάσεις = 2 (1 για την προσκόμιση και 1 για την εκτέλεση).

3. ADD R3, R1

Opcode	R	R	Total
7	6	6	19 bits

19 bit ανά θέση μνήμης, άρα **2 θέσεις μνήμης** (αφού κάθε θέση μνήμης έχει χωρητικότητα 10 bits)

Προσπελάσεις = 1 (1 για την προσκόμιση, ενώ για την εκτέλεσή της δεν χρειάζεται καμία προσπέλαση μνήμης).

4. LOAD1m R1, $X //φέρε περιεχόμενο 1 θέσης μνήμης

Opcode	R	X	Total
7	6	16	29 bits

29 bits, **άρα 3 θέσεις μνήμης** (αφού κάθε θέση μνήμης έχει χωρητικότητα 10 bits)

Προσπελάσεις = 2 (1 προσπέλαση για την προσκόμιση και 1 για την εκτέλεση).

Θέμα Σεπτεμβρίου 2020 (άλλη ομάδα)

Θεωρήστε υπολογιστή ο οποίος διαθέτει 351 εντολές σε επίπεδο γλώσσας μηχανής, 6 καταχωρητές των 144 δυαδικών ψηφίων ο καθένας, κύρια μνήμη με 16 δυαδικά ψηφία ανά θέση μνήμης, και εύρος αρτηρίας δεδομένων 48 δυαδικών ψηφίων. Θεωρήστε ότι το σύστημα της κύριας μνήμης έχει σχεδιαστεί έτσι ώστε να είναι δυνατή η ταυτόχρονη προσπέλαση (για ανάγνωση ή εγγραφή) τόσων διαδοχικών θέσεων μνήμης όσων επιτρέπει το εύρος της αρτηρίας δεδομένων. Για καθεμία από τις ακόλουθες εντολές να δώσετε το πλήθος των θέσεων μνήμης που καταλαμβάνει η εντολή (ένα αριθμό) και το πλήθος προσπελάσεων (ένα αριθμό) που απαιτούνται για την προσκόμιση και την εκτέλεση κάθε εντολής.

1. LOAD1m R1, (R2) / έμμεσος (indirect) τρόπος διευθυνσιοδότησης με χρήση καταχωρητή και προσκόμιση του περιεχομένου 1 θέσης μνήμης. Πλήθος θέσεων μνήμης που καταλαμβάνει η εντολή: .
   Πλήθος προσπελάσεων που απαιτούνται για την προσκόμιση και την εκτέλεση της εντολής: .
2. LOAD3m R1, (R2) / έμμεσος (indirect) τρόπος διευθυνσιοδότησης με χρήση καταχωρητή και προσκόμιση του περιεχομένου 3 θέσεων μνήμης. Πλήθος θέσεων μνήμης που καταλαμβάνει η εντολή: .
   Πλήθος προσπελάσεων που απαιτούνται για την προσκόμιση και την εκτέλεση της εντολής: .
3. ADD R3, R1 / R3 <-- R3 + R1
   Πλήθος θέσεων μνήμης που καταλαμβάνει η εντολή: .
   Πλήθος προσπελάσεων που απαιτούνται για την προσκόμιση και την εκτέλεση της εντολής: .
4. LOAD1m R1, $X / κατ' ευθείαν (direct) τρόπος διευθυνσιοδότησης, το X είναι αριθμός των 16 δυαδικών ψηφίων και προσκόμιση του περιεχομένου 1 θέσης μνήμης.
   Πλήθος θέσεων μνήμης που καταλαμβάνει η εντολή: .
   Πλήθος προσπελάσεων που απαιτούνται για την προσκόμιση και την εκτέλεση της εντολής: .

Λύση

Από το πλήθος των εντολών που δίνονται, μπορούμε να υπολογίσουμε το μέγεθος του κωδικού λειτουργίας και πιο συγκεκριμένα: $2^x \geq 351 \Rightarrow x = 9$ bits (μέγεθος opcode). Από το πλήθος των καταχωρητών που δίνονται, μπορούμε να υπολογίσουμε το μέγεθος του κάθε καταχωρητή και πιο συγκεκριμένα: $2^x \geq 6 \Rightarrow x = 3$ bits.

LOAD1m R1, (R2)

opcode LOAD1m	R1	R2
9	3	3

Μήκος εντολής = 15 bits → **1 θ.μ. καταλαμβάνει η εντολή**.

Επειδή το εύρος του data bus = 48 bits μπορούμε να προσπελάσουμε **ταυτόχρονα έως και 3 διαδοχικές θέσεις μνήμης, αφού η κάθε θέση μνήμης έχει μέγεθος 16 bits**. Η προσκόμιση της εντολής (15 bits) θα γίνει **με μία προσπέλαση** (η οποία μπορεί να μεταφέρει έως 48 bits). Η εκτέλεση της, επειδή περιέχει καταχωρητή σε () δηλ. (R2), θα προσπελάσει μια θ.μ. και

θα προσκομίσει το περιεχόμενο αυτής της θ.μ. (16 bits) στον καταχωρητή R1, αφού στην εντολή υπάρχει η ένδειξη 1m και γιαυτό θα απαιτηθεί άλλη μια προσπέλαση. Άρα συνολικά για την **προσκόμιση** (fetch) και για την **εκτέλεση** (execute) της εντολής **απαιτούνται 2 προσπελάσεις**.

LOAD3m R1, (R2)

opcode LOAD 3m	R1	R2
9	3	3

Μήκος εντολής = 15 bits → 1 θ.μ. καταλαμβάνει η εντολή.

Για **προσκόμιση εντολής** → 1 προσπέλαση (κάθε προσπέλαση μεταφέρει 48 bits και εμείς έχουμε 15 bits).

Για **εκτέλεση εντολής** → 1 προσπέλαση για να προσκομιστούν 3 θ.μ. (αυτό είναι δυνατό να γίνει με μια προσπέλαση)

Άρα, σύνολο **2 προσπελάσεις**.

ADD R3, R1

opcode ADD	R3	R1
9	3	3

Μήκος εντολής = 15 bits → 1 θ.μ. καταλαμβάνει η εντολή.

Η εντολή αυτή **δεν** προσπελαύνει μνήμη.

Προσπελάσεις μνήμης = 1 (μόνο για προσκόμιση εντολής)

LOAD 1m R1, $X

opcode LOAD1m	R1	X
9	3	16

Πλήθος θ.μ.: 2 (28 bits → 32 bits)

Προσπελάσεις: 1 προσπέλαση για την προσκόμιση της εντολής (fetch) + 1 προσπέλαση (εκτέλεση) = 2 προσπελάσεις.

Θέμα Σεπτεμβρίου 2020

Θεωρήστε υπολογιστή ο οποίος διαθέτει 173 εντολές σε επίπεδο γλώσσας μηχανής, 33 καταχωρητές των 12 δυαδικών ψηφίων ο καθένας, κύρια μνήμη με 4 δυαδικά ψηφία ανά θέση μνήμης, και εύρος αρτηρίας δεδομένων 4δυαδικών ψηφίων. Θεωρήστε ότι το σύστημα της κύριας μνήμης έχει σχεδιαστεί έτσι ώστε να είναι δυνατή η ταυτόχρονη προσπέλαση (για ανάγνωση ή εγγραφή) τόσων διαδοχικών θέσεων μνήμης όσων επιτρέπει το εύρος της αρτηρίας δεδομένων. Για καθεμία από τις ακόλουθες εντολές να δώσετε το πλήθος των θέσεων μνήμης που καταλαμβάνει η εντολή (ένα αριθμό) και το πλήθος προσπελάσεων (ένα αριθμό) που απαιτούνται για την προσκόμιση και την εκτέλεση κάθε εντολής.

1. LOAD1m R1, (R2) / έμμεσος (indirect) τρόπος διευθυνσιοδότησης με χρήση καταχωρητή και προσκόμιση του περιεχομένου 1 θέση μνήμης

Πλήθος θέσεων μνήμης που καταλαμβάνει η εντολή: 5

Πλήθος προσπελάσεων που απαιτούνται για την προσκόμιση και την εκτέλεση της εντολής: 6

2. LOAD3m R1, (R2) / έμμεσος (indirect) τρόπος διευθυνσιοδότησης με χρήση καταχωρητή και προσκόμιση του περιεχομένου 3 θέσεων μνήμης

Πλήθος θέσεων μνήμης που καταλαμβάνει η εντολή: 5

Πλήθος προσπελάσεων που απαιτούνται για την προσκόμιση και την εκτέλεση της εντολής: 8

3. ADD R3, R1 / R3 <- - R3 + R1

Πλήθος θέσεων μνήμης που καταλαμβάνει η εντολή: 5 5

Πλήθος προσπελάσεων που απαιτούνται για την προσκόμιση και την εκτέλεση της εντολής:

4. LOAD1m R1, $X / κατ' ευθείαν (direct) τρόπος διευθυνσιοδότησης, το X είναι αριθμός των 16 δυαδικών ψηφίων και προσκόμιση του περιεχομένου 1 θέσης μνήμης

Πλήθος θέσεων μνήμης που καταλαμβάνει η εντολή: 8

Πλήθος προσπελάσεων που απαιτούνται για την προσκόμιση και την εκτέλεση της εντολής: 9

Λύση

Να επαληθευτεί η λύση

Άσκηση 2.10 με ευθυγράμμιση μνήμης

Θεωρούμε δύο υπολογιστές Υ1, και Υ2 οι οποίοι διαθέτουν αρτηρία δεδομένων των 32 δυαδικών ψηφίων και η μνήμη είναι οργανωμένη έτσι ώστε κάθε διεύθυνση να αντιστοιχεί σε μία ψηφιολέξη (byte), αλλά σε μια προσπέλαση να μπορούν να διαβαστούν ή να γραφούν έως και τέσσερις ψηφιολέξεις με διευθύνσεις δ, δ+1, δ+2 και δ+3, όπου το δ είναι ακέραιο πολλαπλάσιο του τέσσερα. Τόσο στον υπολογιστή Υ1 όσο και στον Υ2 οι εντολές είναι των 32 δυαδικών ψηφίων και είναι ευθυγραμμισμένες (aligned). Στον υπολογιστή Υ1 είναι και τα δεδομένα ευθυγραμμισμένα, ενώ στον Υ2 τα δεδομένα δεν είναι και' ανάγκη ευθυγραμμισμένα. Συγκεκριμένα, στον Υ2 τα δεδομένα των τεσσάρων ψηφιολέξεων δεν είναι ευθυγραμμισμένα στο 70% των περιπτώσεων, ενώ τα δεδομένα των δύο ψηφιολέξεων δεν είναι ευθυγραμμισμένα στο 40% των περιπτώσεων. Και οι δύο υπολογιστές έχουν τα ίδια σύνολα εντολών γλώσσας μηχανής. Οι εντολές διακρίνονται σε τέσσερις κατηγορίες Α, Β, Γ, και Δ τα χαρακτηριστικά των οποίων δίνονται στις πρώτες δύο στήλες του Πίνακα 2.10.1. Για κάθε προσπέλαση μνήμης απαιτούνται 4 κύκλοι ρολογιού. Να βρεθεί ο συνολικός χρόνος εκτέλεσης του προγράμματος και στον Υ1 και στον Υ2.

Πίνακας 2.10.1

Είδος εντολής	Πλήθος ψηφιολέξεων δεδομένων που προσπελαύνει η εντολή κατά την εκτέλεσή της	Πλήθος εντολών που εκτελούνται
Α	0	500
Β	ένα δεδομένο της 1 ψηφιολέξης	1000
Γ	δύο δεδομένα των 2 ψηφιολέξεων το καθένα	600
Δ	ένα δεδομένο των 4 ψηφιολέξεων	2000

Λύση

Έχουμε δύο υπολογιστές Υ1 και Υ2. Στον πρώτο έχουμε ευθυγράμμιση εντολών και δεδομένων και στο δεύτερο έχουμε ευθυγράμμιση εντολών και εν' μέρει δεδομένων. Όσον αφορά τις **εντολές** και στους δύο υπολογιστές, κάθε εντολή είναι μεγέθους 32 bits = 4 bytes = 4 θ.μ. (λόγω της οργάνωσης 1 byte/θ.μ.). Αυτό σημαίνει ότι κάθε εντολή καταλαμβάνει 4 θ.μ. Λόγω όμως της **ευθυγράμμισης μνήμης**, σε κάθε προσπέλαση μνήμης μπορούν να προσπελαστούν **έως και 4 διαδοχικές θ.μ. με διευθύνσεις δ, δ+1, δ+2, δ +3**. Αυτό σημαίνει ότι για κάθε εντολή, αρκεί μια προσπέλαση. Όσον αφορά τα δεδομένα, για τον υπολογιστή Υ1, έχουμε πλήρη ευθυγράμμιση, για τον Υ2 δεν έχουμε πλήρη ευθυγράμμιση για δεδομένα μεγέθους δύο και τεσσάρων ψηφιολέξεων. Σε αυτά τα δύο μεγέθη δεδομένων, έχουμε ευθυγράμμιση σε κάποιο ποσοστό.

Το βασικό πλεονέκτημα της ευθυγράμμισης μνήμης είναι ότι δίνει τη δυνατότητα της ταυτόχρονης προσπέλασης πολλών θ.μ. με μια προσπέλαση

T_1 (χρόνος εκτέλεσης προγράμματος στον Y_1) = [(1 εντ. * 1 πρ. + 0 δεδ.) * 500 + (1 εντ. * 1 πρ. + 1 δεδ. * 1 πρ.) * 1000 + (1 εντ. * 1 πρ. + 2 δεδ.* 1 πρ.) * 600 + (1 εντ. * 1 πρ. + 1 δεδ. * 1 πρ.) * 2000] προσπελάσεις * $\frac{4\ \text{κ.ρ.}}{\text{προσπέλαση}}$ = 33200 κ.ρ.

Κανόνας: στην περίπτωση **μη – ευθυγράμμισης μνήμης** οι προσπελάσεις είναι πάντα **κατά μια περισσότερες** σε σχέση με την περίπτωση της ευθυγράμμισης.

T_2 (χρόνος εκτέλεσης προγράμματος στον Y_2) = [(1 εντ. * 1 πρ. + 0 δεδ.) * 500 + (1 εντ. * 1 πρ. + 1 δεδ. * 1 πρ.) * 1000 + (1 εντ. * 1 πρ. + 0.60 * 2 δεδ. * **1 πρ**. + 0.40 * 2 δεδ. * **2 πρ.**) * 600 + (1 εντ. * 1 πρ. + **0.30** * 1 δεδ. * 1 πρ. + **0.70** * 1 δεδ. * **2 πρ.**) * 2000] προσπελάσεις * $\frac{4\ \text{κ.ρ.}}{\text{προσπέλαση}}$ = 40720 κ.ρ.

Συμπέρασμα: παρατηρούμε ότι στην περίπτωση πλήρους ευθυγράμμισης μνήμης (εντολών – δεδομένων) οι κύκλοι ρολογιού που δαπανώνται για την εκτέλεση ενός προγράμματος είναι λιγότεροι, επομένως έχουμε καλύτερη απόδοση. Η ευθυγράμμιση μνήμης σχετίζεται με την οργάνωση χαμηλής τάξης διαφύλλωση μνήμης.

Άσκηση με ευθυγράμμιση μνήμης

Θεωρήστε υπολογιστή ο οποίος διαθέτει αρτηρία δεδομένων των 16 δυαδικών ψηφίων και η μνήμη είναι οργανωμένη έτσι ώστε κάθε διεύθυνση να αντιστοιχεί σε μία ψηφιολέξη (byte), αλλά σε μια προσπέλαση να μπορούν να διαβαστούν ή να γραφούν έως και δύο ψηφιολέξεις με διευθύνσεις δ και δ+1, όπου το δ είναι ακέραιο πολλαπλάσιο του δύο. **Οι εντολές είναι των 32 δυαδικών ψηφίων και είναι ευθυγραμμισμένες (aligned)**. Υποστηρίζει δεδομένα μιας ψηφιολέξης, μισής λέξης (δύο ψηφιολέξεων) και λέξης (τέσσερεις ψηφιολέξεις). Οι εντολές διακρίνονται σε τέσσερις κατηγορίες Α, Β, Γ, και Δ τα χαρακτηριστικά των οποίων δίνονται στις πρώτες δύο στήλες του επόμενου Πίνακα. Θεωρήστε ότι εκτελείται ένα πρόγραμμα του οποίου τα χαρακτηριστικά δίνονται στην τρίτη στήλη του επόμενου Πίνακα. Τα δεδομένα **των τεσσάρων ψηφιολέξεων είναι ευθυγραμμισμένα** ως λέξεις στο 20% των περιπτώσεων και ως μισές λέξεις στο 50% των περιπτώσεων, ενώ στις υπόλοιπες περιπτώσεις δεν είναι καθόλου ευθυγραμμισμένα. Τα δεδομένα των δύο ψηφιολέξεων **δεν** είναι ευθυγραμμισμένα στο 25% των περιπτώσεων.

Πίνακας

Είδος εντολής	**Πλήθος ψηφιοοέξεων δεδομένων που προσπελαύνει η εντολή κατά την εκτέλεσή της**	**Πλήθος εντολών που εκτελούνται**
Α	0	500
Β	ένα δεδομένο των 2 ψηφιολέξεων	1100
Γ	ένα δεδομένο της 1 ψηφιολέξης και ένα δεδομένο των 2 ψηφιολέξεων	600
Δ	ένα δεδομένο των 4 ψηφιολέξεως	2000

Εάν για κάθε προσπέλαση του συστήματος μνήμης απαιτούνται 5 κύκλοι ρολογιού, να υπολογίσετε το πλήθος των κύκλων ρολογιού που δαπανάται για την προσπέλαση της μνήμης κατά την εκτέλεση του προαναφερθέντος προγράμματος.

Λύση

Όταν έχουμε μη-ευθυγραμμισμένα δεδομένα ο αριθμός των προσπελάσεων θα είναι κατά "1" μεγαλύτερος από τον αριθμό των προσπελάσεων των ευθυγραμμισμένων δεδομένων: T = [(1εντ. x 2 πρ. + 0) × 500 + (1εντ. x 2 πρ. + 0.75 × 1 δεδ. x 1 πρ. + 0.25 x 1 δεδ. × 2 πρ.) × 1100 + (1 εντ. x 2 πρ. + 1 δεδ. × 1 πρ. + 0.75 × 1 δεδ. x 1 πρ. + 0.25 × 1 δεδ. x 2 πρ.) × 600 + (1εντ. x 2 πρ. + (0.2 + 0.5) × 2 πρ. + 0.3 × 3 πρ.) × 2000] × 5 κύκλοι ρολογιού.

Παρατήρηση: στην περίπτωση δεδομένων των 4 ψηφιολέξεων, **αθροίζουμε** τα δύο ποσοστά ευθυγράμμισης. Επιπλέον, στην περίπτωση αυτή, χρειαζόμαστε δύο προσπελάσεις μνήμης, αφού σε κάθε προσπέλαση μπορούμε να προσπελάσουμε έως και δύο διαδοχικές θέσεις μνήμης, ενώ το δεδομένο των 4 ψηφιολέξεων καταλαμβάνει 4 διαδοχικές θέσεις μνήμης.

Άσκηση με μετατροπή αριθμού από κινητή σε σταθερή υποδιαστολή

Με τον όρο **πόλωση** εννοούμε τον αριθμό εκείνο που προστίθεται στο μικρότερο αριθμό ενός συστήματος για να τον κάνει «0» και έτσι το σύστημα να απεικονίζει μόνο θετικούς αριθμούς. Για παράδειγμα, σε ένα σύστημα αριθμών από **-7, -6, -5, ……, 5, 6, 7** αν θέλουμε όλοι οι αριθμοί να είναι **θετικοί**, θα πρέπει να θεωρήσουμε ως πόλωση την τιμή **7**, διότι προσθέτοντας την πόλωση αυτή στο -7, θα έχουμε: **0, 1, 2, 3, ….., 14** και πλέον όλοι οι αριθμοί είναι θετικοί. Σε παράσταση κινητής υποδιαστολής, **η πόλωση για απλή ακρίβεια (32 bits) είναι 127** και για **διπλή ακρίβεια** είναι 1023.

Ένας αριθμός σε **παράσταση κινητής υποδιαστολής** αποτελείται από τρία μέρη, όπου ο εκθέτης Ε δείχνει το εύρος του αριθμού, ενώ ο συντελεστής $\Sigma_κ$ δείχνει την ακρίβεια του αριθμού: π – εκθέτης (Ε) – Συντελεστής ($\Sigma_κ$). Για να **μετατρέψουμε έναν αριθμό από παράσταση κινητής υποδιαστολής σε σταθερή, πάντα ξεκινάμε από τον υπολογισμό του εκθέτη Ε**, και ελέγχουμε σε ποιο διάστημα ανήκει, βάσει των τύπων (2.8) του βιβλίου.

Παραδείγματα μετατροπής αριθμών από κινητή σε σταθερή υποδιαστολή

0 10000111 10010000000000000000000 Ζητούμενο είναι ο αντίστοιχος αριθμός σε σταθερή υποδιαστολή

Λύση

Για τη μετατροπή του σε **σταθερή υποδιαστολή**, υπολογίζουμε αρχικά τον εκθέτη Ε, οποίος είναι: $E = 1 \times 2^7 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 128 + 4 + 2 + 1 =$ **135** (λαμβάνουμε υπόψη μας μόνο τους «1»). Επειδή το $0 < E = 135 < 255$, αυτό σημαίνει ότι από τους πέντε τύπους που υπάρχουν στο (2.8), θα λάβουμε υπόψη μας τον τρίτο τύπο, επομένως ο αριθμός σε σταθερή υποδιαστολή $N = (-1)^{π} \times 2^{Ε\text{-πόλωση}} \times 1.\Sigma_κ$. Ο συντελεστής $\Sigma_κ$ επειδή υπάρχει στο κλασματικό μέρος του αριθμού, έχει τους «1» υψωμένους σε αρνητικούς εκθέτες, π.χ. $1011.11_2 = 1 \times 2^3 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} = 8 + 2 + 1 + 0.5 + 0.25 = 11.75$. Στην προκειμένη περίπτωση το $\Sigma_κ =$ 10010000…..0 $= 1 \times 2^{-1} + 1 \times 2^{-4} = 0.5 + 0.0625 = 0.5625$. Επομένως, ο $N = (-1)^{π} \times 2^{Ε\text{-πόλωση}} \times 1.\Sigma_κ = (-1)^0 \times 2^{135-127} \times 1.5625 = 2^8 \times 1.5625 = 256 \times 1.5625 = 400$.

1 11001100 11000000000000000000000 Ζητούμενο είναι ο αντίστοιχος αριθμός σε σταθερή υποδιαστολή

Λύση

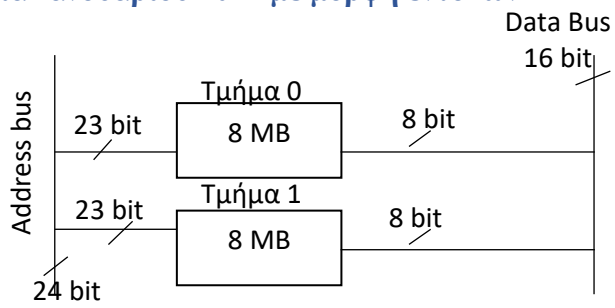
Για τη μετατροπή του σε **σταθερή υποδιαστολή**, υπολογίζουμε αρχικά τον εκθέτη Ε, οποίος είναι: $E = 2^2 + 2^3 + 2^6 + 2^7 = 4 + 8 + 64 + 128 = 204$ → $0 < E = 204 < 255$, οπότε για τη μετατροπή σε σταθερή υποδιαστολή ο τύπος είναι: $N = (-1)^{π} \times 2^{Ε\text{-πόλωση}} \times 1.\Sigma_κ = (-1)^1 \times 2^{204-127} \times 1.75 =$ **$-2^{77} \times 1.75$**, διότι το $\Sigma_κ = 1 \times 2^{-1} + 1 \times 2^{-2} = 0.5 + 0.25 = 0.75$.

0 10000010 11000000000000000000000 Ζητούμενο είναι ο αντίστοιχος αριθμός σε σταθερή υποδιαστολή

Λύση

Για τη μετατροπή του σε **σταθερή υποδιαστολή**, υπολογίζουμε αρχικά τον εκθέτη Ε, οποίος είναι: $E = 2^7 + 2^1 = 130$ → $0 < E = 130 < 255$, οπότε για τη μετατροπή σε σταθερή υποδιαστολή ο τύπος που θα χρησιμοποιηθεί είναι: $N = (-1)^{π} \times 2^{Ε\text{-πόλωση}} \times 1.\Sigma_κ = (-1)^0 \times 2^{130-127} \times 1.75 =$ **$2^3 \times 1.75 = 8 \times 1.75 = 14$**, διότι το $\Sigma_κ = 1x2^{-1} + 1x2^{-2} = 0.5 + 0.25 = 0.75$.

Θέμα Ιανουαρίου 2017 με μορφή εντολών



Δεν υπάρχει πολυπλέκτης, διότι το μέγεθος της αρτηρίας δεδομένων μεγαλώνει, από 8 bit γίνεται 16 bit. Σε κάθε προσπέλαση μνήμης μπορούμε – λόγω ευθυγράμμισης – να προσπελάσουμε με μια προσπέλαση έως και 2 διαδοχικές θέσεις μνήμης και η μεταφορά μέχρι την αρτηρία δεδομένων διαρκεί 20 κ.ρ + 1 κ.ρ. = 21 κ.ρ.

Θεωρήστε ένα ενσωματωμένο υπολογιστή ειδικού σκοπού που βασίζεται στη χρήση γενικού σκοπού. Διαθέτει 16 καταχωρητές με μήκος 32 δυαδικών ψηφίων ο καθένας και το σύνολο εντολών σε επίπεδο γλώσσας μηχανής αποτελείται από 200 εντολές, με μήκος κάθε εντολής ακέραιο πολλαπλάσιο της ψηφιολέξης (byte). Η χωρητικότητα της κύριας μνήμης είναι 16 Mbytes (είναι το μέγιστο μέγεθος που μπορεί να διευθυνσιοδοτηθεί) και κάθε θέση μνήμης είναι των 8 δυαδικών ψηφίων. Στο σύνολο των εντολών περιλαμβάνονται και οι εντολές του Πίνακα 1. Στις εντολές αυτές το U σημαίνει απρόσημο αριθμό, το S αριθμό σε αναπαράσταση συμπληρώματος ως προς 2, εάν δεν υπάρχει ούτε το U ούτε το S σημαίνει ότι χρησιμοποιείται τόσο για απρόσημα όσο και προσημασμένα δεδομένα, το B δηλώνει δεδομένα μεγέθους ψηφιολέξης (byte), το DB δηλώνει δεδομένα μεγέθους δύο ψηφιολέξεων, και το LB δηλώνει δεδομένα τεσσάρων ψηφιολέξεων, εάν δεν δηλώνεται το μέγεθος θεωρείται LB.

Η κύρια μνήμη έχει οργάνωση 2-δρόμων χαμηλής τάξης διαφύλλωσης (2-way low order interleaving) και η αρτηρία δεδομένων είναι των 16 γραμμών. Ο χρόνος προσπέλασης της κύριας μνήμης είναι 20 κύκλοι ρολογιού και για κάθε μεταφορά πληροφορίας μέσω της αρτηρίας δεδομένων απαιτείται 1 κύκλος ρολογιού.

α. Για κάθε μία από τις εντολές σε συμβολική γλώσσα του Πίνακα 1 στην αντίστοιχη στήλη να σχεδιάσετε τη μορφή των εντολών σε επίπεδο γλώσσας μηχανής (δηλαδή τα πεδία από τα οποία αποτελείται κάθε εντολή σε επίπεδο γλώσσας μηχανής) και να δώσετε το εύρος κάθε πεδίου και τη σημασία του.

Στην αντίστοιχη στήλη του Πίνακα 1 για κάθε εντολή να δώσετε το πλήθος των κύκλων ρολογιού που απαιτούνται την προσπέλαση από ή την αποθήκευση των δεδομένων στην κύρια μνήμη. Θεωρήστε ότι τα δεδομένα είναι ευθυγραμμισμένα ως λέξεις των 2 ψηφιολέξεων.

Στην αντίστοιχη στήλη του Πίνακα 2 για κάθε εντολή να δώσετε το περιεχόμενο των καταχωρητών και των σημαιών V = $C_{out} \oplus C_{out-1}$ και C_{out} μετά την εκτέλεση της εντολής. Επίσης να προσθέσετε στην τελευταία στήλη τα δεδομένα που πρέπει. Προσοχή οι εντολές δεν αποτελούν μέρος προγράμματος, είναι ανεξάρτητες και χρησιμοποιείται η αντιστρόφως ανάλογη του μεγέθους απεικόνιση (big-endian).

Στον Πίνακα 2 κατά την εκτέλεση των εντολών ADD R1, R2, R3 και ADD R4, R5, R6 για κάθε μια των περιπτώσεων βασισμένοι στις τιμές των σημαιών V και C_{out} να απαντήσετε και να αιτιολογήσετε αν το περιεχόμενο των R1, R3 είναι σωστό.

Τα περιεχόμενα των καταχωρητών R2, R3, R5 και R6 είναι απρόσημοι αριθμοί.

Τα περιεχόμενα των καταχωρητών R2, R3, R5 και R6 είναι σε αναπαράσταση συμπληρώματος.

Λύση

Πίνακας 1 Εντολές	μορφή εντολής σε επίπεδο γλώσσας μηχανής, μέγεθος και σημασία κάθε πεδίου	πλήθος κύκλων ρολογιού για την προσπέλαση από ή την αποθήκευση των δεδομένων στην κύρια μνήμη
LOAD-UB R, A /R ← A, όπου R καταχωρητής και A διεύθυνση θέση μνήμης (κατ' ευθείαν τρόπος διευθυνσιοδότησης)	op-code | R | A | X 8 | 4 | 24 | 4 5 bytes	21 κ.ρ. διότι έχουμε μια προσπέλαση κύριας μνήμης, αφού μεταφέρεται 1 byte
LOAD-SB R1, (R2) /R1 ← M(R2), R1, R2, καταχωρητές (έμμεσος (indirect) τρόπος διευθυνσιοδότησης με χρήση καταχωρητή)	opcode | R_1 | R_2 8 | 4 | 4 2 bytes	21 κ.ρ. διότι έχουμε μια προσπέλαση κύριας μνήμης, αφού μεταφέρεται 1 byte
STORE-LB R1, (A) /R1 ← M(A), R1 καταχωρητής και A διεύθυνση θέσης μνήμης (έμμεσος (indirect) τρόπος διευθυνσιοδότησης με χρήση της κύριας μνήμης)	opcode | R_1 | A | X 8 | 4 | 24 | 4 5 bytes	42 κ.ρ. διότι έχουμε διπλή προσπέλαση κύριας μνήμης
LOAD-UDB R, #0AB7$_{(16)}$ /άμεσος (immediate) τρόπος διευθυνσιοδότησης	opcode | B | #0AB7 | X 8 | 4 | 16 | 4 4 bytes	0 κ.ρ. δεν γίνεται προσπέλαση κύριας μνήμης
LOAD R1, (R2) R1← M(R2), R1, R2, καταχωρητές (έμμεσος (indirect) τρόπος διευθυνσιοδότησης με χρήση καταχωρητή)	opcode | R_1 | R_2 8 | 4 | 4 2 bytes	21 κ.ρ.
LOAD-SB R, #B7$_{(16)}$ /άμεσος (immediate) τρόπος διευθυνσιοδότησης	opcode | R | #B7 | X 8 | 4 | 8 | 4 3 bytes	0 κ.ρ. δεν γίνεται προσπέλαση κύριας μνήμης
ADD R1, R2, R3 /R1 ←R2+R3, πρόσθεσε τα περιεχόμενα των R2 και R3 και αποθήκευσε το αποτέλεσμα στον R1	opcode | R_1 | R_2 | R_3 | X 8 | 4 | 4 | 4 | 4 3 bytes	0 κ.ρ. δεν γίνεται προσπέλαση κύριας μνήμης

Πίνακας 2 Εντολές	Τιμή σημαιών και περιεχόμενο καταχωρητών πριν την εκτέλεση της εντολής	Τιμή σημαιών και περιεχόμενο καταχωρητών μετά την εκτέλεση της εντολής
LOAD-SB 02, (03) Φορτώνεται στον καταχωρητή 02 το περιεχόμενο της θέσης μνήμης που δείχνει ο καταχωρητής 03	καταχωρητής 02: F0F0F0FF καταχωρητής 03: 00000450	02 ← 51, και γίνεται λόγω επέκτασης προσήμου: 00000051
LOAD-UDB 05, #FAB7$_{(16)}$ Φορτώνεται στον καταχωρητή 05 ο αριθμός #FAB7	καταχωρητής 05: F0F0F0F0 V = 1, C_{out} = 1	05 ←FAB7, και γίνεται λόγω επέκτασης προσήμου: 0000FAB7
LOAD-UB 01, 300$_{(16)}$ Φορτώνεται στον καταχωρητή 01 το περιεχόμενο της θέσης μνήμης με διεύθυνση 300	καταχωρητής 01: F0F0F0FF V = 1, C_{out} = 0	01 ← C2, και λόγω επέκτασης προσήμου, γίνεται 000000C2
LOAD-SB 08, #A3$_{(16)}$	καταχωρητής 08: F0F0F0FF V = 1, C_{out} = 1	08 ← A3, και γίνεται λόγω επέκτασης προσήμου: FFFFFFA3
STORE-LB 04, (570)	καταχωρητής 04: 12347856 V = 0, C_{out} = 1	(570) ←04 και γίνεται (570)←12, (571) ← 34, (572)←78 και (573) ← 56. Ο 04 διατηρεί το περιεχόμενό του
LOAD-SB 06, (07)	καταχωρητής 06: F0F0F0FF καταχωρητής 07: 00000900 V = 0, C_{out} = 1	06 ← (07) = (00000900) = C4 και γίνεται λόγω επέκτασης προσήμου: 06 ←FFFFFFC4
ADD 01, 02, 03	καταχωρητής 01: E0F0F0FF καταχωρητής 02: E0000000 καταχωρητής 03: 20000000 V = 0, C_{out} = 0	01 ← 02+03 = E0000000+20000000 και γίνεται: 01←1)00000000 και οι σημαίες: V=Y=0 και το K = C_{out} =1. Αν τα περιεχόμενα των καταχωρητών θεωρηθούν απρόσημα, το αποτέλεσμα είναι λάθος.
ADD 04, 05, 06	καταχωρητής 04: F0F0F0FF καταχωρητής 05: E0000000 καταχωρητής 06: 90000000 V = 0, C_{out} = 0	04 ← 05+06 = E0000000+90000000 και γίνεται: 04← 1)00000000 και οι σημαίες: V=Y=1 και το K = C_{out} =1. Αν τα περιεχόμενα των καταχωρητών θεωρηθούν απρόσημα ή προσημασμένα, το αποτέλεσμα είναι σε κάθε περίπτωση λάθος.

κύρια μνήμη	
Διεύθυνση θέσης μνήμης στο δεκαεξαδικό	Περιεχόμενο θέσης μνήμης στο δεκαεξαδικό
000300	C2
000450	51
000451	05
000551	F3
000552	C0
000570	12
000571	34
	78
	56
000700	C4
	08
000900	C4

E0000000	1110 0000 0000 …….0000
+	
20000000	0010 0000 0000 …….0000 +
1)00000000	1)0000 0000 0000 …..0000

Από το παραπάνω άθροισμα (είτε απευθείας στο δεκαεξαεδικό ή στο δυαδικό σύστημα) προκύπτει K = C_{out} = 1 και Y = V = $\kappa_{v-1} \oplus \kappa_{v-2}$ = 1 ⊕ 1 = 0. Αν τα περιεχόμενα των καταχωρητών είναι θετικά, τότε επειδή το C_{out} = 1 έχουμε υπερχείλιση και το αποτέλεσμα είναι λάθος, γιατί δεν χωρά να αποθηκευτεί. Αν τα περιεχόμενα των καταχωρητών είναι προσημασμένα, τότε επειδή το V = 0 **δεν** έχουμε υπερχείλιση και το αποτέλεσμα είναι σωστό, γιατί χωρά να αποθηκευτεί. Με ανάλογο τρόπο υπολογίζεται και το δεύτερο άθροισμα.

Θέματα Σεπτεμβρίου 2020 - Φεβρουαρίου - Ιουνίου 2021 με εντολές αρχιτεκτονικής υπολογιστών

Θέμα 1

Η εντολή "ADD R1, R2" μπορεί να ανήκει στο σύνολο εντολών σε επίπεδο συμβολικής γλώσσας ενός επεξεργαστή με αρχιτεκτονική που βασίζεται στη χρήση μηχανισμού στοίβας.

α) Σωστό

β) Λάθος

Λύση

Η εντολή ADD R1, R2 **δεν** μπορεί να ανήκει σε αρχιτεκτονική που βασίζεται στη χρήση μηχανισμού στοίβας, διότι σε μια τέτοια περίπτωση υπάρχει απλά η εντολή ADD, χωρίς τελούμενα.

Θέμα 2

Η εντολή "LOAD R1, [R2]" μπορεί να ανήκει στο σύνολο εντολών σε επίπεδο συμβολικής γλώσσας ενός επεξεργαστή με αρχιτεκτονική που βασίζεται στη χρήση συσσωρευτή.

α) Σωστό

β) Λάθος

Λύση

Η εντολή LOAD R1, [R2] **δεν** μπορεί να ανήκει σε αρχιτεκτονική που βασίζεται στη χρήση συσσωρευτή, διότι σε μια τέτοια περίπτωση υπάρχει απλά η εντολή ADD A, με ένα μόνο τελούμενο

Θέμα 3

Η εντολή "LOAD A" όπου το A είναι διεύθυνση θέσης μνήμης, μπορεί να ανήκει στο σύνολο εντολών σε επίπεδο συμβολικής γλώσσας ενός επεξεργαστή με αρχιτεκτονική που βασίζεται στη χρήση συσσωρευτή.

α) Σωστό

β) Λάθος

Λύση

Η συγκεκριμένη εντολή ανήκει σε αρχιτεκτονική που βασίζεται στη χρήση συσσωρευτή.

Θέμα 4

Η εντολή "LOAD R1, A" όπου το A είναι διεύθυνση θέσης μνήμης, μπορεί να ανήκει στο σύνολο εντολών σε επίπεδο συμβολικής γλώσσας ενός επεξεργαστή με αρχιτεκτονική που βασίζεται στη χρήση συσσωρευτή.

α) Σωστό

β) Λάθος

Λύση

Η συγκεκριμένη εντολή δεν ανήκει σε αρχιτεκτονική που βασίζεται στη χρήση συσσωρευτή, διότι σε μια τέτοια περίπτωση υπονοείται ο Accumulator (A), οπότε η εντολή είναι της μορφής: LOAD A.

Θέμα 5

Η εντολή "ADD" μπορεί να ανήκει στο σύνολο εντολών σε επίπεδο συμβολικής γλώσσας ενός επεξεργαστή με αρχιτεκτονική που βασίζεται στη χρήση καταχωρητών γενικού σκοπού με εντολές καταχωρητή-μνήμης με ένα τελούμενο.

α) Σωστό

β) Λάθος

Λύση

Η συγκεκριμένη εντολή δεν ανήκει σε αρχιτεκτονική που βασίζεται στη χρήση καταχωρητών γενικού σκοπού με εντολές καταχωρητή μνήμης με ένα τελούμενο, διότι απλά δεν περιέχει κανένα τελούμενο. Αυτή ανήκει στην αρχιτεκτονική στοίβας.

Θέμα 6

Η εντολή "LOAD R1, A", όπου το Α είναι διεύθυνση θέσης μνήμης, μπορεί να ανήκει στο σύνολο εντολών σε επίπεδο συμβολικής γλώσσας ενός επεξεργαστή με αρχιτεκτονική που βασίζεται στη χρήση καταχωρητών γενικού σκοπού με εντολές καταχωρητή-μνήμης με τρία τελούμενα.

α) Σωστό
β) Λάθος

Λύση

Η συγκεκριμένη εντολή μπορεί ανήκει σε αρχιτεκτονική που βασίζεται στη χρήση καταχωρητών γενικού σκοπού με εντολές καταχωρητή – μνήμης με τρία τελούμενα, όπως π.χ. ADD R1, A, B, διότι αναφέρεται στο LOAD και η συγκεκριμένη εντολή μπορεί να ανήκει σε αρχιτεκτονική καταχωρητή – καταχωρητή και καταχωρητή – μνήμης είτε με δύο είτε με τρία τελούμενα. Αντίστοιχες παρατηρήσεις ισχύουν και στην περίπτωση της εντολής STORE R1, A.

Θέμα 7

Η εντολή «ADD R1, R2» μπορεί να ανήκει στο σύνολο εντολών σε επίπεδο συμβολικής γλώσσας ενός επεξεργαστή με αρχιτεκτονική που βασίζεται στη χρήση καταχωρητών γενικού σκοπού με εντολές καταχωρητή-καταχωρητή με δύο τελούμενα.

α) Σωστό
β) Λάθος

Λύση

Σωστό (διότι στη συγκεκριμένη αρχιτεκτονική οι εντολές με πράξεις όπως π.χ. ADD, SUB, MUL, DIV γίνονται μεταξύ 2 καταχωρητών)

Θέμα 8

Η εντολή «LOAD R1, A» όπου το Α είναι διεύθυνση θέσης μνήμης, μπορεί να ανήκει στο σύνολο εντολών σε επίπεδο συμβολικής γλώσσας ενός επεξεργαστή με αρχιτεκτονική που βασίζεται στη χρήση συσσωρευτή.

α) Σωστό
β) Λάθος

Λύση

Λάθος, μπορεί να ανήκει σε επεξεργαστή με **αρχιτεκτονική καταχωρητή – καταχωρητή με δύο ή τρία τελούμενα** ή σε επεξεργαστή με **αρχιτεκτονική καταχωρητή – μνήμης με δύο ή τρία τελούμενα**. Αναφορικά με τη χρήση συσσωρευτή, οι εντολές περιέχουν ένα μόνο έντελο, π.χ. LOAD A //Σ ← Α κ.ο.κ.

Θέμα 9

Η εντολή «LOAD A» όπου το Α είναι διεύθυνση θέσης μνήμης, μπορεί να ανήκει στο σύνολο εντολών σε επίπεδο συμβολικής γλώσσας ενός επεξεργαστή με αρχιτεκτονική που βασίζεται στη χρήση καταχωρητών γενικού σκοπού με εντολές καταχωρητή-καταχωρητή με τρία τελούμενα.

α) Σωστό
β) Λάθος

Λύση

Λάθος, ανήκει στην **αρχιτεκτονική συσσωρευτή**. Σε μια τέτοια αρχιτεκτονική που αναφέρεται στην εκφώνηση, η εντολή φορτώματος θα ήταν της μορφής **LOAD R1, A**, όπως και η αντίστοιχη εντολή αποθήκευσης θα ήταν της μορφής **STORE A, R1** ή **STORE R1, A** και μετά μπορούν να ακολουθούν εντολές του είδους ADD R1, R2 (2 τελούμενα) ή του είδους ADD R1, R2, R3 (3 τελούμενα).

Θέμα 10

Η εντολή «LOAD A», όπου το A είναι διεύθυνση θέσης μνήμης, μπορεί να ανήκει στο σύνολο εντολών σε επίπεδο συμβολικής γλώσσας ενός επεξεργαστή με αρχιτεκτονική που βασίζεται στη χρήση καταχωρητών γενικού σκοπού με εντολές καταχωρητή-μνήμης με ένα τελούμενο.

α) Σωστό
β) Λάθος

Λύση

Σωστό, θεωρώντας ότι ο καταχωρητής είναι ο συσσωρευτής και το τελούμενο είναι η θέση μνήμης με διεύθυνση A.

Θέμα 11

Η εντολή «LOAD R1, [R2]» μπορεί να ανήκει στο σύνολο εντολών σε επίπεδο συμβολικής γλώσσας ενός επεξεργαστή με αρχιτεκτονική που βασίζεται στη χρήση καταχωρητών γενικού σκοπού με εντολές καταχωρητή-μνήμης με τρία τελούμενα.

α) Σωστό
β) Λάθος

Λύση

Σωστό. **Κανόνας:** η αρχιτεκτονική **καταχωρητή – καταχωρητή** και **καταχωρητή - μνήμης** με **δύο** ή **τρία** τελούμενα αναφέρεται στις αριθμητικές/λογικές πράξεις που μπορούν να γίνουν και να περιέχουν δύο ή τρία τελούμενα και όχι σε εντολές **LOAD ή STORE, οι οποίες έχουν πάντα μια σταθερή (συγκεκριμένη) μορφή, για παράδειγμα LOAD R1, A ή LOAD R1, [R2] ή STORE R1, A**. Αυτές οι εντολές LOAD, STORE ταιριάζουν με οποιαδήποτε αρχιτεκτονική καταχωρητή – καταχωρητή και καταχωρητή - μνήμης με δύο ή τρία τελούμενα.

Κεφάλαιο 3 – Κατηγορίες αριθμών και εντολές μεταφοράς από/προς μνήμη

Κατηγορίες αριθμών Η/Υ

Οι όροι **προσημασμένοι (signed)** και **συμπλήρωμα ως προς 2 (2's complement)** είναι **ισοδύναμοι**.

Οι όροι **μη- προσημασμένοι (unsigned)** και **θετικοί (positive)** είναι επίσης **ισοδύναμοι**.

I. Προσημασμένοι (συμπλήρωμα ως προς 2) - Χαρακτηριστικά:

α) Η επέκταση προσήμου γίνεται **πάντα** με το MSB, π.χ. **1**101 → **1111** 1101 ή **0**1011010 → **00000000** 01011010. Η **επέκταση προσήμου χρειάζεται σε δύο περιπτώσεις**: i) όταν πρέπει να αθροίσουμε δύο δυαδικούς αριθμούς διαφορετικού μεγέθους, τότε ο **μικρότερος αριθμός παίρνει το πλήθος των bits του μεγαλύτερου**, ii) όταν χρειάζεται να γεμίσουμε έναν καταχωρητή **με μικρότερο περιεχόμενο από αυτό που περιέχει**, π.χ. εισάγουμε 8 bits/16 bits σε ένα καταχωρητή με χωρητικότητα 32 bits. Σε αυτήν την περίπτωση θα πρέπει ο αριθμός που μπαίνει στον καταχωρητή να επεκταθεί και να γίνει 32 bits, προκειμένου ο καταχωρητής να γεμίσει **πλήρως**.

β) Το **MSB** (Most Significant Bit – πιο σημαντικό δυαδικό ψηφίο) δηλώνει το **πρόσημο** του αριθμού και συγκεκριμένα:

"0": θετικός αριθμός

"1": αρνητικός αριθμός

γ) Η υπερχείλιση Y (ή V) χρησιμοποιείται για να ελέγξει την ορθότητα του αποτελέσματος μιας πράξης μεταξύ αριθμών σε συμπλήρωμα ως προς «2» και ανιχνεύεται με τον τύπο: $Y = c_{v-1} \oplus c_{v-2} = c_{out} \oplus c_{out-1} = \kappa_{v-1} \oplus \kappa_{v-2} = \kappa_v \oplus \kappa_{v-1}$, όπου τα c_{v-1}, c_{v-2}, c_{out}, c_{out-2}, κ_{v-1}, κ_{v-2} κ.λ.π. είναι τα κρατούμενα της τελευταίας και της προτελευταίας βαθμίδας της άθροισης. Πιο συγκεκριμένα:

- Αν **Y = 1** → **υπερχείλιση** και το αποτέλεσμα μιας πράξης **δεν** είναι σωστό, διότι δεν χωρά να αποθηκευτεί.
- Αν **Y = 0** → **όχι** υπερχείλιση και το αποτέλεσμα μιας πράξης είναι **σωστό**, διότι χωρά να αποθηκευτεί.

Εναλλακτικά, η ανίχνευση της υπερχείλισης μπορεί να γίνει ως εξής: Αν αθροίσουμε/αφαιρέσουμε δύο ομόσημους και προκύψει αποτέλεσμα ετερόσημο, τότε έχουμε υπερχείλιση, π.χ. αθροίζουμε 2 αρνητικούς και προκύπτει θετικός.

Μη Προσημασμένοι (θετικοί αριθμοί) – Χαρακτηριστικά:

α) Η **επέκταση προσήμου** όταν χρειάζεται να γίνει, γίνεται **πάντα** με το "0", π.χ. 1010 → 00001010 ή **0**1011010 → 00000000 **0**1011010

β) Η **υπερχείλιση** (για την ανίχνευση της ορθότητας μιας πράξης μεταξύ θετικών αριθμών) ανιχνεύεται με το **κρατούμενο** από μια πράξη, δηλαδή με το K (ή C),

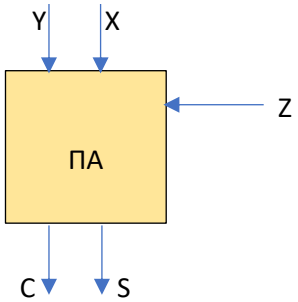
- Αν **K = 1**: **υπερχείλιση**, το αποτέλεσμα **δεν** είναι σωστό, διότι δεν χωρά να αποθηκευτεί.
- Αν **K = 0**: όχι υπερχείλιση και το αποτέλεσμα είναι **σωστό**, διότι χωρά να αποθηκευτεί.

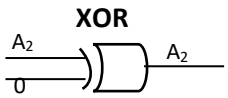
Σημείωση 1: Πολλές φορές από πράξεις που γίνονται μεταξύ δυαδικών αριθμών, θα πρέπει να υπολογίζουμε **τόσο** τη σημαία του κρατούμενου (δηλ. το K ή C), **όσο** και τη σημαία της υπερχείλισης (δηλ. το Y ή το V) καθώς και τη σημαία του μηδενικού αποτελέσματος (δηλ. το M ή Z) και στη συνέχεια, ανάλογα με το είδος των αριθμών που λαμβάνονται υπόψη (δηλ. αν οι αριθμοί είναι προσημασμένοι ή όχι) και προκειμένου να αποφανθούμε για την ορθότητα ενός αποτελέσματος, εξετάζουμε την κατάλληλη σημαία. Δηλαδή, ενώ τις υπολογίζουμε και τις τρεις, στη συνέχεια λαμβάνουμε υπόψη μόνο το Y για προσημασμένους ή μόνο το K για θετικούς. Η σημαία του μηδενικού αποτελέσματος M ενεργοποιείται μόνο στην περίπτωση που προκύψει μηδενικό αποτέλεσμα από μια πράξη.

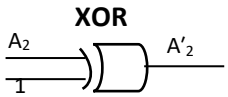
Σημείωση 2: Όταν χρειαστεί να κάνουμε **αφαίρεση μεταξύ προσημασμένων αριθμών**, η αφαίρεση αυτή θα γίνει μέσω πρόσθεσης, χρησιμοποιώντας το συμπλήρωμα ως προς «2». Πιο συγκεκριμένα, $A - B = A + B' + 1$. Η απλή αντιστροφή ενός αριθμού (B → B′) ονομάζεται συμπλήρωμα ως προς «1».

Άθροισμα - Διαφορά προσημασμένων αριθμών

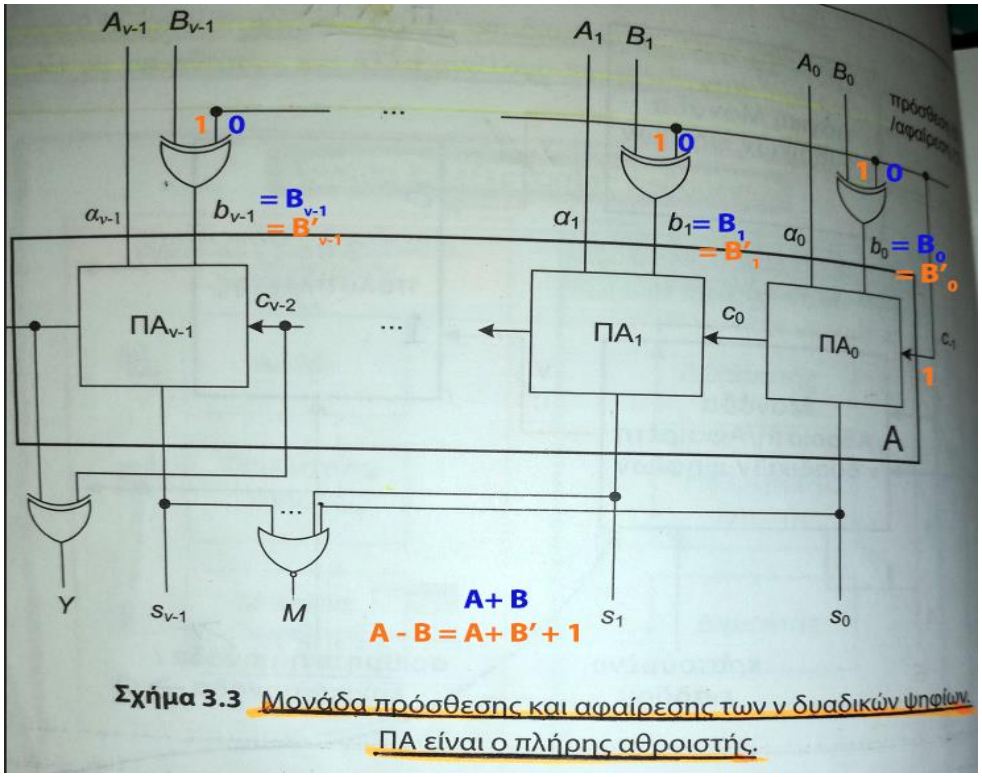
Ένας **πλήρης αθροιστής (ΠΑ)** έχει τρεις εισόδους (οι αριθμοί που αθροίζονται) και δύο εξόδους (S = sum και C = carry).







Επίσης, **μια πύλη XOR** έχει τα εξής χαρακτηριστικά:



$A = A_{v-1}\, A_{v-2} \ldots A_1 A_0$

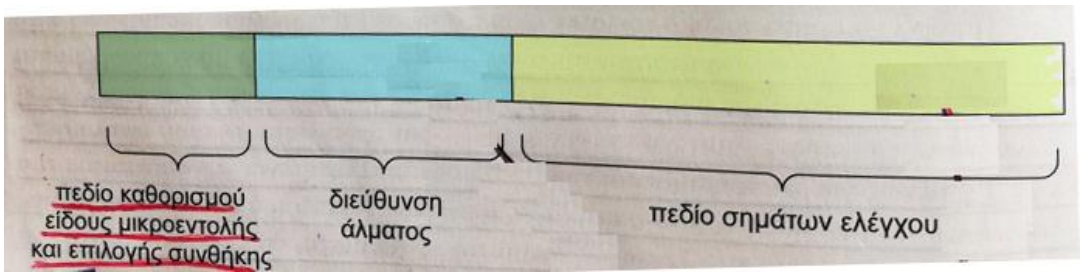
$B = B_{v-1}\, B_{v-2} \ldots B_1 B_0$

Τεχνική μικροπρογραμματισμού – Τεχνικές μείωσης χωρητικότητας μνήμης ελέγχου

- Η μονάδα ελέγχου είναι ένα κύκλωμα που βρίσκεται μέσα στην ΚΜΕ και παράγει τα σήματα ελέγχου, ανάλογα με την εντολή που εκτελείται. Τα σήματα προσδιορίζουν τη λειτουργία συγκεκριμένων εξαρτημάτων της ΚΜΕ, όπως είναι η μνήμη, η ΑΛΜ, οι καταχωρητές κ.λ.π.
- Η **μονάδα ελέγχου υλοποιείται με δύο τρόπους**: ως κλασσικό ακολουθιακό κύκλωμα που παράγει σταθερές συγκεκριμένες ακολουθίες από σήματα ελέγχου και με την τεχνική του μικροπρογραμματισμού.
- Όταν χρησιμοποιείται η τεχνική του **μικροπρογραμματισμού** για τη σχεδίαση της μονάδας ελέγχου, τότε η σχεδίαση της γίνεται πιο συστηματική και τα σήματα ελέγχου οργανώνονται σε λέξεις που έχουν μια καθορισμένη μορφή. Επιπλέον, επειδή οι τιμές αυτών των σημάτων είναι αποθηκευμένες σε μια μνήμη (μικρομνήμη), είναι εύκολο να γίνει διόρθωση σχεδιαστικών λαθών ή τροποποίηση του συνόλου (ρεπερτορίου) εντολών της μηχανής του Η/Υ.
- Αντίθετα όταν η μονάδα ελέγχου υλοποιηθεί ως **κλασσικό ακολουθιακό κύκλωμα** μπορεί να έχει πολλές γραμμές ελέγχου, γεγονός που έχει ως συνέπεια να είναι δύσκολη και δαπανηρή η σχεδίασή της και επιπλέον αν χρειαστεί να

διορθωθούν σχεδιαστικά λάθη ή να τροποποιηθεί το ρεπερτόριο εντολών (instruction set) του επεξεργαστή, θα πρέπει η μονάδα ελέγχου να σχεδιαστεί και να κατασκευαστεί εξ' αρχής.

- Σύμφωνα με την τεχνική του **μικροπρογραμματισμού**, όταν εκτελείται μια κανονική εντολή Assembly π.χ. ADD, SUB, LOAD, STORE, MUL κ.λ.π., εκτελείται αυτόματα στο παρασκήνιο ένα αντίστοιχο μικροπρόγραμμα (συλλογή με μικροεντολές) που υποστηρίζει την εκτέλεση κάθε τέτοιας κανονικής εντολές (μακροεντολή).
- Κάθε **μακροεντολή αποτελείται από ένα μικροπρόγραμμα**, που είναι μια συλλογή μικροεντολών. Αυτές οι μικροεντολές που συνθέτουν το μικροπρόγραμμα είναι αποθηκευμένες σε μια μνήμη που ονομάζεται μικρομνήμη.
- Η μορφή μιας μικροεντολής παρουσιάζεται στη συνέχεια. Αποτελείται από τρία πεδία. Ένας τρόπος για να μειώσουμε τη χωρητικότητα της μνήμης ελέγχου είναι η χρήση περισσότερων της μίας μορφών μικροεντολών. Πιο συγκεκριμένα, οι εντολές άλματος (διακλάδωσης) ελέγχουν μόνο την εσωτερική λειτουργία της μονάδας ελέγχου και δεν χρησιμοποιούν το πεδίο των σημάτων ελέγχου, ενώ οι υπόλοιπες μικροεντολές που δεν είναι εντολές άλματος δεν χρησιμοποιούν το πεδίο που ονομάζεται διεύθυνση άλματος. Επομένως, αντί για τρία πεδία που έχει η μικροεντολή, μπορούν να χρησιμοποιούνται μόνο τα δύο, ανάλογα με το είδος της μικροεντολής που εκτελείται. Το πρώτο πεδίο που καθορίζει το είδος της μικροεντολής υπάρχει πάντα.



- **Πλεονεκτήματα μικροπρογραμματισμού**: Η τεχνική του μικροπρογραμματισμού κάνει το σχεδιασμό της μονάδας ελέγχου πιο συστηματικό, οργανώνοντας τα σήματα ελέγχου σε λέξεις (μικροεντολές), οι οποίες έχουν καλά ορισμένη μορφή. Επειδή τα σήματα ελέγχου είναι πραγματοποιημένα με λογισμικό και όχι με υλικό, είναι εύκολο να γίνει διόρθωση σχεδιαστικών λαθών ή τροποποίηση του συνόλου των εντολών της μηχανής.
- **Μειονεκτήματα μικροπρογραμματισμού:** Η μικροπρογραμματισμένη μονάδα ελέγχου είναι πιο **αργή** από τη μονάδα ελέγχου που υλοποιείται ως ακολουθιακό κύκλωμα, λόγω του χρόνου που απαιτείται για την προσκόμιση των μικροεντολών από τη μνήμη ελέγχου.
- **Υπάρχουν** μικροπρογραμματισμένες και μικροπρογραμματιζόμενες μονάδες ελέγχου. Πιο ευέλικτες είναι οι δεύτερες.
- **Τρεις τεχνικές μπορούν να χρησιμοποιηθούν για τη μείωση της απαιτούμενης χωρητικότητας της μνήμης ελέγχου:**

α. Χρησιμοποίηση περισσότερων της μίας μορφών μικροεντολών

β. Οργάνωση δύο επιπέδων, νανοπρογραμματισμός

γ. Κωδικοποίηση των σημάτων ελέγχου.

- **Φάσεις προσκόμισης και εκτέλεσης:** Τα βήματα που εκτελεί η ΚΜΕ προκειμένου να εκτελέσει μια εντολή αποτελούν τον κύκλο της εντολής. Μπορούμε να διαχωρίσουμε τον κύκλο εντολής σε δύο φάσεις, τη **φάση προσκόμισης** της εντολής και τη **φάση εκτέλεσης** της εντολής. Η φάση προσκόμισης της εντολής αποτελείται από δύο βήματα του κύκλου εντολής, που είναι σταθερά και δεν εξαρτώνται από το είδος της εντολής, ενώ η φάση εκτέλεσης της εντολής αποτελείται από τα υπόλοιπα βήματα που εξαρτώνται από τη συγκεκριμένη εντολή που εκτελείται. Η μονάδα ελέγχου σε κάθε χρονική περίοδο του κύκλου εντολής παράγει ένα σύνολο σημάτων, που καλούνται **σήματα ελέγχου**.
- Κατά τη φάση προσκόμισης η μονάδα ελέγχου παράγει σήματα, που έχουν ως συνέπεια την προσκόμιση μιας εντολής.

- - Κατά τη φάση εκτέλεσης η μονάδα ελέγχου αποκωδικοποιεί μία εντολή και ανάλογα με την εντολή παράγει σήματα που κατευθύνουν τα δεδομένα στις κατάλληλες λειτουργικές μονάδες και επιλέγουν τις λειτουργίες που πρέπει να εκτελεστούν σε συγκεκριμένες χρονικές περιόδους. Διακρίνουμε λοιπόν δύο βασικές υπευθυνότητες της μονάδας ελέγχου, την επιλογή της σειράς εκτέλεσης των εντολών και την ερμηνεία των εντολών. Η επιλογή της σειράς εκτέλεσης των εντολών βασίζεται στη χρήση του μετρητή προγράμματος, ο οποίος περιέχει πάντοτε τη διεύθυνση της θέσης μνήμης που περιέχει την επόμενη προς εκτέλεση εντολή. Η λειτουργία του μετρητή προγράμματος βασίζεται στο γεγονός ότι οι εντολές ενός προγράμματος εκτελούνται συνήθως σειριακά η μία μετά την άλλη με τη σειρά που εμφανίζονται στο πρόγραμμα. Βέβαια, υπάρχουν και περιπτώσεις αλλαγής της σειράς εκτέλεσης των εντολών του προγράμματος.
- - Από τη στιγμή που η φάση προσκόμισης είναι η ίδια για όλες τις εντολές, δηλαδή σε κάθε χρονική περίοδο της φάσης προσκόμισης παράγονται σήματα ελέγχου που δεν εξαρτώνται από το είδος της εντολής που θα προσκομιστεί. Επομένως, δεν υπάρχει λόγος στο μικροπρόγραμμα που αντιστοιχεί σε κάθε εντολή να υπάρχουν και μικροεντολές για την υλοποίηση της φάσης προσκόμισης της εντολής. Η φάση προσκόμισης εντολών υλοποιείται από ένα ξεχωριστό μμικροπρόγραμμα.

Άσκηση 3.1 για άθροιση προσημασμένων αριθμών

Οι αριθμοί που ακολουθούν είναι σε αναπαράσταση **συμπληρώματος ως προς 2**.

α. 10000110
β. 11000000
γ. 01111100
δ. 01000001

Να εκτελεστούν οι πράξεις α + β, γ + δ, α + γ και β + δ σε κάθε μία από τις επόμενες περιπτώσεις, να δοθεί το αποτέλεσμα σε δυαδική και να σχολιαστεί αν το αποτέλεσμα είναι σωστό ή όχι και γιατί:

1. Για την πράξη χρησιμοποιείται αθροιστής των 8 δυαδικών ψηφίων
2. Για την πράξη χρησιμοποιείται αθροιστής των 16 δυαδικών ψηφίων

Λύση

1. Αφού οι αριθμοί είναι των **8 δυαδικών ψηφίων** και ο **αθροιστής είναι των 8 δυαδικών ψηφίων** θα εκτελέσουμε κατ' ευθείαν την πράξη. Υπενθυμίζουμε ότι στην πρόσθεση μεταξύ αριθμών σε αναπαράσταση συμπληρώματος ως προς 2, το κρατούμενο εξόδου το αγνοούμε. Η υπερχείλιση σ' αυτή την περίπτωση δίνεται από τη σχέση $Y = \kappa_{v-1} \oplus \kappa_{v-2}$.

α. 10000110

β. 11000000

1) 01000110
Από την πράξη προκύπτει ότι το K = 1, το $Y = \kappa_{v-1} \oplus \kappa_{v-2}. = 1 \oplus 0 = 1$ → Το αποτέλεσμα είναι λάθος.

γ. 01111100

δ. 01000001

10111101
Από την πράξη προκύπτει ότι το K = 0, το $Y = \kappa_{v-1} \oplus \kappa_{v-2}. = 0 \oplus 1 = 1$ → Το αποτέλεσμα είναι λάθος.

α. 10000110

γ. 01111100

1)00000010
Από την πράξη προκύπτει ότι το K = 1, το $Y = \kappa_{v-1} \oplus \kappa_{v-2}. = 1 \oplus 1 = 0$ → Το αποτέλεσμα είναι σωστό.

β. 11000000

δ. 01000001

1)00000001

Από την πράξη προκύπτει ότι το K = 1, το Y = $\kappa_{v-1} \oplus \kappa_{v-2}$. = 1 ⊕ 1 = 0 → Το αποτέλεσμα είναι σωστό.

2. Αφού οι αριθμοί είναι των **8 δυαδικών ψηφίων** και ο **αθροιστής είναι των 16 δυαδικών ψηφίων** θα πρέπει στην περίπτωση αυτή να κάνουμε επέκταση προσήμου, επεκτείνοντας το αριστερότερο bit τόσες θέσεις προς τα αριστερά, όσες χρειάζεται (εδώ 8 θέσεις)

α. 1111111110000110

β. 1111111111000000

1)1111111101000110

Από την πράξη προκύπτει ότι το K = 1, το Y = $\kappa_{v-1} \oplus \kappa_{v-2}$. = 1 ⊕ 1 = 0 → Το αποτέλεσμα είναι σωστό.

γ. 0000000001111100

δ. 0000000001000001

0000000010111101

Από την πράξη προκύπτει ότι το K = 0, το Y = $\kappa_{v-1} \oplus \kappa_{v-2}$. = 0 ⊕ 0 = 0 → Το αποτέλεσμα είναι σωστό.

α. 1111111110000110

γ. 0000000001111100

1)0000000000000010

Από την πράξη προκύπτει ότι το K = 1, το Y = $\kappa_{v-1} \oplus \kappa_{v-2}$. = 1 ⊕ 1 = 0 → Το αποτέλεσμα είναι σωστό.

β. 1111111111000000

δ. 0000000001000001

1)0000000000000001

Από την πράξη προκύπτει ότι το K = 1, το Y = $\kappa_{v-1} \oplus \kappa_{v-2}$. = 1 ⊕ 1 = 0 → Το αποτέλεσμα είναι σωστό.

Άσκηση 3.2 για άθροιση προσημασμένων αριθμών

Οι αριθμοί που ακολουθούν είναι σε αναπαράσταση **συμπληρώματος ως προς 2**.

Ένας επεξεργαστής υποστηρίζει τόσο θετικά όσο και προσημασμένα ακέραια αριθμητικά δεδομένα μεγέθους μιας ψηφιολέξης (byte), μισής λέξης (δύο ψηφιολέξεις) και λέξης (τέσσερις ψηφιολέξεις). Όλες οι αριθμητικές πράξεις γίνονται με μία μονάδα των ν=32 δυαδικών ψηφίων. Ο καταχωρητής κατάστασης μεταξύ των άλλων διαθέτει τις σημαίες K (κρατούμενο εξόδου), Y (υπερχείλιση, $Y := \kappa_v \oplus \kappa_{v-1}$) και M (μηδενικό αποτέλεσμα). Να δείξετε πως εκτελείται στην αριθμητική μονάδα του υπολογιστή κάθε μια από τις κάτωθι πράξεις, ποιες είναι οι τιμές που λαμβάνουν οι σημαίες K, Y και M σε κάθε περίπτωση και βασιζόμενοι στις τιμές των σημαιών να αποφανθείτε αν το αποτέλεσμα είναι σωστό.

Δεδομένα χωρίς πρόσημο στο δυαδικό:

μισής λέξης α = 1111111110000000

μιας λέξης β = 11111111111111111000000000000000

γ = 10000000000000000 00000000000000

δ = 01110000000000000000000000000000

ε = 00010000000000000000000000000000

Δεδομένα σε μορφή συμπληρώματος ως προς 2:

μισής λέξης ζ = 1111111110000000

μιας λέξης η = 11111111111111111000000000000000

θ = 10000000000000000000000000000000

1η πράξη: α + β, 2η πράξη: β + γ, 3η πράξη: δ + ε, 4η πράξη ζ + η και 5η πράξη: η + θ.

Λύση

1η πράξη: α + β

Αφού οι αριθμοί είναι χωρίς πρόσημο ο α μετά την επέκταση περιοχής (προσήμου) θα γίνει:

α = 00000000000000001111111110000000, οπότε:

α 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0

β + 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

απ 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0

K = 1, Y = 0, M = 0

Αφού οι αριθμοί **είναι χωρίς πρόσημο** η τιμή του κρατούμενου δηλώνει την ύπαρξη υπερχείλισης. Επειδή **K = 1** έχουμε υπερχείλιση, επομένως το αποτέλεσμα **δεν** χωράει σε 32 δυαδικά ψηφία.

2η πράξη: β + γ

β 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

γ + 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

απ 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

K = 1, Y = 1, M = 0

Αφού οι αριθμοί είναι χωρίς πρόσημο η τιμή του κρατούμενου δηλώνει την ύπαρξη υπερχείλισης. Επειδή K = 1 έχουμε υπερχείλιση, επομένως το αποτέλεσμα **δεν** χωράει σε 32 δυαδικά ψηφία.

3η πράξη: δ + ε

δ 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ε + 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

απ 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

K = 0, Y = 1, M = 0

Αφού οι αριθμοί είναι χωρίς πρόσημο η τιμή του κρατούμενου δηλώνει την ύπαρξη υπερχείλισης. Επειδή K = 0 δεν έχουμε υπερχείλιση, επομένως το αποτέλεσμα είναι σωστό.

4η πράξη: ζ + η

Αφού οι αριθμοί είναι σε αναπαράσταση συμπληρώματος ως προς 2, ο ζ μετά την επέκταση προσήμου θα γίνει:

ζ = 11111111111111111111111110000000, οπότε:

ζ 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0

η + 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

απ 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0

K = 1, Y = 0, M = 0

Αφού οι αριθμοί είναι σε αναπαράσταση συμπληρώματος ως προς 2 η τιμή της σημαίας Y δηλώνει την ύπαρξη υπερχείλισης. Επειδή Y = 0 **δεν** έχουμε υπερχείλιση, επομένως το αποτέλεσμα χωράει στα 32 δυαδικά ψηφία, το κρατούμενο εξόδου αγνοείται.

5η πράξη: η + θ

η 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

θ + 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

απ 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

K = 1, Y = 1, M = 0

Αφού οι αριθμοί είναι σε αναπαράσταση συμπληρώματος ως προς 2 η τιμή της σημαίας Y δηλώνει την ύπαρξη υπερχείλισης. Επειδή Y = 1 έχουμε υπερχείλιση, επομένως το αποτέλεσμα δε χωράει σε 32 δυαδικά ψηφία.

Θέμα 2013 με πράξεις προσημασμένων/μη προσημασμένων αριθμών

Ένας επεξεργαστής υποστηρίζει τόσο θετικά όσο και προσημασμένα ακέραια αριθμητικά δεδομένα μεγέθους μιας ψηφιολέξης (byte). Όλες οι αριθμητικές πράξεις γίνονται σε δυαδικό αθροιστή των 8 δυαδικών ψηφίων. Ο καταχωρητής κατάστασης μεταξύ των άλλων διαθέτει τις σημαίες K (κρατούμενο εξόδου), Y (υπερχείλιση, $Y = \kappa_8 \oplus \kappa_7$) και M (μηδενικό αποτέλεσμα). Να δείξετε πως εκτελούνται στην αριθμητική μονάδα του υπολογιστή οι πράξεις α + β και α + γ με α = 11100000, β = 00100000 και γ = 10000000 για κάθε μία από τις κάτωθι περιπτώσεις. Να δώσετε τις τιμές που λαμβάνουν οι σημαίες K, Y και M σε κάθε περίπτωση και βασιζόμενοι στις τιμές των σημαιών να αποφανθείτε αν το αποτέλεσμα είναι σωστό.

i. Τα α, β και γ είναι αριθμοί χωρίς πρόσημο.

ii. Τα α, β και γ είναι σε παράσταση συμπληρώματος ως προς 2.

Λύση

α = 11100000 **K = 1, M = 1, Y = 0**

β = 00100000 +

1)00000000

α = 11100000 **K = 1, M = 0, Y = 1**

γ = 10000000 +

1)01100000

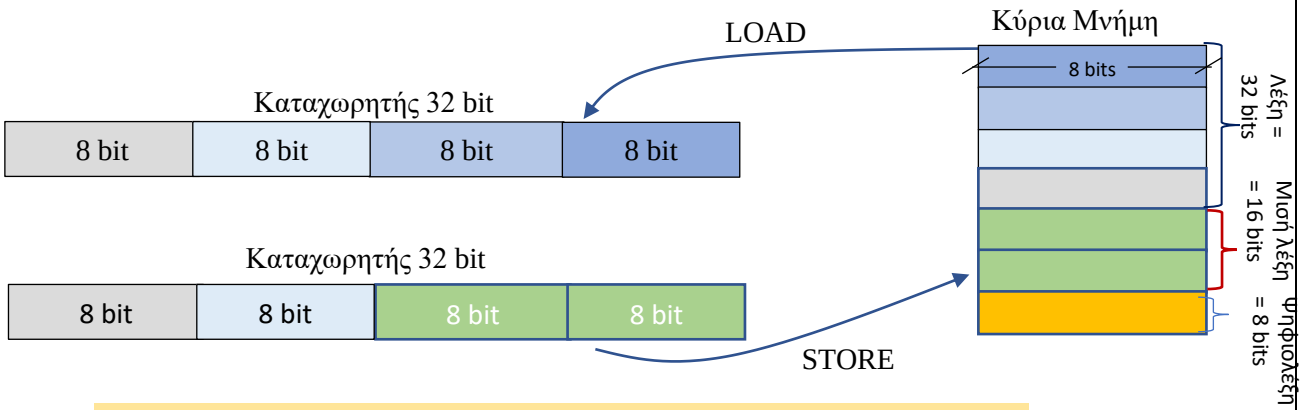
i. Επειδή οι αριθμοί είναι **χωρίς πρόσημο**, κοιτάμε τη σημαία (flag) του K για να αποφανθούμε για την ορθότητα της πράξης. Αφού το K = 1, η πράξη είναι λάθος και στις δύο αθροίσεις.

ii. Επειδή οι αριθμοί είναι **με πρόσημο**, κοιτάμε τη σημαία (flag) του Y για να αποφανθούμε για την ορθότητα της πράξης. Αφού το Y = 0, η πρώτη πράξη είναι σωστή, ενώ η δεύτερη είναι λάθος.

Επεξηγήσεις για εντολές LOAD/STORE

LOAD

Όταν μεταφέρουμε **πληροφορία (δεδομένα) από τη μνήμη σε κάποιον από τους καταχωρητές γενικού σκοπού (δηλαδή όταν εκτελούμε την εντολή LOAD),** έχει σημασία αν χρειάζεται να γίνει επέκταση προσήμου ή όχι στον καταχωρητή γενικού σκοπού. Αν δεν χρειάζεται να γίνει επέκταση προσήμου, π.χ. στην περίπτωση μεταφοράς (φόρτωσης) μιας **λέξης** (32 bit) δεδομένων σε ένα 32 – bit καταχωρητή, τότε θα έχουμε **ίδιες εντολές μεταφοράς δεδομένων και για προσημασμένους και για μη – προσημασμένους (θετικούς) αριθμούς.** Σε διαφορετική περίπτωση, όταν π.χ. μεταφέρουμε μια ψηφιολέξη ή μια μισή λέξη δεδομένων από τη μνήμη σε ένα 32 – bit καταχωρητή, απαιτούνται **διαφορετικές εντολές μεταφοράς δεδομένων, γιατί τότε γίνεται επέκταση προσήμου και η επέκταση αυτή λαμβάνει χώρα με διαφορετικό τρόπο στους προσημασμένους και στους μη – προσημασμένους (θετικούς) αριθμούς. Στην πρώτη περίπτωση γίνεται επέκταση προσήμου με βάση το πιο σημαντικό ψηφίο (MSB), ενώ στη δεύτερη περίπτωση γίνεται επέκταση μόνο με μηδενικά.**



Συμπέρασμα: Όταν γεμίζει **πλήρως** ένας καταχωρητής, τότε **δεν** γίνεται καμία επέκταση προσήμου. Αυτό συμβαίνει όταν φορτώνουμε μια λέξη σε έναν καταχωρητή των 32 bits. Αντίθετα, όταν ένας καταχωρητής γεμίζει **εν' μέρει (με μισή λέξη ή με ψηφιολέξη),** αυτό έχει ως αποτέλεσμα να γίνεται επέκταση προσήμου και επομένως οι εντολές που πρέπει να γίνουν για να φορτωθεί ένας καταχωρητές πλήρως, είναι **διαφορετικές** μεταξύ τους, ανάλογα με το αν φορτώνουμε προσημασμένα ή θετικά δεδομένα.

STORE

Όταν μεταφέρουμε **πληροφορία (δεδομένα) από κάποιον από τους καταχωρητές γενικού σκοπού στη μνήμη** (δηλαδή όταν εκτελούμε την εντολή **STORE**), **δεν γίνεται επέκταση προσήμου, διότι στην περίπτωση αυτή η αποθήκευση γίνεται στην κύρια μνήμη** και εκεί δεν γίνεται επέκταση προσήμου. **Το κριτήριο ομαδοποίησης των εντολών (δηλαδή όμοιες εντολές) είναι το μέγεθος των δεδομένων που αποθηκεύονται και όχι το είδος τους**. Αυτό σημαίνει ότι όταν μεταφέρουμε από κάποιον καταχωρητή γενικού σκοπού στην κύρια μνήμη προσημασμένα και θετικά (μη – προσημασμένα) δεδομένα μεγέθους **μιας ψηφιολέξης, τότε θα έχουμε ίδιες εντολές**. Ομοίως, όταν μεταφέρουμε από κάποιον καταχωρητή γενικού σκοπού στην κύρια μνήμη προσημασμένα και θετικά (μη – προσημασμένα) δεδομένα μεγέθους **μισής λέξης, τότε θα έχουμε επίσης τις ίδιες εντολές**. Ομοίως, όταν μεταφέρουμε από κάποιον καταχωρητή γενικού σκοπού στην κύρια μνήμη προσημασμένα και θετικά (μη – προσημασμένα) δεδομένα μεγέθους μιας **λέξης, τότε θα έχουμε επίσης ίδιες εντολές**.

Συμπέρασμα: Στην εντολή STORE, το κριτήριο είναι ότι όταν μεταφέρουμε τμήματα μνήμης από κάποιον καταχωρητή προς την Κύρια Μνήμη, έχει σημασία το **μέγεθος των δεδομένων** που αποθηκεύονται και όχι το είδος τους.

Προσοχή! Όταν μεταφέρουμε από κάποιον καταχωρητή γενικού σκοπού στη μνήμη (**STORE**) **προσημασμένα** δεδομένα μεγέθους **μιας ψηφιολέξης**, **μισής λέξης** ή **μιας λέξης**, θα έχουμε **διαφορετικές** εντολές, γιατί στην πρώτη περίπτωση η αποθήκευση γίνεται σε **μία θέση μνήμης**, στη δεύτερη περίπτωση η αποθήκευση γίνεται **σε δύο θέσεις μνήμης** και στην τρίτη περίπτωση η αποθήκευση γίνεται σε **τέσσερις θέσεις μνήμης**. **Αντίστοιχα ισχύουν όταν αποθηκεύουμε και θετικά δεδομένα από καταχωρητή γενικού σκοπού στη μνήμη.**

Προσοχή! Όταν μεταφέρουμε προσημασμένα δεδομένα μεγέθους μιας ψηφιολέξης, μισής λέξης ή μιας λέξης **από τη μνήμη σε κάποιον από τους καταχωρητές γενικού σκοπού** (**LOAD**), τότε θα έχουμε **διαφορετικές εντολές μεταφοράς δεδομένων, διότι η επέκταση προσήμου θα γίνει μεν με τον ίδιο τρόπο, αλλά με διαφορετικό αριθμό από δυαδικά ψηφία κάθε φορά. Πιο συγκεκριμένα, όταν φορτώνουμε προσημασμένα δεδομένα μεγέθους μιας ψηφιολέξης (δηλαδή 8 bits), η επέκταση προσήμου θα γίνει με 24 bits (με βάση το MSB), με δεδομένο βέβαια ότι διαθέτουμε καταχωρητές των 32 bits. Στην περίπτωση που φορτώνουμε προσημασμένα δεδομένα μεγέθους μισής λέξης (δηλαδή 16 bits), η επέκταση προσήμου θα γίνει με 16 bits (με βάση το MSB), με δεδομένο βέβαια ότι διαθέτουμε καταχωρητές των 32 bits. Όταν φορτώνουμε προσημασμένα δεδομένα μεγέθους μιας λέξης (δηλαδή 32 bits) τότε δεν θα γίνει καμία επέκταση προσήμου, με δεδομένο βέβαια ότι διαθέτουμε καταχωρητές των 32 bits. Αντίστοιχα ισχύουν όταν φορτώνουμε θετικά δεδομένα μεγέθους μιας ψηφιολέξης (τότε θα έχουμε επέκταση προσήμου με 24 «0») κ.ο.κ.**

U = Unsigned ≡ P = Positive και **S = Signed**.

Οι εντολές **LOADBP** (ή LOADBU) και **LOADBS** όταν χρησιμοποιούνται για να γεμίσουν έναν καταχωρητή 32 – bit, **δεν** είναι ισοδύναμες (ίδιες), λόγω της αναμενόμενης επέκτασης προσήμου, που θα λάβει χώρα.

Οι εντολές **LOADHP** (ή LOADHU) και **LOADHS** όταν χρησιμοποιούνται για να γεμίσουν έναν καταχωρητή 32 – bit, **δεν** είναι ισοδύναμες (είναι διαφορετικές), λόγω της αναμενόμενης επέκτασης προσήμου, που θα λάβει χώρα.

Οι εντολές **LOADWP** (ή LOADWU) και **LOADWS** όταν χρησιμοποιούνται για να γεμίσουν έναν καταχωρητή 32 – bit, είναι **ισοδύναμες**, διότι **δεν** θα γίνει επέκταση προσήμου.

Οι εντολές **STOREBP** (ή STOREBU) και **STOREBS** όταν χρησιμοποιούνται για να γεμίσουν τη μνήμη, είναι **ισοδύναμες**, επειδή καταλαμβάνουν το ίδιο πλήθος θ.μ.

Οι εντολές **STOREHP** (ή STOREHU) και **STOREHS** όταν χρησιμοποιούνται για να γεμίσουν τη μνήμη, είναι **ισοδύναμες**, επειδή καταλαμβάνουν το ίδιο πλήθος θ.μ.

Οι εντολές **STOREWP** (ή STOREWU) και **STOREWS** όταν χρησιμοποιούνται για να γεμίσουν τη μνήμη, είναι **ισοδύναμες**, επειδή καταλαμβάνουν το ίδιο πλήθος θ.μ.

Θέμα Φεβρουαρίου 2016 με πράξεις προσημασμένων/μη προσημασμένων αριθμών

Θέμα 1 με θύρες ανάγνωσης/εγγραφής

Σε ένα επεξεργαστή που βασίζεται στη χρήση καταχωρητών γενικού σκοπού με εντολές καταχωρητή-καταχωρητή με δύο τελούμενα, οι καταχωρητές γενικού σκοπού (register file) έχουν

α) Τρεις πόρτες ανάγνωσης
β) Δύο πόρτες ανάγνωσης και μία πόρτα εγγραφής
γ) Μία πόρτα ανάγνωσης και μία πόρτα εγγραφής
δ) Μία πόρτα ανάγνωσης και δύο πόρτες εγγραφής
ε) Δύο πόρτες ανάγνωσης
ζ) Δύο πόρτες ανάγνωσης και δύο πόρτες εγγραφής
η) Μία πόρτα ανάγνωσης

Λύση

Η σωστή απάντηση είναι η "β", είτε έχουμε δύο τελούμενα (π.χ. ADD R1, R2) είτε τρία τελούμενα (π.χ. ADD R1, R2, R3) οι σύγχρονοι επεξεργαστές διαθέτουν καταχωρητές με **δύο θύρες ανάγνωση και μια θύρα εγγραφής** και αυτό αιτιολογείται από τη **δυνατότητα** εκτέλεσης εντολών των προαναφερόμενων ειδών. Ο καταχωρητής στον οποίο εγγράφεται κάθε φορά το αποτέλεσμα μιας πράξης τοποθετείται στη θύρα εγγραφής.

Θέμα 2 με υπερχείλιση/κρατούμενο

Θεωρήστε αριθμούς των κ δυαδικών ψηφίων. Ποια/ες από τις παρακάτω απαντήσεις είναι σωστή/ες; ($Y = C_{out} \oplus C_{out-1}$, C_{out}: κρατούμενο εξόδου)

α) Το αποτέλεσμα της άθροισης δύο αριθμών σε παράσταση συμπληρώματος ως προς 2 δε χωράει σε κ δυαδικά ψηφία όταν $Y = 0$ και C_{out} =

β) Το αποτέλεσμα της άθροισης δύο αριθμών χωρίς πρόσημο δε χωράει σε κ δυαδικά ψηφία όταν $Y = 1$ και $C_{out} = 0$

γ) Το αποτέλεσμα της άθροισης δύο αριθμών σε παράσταση συμπληρώματος ως προς 2 δε χωράει σε κ δυαδικά ψηφία όταν $Y = 1$ και $C_{out} = 0$

δ) Το αποτέλεσμα της άθροισης δύο αριθμών χωρίς πρόσημο δε χωράει σε κ δυαδικά ψηφία όταν $Y = 0$ και $C_{out} = 1$

ε) Το αποτέλεσμα της άθροισης δύο αριθμών χωρίς πρόσημο δε χωράει σε κ δυαδικά ψηφία όταν $Y = 0$ και $C_{out} = 0$

ζ) Το αποτέλεσμα της άθροισης δύο αριθμών σε παράσταση συμπληρώματος ως προς 2 δε χωράει σε κ δυαδικά ψηφία όταν $Y = 0$ και $C_{out} = 1$

Λύση

Η σωστή απάντηση είναι "γ", "δ", η υπερχείλιση ελέγχεται με το κρατούμενο C_{out} στους μη – προσημασμένους (θετικούς) αριθμούς και με την υπερχείλιση Y στους προσημασμένους (συμπλήρωμα ως προς 2) αριθμούς.

Θέμα 3 με καταχωρητές γενικού σκοπού

Σε ένα επεξεργαστή για λόγους απόδοσης συμφέρει

α) να έχει δύο καταχωρητές γενικού σκοπού
β) να μην έχει κανένα καταχωρητή γενικού σκοπού
γ) να έχει ένα καταχωρητή γενικού σκοπού
δ) να έχει πολλούς καταχωρητές γενικού σκοπού

Λύση

Η σωστή απάντηση είναι η "δ", διότι αφενός το κόστος των καταχωρητών έχει μειωθεί σε σημαντικό βαθμό και αφετέρου οι καταχωρητές είναι πολύ πιο γρήγοροι από την κύρια μνήμη.

Θέματα Ιουνίου 2021 με πράξεις προσημασμένων/μη προσημασμένων αριθμών

Θέμα 1

Θεωρήστε ότι εκτελείται η πράξη **A-B** μεταξύ **προσημασμένων** αριθμών $A=0a_7a_6a_5a_4a_3a_2a_1a_0$ και $B=0b_7b_6b_5b_4b_3b_2b_1b_0$ οι οποίοι είναι και οι δύο σε αναπαράσταση συμπληρώματος ως προς 1 ή προσημασμένου μεγέθους.

Σημειώστε ποια από τις ακόλουθες προτάσεις είναι σωστή.

α) Είναι δυνατόν να προκύψει υπερχείλιση

β) Δεν είναι δυνατόν να προκύψει υπερχείλιση

γ) Εξαρτάται από τον τρόπο που χρησιμοποιείται για την αναπαράσταση των αριθμών

Λύση

Όταν ο αριθμοί που λαμβάνουν χώρα σε πράξεις είναι **προσημασμένοι**, τότε για να προκύψει **υπερχείλιση** ανάμεσά τους πρέπει να είναι **ομόσημοι**, κάτι που ισχύει, αφού οι αριθμοί Α και Β που λαμβάνουν μέρος στην πράξη Α – Β είναι και οι δύο θετικοί. Επομένως, **είναι δυνατόν** να προκύψει υπερχείλιση από την εκτέλεση της συγκεκριμένης πράξης.

Θέμα 2

Θεωρήστε ότι εκτελείται η πράξη **A-B** μεταξύ των προσημασμένων αριθμών $A=1a_7a_6a_5a_4a_3a_2a_1a_0$ και $B=1b_7b_6b_5b_4b_3b_2b_1b_0$ οι οποίοι είναι και οι δύο σε αναπαράσταση συμπληρώματος ως προς 1 ή προσημασμένου μεγέθους.

Σημειώστε ποια από τις ακόλουθες προτάσεις είναι σωστή.

α) Είναι δυνατόν να προκύψει υπερχείλιση

β) Δεν είναι δυνατόν να προκύψει υπερχείλιση

γ) Εξαρτάται από τον τρόπο που χρησιμοποιείται για την αναπαράσταση των αριθμών

Λύση

Ισχύει, ότι προαναφέρθηκε, με αποτέλεσμα η σωστή απάντηση να είναι η πρώτη.

Θέμα 3

Θεωρήστε ότι εκτελείται η πράξη Α-Β μεταξύ **προσημασμένων** αριθμών $A=0a_7a_6a_5a_4a_3a_2a_1a_0$ και $B=1b_7b_6b_5b_4b_3b_2b_1b_0$ οι οποίοι είναι και οι δύο σε αναπαράσταση συμπληρώματος ως προς 1 ή προσημασμένου μεγέθους.

Σημειώστε ποια από τις ακόλουθες προτάσεις είναι σωστή.

α) Είναι δυνατόν να προκύψει υπερχείλιση

β) Δεν είναι δυνατόν να προκύψει υπερχείλιση

γ) Εξαρτάται από τον τρόπο που χρησιμοποιείται για την αναπαράσταση των αριθμών

Λύση

Η σωστή απάντηση είναι η δεύτερη. Στην **περίπτωση αυτή οι αριθμοί Α, Β που λαμβάνουν χώρα στην πράξη Α – Β** είναι ετερόσημοι, δεν **μπορεί** να προκύψει υπερχείλιση ανάμεσά τους.

Θέματα Ιουνίου 2021 με εντολές LOAD/STORE

Θέμα 1

Το σύνολο εντολών, σε επίπεδο γλώσσας μηχανής, ενός επεξεργαστή απλού συνόλου εντολών υποστηρίζει τόσο μη προσημασμένα (θετικά) όσο και προσημασμένα (σε αναπαράσταση συμπληρώματος ως προς 2) ακέραια αριθμητικά δεδομένα μεγέθους μιας ψηφιολέξης (byte), μισής λέξης (δύο ψηφιολέξεις), και λέξης (τέσσερις ψηφιολέξεις). Ο επεξεργαστής αυτός διαθέτει **καταχωρητές των 32 δυαδικών ψηφίων** και Αριθμητική Λογική Μονάδα των 32 δυαδικών ψηφίων. Η κύρια μνήμη έχει οργάνωση **μιας ψηφιολέξης ανά θέση μνήμης**. Να απαντήσετε εάν είναι σωστή ή λάθος η ακόλουθη πρόταση:

Το σύνολο εντολών του επεξεργαστή σε επίπεδο γλώσσας μηχανής πρέπει να διαθέτει **διαφορετικές** εντολές μεταφοράς πληροφορίας **από τους καταχωρητές γενικού σκοπού στη μνήμη** για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. μη προσημασμένα δεδομένα μεγέθους μιας λέξης, 2. προσημασμένα δεδομένα μεγέθους μιας λέξης.

α) Σωστό
β) Λάθος

Λύση

Από τη στιγμή που έχουμε εντολή **STORE, το κριτήριο του να έχουμε ίδιες εντολές είναι το μέγεθος** και όχι το είδος των δεδομένων. Από τη στιγμή που έχουμε ίδιο μέγεθος (λέξη) πρέπει να έχουμε και ίδια εντολή.

Θέμα 2

Το σύνολο εντολών, σε επίπεδο γλώσσας μηχανής, ενός επεξεργαστή απλού συνόλου εντολών υποστηρίζει τόσο μη προσημασμένα (θετικά) όσο και προσημασμένα (σε αναπαράσταση συμπληρώματος ως προς 2) ακέραια αριθμητικά δεδομένα μεγέθους μιας ψηφιολέξης (byte), μισής λέξης (δύο ψηφιολέξεις), και λέξης (τέσσερις ψηφιολέξεις). Ο επεξεργαστής αυτός διαθέτει καταχωρητές των 32 δυαδικών ψηφίων και Αριθμητική Λογική Μονάδα των 32 δυαδικών ψηφίων. Η κύρια μνήμη έχει οργάνωση μιας ψηφιολέξης ανά θέση μνήμης. Να απαντήσετε εάν είναι σωστή ή λάθος η ακόλουθη πρόταση:

Μπορεί να χρησιμοποιεί την **ίδια εντολή**, σε επίπεδο γλώσσας μηχανής, για τη μεταφορά πληροφορίας **από τους καταχωρητές γενικού σκοπού στη μνήμη** για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. μη προσημασμένα δεδομένα μεγέθους μιας λέξης, 2. προσημασμένα δεδομένα μεγέθους μιας λέξης.

α) Σωστό
β) Λάθος

Λύση

Από τη στιγμή που έχουμε εντολή **STORE το κριτήριο του να έχουμε ίδιες εντολές είναι το μέγεθος** και όχι το είδος των δεδομένων. Από τη στιγμή που έχουμε ίδιο μέγεθος (λέξη) πρέπει να έχουμε και ίδια εντολή και αφού έχουμε, η απάντηση είναι Σωστή.

Θέμα 3

Το σύνολο εντολών, σε επίπεδο γλώσσας μηχανής, ενός επεξεργαστή απλού συνόλου εντολών υποστηρίζει τόσο μη προσημασμένα (θετικά) όσο και προσημασμένα (σε αναπαράσταση συμπληρώματος ως προς 2) ακέραια αριθμητικά δεδομένα μεγέθους μιας ψηφιολέξης (byte), μισής λέξης (δύο ψηφιολέξεις), και λέξης (τέσσερις ψηφιολέξεις). Ο επεξεργαστής αυτός διαθέτει καταχωρητές των 32 δυαδικών ψηφίων και Αριθμητική Λογική Μονάδα των 32 δυαδικών ψηφίων. Η κύρια μνήμη έχει οργάνωση μιας ψηφιολέξης ανά θέση μνήμης. Να απαντήσετε εάν είναι σωστή ή λάθος η ακόλουθη πρόταση:

Το σύνολο εντολών του επεξεργαστή σε επίπεδο γλώσσας μηχανής πρέπει να διαθέτει **διαφορετικές** εντολές μεταφοράς πληροφορίας **από τη μνήμη σε κάποιο από τους καταχωρητές γενικού σκοπού** για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. μη προσημασμένα δεδομένα μεγέθους μισής λέξης, 2. προσημασμένα δεδομένα μεγέθους μισής λέξης.

α) Σωστό
β) Λάθος

Λύση

Από τη στιγμή που έχουμε εντολή **LOAD το κριτήριο του να έχουμε ίδιες εντολές ή** όχι είναι το είδος και το αν γεμίζει πλήρως ο καταχωρητής ή όχι.

Θέμα 4

Το σύνολο εντολών, σε επίπεδο γλώσσας μηχανής, ενός επεξεργαστή απλού συνόλου εντολών υποστηρίζει τόσο μη προσημασμένα (θετικά) όσο και προσημασμένα (σε αναπαράσταση συμπληρώματος ως προς 2) ακέραια αριθμητικά δεδομένα μεγέθους μιας ψηφιολέξης (byte), μισής λέξης (δύο ψηφιολέξεις), και λέξης (τέσσερις ψηφιολέξεις). Ο επεξεργαστής αυτός διαθέτει καταχωρητές των 32 δυαδικών ψηφίων και Αριθμητική Λογική Μονάδα των 32 δυαδικών ψηφίων. Η κύρια μνήμη έχει οργάνωση μιας ψηφιολέξης ανά θέση μνήμης. Να απαντήσετε εάν είναι σωστή ή λάθος η ακόλουθη πρόταση: Ο επεξεργαστής μπορεί να χρησιμοποιεί την **ίδια εντολή**, σε επίπεδο γλώσσας μηχανής, για τη μεταφορά πληροφορίας **από τους καταχωρητές γενικού σκοπού στη μνήμη** για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. μη προσημασμένα δεδομένα μεγέθους μιας ψηφιολέξης, 2. μη προσημασμένα δεδομένα μεγέθους μισής λέξης, 3. μη προσημασμένα δεδομένα μεγέθους μιας λέξης.

α) Σωστό

β) Λάθος

Λύση

Στην εντολή STORE το κριτήριο του να έχουμε ίδιες εντολές, είναι το ίδιο μέγεθος. Εδώ έχουμε διαφορετικά μεγέθη, άρα και διαφορετικές εντολές

Θέματα Σεπτεμβρίου 2020 με θύρες καταχωρητών και σταθερή/κινητή υποδιαστολή

Θέμα 1

Σε ένα επεξεργαστή (μη υπερβαθμωτό) που βασίζεται στη χρήση καταχωρητών γενικού σκοπού με εντολές **καταχωρητή-καταχωρητή με δύο τελούμενα**, οι καταχωρητές γενικού σκοπού (register file) έχουν μόνο: Δύο πόρτες ανάγνωσης και μία πόρτα εγγραφής.

α) Σωστό

β) Λάθος

Λύση

Σωστό, διότι σε μια εντολή του είδους ADD R1, R2, γίνεται ανάγνωση των περιεχομένων των καταχωρητών R1 και R2 από δύο πόρτες ανάγνωσης, και αποθήκευση του αποτελέσματος σε κάποιον τρίτο καταχωρητή (από μια πόρτα εγγραφής).

Θέμα 2

Σε ένα επεξεργαστή (μη υπερβαθμωτό) που βασίζεται στη χρήση καταχωρητών γενικού σκοπού με εντολές καταχωρητή-καταχωρητή με **τρία τελούμενα**, οι καταχωρητές γενικού σκοπού (register file) έχουν μόνο: Δύο πόρτες ανάγνωσης και δύο πόρτες εγγραφής.

α) Σωστό

β) Λάθος

Λύση

Λάθος, διότι σε μια εντολή του είδους ADD R1, R2, R3 γίνεται ανάγνωση των περιεχομένων των καταχωρητών R1 και R2 από δύο πόρτες ανάγνωσης, και αποθήκευση του αποτελέσματος στον τρίτο καταχωρητή R3 (από μια πόρτα εγγραφής), αν υποθέσουμε ότι η εντολή εκτελείται από δεξιά προς τα αριστερά. Ακόμα όμως και αν εκτελείται αντίθετα, ισχύει η προαναφερόμενη πρόταση.

Συμπέρασμα: είτε έχουμε καταχωρητές με δύο τελούμενα, είτε με τρία, πάντα οι καταχωρητές γενικού σκοπού που χρησιμοποιούνται, έχουν δύο θύρες ανάγνωσης και μια εγγραφής.

Θέμα 3

Σε ένα επεξεργαστή (μη υπερβαθμωτό) που βασίζεται στη χρήση καταχωρητών γενικού σκοπού με εντολές καταχωρητή-καταχωρητή με τρία τελούμενα, οι καταχωρητές γενικού σκοπού (register file) έχουν μόνο: Δύο πόρτες ανάγνωσης και μία πόρτα εγγραφής.

α) Σωστό

β) Λάθος

Λύση

Σωστό, διότι σε μια εντολή του είδους ADD R1, R2, R3 γίνεται ανάγνωση των περιεχομένων των καταχωρητών R1 και R2 από δύο πόρτες ανάγνωσης, και αποθήκευση του αποτελέσματος στον R3 (από μια πόρτα εγγραφής).

Θέμα 4

Έχετε αριθμούς σε αναπαράσταση σταθερής υποδιαστολής των κ δυαδικών ψηφίων. Εάν **αυξήσουμε** το πλήθος των ακέραιων δυαδικών ψηφίων διατηρώντας **σταθερό** το συνολικό πλήθος των δυαδικών ψηφίων κ τότε: Το συνολικό πλήθος των αριθμών που μπορούν να αναπαρασταθούν μειώνεται.

α) Σωστό

β) Λάθος

Λύση

Λάθος. Όταν **αυξάνουμε το πλήθος των ακεραίων ψηφίων ενός αριθμού σε παράσταση σταθερής υποδιαστολής** και κρατάμε **σταθερό το συνολικό πλήθος των δυαδικών ψηφίων του αριθμού αυτού**, τότε αυξάνεται ουσιαστικά το ακέραιο μέρος του αριθμού και μειώνεται το κλασματικό μέρος του αριθμού. Αυτό έχει ως αποτέλεσμα να αυξάνεται η περιοχή των αριθμών που μπορούν να παρασταθούν, αλλά να μειώνεται η ακρίβεια των αριθμών που παριστάνονται. Για παράδειγμα, αν έχουμε αριθμούς σταθερής υποδιαστολής των 5 δυαδικών ψηφίων, τότε: **01111,0** = 15 ←→ 0111,10 = 7.5

Αύξηση ακεραίου μέρους **αριθμού σε παράσταση σταθερής υποδιαστολής → αύξηση πλήθους αριθμών που μπορούν να παρασταθούν (αύξηση περιοχής αριθμών που μπορούν να παρασταθούν) και το αντίστροφο.**

Αύξηση κλασματικού μέρους **αριθμού σε παράσταση σταθερής υποδιαστολής → αύξηση ακρίβειας αριθμών (μείωση περιοχής αριθμών που μπορούν να παρασταθούν) και το αντίστροφο.**

Θέμα 5

Έχετε αριθμούς σε αναπαράσταση σταθερής υποδιαστολής των κ δυαδικών ψηφίων. Εάν **αυξήσουμε** το πλήθος των ακέραιων δυαδικών ψηφίων διατηρώντας **σταθερό** το συνολικό πλήθος των δυαδικών ψηφίων κ τότε: Το πλήθος των ακέραιων αριθμών που μπορούν να αναπαρασταθούν αυξάνεται.

α) Σωστό

β) Λάθος

Λύση

Σωστό

Θέμα 6

Έχετε αριθμούς σε αναπαράσταση σταθερής υποδιαστολής των κ δυαδικών ψηφίων. Εάν αυξήσουμε το πλήθος των **ακέραιων** δυαδικών ψηφίων διατηρώντας **σταθερό** το συνολικό πλήθος των δυαδικών ψηφίων κ τότε: Το **πλήθος των αριθμών** με τιμή μεταξύ 1 και 0 που μπορούν να αναπαρασταθούν παραμένει σταθερό.

α) Σωστό

β) Λάθος

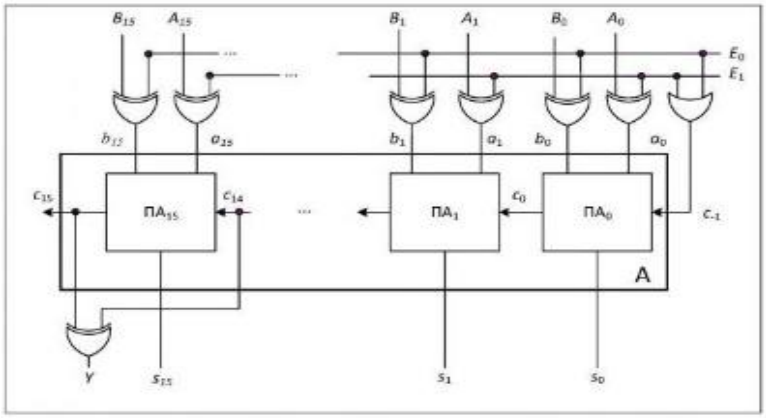
Λύση

Λάθος, όταν η υποδιαστολή στην παράσταση σταθερής υποδιαστολής μετακινείται αριστερά/δεξιά το συνολικό πλήθος των ψηφίων παραμένει σταθερό, όμως το πλήθος των αριθμών με τιμή μεταξύ 1 και 0 που μπορούν να αναπαρασταθούν, αυξάνεται π.χ. **01111,0** ←→ 0111,10 ←→ 011,110

Θέματα Ιουνίου 2020 με πράξεις προσημασμένων/μη προσημασμένων αριθμών

Θέμα 1

Η κεντρική μονάδα επεξεργασίας επεξεργαστή διαθέτει τον αθροιστή του επόμενου σχήματος. Να απαντήσετε εάν είναι σωστό το ακόλουθο ερώτημα: Για την πράξη A + B εάν οι αριθμοί είναι χωρίς πρόσημο (θετικοί) και Y=0 τότε το αποτέλεσμα της πράξης δεν ισούται με την τιμή στις εξόδους S_{15}, S_{14}, …S_0 του αθροιστή.



Υπενθυμίζουμε ότι στις πύλες XOR ισχύει ότι:

Το γεγονός ότι οι πύλες XOR είναι στις δύο εισόδους A_i και B_i σημαίνει ότι εκτός των πράξεων A+B και A-B, μπορούν να γίνουν επιπλέον οι πράξεις B + A και B – A. Πιο συγκεκριμένα:

$E_0 = 0$ και $E_1 = 0$ → A + B (ή B +A).

$E_0 = 0$ και $E_1 = 1$ → B – A = B + A’ + 1

$E_0 = 1$ και $E_1 = 0$ → A – B = A + B’ + 1

$E_0 = 1$ και $E_1 = 1$ → –A – B ή – B –A

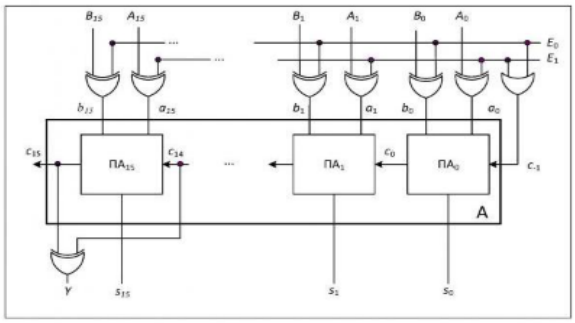
α) Σωστό

β) Λάθος

γ) Δεν έχω αρκετή πληροφορία για να εξάγω συμπέρασμα

Θέμα 2

Η κεντρική μονάδα επεξεργασίας επεξεργαστή διαθέτει τον αθροιστή του επόμενου σχήματος. Να απαντήσετε εάν είναι σωστό το ακόλουθο ερώτημα: Για την πράξη B + A εάν οι αριθμοί είναι σε **αναπαράσταση συμπληρώματος ως προς 2** και Y = 1 τότε το αποτέλεσμα της πράξης δεν ισούται με την τιμή στις εξόδους S_{15}, S_{14}, …S_0 του αθροιστή.

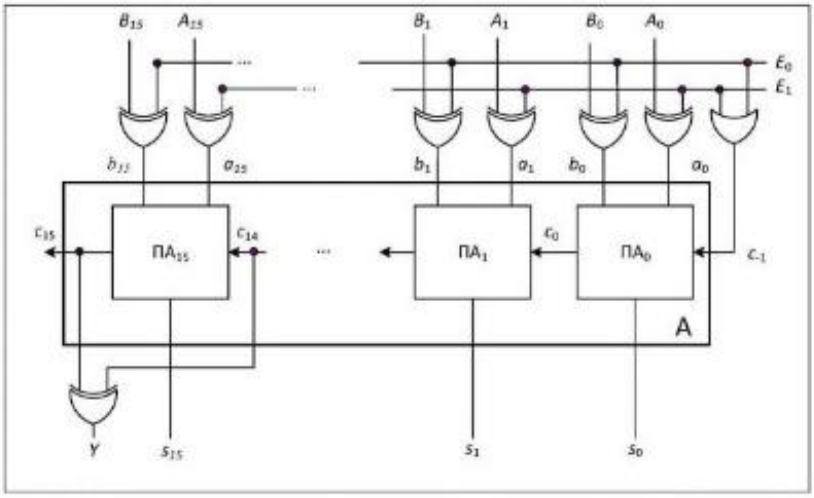


α) Σωστό

β) Λάθος
γ) Δεν έχω αρκετή πληροφορία για να εξάγω συμπέρασμα

Θέμα 3

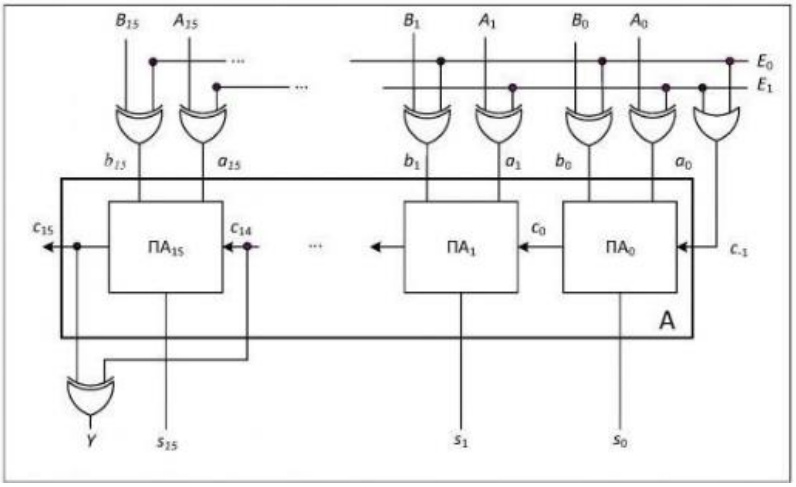
Η κεντρική μονάδα επεξεργασίας επεξεργαστή διαθέτει τον αθροιστή του επόμενου σχήματος. Να απαντήσετε εάν είναι σωστό το ακόλουθο ερώτημα: Για την πράξη A - B εάν οι αριθμοί είναι **χωρίς πρόσημο (θετικοί)** και $C_{15} = 1$ τότε το αποτέλεσμα της πράξης δεν ισούται με την τιμή στις εξόδους S_{15}, S_{14}, …S_0 του αθροιστή.



α) Σωστό
β) Λάθος
γ) Δεν έχω αρκετή πληροφορία για να εξάγω συμπέρασμα

Θέμα 4

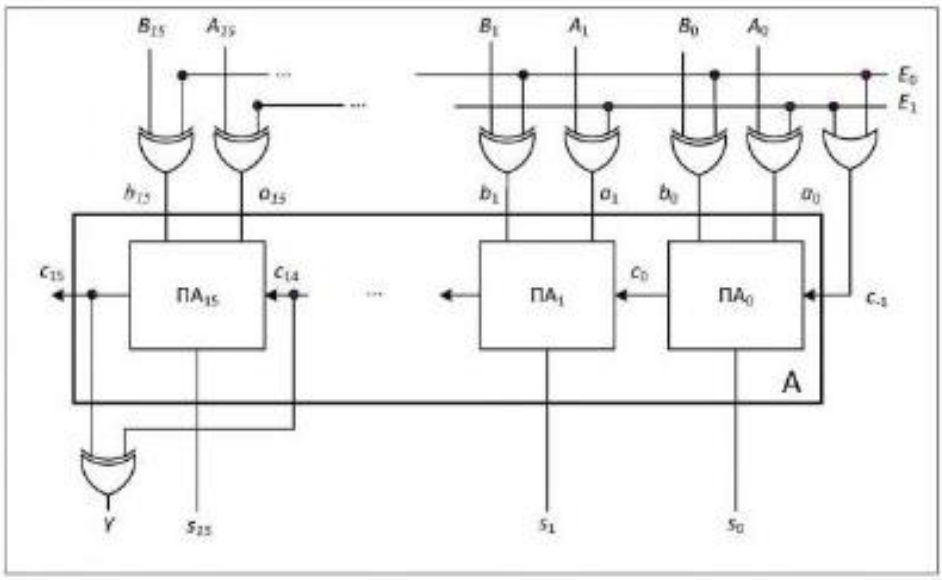
Η κεντρική μονάδα επεξεργασίας επεξεργαστή διαθέτει τον αθροιστή του επόμενου σχήματος. Να απαντήσετε εάν είναι σωστό το ακόλουθο ερώτημα: Για την πράξη B+A εάν οι αριθμοί είναι **χωρίς πρόσημο (θετικοί)** και Y = 1 τότε το αποτέλεσμα της πράξης δεν ισούται με την τιμή στις εξόδους S15, S14, …S0 του αθροιστή.



α) Σωστό
β) Λάθος
γ) Δεν έχω αρκετή πληροφορία για να εξάγω συμπέρασμα

Θέμα 5

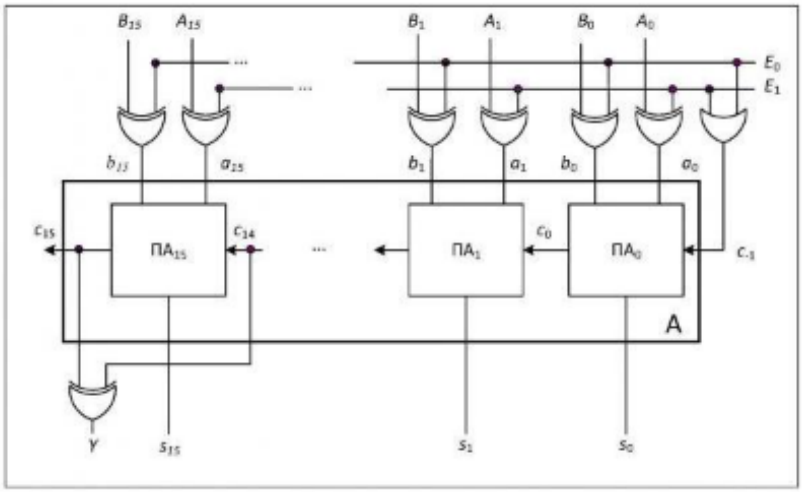
Η κεντρική μονάδα επεξεργασίας επεξεργαστή διαθέτει τον αθροιστή του επόμενου σχήματος. Να απαντήσετε εάν είναι σωστό το ακόλουθο ερώτημα: Για την πράξη B - A εάν οι αριθμοί είναι **χωρίς πρόσημο (θετικοί)** και **Y = 0** τότε το αποτέλεσμα της πράξης δεν ισούται με την τιμή στις εξόδους S_{15}, S_{14}, …S_0 του αθροιστή.



α) Σωστό
β) Λάθος
γ) Δεν έχω αρκετή πληροφορία για να εξάγω συμπέρασμα

Θέμα 6

Η κεντρική μονάδα επεξεργασίας επεξεργαστή διαθέτει τον αθροιστή του επόμενου σχήματος. Να απαντήσετε εάν είναι σωστό το ακόλουθο ερώτημα: Για την πράξη B + A εάν οι αριθμοί είναι σε **αναπαράσταση συμπληρώματος ως προς 2** και **Y = 1** τότε το αποτέλεσμα της πράξης δεν ισούται με την τιμή στις εξόδους S_{15}, S_{14}, …S_0 του αθροιστή.



α) Σωστό
β) Λάθος
γ) Δεν έχω αρκετή πληροφορία για να εξάγω συμπέρασμα

Θέμα 7

Το σύνολο εντολών, σε επίπεδο γλώσσας μηχανής, ενός επεξεργαστή απλού συνόλου εντολών υποστηρίζει τόσο μη προσημασμένα (θετικά) όσο και προσημασμένα (σε αναπαράσταση συμπληρώματος ως προς 2) ακέραια αριθμητικά δεδομένα μεγέθους μιας ψηφιολέξης (byte), μισής λέξης (δύο ψηφιολέξεις), και λέξης (τέσσερις ψηφιολέξεις). Ο επεξεργαστής αυτός διαθέτει καταχωρητές των 32 δυαδικών ψηφίων και Αριθμητική Λογική Μονάδα των 32 δυαδικών ψηφίων. Η κύρια μνήμη έχει οργάνωση μιας ψηφιολέξης ανά θέση μνήμης. Κατά τη μεταφορά πληροφορίας από τη μνήμη σε κάποιο καταχωρητή γενικού σκοπού η πληροφορία γράφεται στο λιγότερο σημαντικό τμήμα του καταχωρητή. Να απαντήσετε στην ακόλουθη ερώτηση: **Δεν** μπορεί να χρησιμοποιεί την **ίδια εντολή**, σε επίπεδο γλώσσας μηχανής, για τη μεταφορά πληροφορίας **από τη μνήμη σε κάποιο από τους καταχωρητές** γενικού σκοπού για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. μη προσημασμένα δεδομένα μεγέθους μιας ψηφιολέξης, 2. προσημασμένα δεδομένα μεγέθους μιας ψηφιολέξης.

α) Σωστό

β) Λάθος

Θέμα 8

Το σύνολο εντολών, σε επίπεδο γλώσσας μηχανής, ενός επεξεργαστή απλού συνόλου εντολών υποστηρίζει τόσο μη προσημασμένα (θετικά) όσο και προσημασμένα (σε αναπαράσταση συμπληρώματος ως προς 2) ακέραια αριθμητικά δεδομένα μεγέθους μιας ψηφιολέξης (byte), μισής λέξης (δύο ψηφιολέξεις), και λέξης (τέσσερις ψηφιολέξεις). Ο επεξεργαστής αυτός διαθέτει καταχωρητές των 32 δυαδικών ψηφίων και Αριθμητική Λογική Μονάδα των 32 δυαδικών ψηφίων. Η κύρια μνήμη έχει οργάνωση μιας ψηφιολέξης ανά θέση μνήμης. Κατά τη μεταφορά πληροφορίας από τη μνήμη σε κάποιο καταχωρητή γενικού σκοπού η πληροφορία γράφεται στο λιγότερο σημαντικό τμήμα του καταχωρητή. Να απαντήσετε στην ακόλουθη ερώτηση: **Μπορεί να χρησιμοποιεί την ίδια εντολή** σε επίπεδο γλώσσα μηχανής, για τη μεταφορά πληροφορίας από τους **καταχωρητές γενικού σκοπού στη μνήμη** για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. μη προσημασμένα δεδομένα μεγέθους μιας λέξης, 2. προσημασμένα δεδομένα μεγέθους μιας λέξης.

α) Σωστό

β) Λάθος

Άσκηση 3.3 βιβλίου για εντολές LOAD/STORE

Το σύνολο εντολών, σε επίπεδο γλώσσας μηχανής, ενός επεξεργαστή απλού συνόλου εντολών υποστηρίζει τόσο θετικά όσο και προσημασμένα (σε αναπαράσταση συμπληρώματος ως προς 2) ακέραια αριθμητικά δεδομένα μεγέθους μιας ψηφιολέξης (byte), μισής λέξης (δύο ψηφιολέξεις), και λέξης (τέσσερις ψηφιολέξεις). Ο επεξεργαστής αυτός διαθέτει καταχωρητές των 32 δυαδικών ψηφίων και Αριθμητική Λογική Μονάδα των ν-32 δυαδικών ψηφίων. Στο σύνολο εντολών του σε επίπεδο γλώσσας μηχανής διαθέτει διαφορετικές εντολές μεταφοράς πληροφορίας από την μνήμη σε κάποιο από τους καταχωρητές γενικού σκοπού για κάθε ένα από τα ακόλουθα είδη δεδομένων: α. 1 θετικά δεδομένα μεγέθους μιας ψηφιολέξης, α.2 προσημασμένα δεδομένα μεγέθους μιας ψηφιολέξης, β.1 θετικά δεδομένα μεγέθους μισής λέξης και β.2 προσημασμένα δεδομένα μεγέθους μισής λέξης. Κατά τη μεταφορά πληροφορίας από τη μνήμη σε κάποιο καταχωρητή γενικού σκοπού η πληροφορία γράφεται στο λιγότερο σημαντικό τμήμα του καταχωρητή. ΕΙ κυρία μνήμη έχει οργάνωση μιας ψηφιολέξης ανά θέση μνήμης.

α. Πιστεύετε ότι πρέπει να διαθέτει διαφορετικές εντολές για την ανάγνωση από τη μνήμη μιας θετικής λέξης δεδομένων και μιας προσημασμένης λέξης δεδομένων;

β. Πιστεύετε ότι στον εν λόγω υπολογιστή απαιτείται η ύπαρξη **διαφορετικών αριθμητικών εντολών** για πράξεις μεταξύ αριθμητικών δεδομένων κάθε μιας από τις κατηγορίες που αναφέρθηκαν πιο πάνω;

γ. Ο καταχωρητής κατάστασης μεταξά των άλλων διαθέτει τις σημαίες K (κρατούμενο εξόδου), Y (υπερχείλιση) και M (μηδενικό αποτέλεσμα). Υπάρχουν εντολές BRK (Branch if K), BRY (Branch if Y) και BRM (Branch if M). Ποια εντολή θα χρησιμοποιήσετε σε κάθε περίπτωση δεδομένων για την ανίχνευση υπερχείλισης;

δ. Πιστεύετε ότι το σύνολο των εντολών πρέπει να περιέχει διαφορετική εντολή για την αποθήκευση του περιεχομένου κάποιου καταχωρητή στη μνήμη ανάλογα του είδους του δεδομένου που περιέχει;

Να αιτιολογήσετε τις απαντήσεις σας.

Λύση

α. Για δεδομένα μιας ψηφιολέξης ή μισής λέξης χρειάζονται διαφορετικές εντολές για θετικά και προσημασμένα δεδομένα ώστε κατά την αποθήκευση στον 32 – bit καταχωρητή, να γίνει η σωστή επέκταση προσήμου. Για δεδομένα μιας λέξης, δηλαδή 4 ψηφιολέξεων, δεν χρειάζεται να γίνει επέκταση προσήμου, επειδή ο καταχωρητής γεμίζει πλήρως, επομένως **δεν** χρειάζονται διαφορετικές εντολές ανάγνωσης από τη μνήμη για θετικά και προσημασμένα δεδομένα μιας λέξης.

β. **Δεν** χρειάζονται διαφορετικές εντολές για δεδομένα μιας ψηφιολέξης, μισής λέξης και μιας λέξης διότι χρησιμοποιείται η ίδια μονάδα (ΑΛΜ) για την εκτέλεση της αριθμητικής εντολής.

γ. Στην περίπτωση που τα δεδομένα που χρησιμοποιήσαμε είναι θετικοί αριθμοί θα χρησιμοποιήσουμε την εντολή BRK, ενώ αν τα δεδομένα είναι σε παράσταση συμπληρώματος ως προς 2, θα χρησιμοποιήσουμε την εντολή BRY.

δ. Χρειάζονται διαφορετικές εντολές για δεδομένα μιας ψηφιολέξης, μισής λέξης και μιας λέξης διότι, στην πρώτη περίπτωση μόνο τα 8 λιγότερο σημαντικά δυαδικά ψηφία του καταχωρητή θα αποθηκευτούν σε μια θέση μνήμης, στη δεύτερη περίπτωση τα 16 λιγότερο σημαντικά δυαδικά ψηφία του καταχωρητή θα αποθηκευτούν σε δύο διαδοχικές θέσεις μνήμης και στην τρίτη περίπτωση όλο το περιεχόμενο του καταχωρητή θα αποθηκευτεί σε 4 διαδοχικές θέσεις μνήμης. Δεν χρειάζονται διαφορετικές εντολές για θετικά και προσημασμένα δεδομένα.

Θέματα Φεβρουαρίου 2014 με εντολή LOAD και πράξεις αριθμών

Θέμα 1

Θεωρήστε ένα επεξεργαστή **με καταχωρητές των 32 δυαδικών ψηφίων** και μνήμη με οργάνωση **8 δυαδικών ψηφίων ανά θέση μνήμης**. Ο συγκεκριμένος επεξεργαστής υποστηρίζει μεταξύ των άλλων δεδομένα μιας ψηφιολέξης σε αναπαράσταση συμπληρώματος ως προς 2 και δεδομένα μιας ψηφιολέξης χωρίς πρόσημο. Στο σύνολο των εντολών που περιλαμβάνονται οι κάτωθι δύο εντολές:

LOAD-UB r, $xxxx	μεταφέρει στον καταχωρητή r τα δεδομένα που είναι αποθηκευμένα στη θέση μνήμης με διεύθυνση $xxxx_{(16)}$, θεωρεί ότι τα δεδομένα είναι μη προσημασμένος αριθμός (U = unsigned).
LOAD-SB r, $xxxx	μεταφέρει στον καταχωρητή r τα δεδομένα που είναι αποθηκευμένα στη θέση μνήμης με διεύθυνση $xxxx_{(16)}$, θεωρεί ότι τα δεδομένα είναι σε παράσταση συμπληρώματος ως προς 2 (S = Signed).

Θεωρήστε ότι τα περιεχόμενα της μνήμης είναι τα ακόλουθα (στο δεκαεξαδικό):

Διεύθυνση Μνήμης	Περιεχόμενα Μνήμης
8001	6F
8002	80
8003	C0
8004	0E

Να δώσετε το περιεχόμενο του καταχωρητή μετά την εκτέλεση κάθε μιας από τις ακόλουθες εντολές:

"LOAD-UB r1, $8001", "LOAD-SB r2, $8002", "LOAD-UB r3, $8003" και "LOAD-SB r4, $8004"

Λύση

Όταν εκτελούμε κάθε μια από τις εντολές που αναφέρονται, ο κάθε καταχωρητής γεμίζει με ένα περιεχόμενο των 8 bits. Επειδή όμως το περιεχόμενό του είναι των 32 bits, πρέπει να γίνει **επέκταση προσήμου** και η επέκταση αυτή γίνεται με διαφορετικό τρόπο στους προσημασμένους και με διαφορετικό τρόπο στους μη – προσημασμένους αριθμούς. Πιο συγκεκριμένα: r1 ← 0000006F ή r1 ← 0000000…….00000110**1**111, r2 ← FFFFFF80 ή r2 ← 1111111…….1111**1**1110000 r3← 000000C0 ή r3← 0000000…….00001100**0**000, r4 ← 0000000E ή r4 ←0000000…….0000**0**0001110

Θέμα 2

Ποια/ές από τις παρακάτω απαντήσεις είναι σωστή/στές: ($Y = C_{out} \oplus C_{out-1}$, C_{out}: κρατούμενο εξόδου)

1) Το αποτέλεσμα της άθροισης δύο αριθμών χωρίς πρόσημο δε χωράει σε κ δυαδικά ψηφία όταν $Y = 1$ και $C_{out} = 0$
2) Το αποτέλεσμα της άθροισης δύο αριθμών σε παράσταση συμπληρώματος ως προς 2 δε χωράει σε κ δυαδικά ψηφία όταν $Y = 0$ και $C_{out} = 0$
3) Το αποτέλεσμα της άθροισης δύο αριθμών χωρίς πρόσημο δε χωράει σε κ δυαδικά ψηφία όταν $Y = 0$ και $C_{out} = 0$
4) Το αποτέλεσμα της άθροισης δύο αριθμών χωρίς πρόσημο δε χωράει σε κ δυαδικά ψηφία όταν $Y = 1$ και $C_{out} = 1$
5) Το αποτέλεσμα της άθροισης δύο αριθμών σε παράσταση συμπληρώματος ως προς 2 δε χωράει σε κ δυαδικά ψηφία όταν $Y = 0$ και $C_{out} = 1$
6) Το αποτέλεσμα της άθροισης δύο αριθμών σε παράσταση συμπληρώματος ως προς 2 δε χωράει σε κ δυαδικά ψηφία όταν $Y = 1$ και $C_{out} = 0$

Θέματα Σεπτεμβρίου 2016 με εντολές LOAD/STORE

Θέμα 1

Το σύνολο εντολών, σε επίπεδο γλώσσας μηχανής, ενός επεξεργαστή απλού συνόλου εντολών υποστηρίζει τόσο θετικά όσο και προσημασμένα (σε αναπαράσταση συμπληρώματος ως προς 2) ακέραια αριθμητικά δεδομένα μεγέθους μιας ψηφιολέξης (byte), μισής λέξης (δύο ψηφιολέξεις), και λέξης (τέσσερις ψηφιολέξεις). Ο επεξεργαστής αυτός **διαθέτει καταχωρητές των 32 δυαδικών ψηφίων** και Αριθμητική Λογική Μονάδα των ν=32 δυαδικών ψηφίων. **Η κύρια μνήμη έχει οργάνωση μιας ψηφιολέξης ανά θέση μνήμης**. Κατά τη μεταφορά πληροφορίας από τη μνήμη σε κάποιο καταχωρητή γενικού σκοπού η πληροφορία γράφεται στο λιγότερο σημαντικό τμήμα του καταχωρητή.

Να σημειώσετε ποιες από τις ακόλουθες προτάσεις ισχύουν και να αιτιολογήσετε κάθε μια το πολύ σε 3 γραμμές.

α. Το σύνολο εντολών του σε επίπεδο γλώσσας μηχανής πρέπει να διαθέτει **διαφορετικές εντολές** μεταφοράς πληροφορίας **από τη μνήμη σε κάποιο από τους καταχωρητές** γενικού σκοπού για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. θετικά δεδομένα μεγέθους μιας λέξης, 2. προσημασμένα δεδομένα μεγέθους μιας λέξης.

β. Μπορεί να χρησιμοποιεί την **ίδια εντολή**, σε επίπεδο γλώσσας μηχανής, για τη μεταφορά πληροφορίας **από τη μνήμη σε κάποιο από τους καταχωρητές** γενικού σκοπού για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. θετικά δεδομένα μεγέθους μιας ψηφιολέξης, 2. προσημασμένα δεδομένα μεγέθους μιας ψηφιολέξης.

γ. Μπορεί να χρησιμοποιεί την **ίδια εντολή**, σε επίπεδο γλώσσας μηχανής, για τη μεταφορά πληροφορίας **από τη μνήμη σε κάποιο από τους καταχωρητές γενικού σκοπού** για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. θετικά δεδομένα μεγέθους μισής λέξης, 2. προσημασμένα δεδομένα μεγέθους μισής λέξης.

δ. Το σύνολο εντολών του σε επίπεδο γλώσσας μηχανής πρέπει να διαθέτει **διαφορετικές εντολές** μεταφοράς πληροφορίας **από τη μνήμη σε κάποιο από τους καταχωρητές** γενικού σκοπού για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. θετικά δεδομένα μεγέθους μιας ψηφιολέξης, 2. προσημασμένα δεδομένα μεγέθους μιας ψηφιολέξης.

ε. Το σύνολο εντολών του σε επίπεδο γλώσσας μηχανής πρέπει να διαθέτει **διαφορετικές εντολές** μεταφοράς πληροφορίας **από τους καταχωρητές γενικού σκοπού στη μνήμη** για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. θετικά δεδομένα μεγέθους μιας ψηφιολέξης, 2. προσημασμένα δεδομένα μεγέθους μιας ψηφιολέξης.

ζ. Μπορεί να χρησιμοποιεί την **ίδια εντολή**, σε επίπεδο γλώσσας μηχανής, για τη μεταφορά πληροφορίας **από τη μνήμη σε κάποιο από τους καταχωρητές** γενικού σκοπού για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. θετικά δεδομένα μεγέθους μιας λέξης, 2. προσημασμένα δεδομένα μεγέθους μιας λέξης.

η. Το σύνολο εντολών του σε επίπεδο γλώσσας μηχανής πρέπει να διαθέτει **διαφορετικές εντολές** μεταφοράς πληροφορίας **από τη μνήμη σε κάποιο από τους καταχωρητές** γενικού σκοπού για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. θετικά δεδομένα μεγέθους μισής λέξης, 2. προσημασμένα δεδομένα μεγέθους μισής λέξης.

θ. Το σύνολο εντολών του σε επίπεδο γλώσσας μηχανής πρέπει να διαθέτει διαφορετικές εντολές μεταφοράς πληροφορίας από τους καταχωρητές γενικού σκοπού στη μνήμη για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. θετικά δεδομένα μεγέθους μισής λέξης, 2. προσημασμένα δεδομένα μεγέθους μισής λέξης.

ι. Μπορεί να χρησιμοποιεί την ίδια εντολή, σε επίπεδο γλώσσας μηχανής, για τη μεταφορά πληροφορίας από τους καταχωρητές γενικού σκοπού στη μνήμη για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. θετικά δεδομένα μεγέθους μιας ψηφιολέξης, 2. προσημασμένα δεδομένα μεγέθους μιας ψηφιολέξης.

κ. Το σύνολο εντολών του σε επίπεδο γλώσσας μηχανής πρέπει να διαθέτει διαφορετικές εντολές μεταφοράς πληροφορίας θετικά μεγέθους μιας λέξης, 2. προσημασμένα δεδομένα μεγέθους μιας λέξης.

λ. Μπορεί να χρησιμοποιεί την **ίδια εντολή**, σε επίπεδο γλώσσας μηχανής, για τη μεταφορά πληροφορίας **από τη μνήμη σε κάποιο από τους καταχωρητές γενικού σκοπού** για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. θετικά δεδομένα μεγέθους μιας ψηφιολέξης, 2. θετικά δεδομένα μισής λέξης, 3. θετικά δεδομένα μεγέθους μιας λέξης.

μ. Το σύνολο εντολών του σε επίπεδο γλώσσας μηχανής πρέπει να διαθέτει διαφορετικές εντολές μεταφοράς πληροφορίας από τη μνήμη σε κάποιο από τους καταχωρητές γενικού σκοπού για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. θετικά δεδομένα μεγέθους μιας ψηφιολέξης, 2. θετικά δεδομένα μεγέθους μισής λέξης, 3. θετικά δεδομένα μεγέθους μιας λέξης.

ν. Μπορεί να χρησιμοποιεί την ίδια εντολή, σε επίπεδο γλώσσας μηχανής, για τη μεταφορά πληροφορίας από τους καταχωρητές γενικού σκοπού στη μνήμη για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. θετικά δεδομένα μεγέθους μισής λέξης, 2. προσημασμένα δεδομένα μεγέθους μισής λέξης.

ξ. Μπορεί να χρησιμοποιεί την ίδια εντολή, σε επίπεδο γλώσσας μηχανής, για τη μεταφορά πληροφορίας από τους καταχωρητές γενικού σκοπού στη μνήμη για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. θετικά δεδομένα μεγέθους μιας λέξης, 2. προσημασμένα δεδομένα μεγέθους μιας λέξης.

ο. Το σύνολο εντολών του σε επίπεδο γλώσσας μηχανής πρέπει να διαθέτει διαφορετικές εντολές μεταφοράς πληροφορίας από τη μνήμη σε κάποιο από τους καταχωρητές γενικού σκοπού για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. προσημασμένα δεδομένα μεγέθους μιας ψηφιολέξης, 2. προσημασμένα δεδομένα μεγέθους μισής λέξης, 3. προσημασμένα δεδομένα μεγέθους μιας λέξης.

π. Το σύνολο εντολών του σε επίπεδο γλώσσας μηχανής πρέπει να διαθέτει διαφορετικές εντολές μεταφοράς πληροφορίας από τους καταχωρητές γενικού σκοπού στη μνήμη για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. θετικά δεδομένα μεγέθους μιας ψηφιολέξης, 2. θετικά δεδομένα μεγέθους μισής λέξης, 3. θετικά δεδομένα μεγέθους μιας λέξης.

ρ. Μπορεί να χρησιμοποιεί την ίδια εντολή, σε επίπεδο γλώσσας μηχανής, για τη μεταφορά πληροφορίας από τους καταχωρητές γενικού σκοπού στη μνήμη για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. προσημασμένα δεδομένα μεγέθους μιας ψηφιολέξης, 2. προσημασμένα δεδομένα μεγέθους μισής λέξης, 3. προσημασμένα δεδομένα μεγέθους μιας λέξης.

σ. Το σύνολο εντολών του σε επίπεδο γλώσσας μηχανής πρέπει να διαθέτει διαφορετικές εντολές μεταφοράς πληροφορίας από τους καταχωρητές γενικού σκοπού στη μνήμη για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. προσημασμένα δεδομένα μεγέθους μιας ψηφιολέξης, 2. προσημασμένα δεδομένα μεγέθους μισής λέξης, 3. προσημασμένα δεδομένα μεγέθους μιας λέξης.

τ. Μπορεί να χρησιμοποιεί την ίδια εντολή, σε επίπεδο γλώσσας μηχανής, για τη μεταφορά πληροφορίας από τη μνήμη σε κάποιο από τους καταχωρητές γενικού σκοπού για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. προσημασμένα δεδομένα μεγέθους μιας ψηφιολέξης, 2. προσημασμένα δεδομένα μεγέθους μισής λέξης, 3. προσημασμένα δεδομένα μεγέθους μιας λέξης.

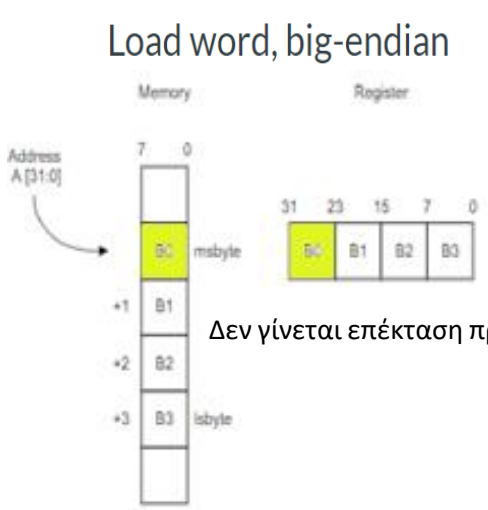
χ. Μπορεί να χρησιμοποιεί την ίδια εντολή, σε επίπεδο γλώσσας μηχανής, για τη μεταφορά πληροφορίας από τους καταχωρητές γενικού σκοπού στη μνήμη για κάθε ένα από τα ακόλουθα είδη δεδομένων: 1. θετικά δεδομένα μεγέθους μιας ψηφιολέξης, 2. θετικά δεδομένα μεγέθους μισής λέξης, 3. θετικά δεδομένα μεγέθους μιας λέξης.

Λύση

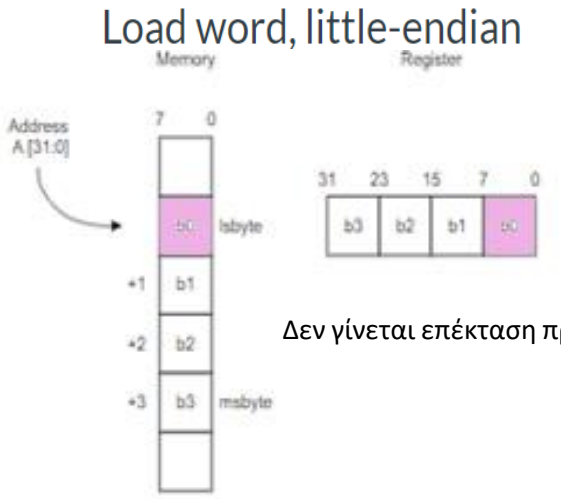
α. Λ
β. Λ
γ. Λ
δ. Σ
ε. Λ
ζ. Σ
η. Σ
θ. Λ
ι. Σ
κ. Λ
λ. Λ
μ. Σ
ν. Σ
ξ. Σ
ο. Σ
π. Σ
ρ. Λ
σ. Σ
τ. Λ
χ. Λ

Αιτιολόγηση: Όλες οι εντολές που αναφέρονται από μεταφορά από την κύρια μνήμη σε καταχωρητές (LOAD) εξαρτώνται από το **αν γεμίζει πλήρως ή όχι ο καταχωρητής γενικού σκοπού**. Για να γεμίσει πλήρως, πρέπει να φορτωθεί με μια λέξη (32 bits) από τη στιγμή που έχει χωρητικότητα 32 bits. Σε οποιαδήποτε άλλη περίπτωση, θα γίνει επέκταση προσήμου, με αποτέλεσμα να χρειάζονται διαφορετικές εντολές για να φορτωθεί. Όλες οι εντολές που αναφέρονται από μεταφορά από καταχωρητές σε κύρια μνήμη (STORE) εξαρτώνται από το **μέγεθος των δεδομένων** που μεταφέρονται. Σε περίπτωση μεταφοράς δεδομένων ίδιου μεγέθους, έχουμε ίδιες εντολές (ανεξάρτητα από το είδος των δεδομένων) Αντίθετα, σε οποιαδήποτε άλλη περίπτωση, θα χρησιμοποιηθούν διαφορετικές εντολές.

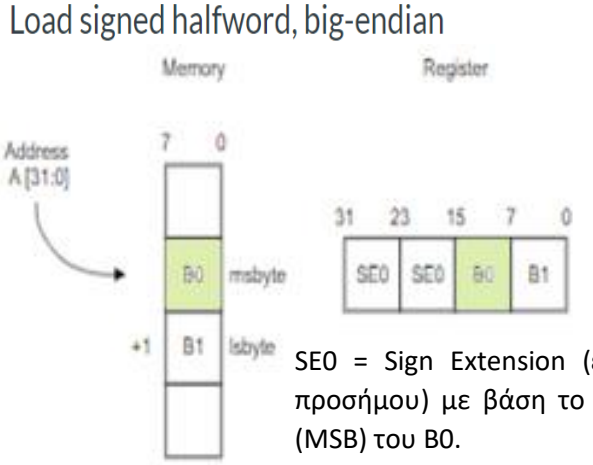
Διαφορά little – endian και big – endian αποθήκευσης



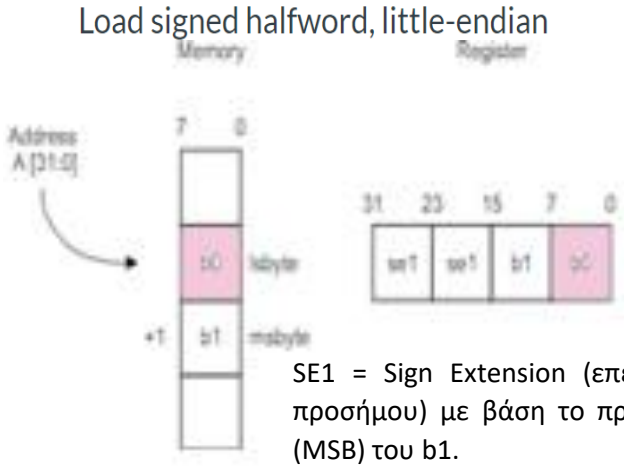
Δεν γίνεται επέκταση προσήμου



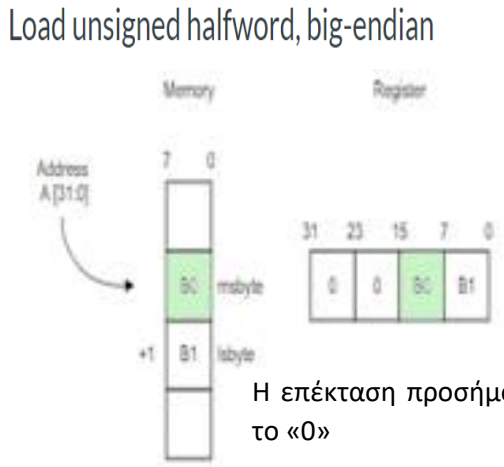
Δεν γίνεται επέκταση προσήμου



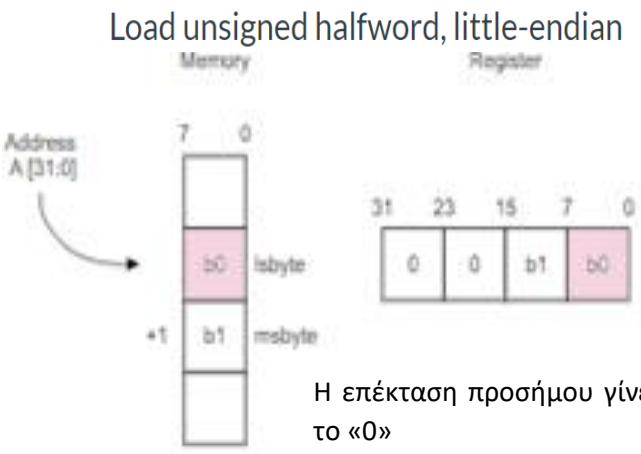
SE0 = Sign Extension (επέκταση προσήμου) με βάση το πρόσημο (MSB) του B0.



SE1 = Sign Extension (επέκταση προσήμου) με βάση το πρόσημο (MSB) του b1.

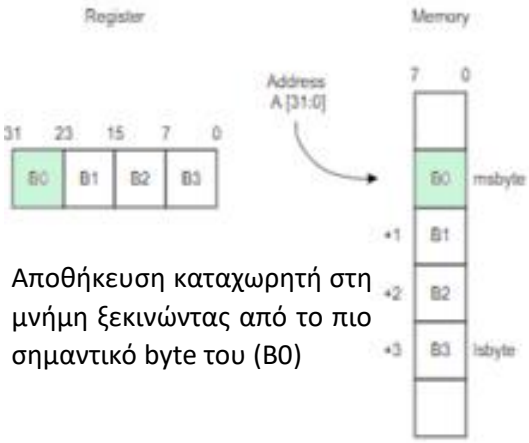


Η επέκταση προσήμου γίνεται με το «0»



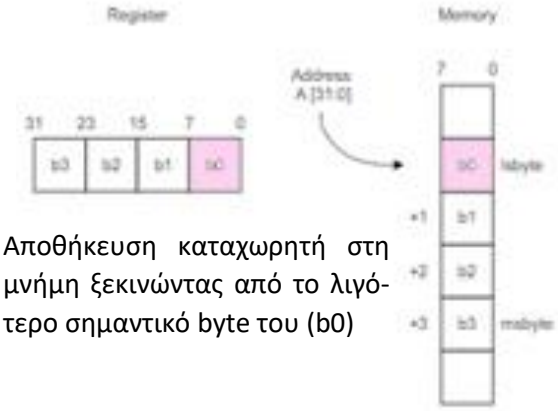
Η επέκταση προσήμου γίνεται με το «0»

Store word, big-endian



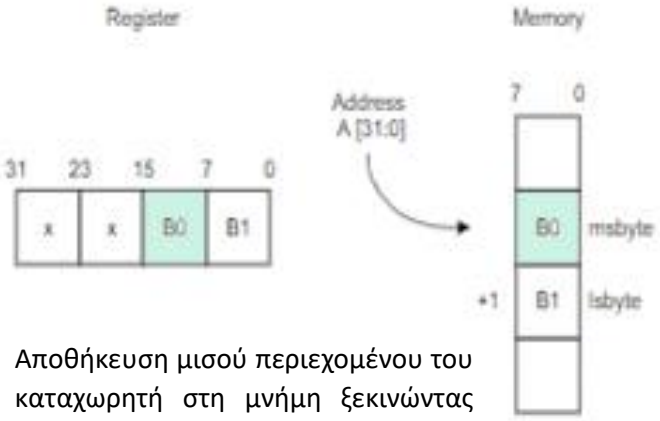
Αποθήκευση καταχωρητή στη μνήμη ξεκινώντας από το πιο σημαντικό byte του (B0)

Store word, little-endian



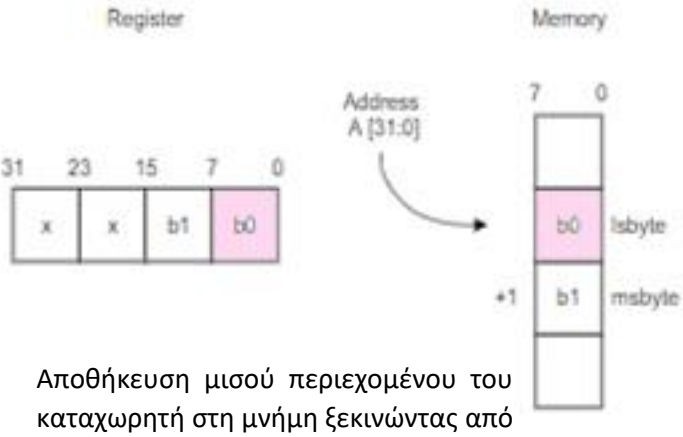
Αποθήκευση καταχωρητή στη μνήμη ξεκινώντας από το λιγό-τερο σημαντικό byte του (b0)

Store halfword, big-endian



Αποθήκευση μισού περιεχομένου του καταχωρητή στη μνήμη ξεκινώντας από το πιο σημαντικό byte του (B0)

Store halfword, little-endian



Αποθήκευση μισού περιεχομένου του καταχωρητή στη μνήμη ξεκινώντας από το πιο λιγότερο byte του (b0)

Άσκηση με big – endian και little – endian

Θεωρήστε επεξεργαστή με 16 καταχωρητές των 32 δυαδικών ψηφίων ο καθένας που στο σύνολο των εντολών του περιλαμβάνονται οι κάτωθι εντολές:

εντολές	σχόλια
LOADSB R1, A	Φόρτωσε προσημασμένη ψηφιολέξη
LOADSHW R1, A	Φόρτωσε προσημασμένη ψηφιολέξη
LOADW R1, A	Φόρτωσε λέξη
LOAUB R1, A	Φόρτωσε απρόσημη ψηφιολέξη
LOAUHW R1, A	Φόρτωσε απρόσημη μισή λέξη
STOREB A, R1	αποθήκευσε ψηφιολέξη
STOREHW A, R1	αποθήκευσε μισή λέξη
STOREW A, R1	αποθήκευσε λέξη

Στον παρακάτω πίνακα δίνονται τα περιεχόμενα τμήματος της κύριας μνήμης. Θεωρήστε ότι χρησιμοποιείται απεικόνιση αντιστρόφως ανάλογη του μεγέθους (**big endian**). Οι προσημασμένοι αριθμοί είναι σε αναπαράσταση συμπληρώματος ως προς 2.

Υπενθυμίζεται: ψηφιολέξη: 8 δυαδικά ψηφία, μισή λέξη: 16 δυαδικά ψηφία, λέξη: 32 δυαδικά ψηφία

μνήμη		
διεύθυνση θέσης μνήμης στο δεκαεξαδικό	περιεχόμενο θέσης μνήμη, στο δεκαεξαδικό πριν την εκτέλεση των εντολών	περιεχόμενο θέσης μνήμης στο δεκαεξαδικό μετά την εκτέλεση των εντολών
100	87	87
101	00	00
102	11	11
103	E3	
104	E4	……
105	00	
106	AB	……
107	F0	……
108	01	
109	80	… .
10A	07	… …
10B	32	11
10C	AE	6A
10D	BC	23
10E	10	AE
10F	C4	00
110	89	24
111	1A	56
112	A0	

Να συμπληρώσετε στην μνήμη και στον κάτωθι πίνακα (στους καταχωρητές) το αποτέλεσμα της εκτέλεση των εντολών

εντολή	Καταχωρητής	Περιεχόμενο του καταχωρητή στο δεκαεξαδικό πριν την εκτέλεση της εντολής	Περιεχόμενο του καταχωρητή στο δεκαεξαδικό μετά την εκτέλεση της εντολής
LOADSB R1, $10A_{(16)}$	R1=	0F 0F 0F 0F	00000007
LOADSHW R2, $103_{(16)}$	R2=	0F 0F 0F 0F	FFFFE3E4
LOADW R3, $105_{(16)}$	R3=	0F 0F 0F 0F	00ABF001
LOADUB R4, $106_{(16)}$	R4=	0F 0F 0F 0F	000000AB
LOADUHW R5, $103_{(16)}$	R5=	0F 0F 0F 0F	0000E3E4
STOREB $10B_{(16)}$, R6	R6=	00 FF 00 11	00FF0011
STOREHW $110_{(16)}$, R7	R7=	FF 09 24 56	FF092456
STOREW $10C_{(16)}$,R8	R8=	6A 23 AE 00	6A23AE00
LOADSB R9, $109_{(16)}$	R9=	0F 0F 0F 0F	FFFFFF80

Θέμα Φεβρουαρίου 2021 με little – endian και big – endian

Θεωρήστε υπολογιστή του οποίου ο επεξεργαστής έχει 16 καταχωρητές των 32 δυαδικών ψηφίων ο καθένας και κύρια μνήμη με οργάνωση 8 δυαδικών ψηφίων ανά θέση μνήμης. Το τρέχον περιεχόμενο της μνήμης δίνεται στο σχήμα. Οι προσημασμένοι αριθμοί είναι σε αναπαράσταση συμπληρώματος ως προς 2. Μισή λέξη: 16 δυαδικά ψηφία. Για κάθε μια των κάτωθι περιπτώσεων να δώσετε το περιεχόμενο του καταχωρητή μετά την εκτέλεση της εντολής στο δεκαεξαδικό χρησιμοποιώντας κεφαλαίους χαρακτήρες. Προσοχή να μην παρεμβάλλονται κενά ή άλλα σύμβολα.

κύρια μνήμη	
διεύθυνση θέσης μνήμης στο δεκαεξαδικό	περιεχόμενο θέσης μνήμης στο δεκαεξαδικό
110	87
111	13
112	E1
113	73
114	94
115	5F
116	A0
117	16
118	C1
119	56
11A	85
11B	32
11C	AE
11D	4C
11E	F0
11F	64
120	89
121	1A
122	A0
123	65
124	8F
125	10
126	92
127	34
128	B3
129	32
12A	F1
12B	20
12C	F4
12D	F3
12E	78

κύρια μνήμη	
διεύθυνση θέσης μνήμης στο δεκαεξαδικό	περιεχόμενο θέσης μνήμης στο δεκαεξαδικό
227	98
228	FA
229	19
22A	FE
22B	A2
22C	41
22D	F3
22E	64
22F	85
230	1F
231	AE
232	68
233	8A
234	1B
235	98
236	5C
237	A1
238	1D
239	C3
23A	58
23B	89
23C	37
23D	97
23E	13
23F	E5
240	63
241	9A
242	5C
243	AD
244	46
245	95

Απεικόνιση αντιστρόφως ανάλογη του μεγέθους (big endian) LOADSHW R1, 113 //Φόρτωσε προσημασμένη μισή λέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διευθυνσιοδότησης (direct addressing mode) R1=[ ]. Απεικόνιση ανάλογη του μεγέθους (little endian) LOADSHW R2, 22E //Φόρτωσε προσημασμένη μισή λέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διευθυνσιοδότησης (direct addressing mode) R2=[ ].

Λύση

R1 = 00007394, R2 = FFFF8564

Παρατήρηση 1: σε περίπτωση που υπήρχαν οι εντολές **LOADUHW R1, 113** και **LOADUHW R2, 22E**, τότε τα περιεχόμενα των καταχωρητών R1 και R2 θα ήταν R1 = 00007394 και R2 = 00008564. Δηλαδή **στην περίπτωση των μη – προσημασμένων αριθμών (U = unsigned) η επέκταση προσήμου γίνεται με βάση το «0»** και όχι με βάση το MSB όπως πριν. Αντίθετα, στην περίπτωση των προσημασμένων αριθμών (**S** = Signed), η **επέκταση προσήμου γίνεται με βάση το MSB.**

Παρατήρηση 2: σε περίπτωση που υπήρχε η εντολή **STOREW R1, 23C** και με δεδομένο ότι το περιεχόμενο του R1 ήταν R1 = 00007394 και η αποθήκευση ήταν **little – endian**, τότε:

23C: 94
23D: 73
23E: 00
23F: 00

Παρατήρηση 3: σε περίπτωση που υπήρχε η εντολή **STOREHW R1, 23C** και με δεδομένο ότι το περιεχόμενο του R1 ήταν R1 = 00007394 και η αποθήκευση ήταν little – endian, τότε:

23C: 94
23D: 73

Θέμα Σεπτεμβρίου 2021 με little – endian και big – endian

Θεωρήστε υπολογιστή του οποίου ο επεξεργαστής έχει 32 καταχωρητές των 16 δυαδικών ψηφίων ο καθένας και κύρια μνήμη με οργάνωση 8 δυαδικών ψηφίων ανά θέση μνήμης. Το τρέχον περιεχόμενο της μνήμης δίνεται στο σχήμα. Οι προσημασμένοι αριθμοί είναι σε αναπαράσταση συμπληρώματος ως προς 2. Μισή λέξη: 16 δυαδικά ψηφία. Για κάθε μια των κάτωθι περιπτώσεων, να δώσετε το περιεχόμενο του καταχωρητή στο δεκαεξαδικό (χρησιμοποιώντας κεφαλαίους χαρακτήρες), μετά την εκτέλεση της εντολής. Προσοχή στην απάντησή σας να μην παρεμβάλλονται κενά ή άλλα σύμβολα.

διεύθυνση θέσης μνήμης	περιεχόμενο θέσης μνήμης	διεύθυνση θέσης μνήμης	περιεχόμενο θέσης μνήμης
100	01	110	53
101	8A	111	F1
102	71	112	2B
103	E6	113	F2
104	4B	114	48
105	90	115	A7
106	54	116	79
107	F3	117	92
108	18	118	79
109	87	119	F3
10A	29	11A	5E
10B	94	11B	93
10C	3D	11C	3C
10D	87	11D	DF
10E	7C	11E	31
10F	8A	11F	EF

Α. Απεικόνιση ανάλογη του μεγέθους (little endian)

LOADHW R1, 10B //Φόρτωσε μισή λέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διευθυνσιοδότησης (direct addressing mode)

LOADSB R2, 10B //Φόρτωσε προσημασμένη ψηφιολέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διευθυνσιοδότησης (direct addressing mode)

Λύση

Με δεδομένο ότι οι **καταχωρητές είναι των 16 bits**, τα περιεχόμενα τους μετά την εκτέλεση των εντολών, θα είναι τα εξής: R1 = 3D94, R2 = FF94

Θέμα Σεπτεμβρίου 2020 με little – endian και big – endian

Θεωρήστε υπολογιστή του οποίου ο επεξεργαστής έχει 16 καταχωρητές των 32 δυαδικών ψηφίων ο καθένας και κύρια μνήμη με οργάνωση 8 δυαδικών ψηφίων ανά θέση μνήμης. Το τρέχον περιεχόμενο της μνήμης δίνεται στο σχήμα. Οι προσημασμένοι αριθμοί είναι σε αναπαράσταση συμπληρώματος ως προς 2. Μισή λέξη: 16 δυαδικά ψηφία. Για κάθε μια των κάτωθι περιπτώσεων, να δώσετε το περιεχόμενο του καταχωρητή μετά την εκτέλεση της εντολής στο δεκαεξαδικό χρησιμοποιώντας κεφαλαίους χαρακτήρες. Προσοχή να μην παρεμβάλλονται κενά ή άλλα σύμβολα.

κύρια μνήμη		κύρια μνήμη	
διεύθυνση θέσης μνήμης στο δεκαεξαδικό	περιεχόμενο θέσης μνήμης στο δεκαεξαδικό	διεύθυνση θέσης μνήμης στο δεκαεξαδικό	περιεχόμενο θέσης μνήμης στο δεκαεξαδικό
110	87	227	98
111	13	228	FA
112	E1	229	19
113	73	22A	FE
114	94	22B	A2
115	5F	22C	41
116	A0	22D	F3
117	16	22E	64
118	C1	22F	85
119	56	230	1F
11A	85	231	AE
11B	32	232	68
11C	AE	233	8A
11D	4C	234	1B
11E	F0	235	98
11F	64	236	5C
120	89	237	A1
121	1A	238	1D
122	A0	239	C3
123	65	23A	58
124	8F	23B	89
125	10	23C	37
126	92	23D	97
127	34	23E	13
128	B3	23F	E5
129	32	240	63
12A	F1	241	9A
12B	20	242	5C
12C	F4	243	AD
12D	F3	244	46
12E	78	245	95

Απεικόνιση αντιστρόφως ανάλογη του μεγέθους (big endian) LOADSHW R1, 110 //Φόρτωσε προσημασμένη μισή λέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διευθυνσιοδότησης (direct addressing mode) R1=[ ]. Απεικόνιση ανάλογη του μεγέθους (little endian) LOADSHW R2, 22B //Φόρτωσε προσημασμένη μισή λέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διευθυνσιοδότησης (direct addressing mode) R2=[ ].

Λύση

R1 ← FFFF8713, διότι 8 = **1**000 (αν είχαμε LOADUHW R1, 110 τότε R1 ← 00008713)

R2 ← 000041A2, διότι 4 = **0**100 (αν είχαμε LOADUHW R2, 22B τότε R2 ← 000041A2)

Θέμα προόδου 2021 με little – endian και big – endian

Θεωρήστε υπολογιστή του οποίου ο επεξεργαστής έχει 32 καταχωρητές των 16 δυαδικών ψηφίων ο καθένας και κύρια μνήμη με οργάνωση 8 δυαδικών ψηφίων ανά θέση μνήμης. Το τρέχον περιεχόμενο της μνήμης δίνεται στο σχήμα. Οι προσημασμένοι αριθμοί είναι σε αναπαράσταση συμπληρώματος ως προς 2. Μισή λέξη: 16 δυαδικά ψηφία. Για κάθε μια των κάτωθι περιπτώσεων, να δώσετε το περιεχόμενο του καταχωρητή στο δεκαεξαδικό (χρησιμοποιώντας κεφαλαίους χαρακτήρες), μετά την εκτέλεση της εντολής. Προσοχή στην απάντησή σας να μην παρεμβάλλονται κενά ή άλλα σύμβολα.

διεύθυνση θέσης μνήμης	περιεχόμενο θέσης μνήμης	διεύθυνση θέσης μνήμης	περιεχόμενο θέσης μνήμης
100	01	110	53
101	8A	111	F1
102	71	112	2B
103	E6	113	F2
104	4B	114	48
105	90	115	A7
106	54	116	79
107	F3	117	92
108	18	118	79
109	87	119	F3
10A	29	11A	5E
10B	94	11B	93
10C	3D	11C	3C
10D	87	11D	DF
10E	7C	11E	31
10F	8A	11F	EF

A. Απεικόνιση ανάλογη του μεγέθους (little endian)

LOADHW R1, 112 //Φόρτωσε μισή λέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διευθυνσιοδότησης (direct addressing mode) R1 = .

LOADSB R2, 112 //Φόρτωσε προσημασμένη ψηφιολέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διευθυνσιοδότησης (direct addressing mode) R2 = .

LOADUB R3, 112 //Φόρτωσε μη προσημασμένη ψηφιολέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διευθυνσιοδότησης (direct addressing mode) R3 = .

B. Απεικόνιση αντιστρόφως ανάλογη του μεγέθους (big endian)

LOADHW R4, 112 //Φόρτωσε μισή λέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διευθυνσιοδότησης (direct addressing mode) R4 = .

LOADSB R5, 113 //Φόρτωσε προσημασμένη ψηφιολέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διευθυνσιοδότησης (direct addressing mode) R5 = .

LOADUB R6, 113 //Φόρτωσε μη προσημασμένη ψηφιολέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διευθυνσιοδότησης (direct addressing mode) R6 = .

Λύση

R1 ← F22B, επειδή οι καταχωρητές είναι των 16 bits δεν γίνεται καμία επέκταση προσήμου.

R2 ← 002B, η επέκταση προσήμου γίνεται με το “0”, διότι το 2 = 0010 (S).

R3 ← 002B, η επέκταση προσήμου γίνεται με το “0”, λόγω θετικών αριθμών (U).

R4 ← 2BF2, επειδή οι καταχωρητές είναι των 16 bits δεν γίνεται καμία επέκταση προσήμου.

R5 ← FFF2, η επέκταση προσήμου γίνεται με το “1”, διότι το F = 1111 (S).

R6 ← 00F2, η επέκταση προσήμου γίνεται με το “0”, λόγω θετικών αριθμών (U).

Θέμα Ιουνίου 2020

Θεωρήστε υπολογιστή του οποίου ο επεξεργαστής έχει 16 καταχωρητές των 32 δυαδικών ψηφίων ο καθένας και κύρια μνήμη με οργάνωση 8 δυαδικών ψηφίων ανά θέση μνήμης. Το τρέχον περιεχόμενο της μνήμης δίνεται στο σχήμα. Οι προσημασμένοι αριθμοί είναι σε αναπαράσταση συμπληρώματος ως προς 2. Μισή λέξη: 16 δυαδικά ψηφία. Για κάθε μια των κάτωθι περιπτώσεων να δώσετε στο περιεχόμενο του καταχωρητή μετά την εκτέλεση της εντολής στο δεκαεξαδικό. Προσοχή να μην παρεμβάλλονται κενά ή άλλα σύμβολα.

κύρια μνήμη	
διεύθυνση θέσης μνήμης στο δεκαεξαδικό	περιεχόμενο θέσης μνήμης στο δεκαεξαδικό
110	87
111	13
112	E1
113	73
114	94
115	5F
116	A0
117	16
118	C1
119	56
11A	85
11B	32
11C	AE
11D	4C
11E	F0
11F	64
120	89
121	1A
122	A0
123	65
124	8F
125	10
126	92
127	34
128	B3
129	32
12A	F1
12B	20
12C	F4
12D	F3
12E	78

κύρια μνήμη	
διεύθυνση θέσης μνήμης στο δεκαεξαδικό	περιεχόμενο θέσης μνήμης στο δεκαεξαδικό
227	98
228	FA
229	19
22A	FE
22B	A2
22C	41
22D	F3
22E	64
22F	85
230	1F
231	AE
232	68
233	8A
234	1B
235	98
236	5C
237	A1
238	1D
239	C3
23A	58
23B	89
23C	37
23D	97
23E	13
23F	E5
240	63
241	9A
242	5C
243	AD
244	46
245	95

Α. Απεικόνιση αντιστρόφως ανάλογη του μεγέθους (big endian) LOADSHW R1, 110 //Φόρτωσε προσημασμένη μισή λέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διευθυνσιοδότησης (direct addressing mode) R1 = [ ].

Β. Απεικόνιση ανάλογη του μεγέθους (little endian) LOADSHW R2, 22B // Φόρτωσε προσημασμένη μισή λέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διευθυνσιοδότησης (direct addressing mode) R2 = [ ].

Λύση

R1 ← FFFF8713, διότι 8 = **1**000 (αν είχαμε LOADUHW R1, 110 τότε R1 ← 00008713)

R2 ← 000041A2, διότι 4 = **0**100 (αν είχαμε LOADUHW R2, 22B τότε R2 ← 000041A2)

Θέμα Σεπτεμβρίου 2020

Θεωρήστε υπολογιστή του οποίου ο επεξεργαστής έχει 16 καταχωρητές των 32 δυαδικών ψηφίων ο καθένας και κύρια μνήμη με οργάνωση 8 δυαδικών ψηφίων ανά θέση μνήμης. Το τρέχον περιεχόμενο της μνήμης δίνεται στο σχήμα. Οι προσημασμένοι αριθμοί είναι σε αναπαράσταση συμπληρώματος ως προς 2. Μισή λέξη: 16 δυαδικά ψηφία. Για κάθε μια των κάτωθι περιπτώσεων να δώσετε στο περιεχόμενο του καταχωρητή μετά την εκτέλεση της εντολής στο δεκαεξαδικό χρησιμοποιώντας κεφαλαίους χαρακτήρες. Προσοχή να μην παρεμβάλλονται κενά ή άλλα σύμβολα.

κύρια μνήμη	
διεύθυνση θέσης μνήμης στο δεκαεξαδικό	περιεχόμενο θέσης μνήμης στο δεκαεξαδικό
110	87
111	13
112	E1
113	73
114	94
115	5F
116	A0
117	16
118	C1
119	56
11A	85
11B	32
11C	AE
11D	4C
11E	F0
11F	64
120	89
121	1A
122	A0
123	65
124	8F
125	10
126	92
127	34
128	B3
129	32
12A	F1
12B	20
12C	F4
12D	F3
12E	78

κύρια μνήμη	
διεύθυνση θέσης μνήμης στο δεκαεξαδικό	περιεχόμενο θέσης μνήμης στο δεκαεξαδικό
227	98
228	FA
229	19
22A	FE
22B	A2
22C	41
22D	F3
22E	64
22F	85
230	1F
231	AE
232	68
233	8A
234	1B
235	98
236	5C
237	A1
238	1D
239	C3
23A	58
23B	89
23C	37
23D	97
23E	13
23F	E5
240	63
241	9A
242	5C
243	AD
244	46
245	95

Απεικόνιση αντιστρόφως ανάλογη του μεγέθους (big endian) LOADSHW R1, 113 //Φόρτωσε προσημασμένη μισή λέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διευθυνσιοδότησης (direct addressing mode) R1 = [ ].

Απεικόνιση ανάλογη του μεγέθους (little endian) LOADSHW R2, 22E // Φόρτωσε προσημασμένη μισή λέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διευθυνσιοδότησης (direct addressing mode) R2 = [ ].

Λύση

R1 = 00007394

R2 = FFFF8564

Θέμα Σεπτεμβρίου 2020

Θεωρήστε υπολογιστή του οποίου ο επεξεργαστής έχει 16 καταχωρητές των 32 δυαδικών ψηφίων ο καθένας και κύρια μνήμη με οργάνωση 8 δυαδικών ψηφίων ανά θέση μνήμης. Το τρέχον περιεχόμενο της μνήμης δίνεται στο σχήμα. Οι προσημασμένοι αριθμοί είναι σε αναπαράσταση συμπληρώματος ως προς 2. Μισή λέξη: 16 δυαδικά ψηφία. Για κάθε μια των κάτωθι περιπτώσεων να δώσετε στο περιεχόμενο του καταχωρητή μετά την εκτέλεση της εντολής στο δεκαεξαδικό. Προσοχή να μην παρεμβάλλονται κενά ή άλλα σύμβολα.

κύρια μνήμη	
διεύθυνση θέσης μνήμης στο δεκαεξαδικό	περιεχόμενο θέσης μνήμης στο δεκαεξαδικό
110	87
111	13
112	E1
113	73
114	94
115	5F
116	A0
117	16
118	C1
119	56
11A	85
11B	32
11C	AE
11D	4C
11E	F0
11F	64
120	89
121	1A
122	A9
123	65
124	8E
125	10
126	92
127	34
128	B3
129	32
12A	F1
12B	20
12C	F4
12D	F3
12E	78

κύρια μνήμη	
διεύθυνση θέσης μνήμης στο δεκαεξαδικό	περιεχόμενο θέσης μνήμης στο δεκαεξαδικό
227	98
228	FA
229	19
22A	FE
22B	A2
22C	41
22D	F3
22E	64
22F	85
230	1F
231	AE
232	68
233	8A
234	1B
235	98
236	5C
237	A1
238	1D
239	C3
23A	58
23B	89
23C	37
23D	97
23E	13
23F	E5
240	63
241	9A
242	5C
243	AD
244	46
245	95

Α. Απεικόνιση αντιστρόφως ανάλογη του μεγέθους (big endian) LOADSHW R1, 128 //Φόρτωσε προσημασμένη μισή λέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διευθυνσιοδότησης (direct addressing mode) R1 = [ ].

Β. Απεικόνιση ανάλογη του μεγέθους (little endian) LOADSHW R2, 243 // Φόρτωσε προσημασμένη μισή λέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διευθυνσιοδότησης (direct addressing mode) R2 = [ ].

Λύση

FFFFB332

000046AD

Άσκηση με μετατροπή από σταθερή σε κινητή υποδιαστολή

Θεωρήστε υπολογιστή που διαθέτει εντολές επεξεργασίας προσημασμένων ακέραιων αριθμών των 8 και 16 δυαδικών ψηφίων σε παράσταση συμπληρώματος ως προς 2 και αριθμούς σε παράσταση κινητής υποδιαστολής σύμφωνα με το πρότυπο IEEE 754 απλής ακρίβειας. Ο υπολογιστής διαθέτει μόνο ένα δυαδικό αθροιστή των 16 δυαδικών ψηφίων που μπορεί να εκτελέσει και προσθέσεις μεταξύ αριθμών σε παράσταση συμπληρώματος ως προς 2.

1. Να δώσετε την παράσταση στον υπολογιστή των αριθμών: $\alpha = +2A_{(16)}$, $\beta = -7E_{(16)}$, $\gamma = +F3_{(16)}$ και $\delta = +1A9F_{(16)}$ με 8 δυαδικά ψηφία.
2. Να δείξετε πως εκτελείται η πράξη β+δ στον υπολογιστή και να σχολιάσετε το αποτέλεσμα.
3. Να δώσετε την παράσταση ε. – $BF0091_{(16)}$ σε παράσταση κινητής υποδιαστολής.

Λύση

1. α = +2A = 0010 1010, προσημασμένος αριθμός, είναι θετικός, αφού ξεκινά με “0”.

β = -7E → 7E = 0111 1110 → 1000 0001 + 1 = 1000 0010, προσημασμένος αριθμός, είναι αρνητικός, αφού ξεκινά με “1”.

γ = +F3 = 1111 0011 **δεν** μπορεί να αναπαρασταθεί σωστά με 8 bits, διότι αφού είναι θετικός, θα έπρεπε να ξεκινά με "0". Για τη σωστή αναπαράστασή του απαιτούνται περισσότερα από 8 bits.

δ = +1A9F δεν μπορεί να παρασταθεί σωστά με 8 bits. Αν υποθέσουμε ότι η αποθήκευση είναι μορφής big endian/little endian, θα αποθηκευτεί μόνο το μισό περιεχόμενο του αριθμού, δηλαδή 1A/9F.

2. Επειδή ο αθροιστής είναι των 16 bits θα πρέπει οι αριθμοί που θα προστεθούν μέσω αυτού, να διαθέτουν αυτό το μέγεθος.

β = -7E = 1000 0010 → 1111 1111 1000 0010

δ = 1A9F → 0001 1010 1001 1111

1) 0001101000100001

Για τον έλεγχο του αποτελέσματος πρέπει να ελέγξουμε τη σημαία (flag) της υπερχείλισης (Y ή V), διότι οι αριθμοί που αθροίζονται είναι προσημασμένοι. Η σημαία $Y = V = \kappa_{v-1} \oplus \kappa_{v-2} = 1 \oplus 1 = 0$, άρα δεν υπάρχει υπερχείλιση και το αποτέλεσμα της πράξης είναι σωστό. Εναλλακτικά, μπορούμε να διαπιστώσουμε άμεσα, χωρίς καμία πράξη, ότι η πράξη είναι σωστή καθώς οι αριθμοί που αθροίζονται είναι ετερόσημοι. Σε μια τέτοια περίπτωση αποκλείεται να υπάρχει υπερχείλιση.

3. Για να μετατραπεί ένας αριθμός από παράσταση σταθερής υποδιαστολής σε παράσταση κινητής υποδιαστολής θα πρέπει πρώτα να τον **κανονικοποιήσουμε**, δηλαδή θα πρέπει να τον μετατρέψουμε στη μορφή **1.□□□ ….□□□**, όπου το σύμβολο του □ δηλώνει δυαδικό ψηφίο. Ο αριθμός που θα μετατρέψουμε σε κινητή υποδιαστολή είναι ο -BF0091. Θα αγνοήσουμε καταρχήν το πρόσημο του αριθμού, το οποίο θα το λάβουμε υπόψη στο τέλος. Ο υπόλοιπος αριθμός είναι ο BF0091 = 1011 1111 0000 0000 1001 0001 = 1011 1111 0000 0000 1001 0001.0 x 2^0 = 1.011 1111 0000 0000 1001 0001 x 2^{23} διότι όσες θέσεις μετακινείται η υποδιαστολή προς τα αριστερά, τόσες θέσεις αυξάνεται αντίστοιχα ο εκθέτης. Στη συνέχεια θα πρέπει να προσθέσουμε την πόλωση στον εκθέτη, δηλαδή την τιμή 127 (για απλή ακρίβεια), οπότε αυτός θα γίνει 23 + 127 = 150 = 10010110, διότι 1 x 2^7 + 1 x 2^4 + 1 x 2^2 +1 x 2^1 = 150. Η τελική απάντηση είναι:

1	10010110	011 1111 0000 0000 1001 0001
πρόσημο	εκθέτης	συντελεστής

Το πρόσημο είναι "1", διότι ο αριθμός είναι αρνητικός.

Για πρότυπο IEEE754 υπάρχουν δύο ακρίβειες με τις οποίες μπορούν να παρασταθούν οι αριθμοί κινητής υποδιαστολής. Η απλή ακρίβεια εμφανίζει ένα αριθμό κινητής υποδιαστολής με 32 bits, ενώ η διπλή ακρίβεια εμφανίζει ένα αριθμό κινητής υποδιαστολής με 64 bits. Στην πρώτη περίπτωση η πόλωση είναι το 127, ενώ στη δεύτερη περίπτωση είναι το 1023. Στη μετατροπή από κινητή υποδιαστολή σε σταθερή, η πόλωση αφαιρείται από τον εκθέτη, ενώ στη μετατροπή από σταθερή σε κινητή υποδιαστολή, η πόλωση προστίθεται στον εκθέτη.

Θέμα προόδου με άθροισμα αριθμών σε μορφή IEEE 754 – Ιανουάριος 2022

Δίνονται οι αριθμοί A = 0**10000110**0001000000000000000000 και B = 0**10000101**0011000000000000000000 σε αναπαράσταση κινητής υποδιαστολής, σύμφωνα με το πρότυπο IEEE 754. Να υπολογίσετε και να δώσετε το **άθροισμά** τους σε αναπαράσταση κινητής υποδιαστολής, σύμφωνα με το ίδιο πρότυπο.

Λύση

Ένας τρόπος είναι να μετατρέψουμε τους αριθμούς σε παράσταση σταθερής υποδιαστολής, να τους αθροίσουμε σε αυτή τη μορφή και μετά να μετατρέψουμε το αποτέλεσμα πίσω σε παράσταση κινητής υποδιαστολής. Πιο συγκεκριμένα, υπολογίζουμε για τον κάθε αριθμό την τιμή του εκθέτη του, και έχουμε: Για τον αριθμό A, ο E = 128 + 6 = 134 και για τον αριθμό B, ο E = 128 + 5 = 133. Επειδή και για τους δύο αριθμούς ο 0 < E < 255, ισχύει ο τύπος: $N = (-1)^{\pi} * 2^{E-\text{πόλωση}} * 1.\Sigma_{\kappa}$ οπότε για τον αριθμό A έχουμε ότι είναι: $N = (-1)^0 * 2^{134-127} * 1.0625 = 2^7 * 1.0625 = 136$, όπου $\Sigma_{\kappa} = 1/16 = 0.0625$ ενώ για τον αριθμό B έχουμε ότι είναι: $N = (-1)^0 * 2^{133-127} * 1.1875 = 2^6 * 1.1875 = 76$, όπου $\Sigma_{\kappa} = 1/8 + 1/16 = 0.1875$. Το άθροισμά τους

σε παράσταση σταθερής υποδιαστολής είναι 136 + 76 = 212. Στη συνέχεια, πρέπει να μετατρέψουμε το άθροισα αυτό πίσω σε αναπαράσταση κινητής υποδιαστολής, σύμφωνα με το πρότυπο IEEE 754. Θα έχουμε ότι 212 = 1101 0100 = 1101 0100.0 x 2^0 = 1.1010100 x 2^7 Τ Τώρα η πόλωση πρέπει να προστεθεί στον εκθέτη, ο οποίος θα γίνει 7 + 127 = 134 = 1000 0110, οπότε ο αριθμός είναι:

0 **1000 0110** **10101000000000000000000**

Ένας **δεύτερος** τρόπος για να προσθέσουμε τους δύο αριθμούς απευθείας στην παράσταση κινητής υποδιαστολής είναι να ελέγξουμε τους εκθέτες τους και εφόσον δεν είναι ίσοι, όπως συμβαίνει στην προκειμένη περίπτωση, θα πρέπει ο μικρότερος εκθέτης να πάρει την τιμή του μεγαλύτερου εκθέτη και αυτό θα γίνει με ολίσθηση του αριθμού με το μικρότερο εκθέτη προς τα δεξιά τόσες θέσεις όσο είναι η διαφορά των εκθετών (με άλλα λόγια η υποδιαστολή στον αριθμό με το μικρότερο εκθέτη θα μετακινηθεί προς τα αριστερά τόσες θέσεις όσο είναι η διαφορά των εκθετών), στην προκειμένη περίπτωση αυτή ισούται με 134 – 133 = 1 θέση. Δηλαδή ο A = 2^{134} x 1.**00010000000000000000000** και ο B = 2^{133} x 1.**00110000000000000000000**, οπότε ο B θα γίνει: B = 2^{134} x 0.**10011000000000000000000** και στη συνέχεια οι αριθμοί μπορούν να αθροιστούν αναφορικά με τους συντελεστές τους, αφού απέκτησαν **ίδιο** εκθέτη. Επομένως, από το άθροισμα των συντελεστών τους θα έχουμε:

1.**00010000000000000000000**
0.**10011000000000000000000**
1.10101000000000000000000

Παρατηρούμε ότι καταλήξαμε στο ίδιο αποτέλεσμα με πριν, δηλ. το άθροισμα των δύο αριθμών σε μορφή κινητής υποδιαστολής σύμφωνα με το πρότυπο IEEE 754 είναι: 0 **1000 0110** **10101000000000000000000.**

1η Άσκηση με τεχνική μικροπρογραμματισμού δύο επιπέδων

Θεωρήστε ένα μικροεπεξεργαστή με μονάδα ελέγχου που έχει σχεδιαστεί με την τεχνική του **μικροπρογραμματισμού ενός επιπέδου**. Το σύνολο των μικροπρογραμμάτων καταλαμβάνει 10.000 μικροεντολές της μνήμης ελέγχου και υπάρχουν 500 διαφορετικές μικροεντολές.

α) Ποιο είναι το πλεονέκτημα και ποιο το μειονέκτημα χρήσης και **δεύτερου επιπέδου μικροπρογραμματισμού**; Μπορείτε να χρησιμοποιήσετε αυτή την τεχνική στον παραπάνω μικροεπεξεργαστή;

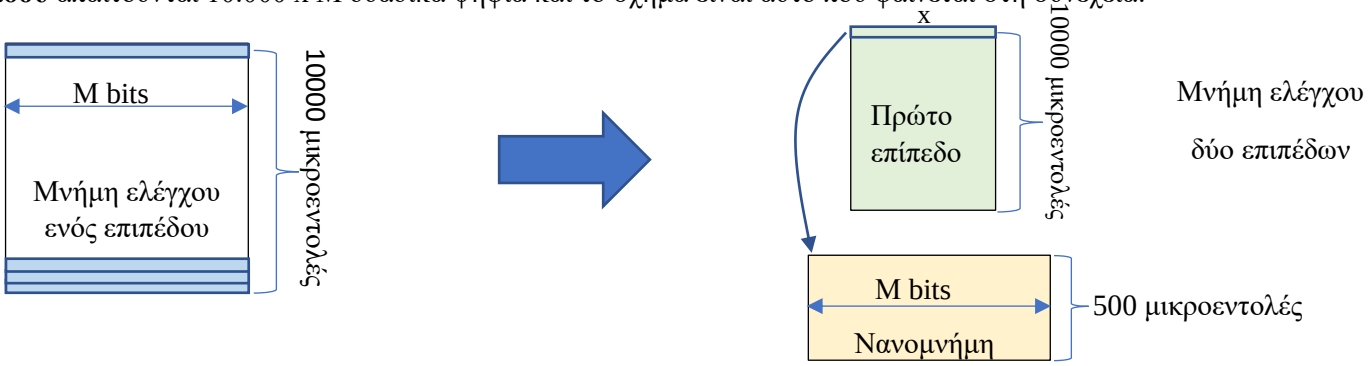
β) Καθορίστε τον αριθμό των δυαδικών ψηφίων που πρέπει να έχει η μικροεντολή προκειμένου να επιτυγχάνουμε μείωση του απαιτούμενου αποθηκευτικού χώρου με τη χρήση του 2ου επιπέδου κατά τουλάχιστον i) 50%, ii) 75% και iii) 90%.

Λύση

Ερώτημα (α): Εάν χρησιμοποιήσουμε και δεύτερο επίπεδο μικροπρογραμματισμού τότε μπορούμε να επιτύχουμε μείωση της χωρητικότητας που απαιτείται για τη μνήμη ελέγχου ενός επιπέδου. Για να μπορεί να χρησιμοποιηθεί αυτή η τεχνική θα πρέπει να εμφανίζονται στα μικροπρογράμματα συχνά όμοιες μικροεντολές. Τότε στη μνήμη ελέγχου δεύτερου επιπέδου (νανομνήμη) αποθηκεύονται όλες οι διαφορετικές μικροεντολές (νανοεντολές) ενώ στην μνήμη ελέγχου πρώτου επιπέδου αποθηκεύονται τα μικροπρογράμματα που τώρα αποτελούνται από διευθύνσεις της νανομνήμης. Αυτό το χαρακτηριστικό ισχύει στην περίπτωση του παραπάνω μικροεπεξεργαστή αφού τα μικροπρογράμματα καταλαμβάνουν χώρο 10.000 μικροεντολών και απαρτίζονται μόνο από 500 διαφορετικές μικροεντολές. Άρα κατά μέσο όρο κάθε μικροεντολή επαναλαμβάνεται 10.000/500= 20 φορές, αριθμός αρκετά μεγάλος για να χρησιμοποιηθεί η τεχνική του δεύτερου επιπέδου.

Το μειονέκτημα της χρήσης και δεύτερου επιπέδου είναι ότι απαιτούνται δύο προσπελάσεις, μία στη μνήμη του πρώτου επιπέδου και μία στην μνήμη του δεύτερου. Αντίθετα με τη χρήση ενός επιπέδου απαιτείται μόνο μία προσπέλαση οπότε είναι ταχύτερη η εκτέλεση κάθε μικροεντολής.

Ερώτημα (β): Έστω ότι το μήκος της κάθε μικροεντολής είναι Μ. Τότε για την υλοποίηση της μνήμης ελέγχου **ενός επιπέδου** απαιτούνται 10.000 x Μ δυαδικά ψηφία και το σχήμα είναι αυτό που φαίνεται στη συνέχεια:



Όταν η μνήμη ελέγχου υλοποιείται με την τεχνική του νανοπρογραμματισμού, τότε το δεύτερο επίπεδο αποτελείται από 500 x Μ δυαδικά ψηφία, όσα ακριβώς απαιτούνται για την αποθήκευση των 500 διαφορετικών μικροεντολών. Το πρώτο επίπεδο έχει 10.000 θέσεις, όσες απαιτούνται για τα μικροπρογράμματα στην περίπτωση του ενός επιπέδου, μόνο που τώρα το **μήκος κάθε θέσης είναι διαφορετικό** αφού είναι δείκτης στην νανομνήμη. Αφού λοιπόν η νανομνήμη έχει 500 θέσεις, κάθε δείκτης θα έχει μήκος $2^x \geq 500 \Rightarrow x = 9$ δυαδικά ψηφία. Επομένως στην περίπτωση του νανοπρογραμματισμού απαιτούνται 10.000 x 9 + 500 x Μ δυαδικά ψηφία. Ο λόγος μείωσης του απαιτούμενου αποθηκευτικού χώρου για τις δύο τεχνικές είναι: $\Lambda = \frac{\text{αρχική χωρητικότητα} - \text{τελική χωρητικότητα}}{\text{αρχική χωρητικότητα}} = \frac{10000 \text{ x } M - (10.000 \text{ x } 9 + 500 \text{ x } M)}{10000 \text{ x } M}$.

i) Πρέπει $\Lambda \geq 50\% \Rightarrow \frac{10000 \text{ x } M - (10.000 \text{ x } 9 + 500 \text{ x } M)}{10000 \text{x} M} \geq 0.5 \Rightarrow M \geq 20$.
ii) Ομοίως για $\Lambda \geq 75\%$, πρέπει $M \geq 45$ και iii) Ομοίως για $\Lambda \geq 90\%$, πρέπει $M \geq 180$.

2η Άσκηση με τεχνική μικροπρογραμματισμού δύο επιπέδων – Θέμα Σεπτέμβριος 2021

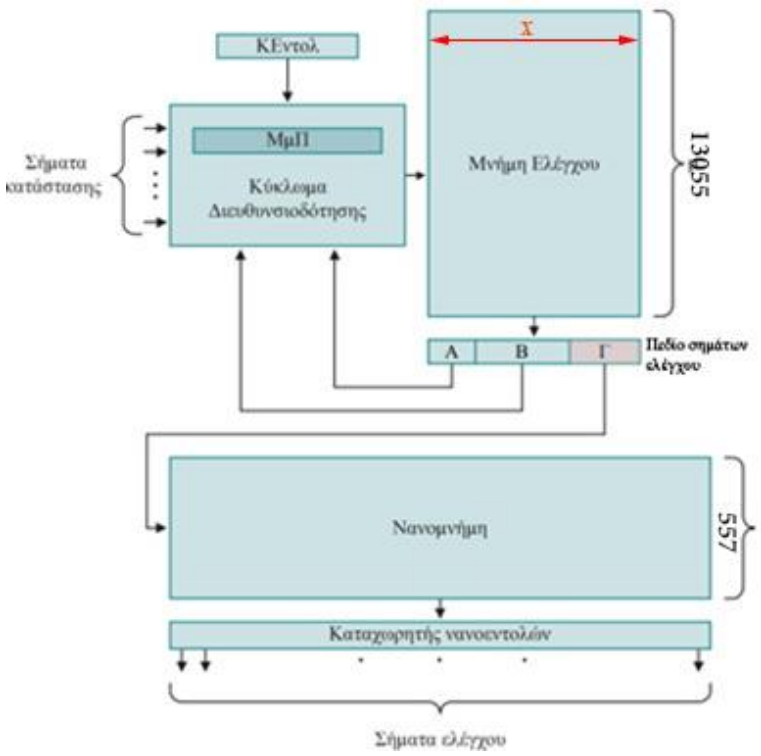
Θεωρήστε ένα μικροεπεξεργαστή με μονάδα ελέγχου που έχει σχεδιαστεί με την τεχνική του μικροπρογραμματισμού δύο επιπέδων (νανοπρογραμματισμός). Το σύνολο των μικροπρογραμμάτων αποτελείται από 13055 μικροεντολές και υπάρχουν 557 διαφορετικές μικροεντολές. Το μήκος κάθε μικροεντολής είναι 119 δυαδικά ψηφία εκ' των οποίων τα 117 είναι σήματα ελέγχου. Να υπολογίσετε την επί τοις εκατό μείωση του απαιτούμενου αποθηκευτικού χώρου σε σχέση με την περίπτωση που η μονάδα ελέγχου είχε σχεδιαστεί με την τεχνική του μικροπρογραμματισμού ενός επιπέδου.

Να δώσετε μόνο το **ακέραιο** μέρος της τιμή που υπολογίσατε [ ]%

Λύση

Για **ένα επίπεδο** ισχύει ότι η χωρητικότητα του είναι 13055 μικροεντολές * 119 bits = 1553545 bits.

Για τα **δύο επίπεδα** ισχύει ότι στο 2ο επίπεδο υπάρχουν 557 διαφορετικές μικροεντολές. Αυτό σημαίνει ότι το πρώτο επίπεδο πρέπει να έχει ένα μήκος μικροεντολής x bits όπου: $2^x \geq 557 \Rightarrow x = 10$ bits. Η **συνολική** χωρητικότητα και των δύο επιπέδων θα είναι: Χωρητικότητα 1ου επιπέδου + Χωρητικότητα 2ου επιπέδου = 13055 * (10 + 2) bits + 557 μικροεντολές * 117 bits = 221829. Δηλαδή, στην περίπτωση χρήσης της νανομνήμης (δύο επιπέδων) για να υπολογίσουμε τη χωρητικότητα του 1ου επιπέδου θα πρέπει να πολλαπλασιάσουμε το πλήθος των συνολικών μικροεντολών με το άθροισμα (πλήθος bits που απαιτούνται για την προσπέλαση της νανομνήμης + (διαφορά bits μικροεντολής – bits σημάτων ελέγχου)) και για να υπολογίσουμε τη χωρητικότητα του 2ου επιπέδου θα πρέπει να πολλαπλασιάσουμε το πλήθος των διαφορετικών μικροεντολών με τα bits (σήματα) ελέγχου. Το ποσοστό μείωσης του απαιτούμενου αποθηκευτικού χώρου με τη χρήση της τεχνικής της νανομνήμης είναι: $\Lambda = \frac{\text{αρχική χωρητικότητα} - \text{τελική χωρητικότητα}}{\text{αρχική χωρητικότητα}} = \frac{1553545 - 221829}{1553545} = 0.857 = 85.7\% \rightarrow 85\%$



Σημείωση: σε τέτοιου είδους ασκήσεις ισχύει ο τύπος: Για υλοποίηση με 1 επίπεδο: α = συνολικές εντολές * μέγεθος εντολής, ενώ για 2 επίπεδα: β = (μέγεθος εντολής - σήματα ελέγχου + ceiling(log2(διαφορετικές εντολές))) * συνολικές εντολές + διαφορετικές εντολές * σήματα ελέγχου. Το πρώτο μέλος του αθροίσματος είναι το πρώτο επίπεδο, και αντίστοιχα για το δεύτερο. Άρα η μείωση είναι (α-β)/α.

3η Άσκηση με τεχνική μικροπρογραμματισμού δύο επιπέδων από ΕΑΠ

Θεωρήστε ότι σε ένα επεξεργαστή η μονάδα ελέγχου έχει σχεδιαστεί με την τεχνική του μικροπρογραμματισμού ενός επιπέδου. Θεωρήστε ότι κάθε μικροεντολή είναι των 100 δυαδικών ψηφίων, το σύνολο των μικροπρογραμμάτων καταλαμβάνει 2048 θέσεις της μνήμης ελέγχου και υπάρχουν 200 διαφορετικές μικροεντολές. Αν υλοποιήσουμε τη μνήμη ελέγχου με την τεχνική του μικροπρογραμματισμού δύο επιπέδων (νανοπρογραμματισμός), τι θα κερδίσουμε και τι θα χάσουμε σε ταχύτητα και συνολική χωρητικότητα της μνήμης ελέγχου;

Λύση

Για την υλοποίηση της μνήμης ελέγχου **ενός επιπέδου** απαιτούνται 2048 * 100 = **204800** δυαδικά ψηφία. Όταν η μνήμη ελέγχου υλοποιείται με την τεχνική του νανοπρογραμματισμού, το δεύτερο επίπεδο αποτελείται από 200 θέσεις, όσες είναι οι διαφορετικές μικροεντολές, των 100 δυαδικών ψηφίων η καθεμία, όσο το μήκος της κάθε μικροεντολής. Το πρώτο επίπεδο έχει 2048 θέσεις, όσες θέσεις καταλαμβάνουν τα μικροπρογράμματα στην περίπτωση ενός επιπέδου, των 8 δυαδικών ψηφίων η καθεμία, διότι $2^x \geq 200 \Rightarrow x = 8$ bits. Επομένως στην περίπτωση του νανοπρογραμματισμού απαιτούνται 2048 * 8 bits + 200 * 100 bits = **36384** δυαδικά ψηφία. Παρατηρούμε λοιπόν ότι στην περίπτωση που χρησιμοποιούμε την τεχνική του νανοπρογραμματισμού απαιτείται λιγότερο από το ένα πέμπτο της χωρητικότητας που απαιτείται στην περίπτωση της μνήμης ελέγχου ενός επιπέδου. Όσον αφορά όμως στην ταχύτητα στην περίπτωση της μνήμης ελέγχου ενός επιπέδου για να εκτελεστεί μία μικροεντολή αρκεί μία προσπέλαση της μνήμης ελέγχου, ενώ στην περίπτωση του νανοπρογραμματισμού χρειάζονται δύο προσπελάσεις, μία της μνήμης του πρώτου επιπέδου και μία της μνήμης του δεύτερου.

4η Άσκηση με τεχνική μικροπρογραμματισμού δύο επιπέδων από το μάθημα

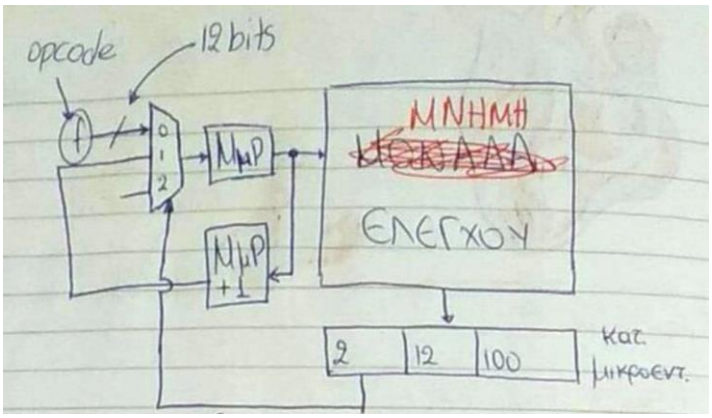
Να θεωρήσετε μονάδα ελέγχου με την τεχνική του μικροπρογραμματισμού **250** εντολών γλώσσας μηχανής. Κάθε εντολή καθορίζει τιμή **100** σημάτων ελέγχου. Το μικροπρόγραμμα **προσκόμισης εντολής** αποτελείται από 5 μικροεντολές. Κάθε μικροπρόγραμμα **εκτέλεσης εντολών** αποτελείται από 10 μικροεντολές. Να υπολογισθεί: α) το μέγεθος μνήμης ελέγχου, β) να σχεδιαστεί η μνήμη ελέγχου, γ) τρόπος μείωσης χωρητικότητας απαιτούμενης μνήμης ελέγχου, να υπολογισθεί η χωρητικότητα της νέας μνήμης ελέγχου, να σχεδιαστεί η νέα μνήμη ελέγχου.

Λύση

α) Για να υπολογίσουμε **το μέγεθος της μνήμης ελέγχου** θα προσθέσουμε τις 5 μικροεντολές από το μικροπρόγραμμα προσκόμισης εντολής και τις 250 εντολές γλώσσας μηχανής. Πιο συγκεκριμένα, θα έχουμε ότι: Μέγεθος μνήμης ελέγχου = 5 μικροεντολές + 250 εντολές * 10 $\frac{\text{μικροεντολές}}{\text{εντολή}}$ = 2505 μικροεντολές. Για την κωδικοποίηση όλων των μικροεντολών απαιτούνται $2^x \geq 2505 \Rightarrow x = 12$ bits. Στη συνέχεια φαίνεται η μορφή μιας μικροεντολής, που είναι:

2 bits	12 bits	100 bits
	διεύθυνση	σήματα ελέγχου

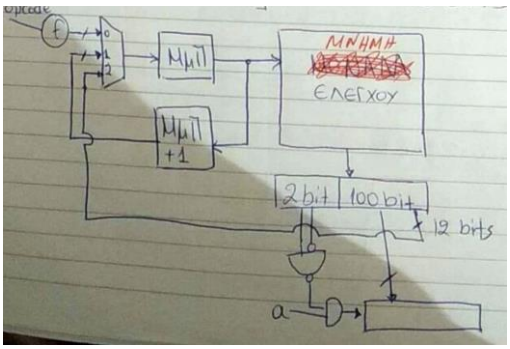
Χωρητικότητα Μνήμης Ελέγχου = 2505 x 114 = 285.570 bits



f → ΜμΠ 00
ΜμΠ+1 → ΜμΠ 01
διεύθυνση → ΜμΠ 10

2 bits	100 bits
00	σήματα ελέγχου
01	σήματα ελέγχου
10	διεύθυνση

ΧΜΕ = [5 + 250(10+1)] · 102 = 281.570 bits



Θέμα προόδου με τεχνική μικροπρογραμματισμού δύο επιπέδων – Ιανουάριος 2022

Θεωρήστε ένα μικροεπεξεργαστή με μονάδα ελέγχου που έχει σχεδιαστεί με την τεχνική του μικροπρογραμματισμού δύο επιπέδων (νανοπρογραμματισμός). Το σύνολο των μικροπρογραμμάτων αποτελείται από 45270 μικροεντολές και υπάρχουν 1005 διαφορετικές μικροεντολές. Το μήκος κάθε μικροεντολής είναι 45 δυαδικά ψηφία εκ' των οποίων τα 43 είναι σήματα ελέγχου. Να υπολογίσετε την επί τοις εκατό μείωση του απαιτούμενου αποθηκευτικού χώρου σε σχέση με την περίπτωση που η μονάδα ελέγχου είχε σχεδιαστεί με την τεχνική του μικροπρογραμματισμού ενός επιπέδου.

Να δώσετε το αποτέλεσα με τη μορφή ακέραιο μέρος τελεία και ένα κλασματικό ψηφίο εφαρμόζοντας την τεχνική της αποκοπής, π.χ. για την τιμή 45.489 να δώσετε την τιμή 45.4 [ ]%

Λύση

Για **ένα επίπεδο** ισχύει ότι η χωρητικότητα του είναι 45270 μικροεντολές * 45 bits = 2.037.150 bits.

Για τα **δύο επίπεδα** ισχύει ότι στο 2ο επίπεδο υπάρχουν 557 διαφορετικές μικροεντολές. Αυτό σημαίνει ότι το πρώτο επίπεδο πρέπει να έχει ένα μήκος μικροεντολής x bits όπου: $2^x \geq 1005 \Rightarrow x = 10$ bits. Η **συνολική** χωρητικότητα και των δύο επιπέδων θα είναι: Χωρητικότητα 1ου επιπέδου + Χωρητικότητα 2ου επιπέδου = 45270 μικροεντολές * (10 + 2) bits + 1055 * 43 bits = 588605. Στο άθροισμα 10 + 2 bits, τα 10 bits χρειάζονται για τη διευθυνσιοδότηση των διαφορετικών μικροεντολών και τα 2 bits προέρχονται από τη διαφορά 45 – 43 bits. Το ποσοστό μείωσης του απαιτούμενου αποθηκευτικού χώρου με τη χρήση της τεχνικής της νανομνήμης είναι: $\Lambda = \frac{\text{αρχική χωρητικότητα} - \text{τελική χωρητικότητα}}{\text{αρχική χωρητικότητα}} = \frac{2.037.150 - 588605}{2.037.150} =$ 0.711 = 71.1%

Κεφάλαιο 5 – Μνήμες

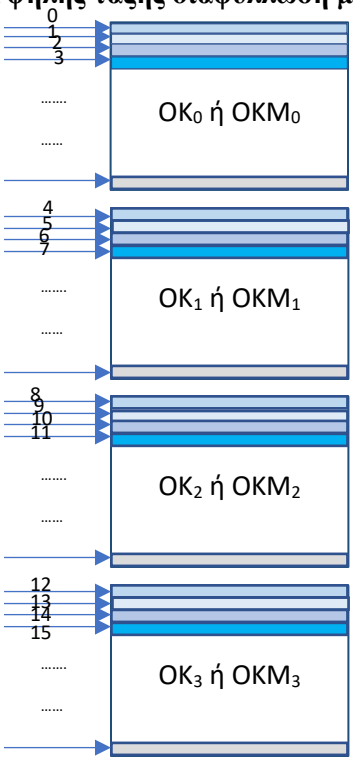
Επεξηγήσεις κεφαλαίου 5

- Σε μια ιεραρχική μνήμη, **οι σχέσεις που διέπουν τα επίπεδα μνήμης μεταξύ τους** είναι: κόστος $K_i > K_{i+1}$ (λόγω τεχνολογίας κατασκευής), χρόνος προσπέλασης $\tau_i < \tau_{i+1}$ (λόγω τεχνολογίας κατασκευής) και χωρητικότητα $X_i < X_{i+1}$ (διότι όσο απομακρυνόμαστε από την ΚΜΕ τόσο μεγαλώνει η χωρητικότητα του κάθε επιπέδου μνήμης).

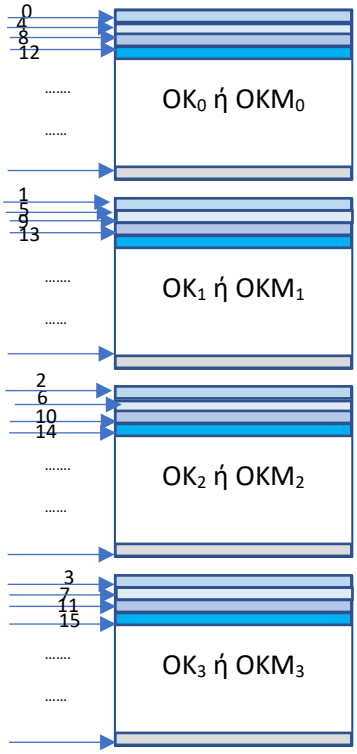
- **Όταν ζητείται ο μέσος χρόνος προσπέλασης (ανάγνωσης ή εγγραφής) από ιεραρχικό σύστημα μνήμης**, τότε χρησιμοποιήσουμε τον τύπο από το Κεφάλαιο 5.2, δηλ. τον τύπο: $T = \sum_{i=1}^{n}(E_i - E_{i-1})$ x T_iκαι όχι τύπο από το Κεφάλαιο 1. Στον τύπο αυτό θεωρούμε ότι ο **λόγος επιτυχίας του τελευταίου επιπέδου της ιεραρχικής μνήμης είναι πάντα «1», δηλ. $E_v = 1$**, επειδή εκεί υπάρχει σίγουρα η ζητούμενη πληροφορία και ο λόγος επιτυχίας του επεξεργαστή είναι 0, δηλ. $E_0 = 0$.

- Στο σχεδιασμό κύριας μνήμης με οργάνωση **υψηλής τάξης ή χαμηλής διαφύλλωσης, χρησιμοποιούμε ΟΚΜ** (Ολοκληρωμένα Κυκλώματα Μνήμης) ή απλώς **ΟΚ** (Ολοκληρωμένα Κυκλώματα ή chip), τα οποία συνδέουμε πάνω στην αρτηρία (δίαυλο) διευθύνσεων και δεδομένων του υπολογιστικού συστήματος. Η σημασία του διαύλου διευθύνσεων (address bus) είναι για τον εντοπισμό (προσπέλαση) μιας θέσης μνήμης (θ.μ.) και η σημασία του διαύλου δεδομένων (data bus) είναι για τον εγγραφή/ανάγνωση της θ.μ. που νωρίτερα προσπελάστηκε (εντοπίστηκε) από την αρτηρία διευθύνσεων.

Υψηλής τάξης διαφύλλωση μνήμης



Χαμηλής τάξης διαφύλλωση μνήμης



Στην οργάνωση υψηλής τάξης διαφύλλωση μνήμης, **διαδοχικές** διευθύνσεις μνήμης προσπελαύνουν το **ίδιο** ΟΚ. Αυτό σημαίνει ότι για να προσπελαστούν οι διευθύνσεις μνήμης 0 – 3, θα πρέπει το OK_0 να προσπελαστεί τέσσερις συνολικά φορές, μία για κάθε διεύθυνση. Αντίθετα, στην οργάνωση χαμηλής τάξης διαφύλλωση μνήμης, διαδοχικές διευθύνσεις μνήμης προσπελαύνονται σε διαφορετικά ΟΚ. Αυτό σημαίνει ότι για να προσπελαστούν οι διευθύνσεις μνήμης 0 – 3, αρκεί **μια μόνο προσπέλαση** και όχι τέσσερις όπως πριν, αφού στην περίπτωση αυτή, η διεύθυνση 0 προσπελαύνεται από το OK_0, η διεύθυνση 1 προσπελαύνεται από το OK_1 η διεύθυνση 2 προσπελαύνεται από το OK_2 και η διεύθυνση 3 προσπελαύνεται από το OK_3. Επομένως, στη δεύτερη περίπτωση έχουμε **ταχύτερη** προσπέλαση της κύριας μνήμης, αφού με μια μόνο προσπέλαση μπορούν να προσπελαστούν ταυτόχρονα διαφορετικές διευθύνσεις μνήμης.

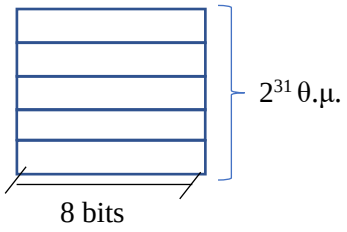
- Η οργάνωση **υψηλής τάξης διαφύλλωση μνήμης** έχει το **πλεονέκτημα** ότι επιτρέπει την **εύκολη επέκταση** του συστήματος μνήμης, προσθέτοντας ένα ή περισσότερα τμήματα μνήμης, όπου κάθε τμήμα αποτελείται από ένα ή περισσότερα ΟΚ, δηλαδή κάθε τμήμα είναι μια συλλογή ΟΚΜ. Το **μειονέκτημα** αυτής της οργάνωσης είναι μπορεί να δημιουργήσει **καθυστερήσεις** σε κάποιες περιπτώσεις, όπως π.χ. στην περίπτωση συστημάτων μερικώς επικαλυπτόμενων λειτουργιών.

- Η οργάνωση **χαμηλής τάξης διαφύλλωση μνήμης** έχει το **πλεονέκτημα** ότι επιτρέπει την **προσπέλαση ενός διανύσματος από στοιχεία δεδομένων ή ενός μπλοκ πληροφορίας σε επεξεργαστές με κρυφή μνήμη**. Το **μειονέκτημα** αυτής της οργάνωσης είναι ότι η απόδοσή της πέφτει γρήγορα όταν απαιτείται η προσπέλαση λέξεων (θ.μ.) με **μη - διαδοχικές** διευθύνσεις.

- **Κάθε ΟΚΜ (ΟΚ) έχει μια χωρητικότητα και μια εσωτερική οργάνωση (μέγεθος κάθε θ.μ.)** που διατυπώνεται ως εξής: **Διαθέτουμε ΟΚΜ των 2GB με οργάνωση μιας ψηφιολέξης ανά θέση μνήμης**. Θα πρέπει να μετατρέπουμε τη χωρητικότητα του ΟΚΜ σε θ.μ. με μια απλή μέθοδο των τριών.

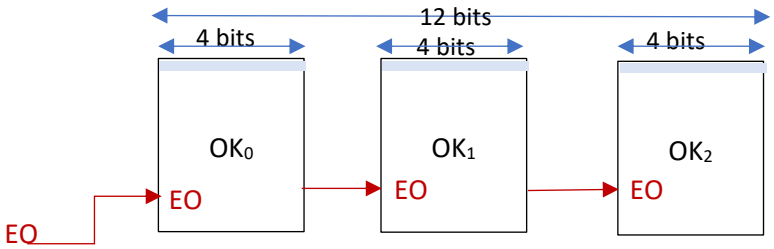
1 θ.μ.	1 ψηφιολέξη (byte)
x;	2GB

$x = 2 \times 2^{30}$ bytes/1 byte = 2^{31} θ.μ.



- Στο **σχεδιασμό** κύριας μνήμης (ΚΜ) με οργάνωση **υψηλής τάξης διαφύλλωση**, θα πρέπει να εφαρμόσουμε κατά το σχεδιασμό της, οριζόντια και κατακόρυφη επανάληψη.

- Πιο συγκεκριμένα, **η οριζόντια επανάληψη ΟΚΜ έχει ως στόχο το επιθυμητό εύρος της θ.μ. και εξαρτάται από την εσωτερική οργάνωση των ΟΚΜ που διαθέτουμε αλλά και της Κ.Μ. (Κύριας Μνήμης) που θα σχεδιάσουμε.** Για παράδειγμα, αν υποθέσουμε ότι έχουμε ΟΚΜ των 4 bits/θ.μ. → Κ.Μ. των 12 bits/θ.μ., πρέπει να χρησιμοποιήσουμε οριζόντια 3 ΟΚΜ, τα οποία να επιλέγονται ταυτόχρονα με ένα κοινό σήμα επιλογής



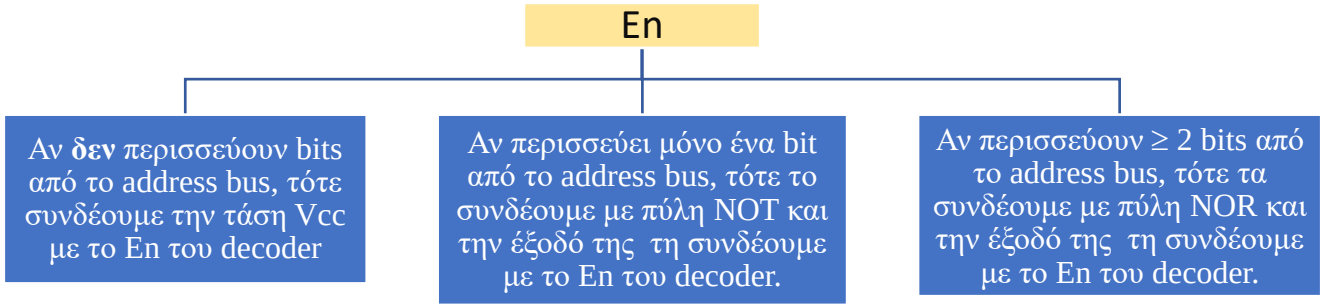
- Πιο συγκεκριμένα, **η κατακόρυφη επανάληψη ΟΚΜ έχει ως στόχο** το **επιθυμητό πλήθος θ.μ.** και προκύπτει από το κλάσμα: $\frac{\text{θ.μ. που θέλουμε να έχουμε στο τελικό σύστημα ΚΜ}}{\text{θ.μ. που ήδη έχουμε στα ΟΚΜ που διαθέτουμε}}$. Για το σωστό υπολογισμό της κατακόρυφης επανάληψης, απαιτείται η προγενέστερη **μετατροπή των χωρητικοτήτων των ΟΚΜ και της ΚΜ σε θέσεις μνήμης.**

- Στο σχεδιασμό κύριας μνήμης με οποιαδήποτε οργάνωση, **το μέγεθος της αρτηρίας διευθύνσεων όταν δίνεται από την εκφώνηση της άσκησης, τοποθετείται πάνω στο σχήμα. Όταν όμως δεν δίνεται, τότε χρησιμοποιούμε ως μέγεθος του address bus το μέγεθος της Κ.Μ. που θα σχεδιάσουμε.** Αν για παράδειγμα, θέλουμε να σχεδιάσουμε σύστημα Κ.Μ. 1GB με οργάνωση 16 bits/θ.μ. και δεν δίνεται το μέγεθος της αρτηρίας διευθύνσεων, τότε θα μετατρέψουμε τη χωρητικότητα της ΚΜ σε θέσεις μνήμης, με απλή μέθοδο των τριών:

1 θ.μ.	2 ψηφιολέξεις (bytes)
x;	1 GB

$x = \frac{2^{30}\ bytes}{2\ bytes/θ.μ}* = 2^{29}$ θ.μ. → address bus = 29 bits. Αυτό είναι το μέγεθος του address bus. Αν όμως δινόταν ότι το μέγεθος της αρτηρίας διευθύνσεων είναι ίσο με 32 γραμμές, τότε το address bus = 32 bits.

- Στο σχεδιασμό κύριας μνήμης με οργάνωση **υψηλής τάξης διαφύλλωσης** (α) πρώτα συνδέουμε τα ΟΚΜ με την αρτηρία δεδομένων, (β) στη συνέχεια τα συνδέουμε με την αρτηρία διευθύνσεων και (γ) τέλος με τον αποκωδικοποιητή (decoder), που θα χρησιμοποιηθεί ως κύκλωμα επιλογής και ο οποίος χρειάζεται σήμα ενεργοποίησης (επιλογής), που είναι θετικής λογικής. **Αν δεν περισσεύουν κάποια bits από το address bus για να χρησιμοποιηθούν ως είσοδοι στο E_n (Enable) του decoder, τότε συνδέσουμε την τάση λειτουργίας V_{cc} (που αντιστοιχεί σε λογικό «1») με το E_n του decoder, προκειμένου ο αποκωδικοποιητής να είναι μόνιμα επιλεγμένος. Αν περισσεύει ακριβώς ένα bit από το address bus**, τότε το συνδέουμε σε πύλη NOT και στη συνέχεια στο E_n του decoder. **Αν περισσεύουν περισσότερα από ένα bits από το address bus,** τα συνδέουμε σε πύλη NOR και στη συνέχεια στο E_n του decoder.



- Στο σχεδιασμό κύριας μνήμης με οργάνωση **υψηλής τάξης διαφύλλωση**, όταν **η χωρητικότητα της δίνεται σε Μθέσεις μνήμης,** τότε **δεν** χρειάζεται να εφαρμόσουμε απλή μέθοδο των τριών. Για παράδειγμα, αν η χωρητικότητα της κύριας μνήμης είναι των 256 Μθέσεων**, τότε χωρίς να** χρησιμοποιήσουμε **απλή μέθοδο των τριών, λέμε απευθείας ότι η χωρητικότητα της κύριας μνήμης είναι 2^8 x 2^{20} θ.μ. = 2^{28} θ.μ.**

- Στο σχεδιασμό κύριας μνήμης με οργάνωση **υψηλής τάξης διαφύλλωσης, όταν συνδέουμε οριζόντια ΟΚ διαφορετικών χωρητικοτήτων,** θα πρέπει **αυτά να έχουν το ίδιο πλήθος θ.μ.** για να μπορούν να συνδεθούν το ένα δίπλα στο άλλο.

- Όταν **μετατρέπουμε τις χωρητικότητες των ΟΚΜ, ΚΜ σε θέσεις μνήμης**, θα πρέπει να προσέχουμε ότι μονάδες έχουμε πάνω να έχουμε και κάτω. Δηλαδή:

1 θ.μ.	8 bits
x;	2 MB

$x = \frac{2 \text{ x } 2^{20} \text{ x } 2^3 bits}{2^3\ bits} = 2^{21}$ θ.μ.

1 θ.μ.	1 byte
x;	2 MB

$x = \frac{2 \text{ x } 2^{20} bytes}{1\ byte} = 2^{21}$ θ.μ.

1 θ.μ.	8 bits
x;	2 Mbit

$x = \frac{2 \text{ x } 2^{20} bits}{2^3\ bits} = 2^{18}$ θ.μ.

1 θ.μ.	1 bit
x;	2 MB

$x = \frac{2 \text{ x } 2^{20} \text{x } 2^3 bits}{1\ bit} = 2^{24}$ θ.μ.

- Στο σχεδιασμό κύριας μνήμης με οργάνωση **υψηλής τάξης διαφύλλωση, όταν επιλέγουμε σειρές ΟΚ με διαφορετική χωρητικότητα,** θα πρέπει να εντοπίζουμε τη σειρά εκείνη με τη μικρότερη χωρητικότητα και για αυτήν θα πρέπει να χρησιμοποιούμε μια έξοδο του αποκωδικοποιητή, ενώ για τις υπόλοιπες σειρές, ανάλογα με τη χωρητικότητά τους, θα πρέπει να χρησιμοποιούμε περισσότερες εξόδους του αποκωδικοποιητή, συνδυασμένες σε μια πύλη OR. Το πόσες εξόδους χρησιμοποιούμε, εξαρτάται από το πολλαπλάσιο της χωρητικότητάς τους σε σχέση με τη βασική χωρητικότητα.

- Στο σχεδιασμό κύριας μνήμης με οργάνωση υψηλής **τάξης διαφύλλωσης, αν θέλουμε να επεκτείνουμε τη χωρητικότητά της κατά ένα μέγεθος που δεν το διαθέτουμε από τα ΟΚΜ που έχουμε** (π.χ. επέκταση κατά 1 GB τη στιγμή που διαθέτουμε ΟΚΜ των 256 MB), **τότε θα πρέπει να χρησιμοποιήσουμε τμήματα μνήμης**. Ένα τμήμα μνήμης είναι μια συλλογή (ομάδα) ενός ή περισσοτέρων ΟΚΜ.

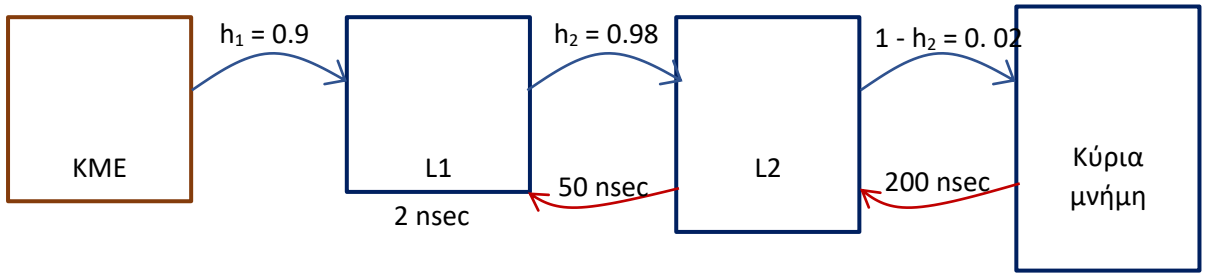
- Όταν από την εκφώνηση μιας άσκησης **τίθενται χρονικοί περιορισμοί** αναφορικά με τη μεταφορά δεδομένων από κύρια μνήμη στην κρυφή, τότε θα πρέπει να χρησιμοποιηθεί η οργάνωση χαμηλής τάξης διαφύλλωση μνήμης, που επιτρέπει την ταυτόχρονη προσπέλαση πολλών θέσεων μνήμης.

- **Κανόνας σχεδιασμού κύριας μνήμης από επιμέρους ΟΚΜ**: αν από τη διαίρεση που γίνεται στην κατακόρυφη επανάληψη προκύψει αποτέλεσμα **κλασματικό**, π.χ. 2.5 τότε δεν υπάρχει περίπτωση να χρειάζονται 2.5 ολοκληρωμένα, ΟΚ. Εάν χρησιμοποιούσαμε 3 θα φτιάχναμε μεγαλύτερη μνήμη από τη ζητούμενη. Εάν χρησιμοποιούσαμε μόνο 2, θα φτιάχναμε μικρότερη. Επειδή η άσκηση θα δίνει περισσότερα είδη ΟΚ, πρέπει να χρησιμοποιήσουμε κάποιο πλήθος από κάθε είδος ώστε η μνήμη που θα σχεδιάσουμε να έχει τα απαιτούμενα χαρακτηριστικά.

1ο Θέμα με ιεραρχία μνήμης

Να υπολογίσετε το μέσο χρόνο ανάγνωσης από ιεραρχικό σύστημα μνήμης που αποτελείται από δύο επίπεδα κρυφής μνήμης L1 και L2 με λόγους επιτυχίας h_1=0,9 και h_2=0,98 αντίστοιχα, και κύρια μνήμη. Ο χρόνος μεταφοράς ενός μπλοκ πληροφορίας από την κύρια μνήμη στην L2 ισούται με 200 ns, από την L2 στην L1 ισούται με 50 ns και ο χρόνος προσπέλασης της L1 ισούται με 2 ns. Θεωρήστε ότι η ΚΜΕ μπορεί να διαβάσει πληροφορία μόνο από την L1 και ότι η κύρια μνήμη περιέχει πάντα τη ζητούμενη πληροφορία.

Λύση



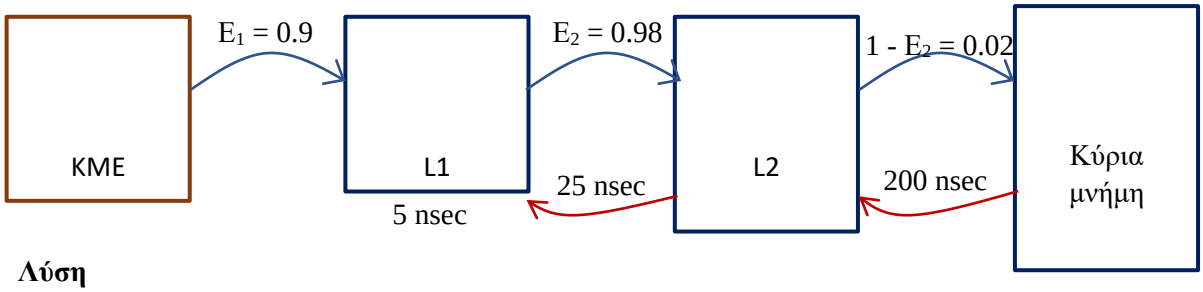
Το μοναδικό επίπεδο μιας ιεραρχικής μνήμης που είναι άμεσα προσπελάσιμο από την ΚΜΕ είναι πάντα το πρώτο επίπεδο κρυφής μνήμης L_1. Αυτό σημαίνει ότι αν η πληροφορία βρεθεί σε κάποιο άλλο επίπεδο ιεραρχικής μνήμης (π.χ. στο L_2) ή στην κύρια μνήμη, τότε θα πρέπει να μεταφερθεί στο επίπεδο L_1 προκειμένου να υποστεί επεξεργασία από την ΚΜΕ. Η διατύπωση ότι η **κύρια μνήμη περιέχει πάντα τη ζητούμενη** πληροφορία σημαίνει ότι το λόγος επιτυχίας $E_v = E_3 = 1$.

Ο **μέσος χρόνος ανάγνωσης (προσπέλασης) από ιεραρχικό σύστημα μνήμης** δίνεται από τον τύπο $T = \sum_{i=1}^{n=3}(E_i - E_{i-1})$ x $T_i = (E_1 - E_0)$ x T_1 + $(E_2 - E_1)$ x T_2 + $(E_3 - E_2)$ x T_3 = (0.9 – 0) x 2 nsec + (0.98 – 0.9) x (50 + 2) nsec + (1 – 0.98) x (200 + 50 + 2)nsec = 0.9 x 2nsec + 0.08 x 52 nsec + 0.02 x 252 nsec = 11 nsec.

2ο Θέμα με ιεραρχία μνήμης

Δίνεται ιεραρχική μνήμη αποτελούμενη από τρία επίπεδα: 1ο επίπεδο κρυφής μνήμης, 2ο επίπεδο κρυφής μνήμης και κύρια μνήμη. Υποθέστε ότι η ΚΜΕ διαβάζει μόνο από το 1ο επίπεδο κρυφής μνήμης. Να υπολογίσετε το μέσο χρόνο προσπέλασης της ιεραρχικής μνήμης λαμβάνοντας υπόψη ότι ο λόγος επιτυχίας του επιπέδου k είναι E_k, με E_1 = 0.9, E_2 = 0.99 και E_3 = 1, ο χρόνος προσπέλασης του 1ου επιπέδου είναι t_1 = 5 nsec και ο χρόνος μεταφοράς ενός μπλοκ πληροφορίας από το επίπεδο k + 1 στο επίπεδο k είναι t_B (k ← k + 1), είναι t_B (1 ← 2) = 25 nsec και t_B (2 ← 3) = 200 nsec

Λύση



Λύση

Ο μέσος χρόνος προσπέλασης της ιεραρχικής μνήμης είναι:

$T = \sum_{i=1}^{v}(E_i - E_{i-1})$. $T_i = \sum_{i=1}^{3}(E_i - E_{i-1})$. $T_i = (E_i - E_0)\ T_1 + (E_2 - E_1)\ T_2 + (E_3 - E_2)$.

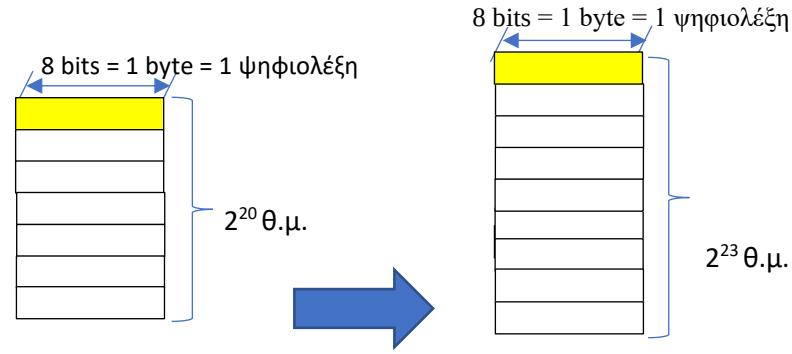
$T_3 = (0{,}9 - 0)\ T_1 + (0{,}98 - 0{,}9)\ T_2 + (1 - 0{,}98)\ T_3 = 0{,}9\ T_1 + 0{,}08\ T_2 + 0{,}01\ T_3$ όπου: $T_1 =$ 5nsec. Το T_2 = 5 + 25 nsec = 30 nsec και το T_3 = 5 + 25 + 200 nsec = 230 nsec. Άρα T = 0.9 x 5 + 0.09 x 30 + 0.01 x 230 nsec = 4.5 + 2.7 + 2.3 = 9.5 nsec

Παράδειγμα 5.6 βιβλίου – Σχεδιασμός υψηλής τάξης διαφύλλωση μνήμης

Έχουμε στη διαθεσή μας ΟΚΜ (Ολοκληρωμένα Κυκλώματα Μνήμης) με χωρητικότητα 1 Μψηφιολέξεις και εσωτερική οργάνωση μιας ψηφιολέξης ανά θέση μνήμης (θ.μ.) και θέλουμε να σχεδιάσουμε **σύστημα κύριας μνήμης (Κ.Μ.) των 8 Μψηφιολέξεων και εσωτερική οργάνωση μιας ψηφιολέξης ανά θέση μνήμης (θ.μ.)**.

Λύση

Διαθέτουμε ΟΚΜ 1 Μψηφιολέξη x 1 ψηφιολέξη/θ.μ. → **Στόχος**: Κ.Μ. 8 Μψηφιολέξεων x 1 ψηφιολέξη/θ.μ.



ΟΚΜ: 1 Mbyte

ΚΜ: 8 Mbyte

Για να πετύχουμε το στόχο μας, **πρέπει να επαναλάβουμε οριζόντια και κατακόρυφα τα ΟΚΜ που μας δίνονται**. Με άλλα λόγια, πρέπει να εφαρμόσουμε **οριζόντια και κατακόρυφη επανάληψη των ΟΚΜ** που έχουμε.

Οριζόντια επανάληψη (στόχος είναι η δημιουργία του **κατάλληλου εύρους της θέσης μνήμης**). Επειδή διαθέτουμε ΟΚΜ με εσωτερική οργάνωση 1 byte/θ.μ. και θέλουμε να σχεδιάσουμε σύστημα Κ.Μ. με την **ίδια** εσωτερική οργάνωση, αρκεί **μόνο** μια οριζόντια επανάληψη.

Κατακόρυφη επανάληψη (στόχος είναι το **επιθυμητό πλήθος θέσεων μνήμης**) και προκύπτει από το κλάσμα: $\frac{\textbf{θ.μ. που θέλουμε να έχουμε στο τελικό σύστημα ΚΜ}}{\textbf{θ.μ. που ήδη έχουμε στα ΟΚΜ που διαθέτουμε}}$. Για σωστό υπολογισμό αυτού του κλάσματος, θα πρέπει **πρώτα** με απλές μεθόδους των τριών, να μετατρέψουμε τις χωρητικότητες τόσο των ΟΚΜ που διαθέτουμε όσο και της Κ.Μ. που θα σχεδιάσουμε, σε θέσεις μνήμης.

1 θ.μ.	1 byte
x;	1 MB

$x = \frac{2^{20}\ \text{bytes}}{1\ \text{byte/θ.μ}} = 2^{20}$ θ.μ.

1 θ.μ.	1 byte
x;	8 MB

$x = \frac{2^3 2^{20}\ \text{bytes}}{1\ \text{byte/θ.μ}}* = 2^{23}$ θ.μ

Επομένως $\frac{\textbf{θ.μ.που θέλουμε να έχουμε στο τελικό σύστημα ΚΜ}}{\textbf{θ.μ.που ήδη έχουμε στα ΟΚΜ που διαθέτουμε}} = \frac{2^{23}}{2^{20}} = 8$ επαναλήψεις ΟΚΜ

Στη συνέχεια προχωράμε στο σχεδιασμό της Κ.Μ. με βάση την οριζόντια και κατακόρυφη επανάληψη που υπολογίσαμε. Στο σχεδιασμό αυτό, θα πρέπει να συνδέσουμε τα ΟΚΜ με την **αρτηρία δεδομένων** και την **αρτηρία διευθύνσεων**.

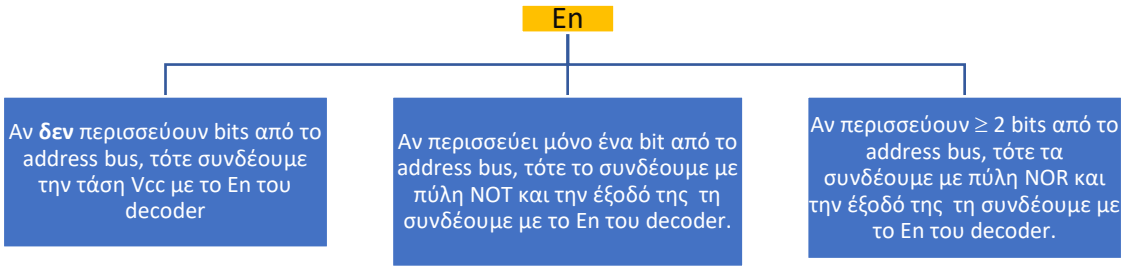
Μεθοδολογία σχεδίασης:

a. **Πρώτα** σχεδιάζουμε οριζόντια και κατακόρυφα τα ΟΚΜ (ή ΟΚ) που διαθέτουμε.

b. Στη συνέχεια, **συνδέουμε τα ΟΚΜ με την αρτηρία δεδομένων** στα δεξιά με διπλά αμφίδρομα βέλη. Αν έχουμε οριζόντια επανάληψη των ΟΚΜ, τότε το data bus επαναλαμβάνεται τόσες φορές όσο είναι η οριζόντια επανάληψη.

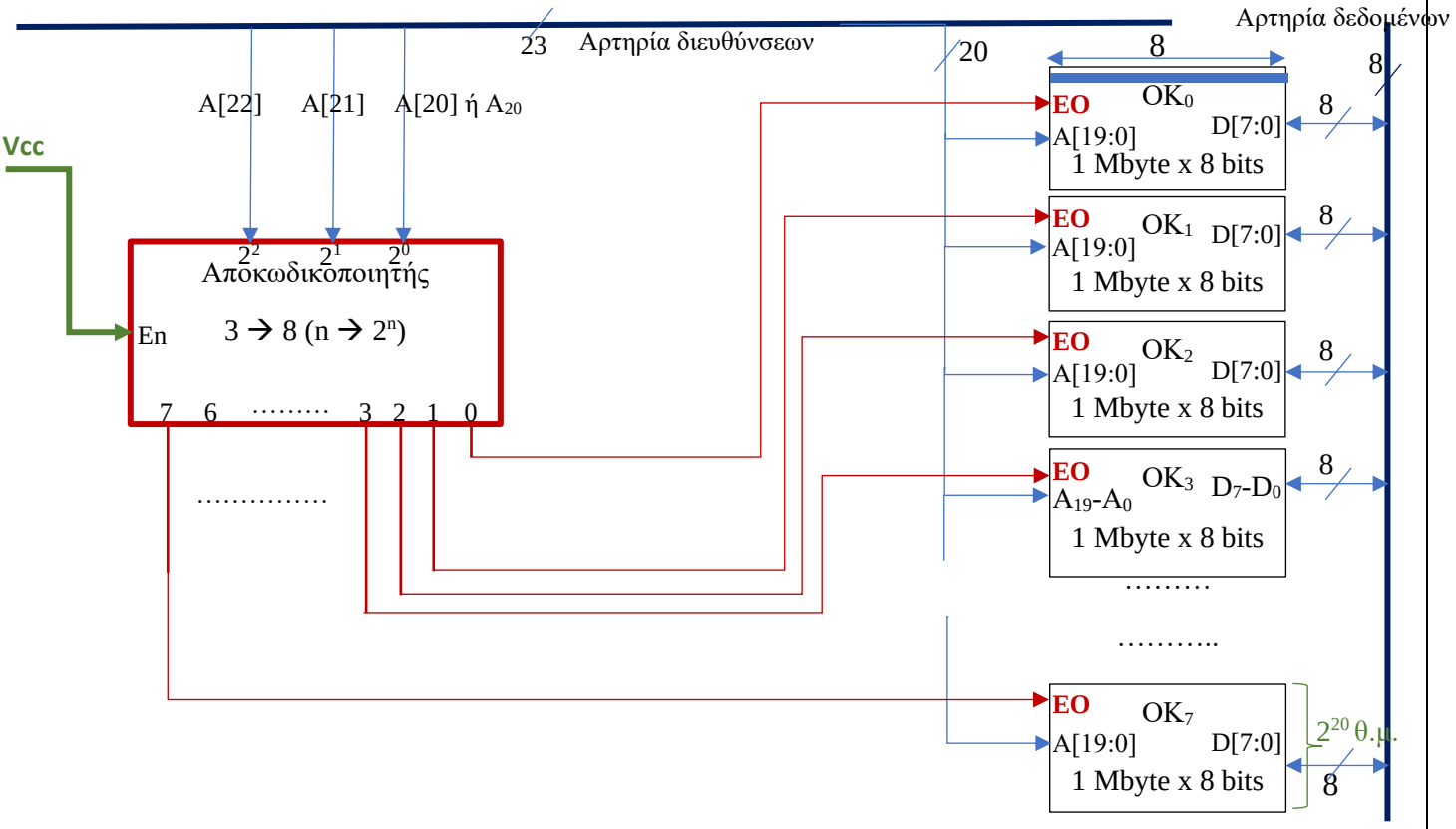
c. Στη συνέχεια, **συνδέουμε τα ΟΚΜ με την αρτηρία διευθύνσεων**, με βέλη που είναι μόνο είσοδοι στα ΟΚΜ.

d. Επιπλέον, είναι απαραίτητη η χρήση ενός **αποκωδικοποιητή** (συνδυαστικό κύκλωμα n εισόδων και 2^n εξόδων), το οποίο θα χρησιμοποιηθεί για να κάνει την επιλογή των ΟΚΜ. Αυτό γίνεται διότι η κάθε σειρά με τα ΟΚ που χρησιμοποιούμε πρέπει να επιλέγεται ξεχωριστά από τις υπόλοιπες. Ε**πομένως, σχεδιάζουμε τον αποκωδικοποιητή και συνδέουμε κάθε έξοδό του με ένα ή περισσότερα ΟΚΜ** που είναι στην ίδια σειρά. **Το μέγεθος του αποκωδικοποιητή προσδιορίζεται από τις εξόδους του** και οι έξοδοί του από τον αριθμό των σειρών. Όσα bits από το address bus περισσεύουν, τα συνδέουμε στον ακροδέκτη En του αποκωδικοποιητή. Κάθε φορά, από τις εξόδους του αποκωδικοποιητή, **ενεργοποιείται μόνο μία**, με βάση τις τιμές των εισόδων του. Επιπλέον, ο αποκωδικοποιητής εκτός των εισόδων του, έχει και μια γραμμή ενεργοποίησης (En) που είναι **θετικής λογικής** και η οποία επιλέγεται ως εξής:



- Αν χρειάζεται (αν ζητείται από την εκφώνηση) μετά το σχεδιασμό του συστήματος Κ.Μ. από τα επιμέρους ΟΚΜ, θα πρέπει να περιγράψουμε και το χάρτη διευθύνσεων (memory map) της Κ.Μ., ο οποίος δείχνει την περιοχή διευθύνσεων μνήμης που καταλαμβάνει κάθε ΟΚΜ.

- Το μέγεθος της αρτηρίας δεδομένων και διευθύνσεων προκύπτει από την ΚΜ που θα σχεδιάσουμε. Στην προκειμένη περίπτωση η Κ.Μ. που θα σχεδιάσουμε θα έχει 2^{23} θ.μ., άρα το address bus = 23. Αν όμως, η εκφώνηση ανέφερε τι η αρτηρία διευθύνσεων έχει 30 δυαδικά ψηφία (γραμμές), τότε θα παίρναμε αυτό το μέγεθος στην αρτηρία διευθύνσεων και όχι το 23.

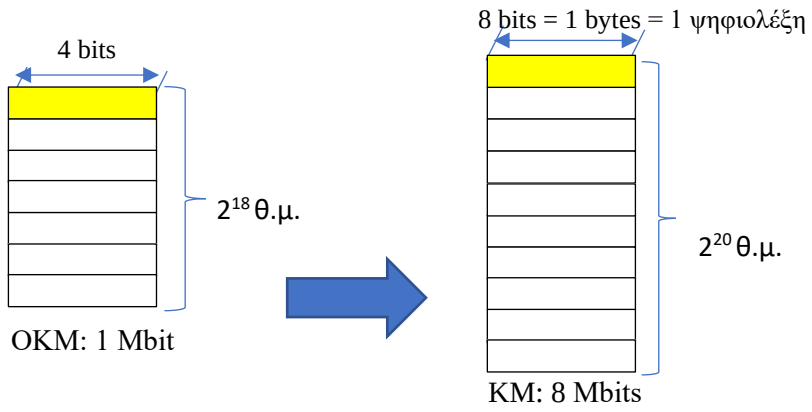


Παράδειγμα 5.7 βιβλίου – Σχεδιασμός υψηλής τάξης διαφύλλωση μνήμης

Έχουμε στη διαθεσή μας ΟΚΜ (Ολοκληρωμένα Κυκλώματα Μνήμης) με χωρητικότητα 1 Μδυαδικών ψηφίων και εσωτερική οργάνωση 4 δυαδικά ψηφία ανά θέση μνήμης (θ.μ.) και θέλουμε να σχεδιάσουμε σύστημα κύριας μνήμης (Κ.Μ.) των 8 Μδυαδικών ψηφίων και εσωτερική οργάνωση μιας ψηφιολέξης ανά θέση μνήμης (θ.μ.).

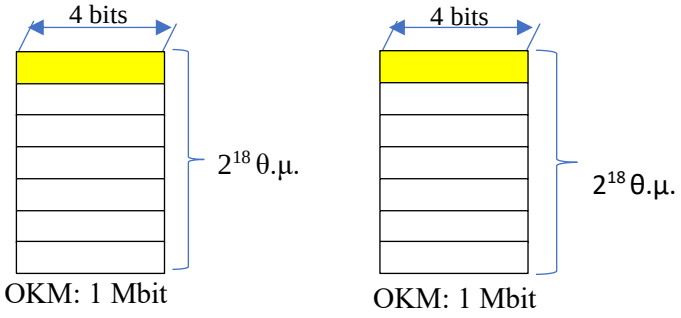
Λύση

Διαθέτουμε ΟΚΜ 1 Mbit x 4 bits/θ.μ. → **Στόχος**: Κ.Μ. 8 Mbits x 1 ψηφιολέξη/θ.μ.



Για να πετύχουμε το στόχο μας, πρέπει να επαναλάβουμε οριζόντια και κατακόρυφα τα ΟΚΜ που μας δίνονται. Με άλλα λόγια, πρέπει να εφαρμόσουμε οριζόντια και κατακόρυφη επανάληψη των ΟΚΜ που έχουμε.

- **Οριζόντια επανάληψη** (στόχος είναι η δημιουργία του **κατάλληλου εύρους της θέσης μνήμης**). Επειδή διαθέτουμε ΟΚΜ με εσωτερική οργάνωση 4 bits/θ.μ. και θέλουμε να σχεδιάσουμε σύστημα Κ.Μ. με την εσωτερική οργάνωση 1 ψηφιολέξη/θ.μ = 8 bits/θ.μ., θα **χρειαστούν 2 οριζόντιες επαναλήψεις**.



- **Κατακόρυφη επανάληψη** (στόχος είναι το **επιθυμητό πλήθος θέσεων μνήμης**) και προκύπτει από το κλάσμα: $\frac{\textbf{θ.μ.που θέλουμε να έχουμε στο τελικό σύστημα ΚΜ}}{\textbf{θ.μ. που ήδη έχουμε στα ΟΚΜ που διαθέτουμε}}$. Για σωστό υπολογισμό αυτού του κλάσματος, θα πρέπει **πρώτα** με απλές μεθόδους των τριών, να μετατρέψουμε τις χωρητικότητες τόσο των ΟΚΜ που διαθέτουμε όσο και της Κ.Μ. που θα σχεδιάσουμε, σε θέσεις μνήμης.

1 θ.μ.	4 bits
x;	1 Mbit

$$x = \frac{2^{20}\ \text{bits}}{4\ \text{bits/θ.μ}} = 2^{18}\ \text{θ.μ.}$$

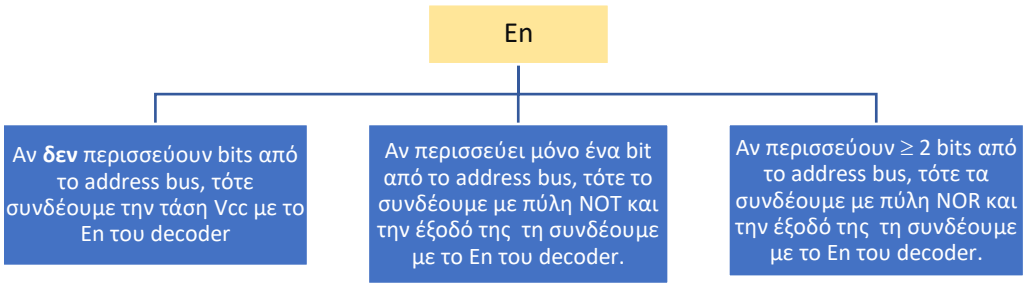
1 θ.μ.	8 bits
x;	8 Mbits

$$x = \frac{2^3 2^{20}\ \text{bits}}{2^3 \text{bits/θ.μ}} * = 2^{20}\ \text{θ.μ}$$

Επομένως $\frac{\textbf{θ.μ. που θέλουμε να έχουμε στο τελικό σύστημα ΚΜ}}{\textbf{θ.μ.που ήδη έχουμε στα ΟΚΜ που διαθέτουμε}} = \frac{2^{20}}{2^{18}} = 4$ επαναλήψεις

Στη συνέχεια προχωράμε στο σχεδιασμό της Κ.Μ. με βάση την οριζόντια και την κατακόρυφη επανάληψη που υπολογίσαμε. Στο σχεδιασμό αυτό, θα πρέπει να συνδέσουμε τα ΟΚΜ με την αρτηρία δεδομένων και την αρτηρία

διευθύνσεων. Επιπλέον, είναι απαραίτητη η χρήση ενός αποκωδικοποιητή (συνδυαστικό κύκλωμα n εισόδων και 2^n εξόδων), το οποίο θα χρησιμοποιηθεί για να κάνει την επιλογή των ΟΚΜ. Κάθε φορά, από τις εξόδους του αποκωδικοποιητή, **ενεργοποιείται μόνο μία**, με βάση τις τιμές των εισόδων του. Επιπλέον, ο αποκωδικοποιητής εκτός των εισόδων του, έχει και μια γραμμή ενεργοποίησης (En) που είναι θετικής λογικής και η οποία επιλέγεται ως εξής:



Μεθοδολογία σχεδίασης:

Ακολουθεί ο σχεδιασμός της κύριας μνήμης με οργάνωση υψηλής τάξης διαφύλλωση μνήμης:

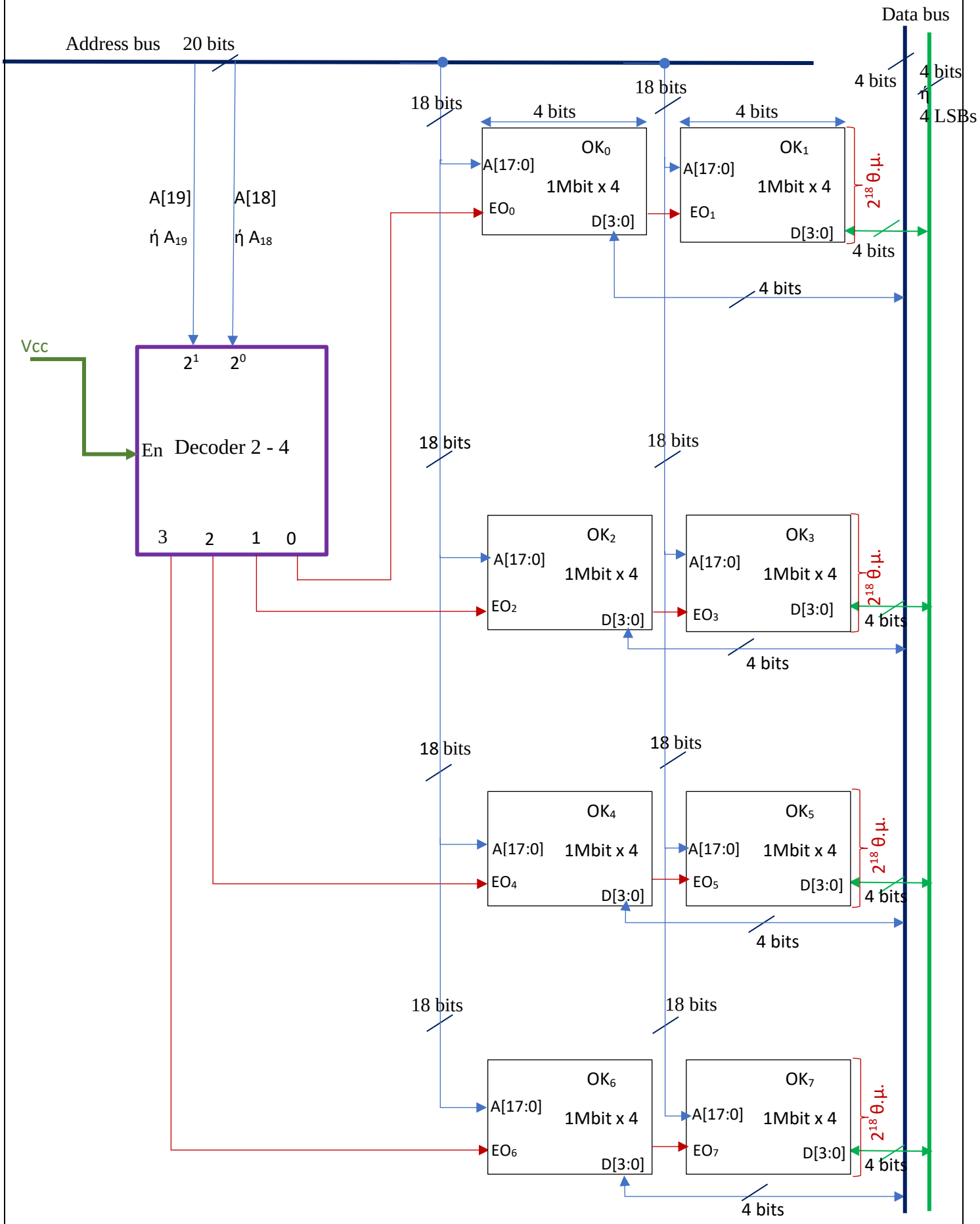
- **Πρώτα** σχεδιάζουμε οριζόντια και κατακόρυφα τα ΟΚΜ (ή ΟΚ) που διαθέτουμε.
- **Στη συνέχεια, συνδέουμε τα ΟΚΜ με την αρτηρία δεδομένων** στα δεξιά με διπλά αμφίδρομα βέλη. Αν έχουμε οριζόντια επανάληψη των ΟΚΜ, τότε το data bus (αρτηρία δεδομένων) επαναλαμβάνεται τόσες φορές όσο είναι η οριζόντια επανάληψη. Στην προκειμένη περίπτωση το data bus επαναλαμβάνεται δύο φορές.
- **Στη συνέχεια συνδέουμε τα ΟΚΜ** με την αρτηρία διευθύνσεων, με βέλη που είναι μόνο είσοδοι στα ΟΚΜ.
- Το μέγεθος της αρτηρίας δεδομένων και διευθύνσεων προκύπτει από την ΚΜ που θα σχεδιάσουμε.
- **Τέλος, σχεδιάζουμε τον αποκωδικοποιητή και συνδέουμε κάθε έξοδό του με ένα ή περισσότερα ΟΚΜ** (αν αυτά είναι στην ίδια σειρά). **Το μέγεθος του αποκωδικοποιητή προσδιορίζεται από τις εξόδους του**. Όσα bits περισσεύουν από το address bus τα συνδέουμε στον ακροδέκτη En του αποκωδικοποιητή, σύμφωνα με τα όσα αναφέρθηκαν προηγουμένως.
- Αν χρειάζεται (αν ζητείται από την εκφώνηση) μετά το σχεδιασμό του συστήματος Κ.Μ. από τα επιμέρους ΟΚΜ, θα πρέπει να περιγράψουμε και το χάρτη διευθύνσεων (memory map) της Κ.Μ., ο οποίος δείχνει την περιοχή διευθύνσεων μνήμης που καταλαμβάνει κάθε ΟΚΜ.

Χάρτης διευθύνσεων μνήμης (memory map)

Ολοκληρωμένο κύκλωμα μνήμης	Διευθύνσεις $A_{19}A_{18}A_{17}$ … A_1A_0	
$OK_0 – OK_1$	από	00000000000000000000
	έως	00111111111111111111
$OK_2 – OK_3$	από	01000000000000000000
	έως	01111111111111111111
$OK_4 – OK_5$	από	10000000000000000000
	έως	10111111111111111111
$OK_6 – OK_7$	από	11000000000000000000
	έως	11111111111111111111

Κάθε ΟΚΜ διαθέτει συνολικά 2^{18} θ.μ., για το λόγο αυτό απαιτούνται 18 bits από το address bus για τη διευθυνσιοδότηση, που είναι από $A_{17} – A_0$. Αυτά τα bits παίρνουν όλες τις τιμές από 00000…….00000 έως 11111……..11111, για να δείξουν ότι καλύπτουν όλο το εύρος τιμών που μπορούν να πάρουν.

Σχεδιασμός Κ.Μ.



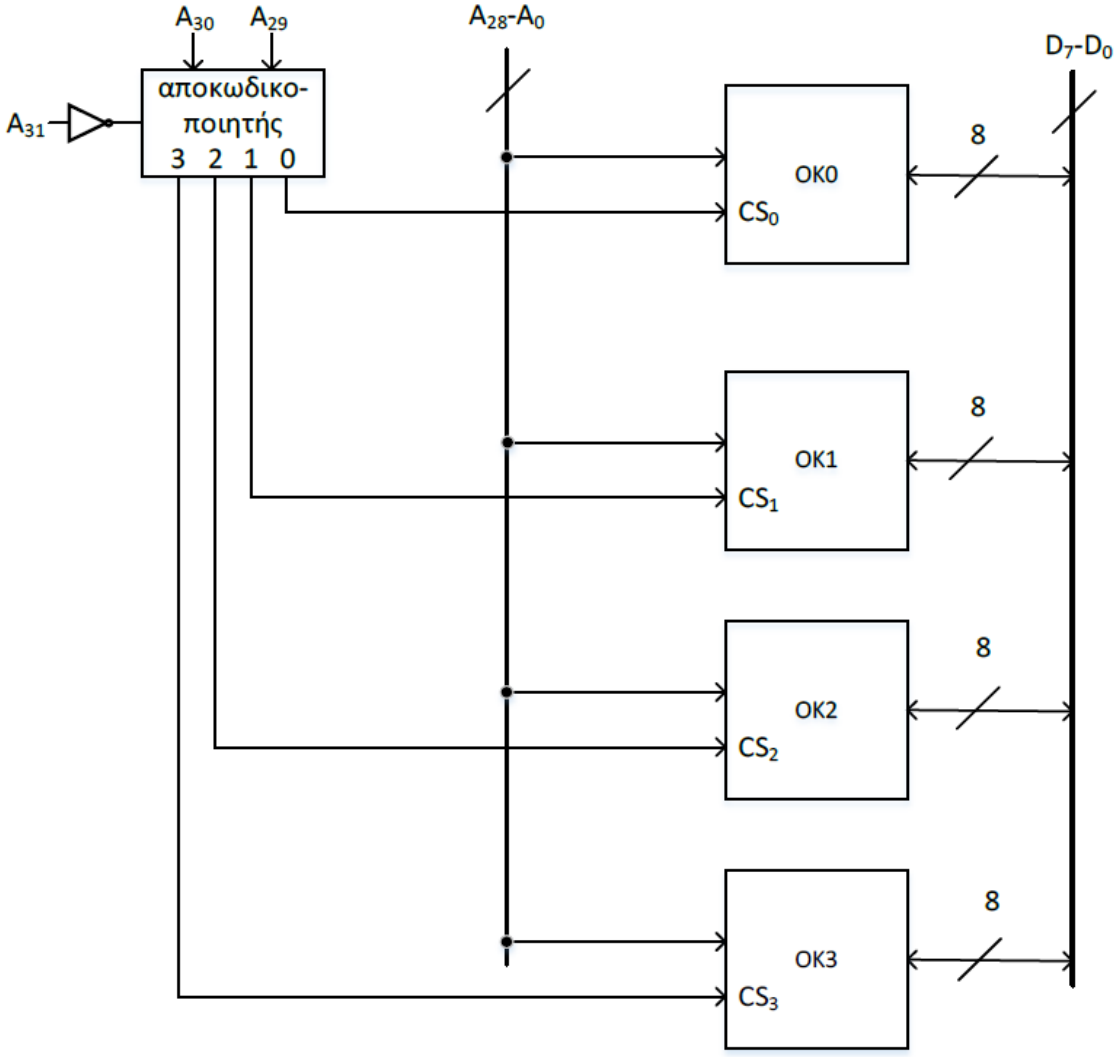
Θέμα Φεβρουαρίου 2016

Έχετε στη διάθεσή σας ολοκληρωμένα κυκλώματα μνήμης με χωρητικότητα 512 MBytes και μία ψηφιολέξη (1 ψηφιολέξη = 1 byte) ανά θέση μνήμης (δηλ. σε κάθε ψηφιολέξη αντιστοιχεί μια διεύθυνση). Να σχεδιάσετε σύστημα κύριας μνήμης με χωρητικότητα 2 GBytes και μία ψηφιολέξη ανά θέση μνήμης. Η αρτηρία διευθύνσεων και η αρτηρία δεδομένων είναι των 32 και των 8 δυαδικών ψηφίων αντίστοιχα. Θεωρήστε ότι έχετε στη διάθεσή σας οποιοδήποτε άλλο κύκλωμα χρειάζεστε. Ένα σχήμα στο οποίο να δίνεται όλη την πληροφορία είναι απαραίτητο και αρκετό.

Λύση

Διαθέτουμε ΟΚΜ 512 MBytes x **1 ψηφιολέξη/θ.μ.** → Στόχος: **ΚΜ 2 GBytes x 1 ψηφιολέξη/θ.μ.** Τα μεγέθη των δύο αρτηριών θα τα πάρουμε από την εκφώνηση της άσκησης. Αυτό σημαίνει ότι το data bus = 8 bits και το address bus = 32 bits. Επειδή έχουμε την ίδια εσωτερική οργάνωση μεταξύ των ΟΚ που διαθέτουμε και της Κ.Μ. που θα κατασκευάσουμε, τα ΟΚ επαναλαμβάνονται οριζόντια μόνο μια φορά. Όσον αφορά την κατακόρυφη επανάληψη, έχουμε $\frac{\text{θ.μ. που θέλουμε να έχουμε στο τελικό σύστημα ΚΜ}}{\text{θ.μ.που ήδη έχουμε στα ΟΚΜ που διαθέτουμε}} = \frac{2^{31}}{2^{29}} = 4$ επαναλήψεις. Τις θέσεις μνήμης τις έχουμε υπολογίσει με απλές μεθόδους των τριών:

1 θ.μ. 1 byte	1 θ.μ. 1 byte
x; 512MB	x; 2GB
$x = 2^{29}$ θ.μ.	$x = 2^{31}$ θ.μ.



Θέμα Ιουνίου 2016

Σ' ένα υπολογιστή ο επεξεργαστής έχει αρτηρία διευθύνσεων των 34 δυαδικών ψηφίων. Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης ΟΚ1 με χωρητικότητα 2048 Μ δυαδικών ψηφίων (Μ bit) και οργάνωση μιας ψηφιολέξης ανά θέση μνήμης και ολοκληρωμένα κυκλώματα μνήμης ΟΚ2 με χωρητικότητα 256 Μ δυαδικών ψηφίων και οργάνωση ενός δυαδικού ψηφίου ανά θέση μνήμης. Όλα τα ολοκληρωμένα κυκλώματα μνήμης είναι τύπου SRAM.

α. Να δώσετε το πλήθος των εισόδων διευθύνσεων και των εισόδων/εξόδων δεδομένων κάθε ολοκληρωμένου.

β. Να σχεδιάσετε σύστημα μνήμης με χωρητικότητα 9.216 Μ δυαδικών ψηφίων με εννέα δυαδικά ψηφία ανά θέση μνήμης, εκ των οποίων 8 δυαδικά ψηφία δεδομένων και 1 δυαδικό ψηφίο ισοτιμίας. Ένα σχήμα στο οποίο να φαίνεται όλη η πληροφορία είναι απαραίτητο και υπεραρκετό, δεν χρειάζεται αιτιολόγηση.

Λύση

Διαθέτουμε ΟΚ1 2048 Mbit x 8 bits/θ.μ. και **ΟΚ2** 256 Mbit x 1 bits/θ.μ. → **Στόχος**: Κ.Μ. 9216 Mbit x 9 bits/θ.μ.

Προϋπόθεση για να συνδέσουμε διαφορετικά ΟΚΜ παράλληλα μεταξύ τους, είναι να έχουν **ίδιο πλήθος θ.μ.**

Οριζόντια επανάληψη: 1 ΟΚ1 + 1 ΟΚ2 ←→ πάντα προσδιορίζεται με βάση την εσωτερική οργάνωση του τελικού συστήματος Κ.Μ. που θέλουμε να σχεδιάσουμε.

Κατακόρυφη επανάληψη: 4 ←→ πάντα προσδιορίζεται με βάση το πλήθος των θ.μ. του τελικού συστήματος Κ.Μ. που θέλουμε να σχεδιάσουμε.

1 θ.μ.	8 bits
x;	2048 Mbit

1 θ.μ.	1 bit
x;	256 Mbit

1 θ.μ.	9 bit
x;	9216 Mbit

$x = \frac{2^{11} x 2^{20}\ bit}{8\ bit/\theta.\mu} = 2^{28}\ \theta.\mu.$

$x = \frac{2^{8}\ x\ 2^{20}\ bit}{1\ bit/\theta.\mu} = 2^{28}\ \theta.\mu$

$x = \frac{9216\ x\ 2^{20}\ bit}{9\ bit/\theta.\mu} = 1024\ x\ 2^{20}\ \theta.\mu. = 2^{30}\ \theta.\mu.$

Επομένως $\frac{\textbf{θ.μ. που θέλουμε να έχουμε στο τελικό σύστημα ΚΜ}}{\textbf{θ.μ.που ήδη έχουμε στα ΟΚΜ που διαθέτουμε}} = \frac{2^{30}}{2^{28}} = 4$ επαναλήψεις

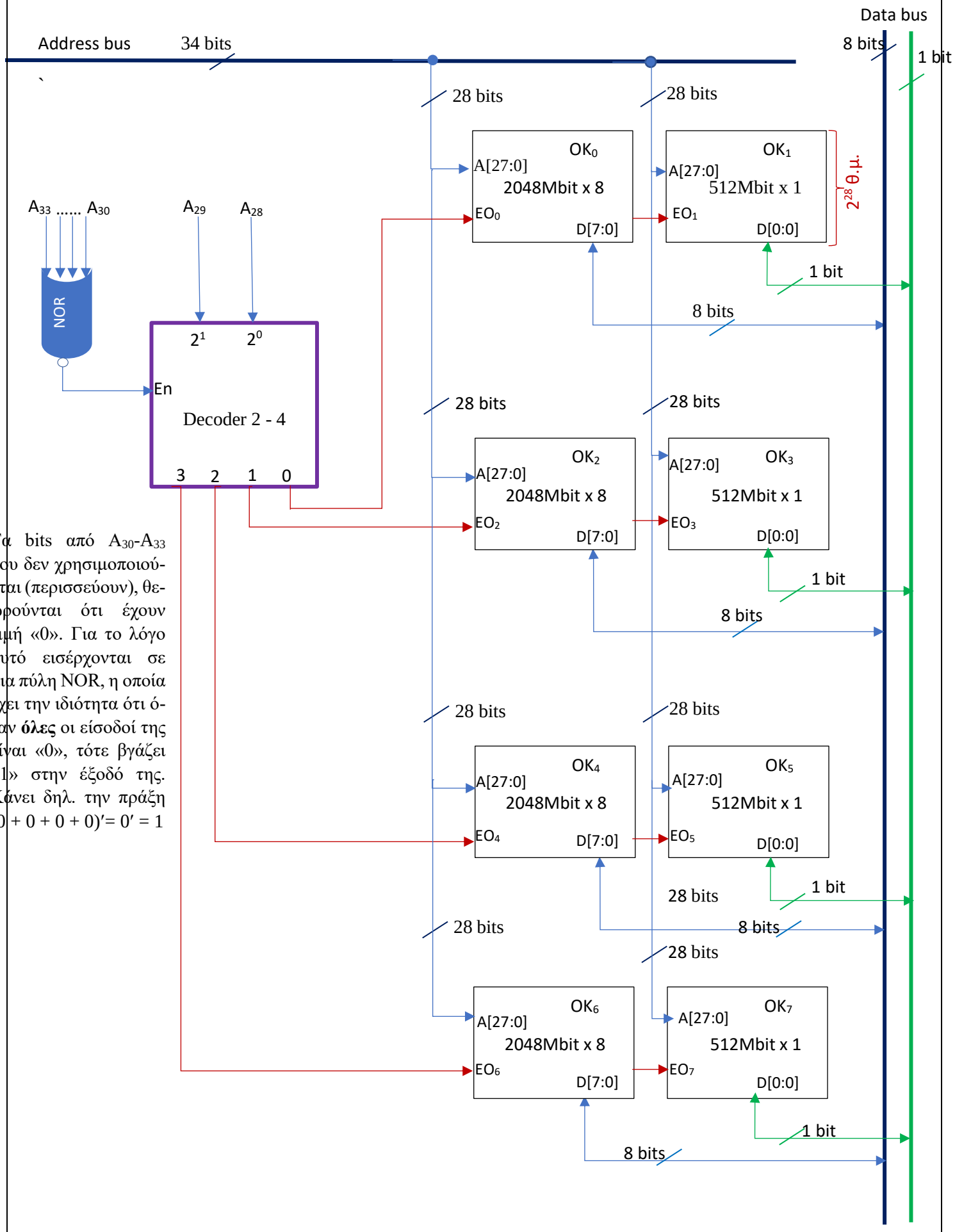
α. Πλήθος εισόδων διευθύνσεων και εισόδων/εξόδων δεδομένων **ΟΚ1**: 28 bits (address bus) και 8 bits (data bus) αντίστοιχα.

Πλήθος εισόδων διευθύνσεων και εισόδων/εξόδων δεδομένων **ΟΚ2**: 28 bits (address bus) και 1 bit (data bus) αντίστοιχα.

β. **Σχεδιασμός συστήματος Κ.Μ. με οργάνωση υψηλής τάξης διαφύλλωση**. Κατά σειρά σχεδιάζουμε τα ΟΚΜ (ΟΚ) με την οριζόντια και κατακόρυφη επανάληψη που έχουμε υπολογίσει νωρίτερα (2 και 4 αντίστοιχα), στη συνέχεια τα συνδέουμε με την αρτηρία δεδομένων στα δεξιά με αμφίδρομα βέλη (γιατί έχουμε μνήμη RAM) και με την αρτηρία διευθύνσεων των 34 bits, από τα οποία θα πάρουμε ως διακλάδωση τα 28 bits για τη διευθυνσιοδότηση των ΟΚΜ.

γ. **Στο τέλος θα σχεδιάσουμε τον αποκωδικοποιητή**, το μέγεθος του οποίου προσδιορίζεται από τον αριθμό των σειρών με ΟΚΜ που επιλέγουμε (στην προκειμένη περίπτωση έχουμε 4 σειρές ΟΚΜ με δύο ΟΚΜ ανά σειρά), και συνδέουμε την κάθε έξοδο του αποκωδικοποιητή με την γραμμή ενεργοποίησης (ΕΟ ή CS) όλων των ΟΚΜ που βρίσκονται στην ίδια σειρά.

Σχεδιασμός Κ.Μ.



Τα bits από A_{30}-A_{33} που δεν χρησιμοποιούνται (περισσεύουν), θεωρούνται ότι έχουν τιμή «0». Για το λόγο αυτό εισέρχονται σε μια πύλη NOR, η οποία έχει την ιδιότητα ότι όταν **όλες** οι είσοδοί της είναι «0», τότε βγάζει «1» στην έξοδό της. Κάνει δηλ. την πράξη $(0 + 0 + 0 + 0)' = 0' = 1$

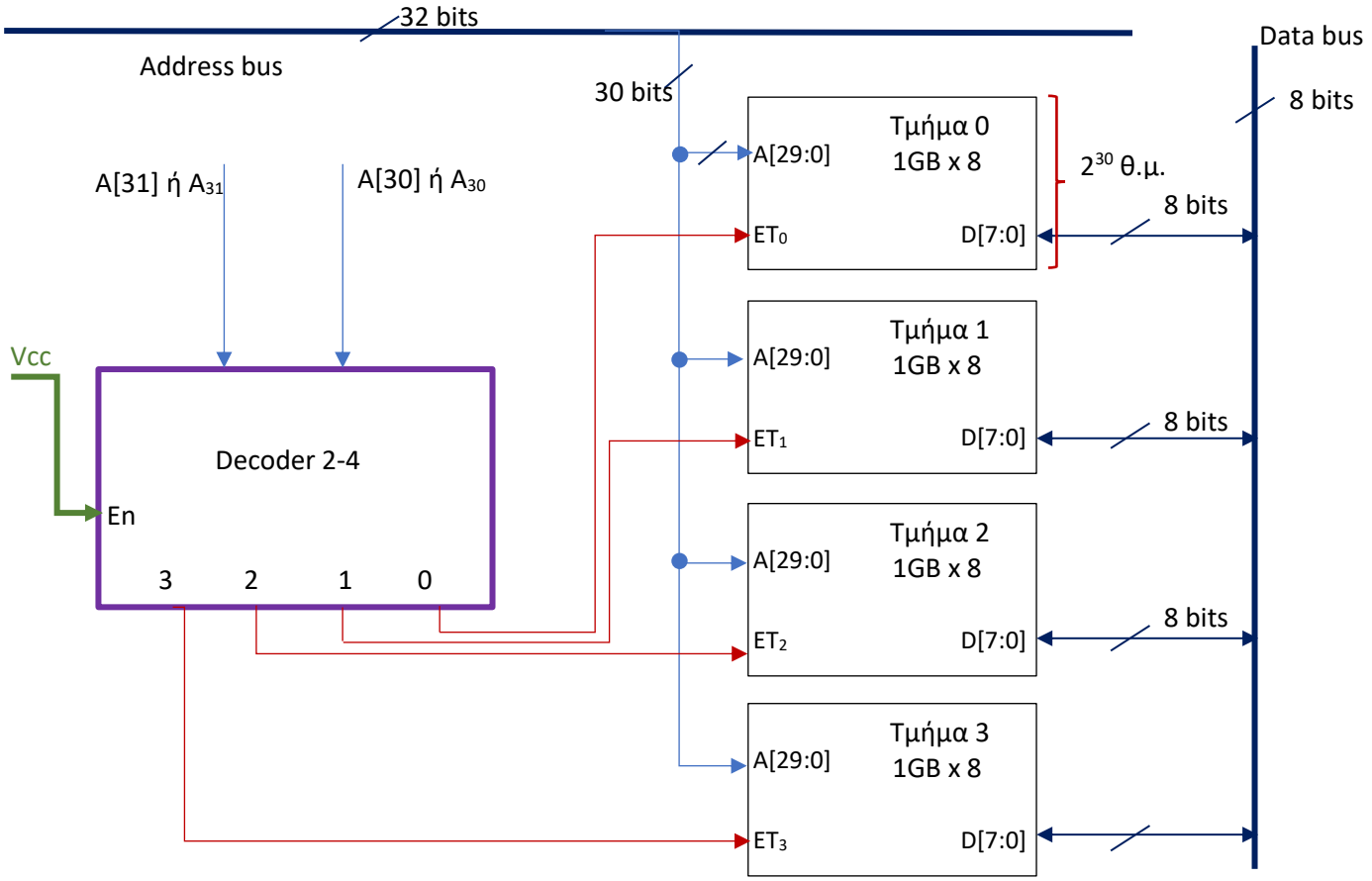
Άσκηση 5.4 με τμήματα μνήμης

Σ' ένα υπολογιστή η αρτηρία διευθύνσεων είναι των 32 δυαδικών ψηφίων. Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα των 256 Μψηφιολέξεων το κάθε ένα με οργάνωση μίας ψηφιολέξης ανά θέση μνήμης και αποκωδικοποιητές με είσοδο ενεργοποίησης. Να σχεδιάσετε σύστημα κύριας μνήμης με οργάνωση μίας ψηφιολέξης ανά θέσης μνήμης έτσι ώστε η κύρια μνήμη να είναι επεκτάσιμη κατά 1 Gψηφιολέξεις μέχρι το μέγιστο της χωρητικότητάς της. Θεωρήστε ότι το εύρος της αρτηρίας δεδομένων είναι 8. Να δώσετε τη δομή του συστήματος μνήμης ώστε να μπορεί να γίνεται η επέκταση μέχρι τη μέγιστη χωρητικότητα καθώς και τη δομή κάθε τμήματος με χωρητικότητα 1G ψηφιολέξεις, το οποίο χρησιμοποιείται κάθε φορά για την επέκταση.

Λύση

Διαθέτουμε ΟΚΜ των 256MB x 8 και θέλουμε να σχεδιάσουμε σύστημα κύριας μνήμης (ΚΜ), η οποία θα είναι επεκτάσιμη κατά 1GB, που αποτελεί **μια μονάδα χωρητικότητας διαφορετικού μεγέθους από αυτή που διαθέτουμε (256 MB)**. Για το λόγο αυτό θα πρέπει να χρησιμοποιήσουμε τμήματα μνήμης, όπου κάθε τμήμα μνήμης αποτελεί μια συλλογή ΟΚΜ και στην προκειμένη περίπτωση θα είναι μεγέθους 1 GB. **Επομένως, ο λόγος για τον οποίο χρησιμοποιούμε τμήματα μνήμης σχετίζεται με την επεκτασιμότητα της Κ.Μ**.

Το μέγιστο της χωρητικότητας (με άλλα λόγια η μέγιστη χωρητικότητα) της Κ.Μ. που θα κατασκευάσουμε θα είναι 2^{32} θ.μ., με δεδομένο ότι το address bus = 32 bits. Επειδή η οργάνωση της Κ.Μ. που θα σχεδιάσουμε θα είναι 1 ψηφιολέξη/θ.μ., αυτό σημαίνει ότι 2^{32} θ.μ. = 2^{32} bytes = 2^{30} x 2^2 bytes = 4 GB. Επομένως, με δεδομένο ότι διαθέτουμε τμήματα 1GB x 8 → Στόχος είναι 4 GB x 8, θα πρέπει να χρησιμοποιήσουμε 4 τμήματα μνήμης.

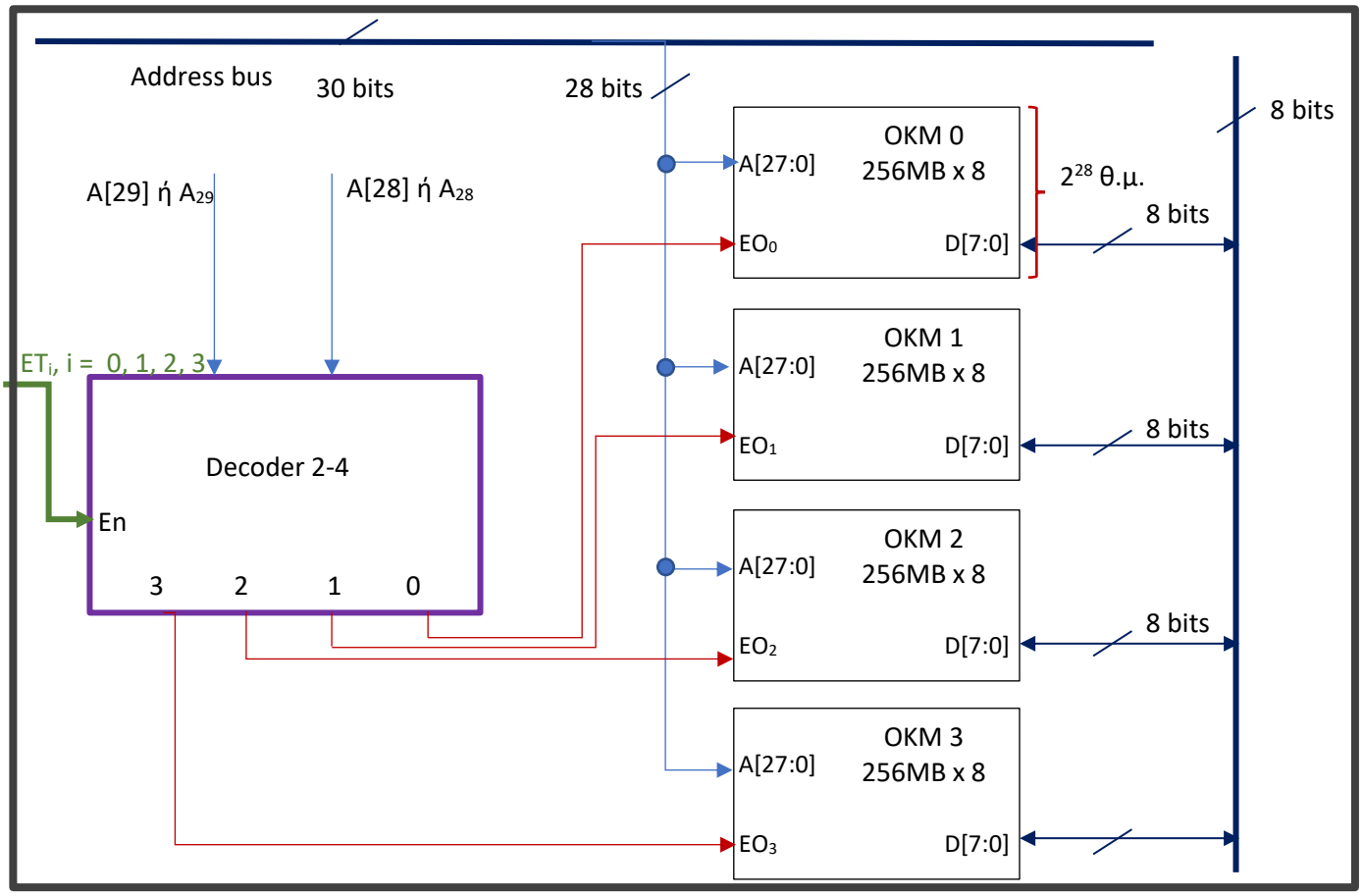


Ο **χάρτης διευθύνσεων μνήμης** για τα 4 τμήματα μνήμης που χρησιμοποιήσαμε είναι ο εξής:

Πίνακας 1: Χάρτης διευθύνσεων μνήμης του συστήματος μνήμης με τα 4 τμήματα

Τμήμα	Πλήθος εισόδων διευθύνσεων	Διευθύνσεις $A_{31}A_{30}A_{29}A_{28} \dots A_1A_0$
T_0	30	από 0 0 0 0 … 0 0 έως 0 0 1 1 … 1 1
T_1	30	από 0 1 0 0 … 0 0 έως 0 1 1 1 … 1 1
T_2	30	από 1 0 0 0 … 0 0 έως 1 0 1 1 … 1 1
T_3	30	από 1 1 0 0 … 0 0 έως 1 1 1 1 … 1 1

Στη συνέχεια, θα πρέπει να δείξουμε την κατασκευή του κάθε τμήματος μνήμης από τα επιμέρους ΟΚΜ που διαθέτουμε. Πιο συγκεκριμένα: Διαθέτουμε **ΟΚΜ 256MB x 8 → Τμήμα 1GB x 8**.



1 θ.μ. 1 byte
x; 256MB
$x = 2^8 \times 2^{20} / 1 = 2^{28}$ θ.μ.

1 θ.μ. 1 byte
x; 1GB
$x = 2^{30}/1 = 2^{30}$ θ.μ.

Επομένως $\frac{\text{θ.μ.που θέλουμε να έχουμε στο τελικό σύστημα ΚΜ}}{\text{θ.μ.που ήδη έχουμε στα ΟΚΜ που διαθέτουμε}} = \frac{2^{30}}{2^{28}} = 4$ επαναλήψεις

Ο χάρτης διευθύνσεων μνήμης για τα 4 ΟΚΜ μέσα σε κάθε τμήμα, είναι ο εξής:

Πίνακας 2: Χάρτης διευθύνσεων μνήμης για ένα τμήμα με τα ΟΚΜ που περιέχει

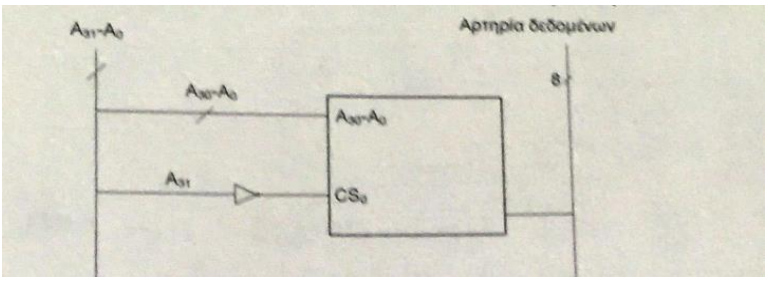
Ολοκληρωμένο κύκλωμα μνήμης	Πλήθος εισόδων διευθύνσεων	Διευθύνσεις $A_{29}A_{28}A_{27}A_{26}\ldots A_1A_0$
OK_0	28	από 0 0 0 0 … 0 0 έως 0 0 1 1 … 1 1
OK_1	28	από 0 1 0 0 … 0 0 έως 0 1 1 1 … 1 1
OK_2	28	από 1 0 0 0 … 0 0 έως 1 0 1 1 … 1 1
OK_3	28	από 1 1 0 0 … 0 0 έως 1 1 1 1 … 1 1

Θέμα Σεπτεμβρίου 2016

Έχετε στη διάθεσή σας ολοκληρωμένα κυκλώματα μνήμης με χωρητικότητα 512 M ψηφιολέξεων (Mbytes) και οργάνωση μιας ψηφιολέξης ανά θέση μνήμης και οποιαδήποτε συνδυαστικά κυκλώματα σας χρειάζονται.

α. Να σχεδιάσετε κύκλωμα μνήμης ΚΜ με χωρητικότητα 2 GBytes και οργάνωση μιας ψηφιολέξης ανά θέση μνήμης. Ένα σχήμα στο οποίο να φαίνεται όλη η πληροφορία είναι απαραίτητο και υπεραρκετό, δεν χρειάζεται αιτιολόγηση.

β. Στο σύστημα μνήμης που δίνεται στο επόμενο σχήμα να διασυνδέσετε κατάλληλα τόσα κυκλώματα μνήμης ΚΜ που σχεδιάσατε στο ερώτημα α ώστε το σύστημα της μνήμης να έχει συνολική χωρητικότητα 4 GBytes. Ένα σχήμα στο οποίο να φαίνεται όλη η πληροφορία είναι απαραίτητο.



Λύση

Διαθέτουμε ΟΚ 512 Μψηφιολέξεις x 8 bits/θ.μ. → **Στόχος**: Κ.Μ. 2GB x 8 bits/θ.μ.

Οριζόντια επανάληψη: 1 ←→ πάντα προσδιορίζεται με βάση την **εσωτερική οργάνωση της Κ.Μ.** που θα σχεδιάσουμε.

Κατακόρυφη επανάληψη: 4 ←→ πάντα προσδιορίζεται με βάση το **πλήθος των θ.μ. της Κ.Μ.** που θα σχεδιάσουμε.

1 θ.μ.	1 byte
x;	512 MB

$x = \frac{2^9 x 2^{20}\ bytes}{1\ byte/\theta.\mu} = 2^{29}$ θ.μ.

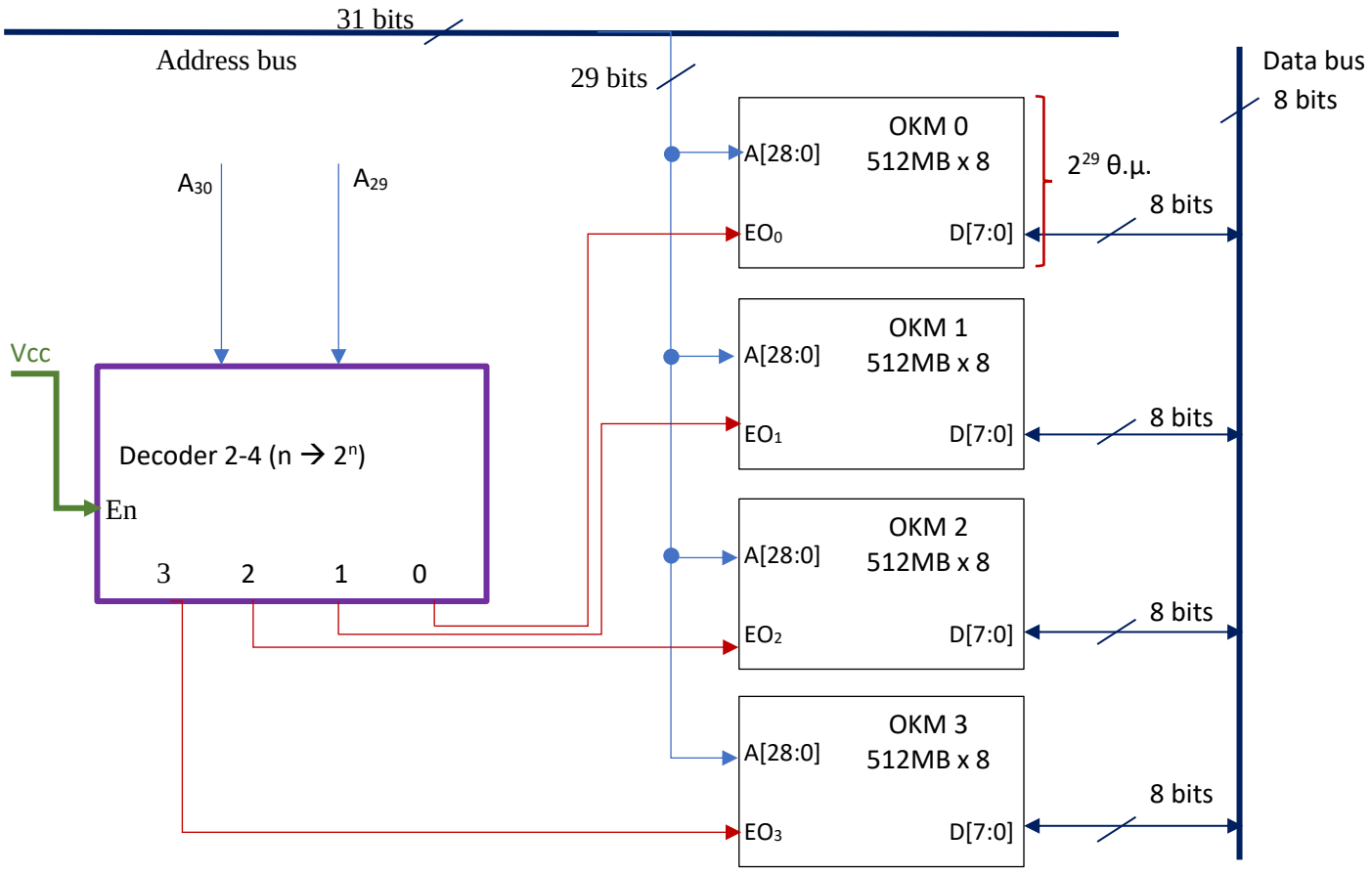
1 θ.μ.	1 byte
x;	2 GB

$x = \frac{2 \times 2^{30}\ bytes}{1\ byte/\theta.\mu}* = 2^{31}$ θ.μ.

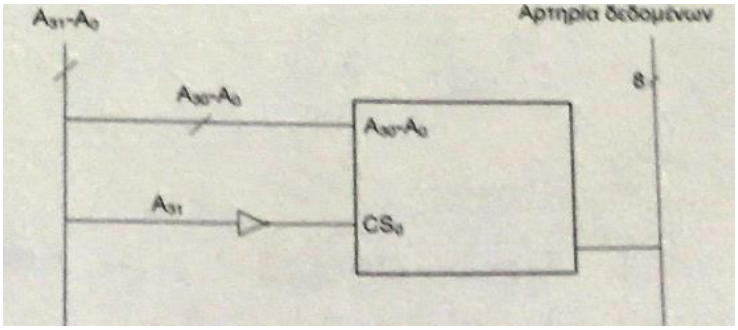
Επομένως $\frac{\textbf{θ.μ.που θέλουμε να έχουμε στο τελικό σύστημα ΚΜ}}{\textbf{θ.μ.που ήδη έχουμε στα ΟΚΜ που διαθέτουμε}} = \frac{2^{31}}{2^{29}} = 4$ επαναλήψεις

α. Στη συνέχεια προχωραμε στο σχεδιασμό της Κ.Μ. με βάση όσα έχουν αναφερθεί σε παρόμοιες ασκήσεις. Από τη στιγμή που η εκφώνηση **δεν** προσδιορίζει το μέγεθος του address bus, θα χρησιμοποιήσουμε ως μέγεθος τα 31 bits της Κ.Μ. που βρήκαμε προηγουμένως. Στην προκειμένη περίπτωση χρησιμοποιούμε όλα τα δυαδικά ψηφία από την αρτηρία διευθύνσεων προκειμένου να διευθυσιοδοτήσουμε τα ΟΚΜ, με αποτέλεσμα να χρησιμοποιούμε την τάση λειτουργίας V_{cc} για την ενεργοποίηση του αποκωδικοποιητή.

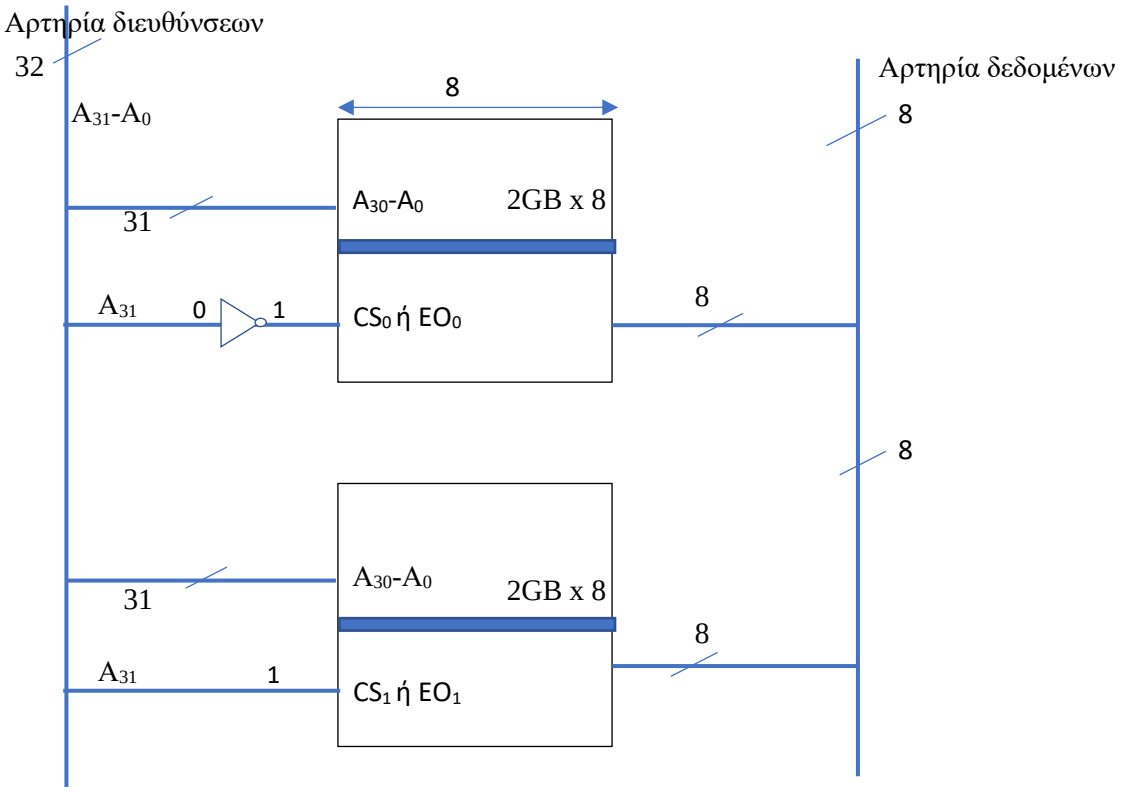
Σχεδιασμός Κ.Μ.



b) Έχουμε ως δεδομένο το ακόλουθο σχήμα.



Από τη στιγμή που το address bus (αρτηρία διευθύνσεων) είναι των 31 – bit ($A_{30} – A_0$), αυτό σημαίνει ότι έχουμε συνολικά 2^{31} θ.μ. και από τη στιγμή που το data bus (αρτηρία δεδομένων) είναι των 8 – bit, αυτό σημαίνει ότι από το ολοκληρωμένο βγαίνουν 8 bit, άρα **κάθε θέση μνήμης σε αυτό είναι των 8 – bit**, **κάτι που σημαίνει ότι έχουμε οργάνωση μιας ψηφιολέξης ανά θέση μνήμης**. Επομένως, η χωρητικότητα του υπάρχοντος συστήματος μνήμης είναι: Χωρητικότητα = πλήθος θ.μ. x μέγεθος κάθε θ.μ. = 2^{31} θ.μ. x 8 bits/θ.μ. = 1 byte/θ.μ. x 2^{31} θ.μ. = 2^{31} byte = 2 x 2^{30} byte = 2 GB. Εμείς θέλουμε χωρητικότητα 4 GB. Επομένως, πρέπει να επεκτείνουμε το υπάρχον σύστημα κατά 2 GB, προκειμένου να αποκτήσει χωρητικότητα 4 GB. Επειδή **δεν** υπάρχει αποκωδικοποιητής, χρησιμοποιείται το bit A_{31} ως bit επιλογής των ΟΚΜ (τόσο του αρχικού, όσο και αυτού που προσθέσαμε). **Επομένως η επιλογή ΟΚΜ μπορεί να γίνει και χωρίς τη χρήση αποκωδικοποιητή**, όπως στην προκειμένη περίπτωση, και συγκεκριμένα με κάποιο bit του address bus, το οποίο στην προκειμένη περίπτωση είναι το A_{31}. **Αν αυτό είναι «0» τότε επιλέγεται το πάνω ΟΚΜ, αν είναι «1», τότε επιλέγεται το κάτω ΟΚΜ**. Στη συνέχεια φαίνεται η επέκταση της υπάρχουσας ΚΜ κατά 2GB έτσι ώστε η συνολική χωρητικότητα να γίνει 4 GB.



Λύση θέματος Ιανουαρίου 2018

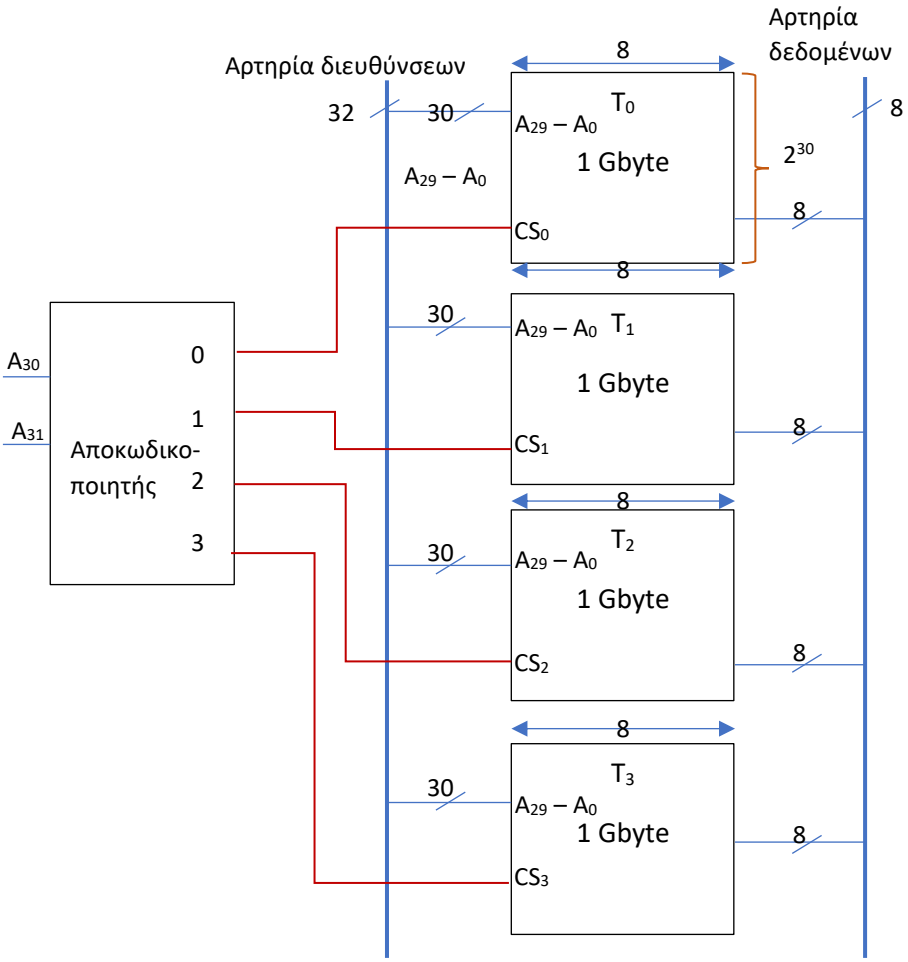
Στο διπλανό σχήμα φαίνεται η δομή της κύριας μνήμης υπολογιστικού συστήματος με αρτηρία διευθύνσεων των 32 δυαδικών ψηφίων, αρτηρία δεδομένων των 8 δυαδικών ψηφίων και οργάνωση κύριας μνήμης μιας ψηφιολέξης (byte) ανά θέση μνήμης.

α. Να γράψτε πάνω στο σχήμα τα δυαδικά ψηφία της αρτηρίας διευθύνσεων και της αρτηρίας δεδομένων που οδηγούν ή οδηγούνται από **κάθε τμήμα Tᵢ** (i= 0, 1, 2, 3) της μνήμης και του αποκωδικοποιητή.

β. Να φτιάξτε το χάρτη διευθύνσεων της κύριας μνήμης στον οποίο να φαίνεται ποιες διευθύνσεις χρησιμοποιούνται από το κάθε τμήμα.

γ. Έχετε στη διάθεσή σας ολοκληρωμένα κυκλώματα με χωρητικότητα **128 Mbytes το καθένα και οργάνωση 4 δυαδικών ψηφίων ανά θέση μνήμης**. Επίσης, έχετε στη διάθεσή σας αποκωδικοποιητές οποιουδήποτε μεγέθους, με είσοδο ενεργοποίησης.

δ. Να δώσετε τη δομή του τμήματος Ti της κύριας μνήμης. Ένα σχήμα στο οποίο να φαίνεται όλη η πληροφορία είναι απαιτούμενο και αρκετό.



Λύση

α. Από τη στιγμή που το εύρος της αρτηρίας δεδομένων είναι των 8 δυαδικών ψηφίων = 1 ψηφιολέξη, σημαίνει ότι από κάθε τμήμα T_i βγαίνουν 8 bits, οπότε κάθε θ.μ. είναι επίσης των 8 δυαδικών ψηφίων. Αυτό πρακτικά σημαίνει ότι το 1 GB = 2^{30} bytes = 2^{30} θ.μ. (δηλ. έχουμε οργάνωση μιας ψηφιολέξης/θ.μ.), επομένως το μέρος του address bus (αρτηρία διευθύνσεων) που χρησιμοποιείται για τη διευθυνσιοδότηση των τμημάτων είναι των 30 δυαδικών ψηφίων ($A_{29} – A_0$). Άρα από τα 32 δυαδικά ψηφία που αποτελούν την αρτηρία διευθύνσεων, τα 30 δυαδικά ψηφία χρησιμοποιούνται για τη διευθυνσιοδότηση των τμημάτων και αυτά που περισσεύουν (2 δυαδικά ψηφία) χρησιμοποιούνται ως είσοδοι στον αποκωδικοποιητή ($A_{31} – A_{30}$). Ο αποκωδικοποιητής **δεν** διαθέτει σήμα ενεργοποίησης, οπότε δεν χρησιμοποιούμε την τάση V_{cc} για να τον επιλέξουμε. Επιπλέον, οι ακροδέκτες επιλογής $CS_0 – CS_3$ αναφέρονται σε επιλογή τμημάτων και όχι σε επιλογή ΟΚΜ, διότι σύμφωνα με την εκφώνηση, στο δοθέν σχήμα απεικονίζονται τμήματα και όχι ολοκληρωμένα κυκλώματα μνήμης. Στη συνέχεια, ακολουθεί η σχεδίαση των 4 τμημάτων με τις λεπτομέρειες που φαίνονται πάνω στο σχήμα.

β. Στη συνέχεια φαίνεται ο πρώτος από τους δύο συνολικά χάρτες διευθύνσεων μνήμης που θα πρέπει να φτιάξουμε. Αυτός αφορά συνολικά όλο το σύστημα της Κ.Μ. Ο δεύτερος χάρτης διευθύνσεων που ακολουθεί, αναφέρεται σε κάθε επιμέρους τμήμα χωριστά.

Πίνακας 3: Χάρτης διευθύνσεων μνήμης του συστήματος μνήμης με τα 4 τμήματα

Τμήμα	Πλήθος εισόδων διευθύνσεων	Διευθύνσεις $A_{31}A_{30}A_{29}A_{28}$.....A_1A_0
T_0	30	00 0000000000.....00000 00 1111111111.....11111
T_1	30	01 0000000000.....00000 01 1111111111.....11111
T_2	30	10 0000000000.....00000 10 1111111111.....11111
T_3	30	11 0000000000.....00000 11 1111111111.....11111

γ1)

Πίνακας 2: Χάρτης διευθύνσεων μνήμης του κάθε τμήματος

Τμήμα	Πλήθος εισόδων διευθύνσεων	Διευθύνσεις $A_{29}A_{28}A_{27}$.....A_1A_0
$OK_0 – OK_1$	28	00 0000000000.....00000 00 1111111111.....11111
$OK_2 – OK_3$	28	01 0000000000.....00000 01 1111111111.....11111
$OK_4 – OK_5$	28	10 0000000000.....00000 10 1111111111.....11111
$OK_6 – OK_7$	28	11 0000000000.....00000 11 1111111111.....11111

γ2) Διαθέτουμε ΟΚΜ 128 MB x 4 → Στόχος: 1 GB x 8 (Τμήμα)

Οριζόντια επανάληψη: 2 (διότι πρέπει να συνδέσουμε 2 ΟΚΜ παράλληλα για να δημιουργήσουμε το εύρος των 8 bits)

Κατακόρυφη επανάληψη: 4 (διότι πρέπει να δημιουργήσουμε το επιθυμητό πλήθος θ.μ.).

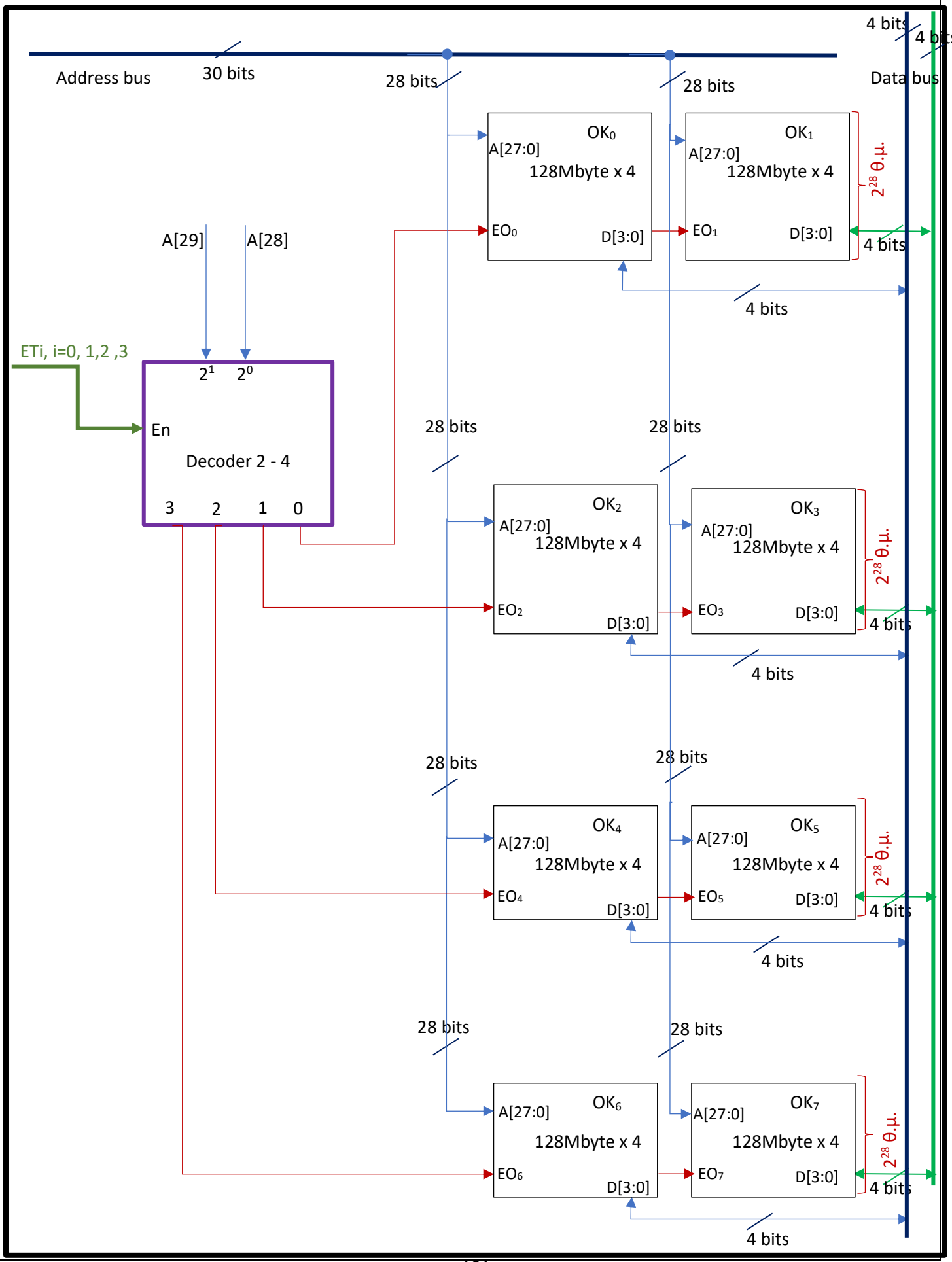
1 θ.μ.	4 bits
x;	128 Mbyte

$$x = \frac{2^7 \, x \, 2^{20} \, x \, 2^3 \, bits}{2^2 \, bits/\theta.\mu} * = 2^{28} \, \theta.\mu.$$

1 θ.μ.	1 byte
x;	1 GB

$$x = \frac{2^{30} \text{ bytes}}{1 \text{ byte}/\theta.\mu} * = 2^{30} \, \theta.\mu.$$

Επομένως $\frac{\text{θ.μ.που θέλουμε να έχουμε στο τελικό σύστημα ΚΜ}}{\text{θ.μ.που ήδη έχουμε στα ΟΚΜ που διαθέτουμε}} = \frac{2^{30}}{2^{28}} = 4$ επαναλήψεις



Θέμα Ιουνίου 2014

Να σχεδιάστε σύστημα κύριας μνήμης με χωρητικότητα 6 Mbytes και οργάνωση 1 byte ανά θέση μνήμης (σε κάθε byte αντιστοιχεί μια διεύθυνση). Η αρτηρία διευθύνσεων είναι των 32 δυαδικών ψηφίων. Έχετε στη διάθεσή σας **4 ολοκληρωμένα κυκλώματα** μνήμης με χωρητικότητα 4 Mbits το καθένα και οργάνωση 2 bits ανά θέση μνήμης και **1 ολοκληρωμένο κύκλωμα** μνήμης με χωρητικότητα 32 Mbits το καθένα και οργάνωση 8 bits ανά θέση μνήμης. Ένα σχήμα στο οποίο να φαίνεται όλη η πληροφορία είναι απαιτούμενο και αρκετό.

Λύση

Διαθέτουμε 4 ΟΚΜ1 των 4Mbits x 2 bits και 1 ΟΚΜ0 των 32Mbits x 8 bits → **Στόχος:** ΚΜ 6MB x 8 bits.

- **Οριζόντια επανάληψη** (στόχος είναι η δημιουργία του **κατάλληλου εύρους της θέσης μνήμης**). Επειδή διαθέτουμε ένα πρώτο είδος ΟΚΜ με εσωτερική οργάνωση 1 byte/θ.μ. και θέλουμε να σχεδιάσουμε σύστημα Κ.Μ. με την **ίδια** εσωτερική οργάνωση, αρκεί **μόνο** μια οριζόντια επανάληψη για το συγκεκριμένο ολοκληρωμένο. Για το λόγο αυτό δίνεται από την εκφώνηση ότι έχουμε 1 ΟΚΜ αυτού του είδους Επιπλέον, διαθέτουμε και ένα δεύτερο είδος ΟΚΜ με εσωτερική οργάνωση 2 bits/θ.μ. Θα πρέπει αυτό να το επαναλάβουμε οριζόντια 4 φορές, για να δημιουργήσουμε και πάλι το επιθυμητό εύρος των 8 bits/θ.μ. Για το λόγο αυτό δίνεται από την εκφώνηση ότι έχουμε 4 ΟΚΜ αυτού του είδους.

- **Κατακόρυφη επανάληψη** (στόχος είναι το **επιθυμητό πλήθος θέσεων μνήμης**) και προκύπτει από το κλάσμα: $\frac{\text{θ.μ. που θέλουμε να έχουμε στο τελικό σύστημα ΚΜ}}{\text{θ.μ. που ήδη έχουμε στα ΟΚΜ που διαθέτουμε}}$. Για σωστό υπολογισμό αυτού του κλάσματος, θα πρέπει πρώτα με απλές μεθόδους των τριών, να μετατρέψουμε τις χωρητικότητες τόσο των ΟΚΜ που διαθέτουμε όσο και της Κ.Μ. που θα σχεδιάσουμε, σε θέσεις μνήμης

1 θ.μ.	2 bits
x;	4 Mbit

$x = \frac{4 \text{ x } 2^{20} \text{ bits}}{2 \text{ bits/θ.μ}} * = 2^{21}$ θ.μ.

1 θ.μ.	8 bits
x;	32 Mbit

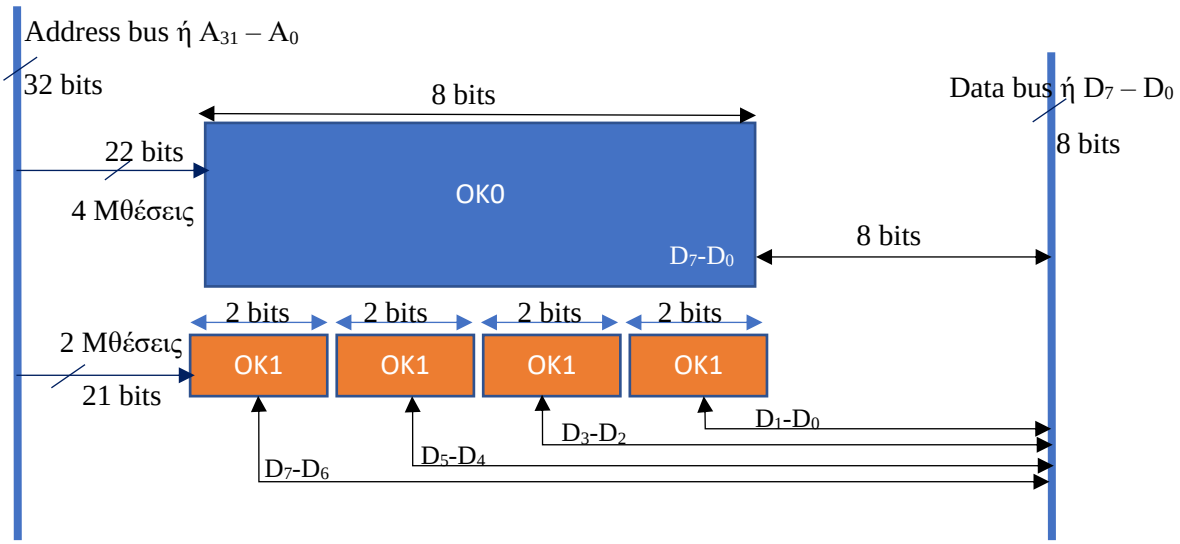
$x = \frac{2^5 2^{20} \text{ bits}}{2^3 \text{ bits/θ.μ.}} * = 2^{22}$ θ.μ.

1 θ.μ.	8 bits
x;	6 MB

$x = \frac{6 \text{ x } 2^3 \text{x } 2^{20} \text{ bits}}{2^3 \text{ bits/θ.μ.}} * = 6 \text{ x } 2^{20}$ θ.μ

- Κάθε ΟΚ1 αφού περιέχει 2^{21} θ.μ., έχει $2^{21} = 2^1 2^{20} = 2$ Μθέσεις των 2 bits η καθεμία.
- Κάθε ΟΚ0 αφού περιέχει 2^{22} θ.μ., έχει $2^{22} = 2^2 2^{20} = 4$ Μθέσεις των 8 bits η καθεμία.
- Η **κύρια μνήμη** που θα σχεδιάσουμε έχει 6 x 2^{20} θ.μ. = 6 Μθέσεις των 8 bits η καθεμία

Ακολουθεί ο σχεδιασμός της κύριας μνήμης



Θέμα Σεπτεμβρίου 2015

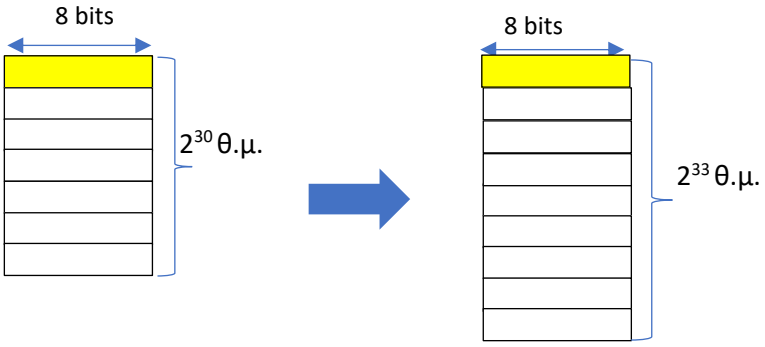
Θεωρήστε υπολογιστικό σύστημα με αρτηρία διευθύνσεων των 36 δυαδικών ψηφίων και αρτηρία δεδομένων των 16 δυαδικών ψηφίων. Να σχεδίασετε σύστημα κύριας μνήμης με χωρητικότητα 8 Gbytes (1 byte = 1 ψηφιολέξη = 8 δυαδικά ψηφία) και οργάνωση μίας ψηφιολέξης ανά θέση μνήμης (δηλαδή σε κάθε μία ψηφιολέξη αντιστοιχεί μια διεύθυνση) ώστε σε κάθε προσπέλαση να προσπελαύνεται το περιεχόμενο δύο διαδοχικών θέσεων μνήμης εκ' των οποίων η μικρότερη διεύθυνση είναι άρτια. Έχετε στη διάθεσή σας ολοκληρωμένα κυκλώματα (ΟΚ) των 1 Gbytes με οργάνωση 8 δυαδικών ψηφίων ανά θέση και ένα αποκωδικοποιητή με είσοδο ενεργοποίησης (όποιον σας χρειάζεται). Να σχεδιάσετε το σύστημα της κύριας μνήμης. Στο σχήμα να δώσετε όλη την απαιτούμενη πληροφορία.

Λύση

Διαθέτουμε ΟΚΜ 1GB x 8 bits/θ.μ. → **Στόχος**: Κ.Μ. 8 GB x 8 bits/θ.μ.

Για να πετύχουμε το στόχο μας, πρέπει να επαναλάβουμε οριζόντια και κατακόρυφα τα ΟΚΜ που μας δίνονται. Με άλλα λόγια, πρέπει να εφαρμόσουμε οριζόντια και κατακόρυφη επανάληψη των ΟΚΜ που έχουμε.

- **Οριζόντια επανάληψη** (στόχος είναι η δημιουργία του **κατάλληλου εύρους της θέσης μνήμης**). Παρότι έχουμε ίδια εσωτερική οργάνωση τόσο στα ΟΚΜ που διαθέτουμε, όσο και στην Κ.Μ. που θα σχεδιάσουμε, το γεγονός ότι σε μια **προσπέλαση μνήμης θέλουμε να προπελαύνονται δύο διαδοχικές θ.μ. ταυτόχρονα**, σημαίνει ότι θα πρέπει να συνδέσουμε δύο ΟΚΜ παράλληλα μεταξύ τους (δηλ. οριζόντια επανάληψη = 2).



- **Κατακόρυφη επανάληψη** (στόχος είναι το **επιθυμητό πλήθος θέσεων μνήμης**) και προκύπτει από το κλάσμα: $\frac{\textbf{θ.μ.που θέλουμε να έχουμε στο τελικό σύστημα ΚΜ}}{\textbf{θ.μ. που ήδη έχουμε στα ΟΚΜ που διαθέτουμε}}$. Για σωστό υπολογισμό αυτού του κλάσματος, θα πρέπει **πρώτα** με απλές μεθόδους των τριών, να μετατρέψουμε τις χωρητικότητες τόσο των ΟΚΜ που διαθέτουμε όσο και της Κ.Μ. που θα σχεδιάσουμε, σε θέσεις μνήμης.

1 θ.μ.	1 byte
x;	1 GB

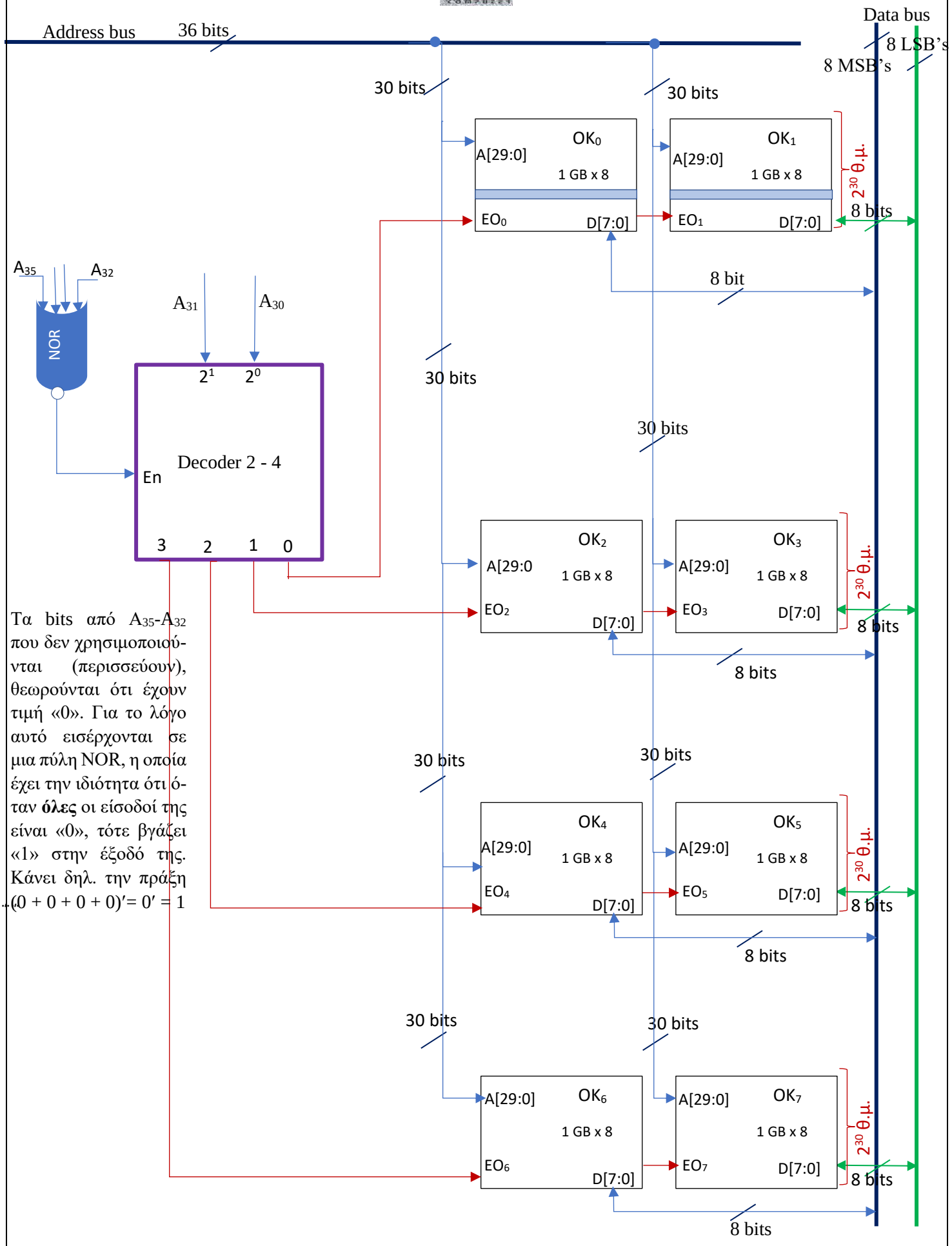
$x = \frac{2^{30}\ bytes}{1\ byte/θ.μ} = 2^{30}$ θ.μ.

1 θ.μ.	1 byte
x;	8 GB

$x = \frac{2^3\ x 2^{30}\ bytes}{1\ byte/θ.μ} = 2^{33}$ θ.μ

Επομένως $\frac{\textbf{θ.μ.που θέλουμε να έχουμε στο τελικό σύστημα ΚΜ}}{\textbf{θ.μ.που ήδη έχουμε στα ΟΚΜ που διαθέτουμε}} = \frac{2^{33}}{2^{30}} = 8$ επαναλήψεις

Στη συνέχεια προχωράμε στο σχεδιασμό της Κ.Μ. με βάση την οριζόντια και την κατακόρυφη επανάληψη που υπολογίσαμε. Στο σχεδιασμό αυτό, θα πρέπει να συνδέσουμε τα ΟΚΜ με την αρτηρία δεδομένων και την αρτηρία διευθύνσεων. Επειδή πρέπει σε κάθε προσπέλαση να προσπελαύνεται το περιεχόμενο δύο διαδοχικών θέσεων μνήμης, συνδέουμε ανά δύο τα ΟΚΜ σε μια γραμμή, έτσι ώστε να επιλέγονται ταυτόχρονα με την ίδια έξοδο του αποκωδικοποιητή. Αν η εκφώνηση ζητούσε την επιλογή 4 διαδοχικών θέσεων μνήμης, θα τα συνδέαμε ανά 4 μαζί.



Τα bits από A_{35}-A_{32} που δεν χρησιμοποιούνται (περισσεύουν), θεωρούνται ότι έχουν τιμή «0». Για το λόγο αυτό εισέρχονται σε μια πύλη NOR, η οποία έχει την ιδιότητα ότι όταν **όλες** οι είσοδοί της είναι «0», τότε βγάζει «1» στην έξοδό της. Κάνει δηλ. την πράξη $(0 + 0 + 0 + 0)' = 0' = 1$

Θέμα Φεβρουαρίου 2021

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης ΟΚ0 με χωρητικότητα 32 Mbits και 8 δυαδικά ψηφία ανά θέση μνήμης καθώς και ολοκληρωμένα κυκλώματα μνήμης ΟΚ1 με χωρητικότητα 1 MBytes και 8 δυαδικά ψηφία ανά θέση μνήμης. Πόσα ολοκληρωμένα κυκλώματα τύπου ΟΚ0 και πόσα τύπου ΟΚ1 χρειάζεστε για να σχεδιάσετε ένα σύστημα μνήμης με χωρητικότητα 176 Mbits και 16 δυαδικά ψηφία ανά θέση ώστε το συνολικό πλήθος των ολοκληρωμένων που χρησιμοποιήσατε να είναι το ελάχιστο; Να δώσετε το αποτέλεσμα ως δύο αριθμούς που χωρίζονται με το σύμβολο + πχ. 4+12 το οποίο σημαίνει 4 τύπου ΟΚ0 και 12 τύπου ΟΚ1. Προσοχή μη βάλετε κενά μεταξύ των συμβόλων.

Αποτέλεσμα: [ ].

Λύση

ΟΚ0

1 θ.μ. 8 bits

x; 32 Mbits

$x = 2^5 \times 2^{20}/2^3 = 2^{22}$ θ.μ.

ΟΚ1

1 θ.μ. 8 bits

x; 1 MB

$x = 2^{20} \times 2^3/2^3 = 2^{20}$ θ.μ.

Κύρια μνήμη

1 θ.μ. 16 bits

x; 176 Mbits

$x = 176 \times 2^{20}/16 = 11 \times 2^{20}$ θ.μ.

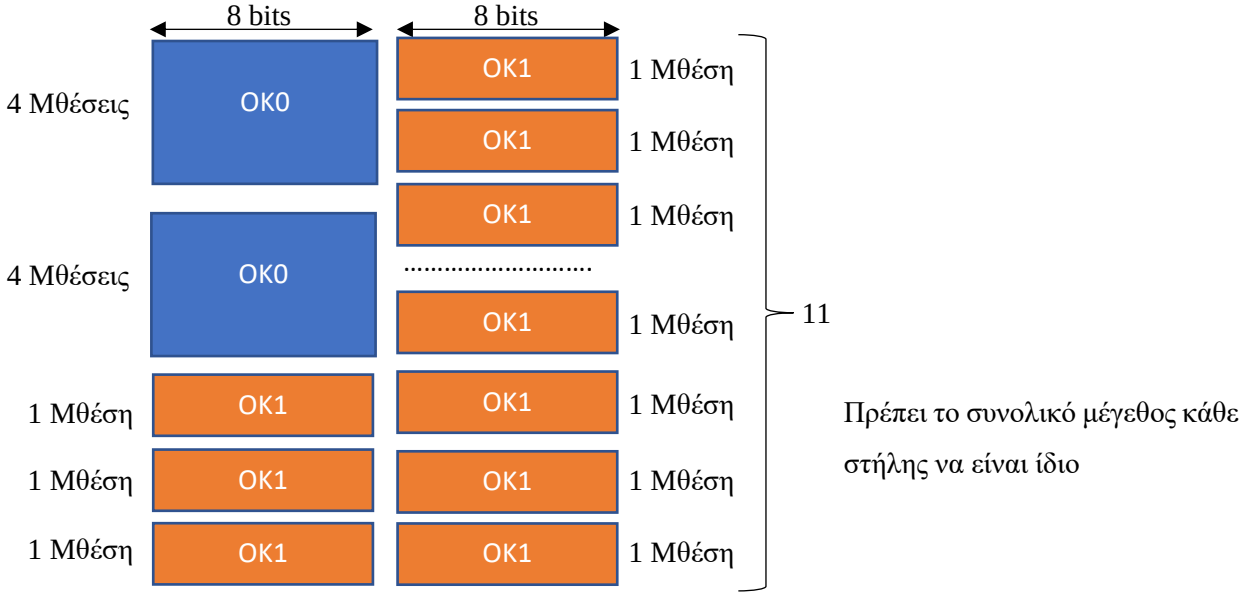
Οριζόντια επανάληψη: **2** (8 bits + 8 bits), πάντα **δημιουργεί το επιθυμητό εύρος μνήμης**

Κατακόρυφη επανάληψη: $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα ΟΚΜ}} = \frac{11 \times 2^{20} \text{ θ.μ.}}{2^{22} \text{ θ.μ.}} = 11 \times 2^{-2} = \frac{11}{4} = 2.75$. **Δεν επιστρέφεται ακέραιο πηλίκο**

Κατακόρυφη επανάληψη: $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα ΟΚΜ}} = \frac{11 \times 2^{20} \text{ θ.μ.}}{2^{20} \text{ θ.μ.}} = 11$. **Επιστρέφεται ακέραιο πηλίκο**

- Κάθε ΟΚ0 αφού περιέχει 2^{22} θ.μ., έχει $2^{22} = 2^2\, 2^{20} = 4$ Μθέσεις του ενός byte η καθεμία.
- Κάθε ΟΚ1 αφού περιέχει 2^{20} θ.μ., έχει 1 Μθέση του ενός byte η καθεμία.
- Η **κύρια μνήμη** που θα σχεδιάσουμε έχει 11×2^{20} θ.μ. = 11 Μθέσεις των 16 bits η καθεμία

Για να σχεδιάσουμε το επιθυμητό μέγεθος της κύριας μνήμης, θα πρέπει να επαναλάβουμε κατακόρυφα 11 ΟΚ1 (έχουμε ακέραιο πηλίκο) και όσον αφορά τα ΟΚ0, θα πρέπει να επαναλάβουμε κατακόρυφα **2 ΟΚ0** (στρογγυλοποίηση προς το μικρότερο ακέραιο) και το δεκαδικό μέρος που περισσεύει, θα το δημιουργήσουμε από τα ΟΚ1, όπως φαίνεται:



Άρα, χρειαζόμαστε 2 ΟΚ0 + 14 ΟΚ1 → 2+14

Επαλήθευση: 2 * 32 Mbits + 14 * 1 MB = 2 * 32 Mbits + 14 * 8 Mbits = 64 Mbits + 112 Mbits = 176 Mbits

1 byte 8 bits

1 MB x;

$x = 2^3 \times 2^{20}$ bits = 8 Mbits

Θέμα Ιουνίου 2021

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης ΟΚ0 με χωρητικότητα 16 Mbits και 8 δυαδικά ψηφία ανά θέση μνήμης καθώς και ολοκληρωμένα κυκλώματα μνήμης ΟΚ1 με χωρητικότητα 0.5 MBytes και 4 δυαδικά ψηφία ανά θέση μνήμης. Πόσα ολοκληρωμένα κυκλώματα τύπου ΟΚ0 και πόσα τύπου ΟΚ1 χρειάζεστε για να σχεδιάσετε ένα σύστημα μνήμης με χωρητικότητα 60 Mbits και 12 δυαδικά ψηφία ανά θέση ώστε το συνολικό πλήθος των ολοκληρωμένων που χρησιμοποιήσατε να είναι το ελάχιστο; Να δώσετε το αποτέλεσμα ως δύο αριθμούς που χωρίζονται με το σύμβολο + πχ. 4+12 το οποίο σημαίνει 4 τύπου ΟΚ0 και 12 τύπου ΟΚ1. Προσοχή μη βάλετε κενά μεταξύ των συμβόλων.

Αποτέλεσμα: .

Λύση

ΟΚ0

1 θ.μ.	8 bits
x;	16 Mbits

$x = 2^4 \times 2^{20}/2^3 = 2^{21}$ θ.μ.

ΟΚ1

1 θ.μ.	4 bits
x	1/2 MB

$x = ½ * 2^{20} \times 2^3/2^2 = 2^{-1} \times 2^{21}$ θ.μ. $= 2^{20}$ θ.μ.

Κύρια μνήμη

1 θ.μ.	12 bits
x	60 Mbits

$x = 60 \times 2^{20}/12 = 5 \times 2^{20}$ θ.μ.

Οριζόντια επανάληψη: **2** (8 bits + 4 bits), πάντα **δημιουργεί το επιθυμητό εύρος μνήμης**

Κατακόρυφη επανάληψη: $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα ΟΚΜ}} = \frac{5 \times 2^{20} \text{ θ.μ.}}{2^{21} \text{ θ.μ.}} = 5 \times 2^{-1} = 2.5$. **Δεν επιστρέφεται ακέραιο πηλίκο**

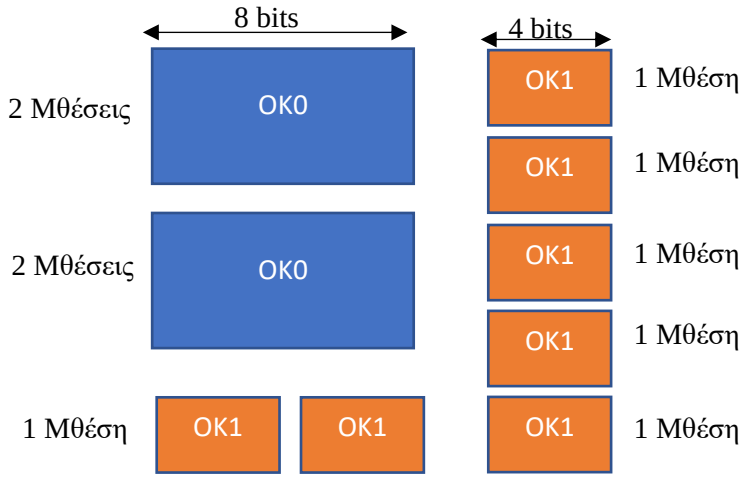
Κατακόρυφη επανάληψη: $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα ΟΚΜ}} = \frac{5 \times 2^{20} \text{ θ.μ}}{2^{20} \text{ θ.μ.}} = 5$. **Επιστρέφεται ακέραιο πηλίκο**

Κάθε **ΟΚ0** αφού περιέχει 2^{21} θ.μ., έχει $2^{21} = 2^1 2^{20}$ θ.μ. = 2 Μθέσεις του ενός byte η καθεμία.

Κάθε **ΟΚ1** αφού περιέχει 2^{20} θ.μ., έχει 1 Μθέση των 4 bits η καθεμία.

Η **κύρια μνήμη** που θα σχεδιάσουμε έχει 5×2^{20} θ.μ. = 5 Μθέσεις των 12 bits η καθεμία

Για να σχεδιάσουμε το επιθυμητό μέγεθος της κύριας μνήμης, θα πρέπει να επαναλάβουμε κατακόρυφα 5 ΟΚ1 (έχουμε ακέραιο πηλίκο) και όσον αφορά τα ΟΚ0, θα πρέπει να επαναλάβουμε κατακόρυφα 2 ΟΚ0 (στρογγυλοποίηση προς το μικρότερο ακέραιο) και το δεκαδικό μέρος που περισσεύει, θα το δημιουργήσουμε από τα ΟΚ1, όπως φαίνεται παρακάτω:



Πρέπει το συνολικό μέγεθος κάθε στήλης να είναι ίδιο

Άρα, χρειαζόμαστε 2 ΟΚ0 + 7 ΟΚ1 → 2+7

Επαλήθευση: 2 * 16 Mbits + 7 * 1/2 MB = 2 * 16 Mbits + 7 * 4 Mbits = 32 Mbits + 28 Mbits = 60 Mbits

1 byte	8 bits
1/2 MB	x;

$x = 2^3 \times 0.5 \times 2^{20}$ bits $= 2^3 \times 2^{-1} \times 2^{20}$ bits $= 2^2 \times 2^{20}$ bits = 4 Mbits.

Θέμα Ιουνίου 2020

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης ΟΚ0 με χωρητικότητα 48Mbits και 6 δυαδικά ψηφία ανά θέση μνήμης καθώς και ολοκληρωμένα κυκλώματα μνήμης ΟΚ1 με χωρητικότητα 1.5 MBytes και 3 δυαδικά ψηφία ανά θέση μνήμης. Πόσα ολοκληρωμένα κυκλώματα τύπου ΟΚ0 και πόσα τύπου ΟΚ1 χρειάζεστε για να σχεδιάσετε ένα σύστημα μνήμης με χωρητικότητα 252Mbits και 9 δυαδικά ψηφία ανά θέση ώστε το συνολικό πλήθος των ολοκληρωμένων που χρησιμοποιήσατε να είναι το ελάχιστο; Να δώσετε το αποτέλεσμα ως δύο αριθμούς που χωρίζονται με το σύμβολο + πχ. 4+12 το οποίο σημαίνει 4 τύπου ΟΚ0 και 12 τύπου ΟΚ1. Προσοχή μη βάλετε κενά μεταξύ των συμβόλων.

Αποτέλεσμα: [ ].

Λύση

ΟΚ0

1 θ.μ.	6 bits
x;	48 Mbits

$x = 48 \times 2^{20}/6 = 8 \times 2^{20}$ θ.μ.

ΟΚ1

1 θ.μ.	3 bits
x	1.5 MB

$x = 1.5 \times 2^{20} \times 2^3/3 = 4 \times 2^{20}$ θ.μ.

Κύρια μνήμη

1 θ.μ.	9 bits
x	252 Mbits

$x = 252 \times 2^{20}/9 = 28 \times 2^{20}$ θ.μ.

Οριζόντια επανάληψη: **2** (6 bits + 3 bits), πάντα **δημιουργεί το επιθυμητό εύρος μνήμης**

Κατακόρυφη επανάληψη: $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα ΟΚΜ}} = \frac{28 \times 2^{20} \text{ θ.μ.}}{8 \times 2^{20} \text{ θ.μ.}} = \frac{28}{8} = 3.5$. **Δεν επιστρέφεται ακέραιο πηλίκο**

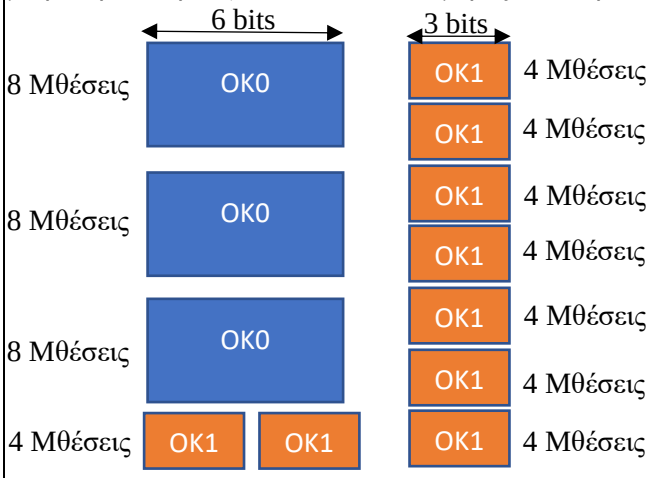
Κατακόρυφη επανάληψη: $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα ΟΚΜ}} = \frac{28 \times 2^{20} \text{ θ.μ.}}{4 \times 2^{20} \text{ θ.μ.}} = 7$. **Επιστρέφεται ακέραιο πηλίκο**

Κάθε **ΟΚ0** αφού περιέχει 8×2^{20} θ.μ., έχει $8 \times 2^{20} = 8$ Μθέσεις των 6 bits η καθεμία.

Κάθε **ΟΚ1** αφού περιέχει 4×2^{20} θ.μ., έχει 4 Μθέσεις των 3 bits η καθεμία.

Η **κύρια μνήμη** που θα σχεδιάσουμε έχει 28×2^{20} θ.μ. = 28 Μθέσεις των 9 bits η καθεμία

Για να σχεδιάσουμε το επιθυμητό μέγεθος της κύριας μνήμης, θα πρέπει να επαναλάβουμε κατακόρυφα 7 ΟΚ1 (έχουμε ακέραιο πηλίκο) και όσον αφορά τα ΟΚ0, θα πρέπει να επαναλάβουμε κατακόρυφα 3 ΟΚ0 (στρογγυλοποίηση προς το μικρότερο ακέραιο) και το δεκαδικό μέρος που περισσεύει, θα το δημιουργήσουμε από τα ΟΚ1, όπως φαίνεται παρακάτω:



Πρέπει το συνολικό μέγεθος κάθε στήλης να είναι ίδιο

Άρα, χρειαζόμαστε 3 ΟΚ0 + 9 ΟΚ1 → 3+9

Επαλήθευση: 3 * 48 Mbits + 9 * 1.5 MB = 144 Mbits + 9 * 12 Mbits = 252 Mbits

1 byte	8 bits
1.5 MB	x;

$x = 1.5 \times 2^{20} \times 2^3 = 12 \times 2^{20}$bits = 12 Mbits

Θέμα Σεπτεμβρίου 2020

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης ΟΚ0 με χωρητικότητα 256 Kbits και οργάνωση οκτώ δυαδικών ψηφίων ανά θέση μνήμης καθώς και ολοκληρωμένα κυκλώματα ΟΚ2 με χωρητικότητα 32 Kbytes και οργάνωση δύο δυαδικών ψηφίων ανά θέση. Πόσα ολοκληρωμένα κυκλώματα τύπου ΟΚ0 και πόσα τύπου ΟΚ2 χρειάζεστε για να σχεδιάσετε ένα σύστημα μνήμης με χωρητικότητα 160 Kbytes και 10 bits ανά θέση: Να δώσετε το αποτέλεσμα ως δύο διψήφιους αριθμούς που χωρίζονται με το σύμβολο + πχ. 01+12 το οποίο σημαίνει 4 τύπου ΟΚ0 και 12 τύπου ΟΚ2. .

Λύση

ΟΚ0

1 θ.μ. 8 bits

x; 256 Kbits

$x = 2^8 \times 2^{10}/2^3 = 2^{15}$ θ.μ.

ΟΚ2

1 θ.μ. 2 bits

x 32 KB

$x = 2^5 \times 2^{10} \times \mathbf{2^3}/2^1 = 2^{18}/2^1 = 2^{17}$ θ.μ.

Κύρια μνήμη

1 θ.μ. 10 bits

x 160 KB

$x = 160 \times 2^{10} \times \mathbf{2^3} /10 = 2^{17}$ θ.μ.

Οριζόντια επανάληψη: **2** (8 bits + 2 bits)

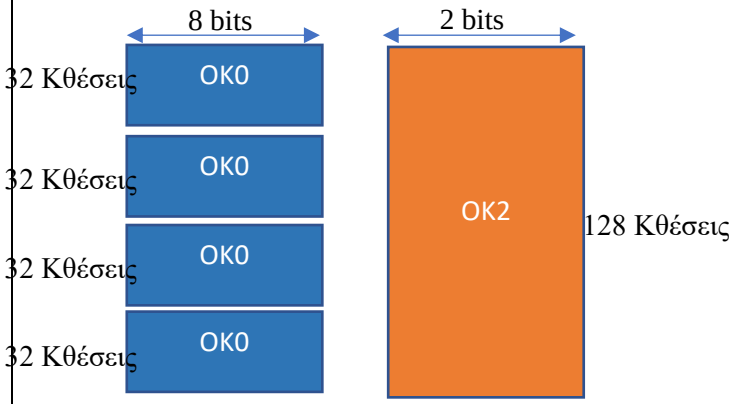
Κατακόρυφη επανάληψη για ΟΚ2: $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα ΟΚΜ}} = \frac{2^{17}\ \text{θ.μ.}}{2^{17}\ \text{θ.μ.}} = 1$ επανάληψη

Κατακόρυφη επανάληψη για ΟΚ0: $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα ΟΚΜ}} = \frac{2^{17}\ \text{θ.μ.}}{2^{15}\ \text{θ.μ.}} = 4$ επαναλήψεις

Κάθε **ΟΚ0** αφού περιέχει 2^{15} θ.μ., έχει $2^5 \times 2^{10} = 32$ Κθέσεις των 8 bits η καθεμία.

Κάθε **ΟΚ1** αφού περιέχει 2^{17} θ.μ., έχει $2^7 \times 2^{10} = 128$ Κθέσεις των 2 bits η καθεμία.

Η **κύρια μνήμη** που θα σχεδιάσουμε έχει 2^{17} θ.μ., έχει $2^7 \times 2^{10} = 128$ Κθέσεις των 10 bits η καθεμία



Πρέπει το συνολικό μέγεθος κάθε στήλης να είναι ίδιο

Επομένως χρειαζόμαστε 4 ΟΚ0 + 1 ΟΚ2 → 04+01.

Επαλήθευση: Έχουμε συνολική χωρητικότητα 4 * 256 Kbits + 1 * 32 Kbytes = 4 * 32 Kbytes + 1 * 32Kbytes = 160 Kbytes

1 byte 8 bits

x; 256 Kbits

$x = 2^8\ 2^{10}/2^3 = 2^{15}$ bytes $= 2^5 \times 2^{10}$ bytes = 32 KBytes

Θέμα Φεβρουαρίου 2021

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης ΟΚ0 με χωρητικότητα 256 Kbits και οργάνωση οκτώ δυαδικών ψηφίων ανά θέση μνήμης καθώς και ολοκληρωμένα κυκλώματα ΟΚ2 με χωρητικότητα 16 Kbytes και οργάνωση δύο δυαδικών ψηφίων ανά θέση. Πόσα ολοκληρωμένα κυκλώματα τύπου ΟΚ0 και πόσα τύπου ΟΚ2 χρειάζεστε για να σχεδιάσετε ένα σύστημα μνήμης με χωρητικότητα 160 Kbytes και 10 bits ανά θέση: Να δώσετε το αποτέλεσμα ως δύο διψήφιους αριθμούς που χωρίζονται με το σύμβολο + πχ. 01+11 το οποίο σημαίνει 4 τύπου ΟΚ0 και 12 τύπου ΟΚ2. .

Λύση

ΟΚ0

1 θ.μ.	8 bits
x;	256 Kbits

$x = 2^8 \times 2^{10}/2^3 = 2^{15}$ θ.μ.

ΟΚ2

1 θ.μ.	2 bits
x	16 KB

$x = 2^4 \times 2^{10} \times \mathbf{2^3}/2^1 = 2^{17}/2^1 = 2^{16}$ θ.μ.

Κύρια μνήμη

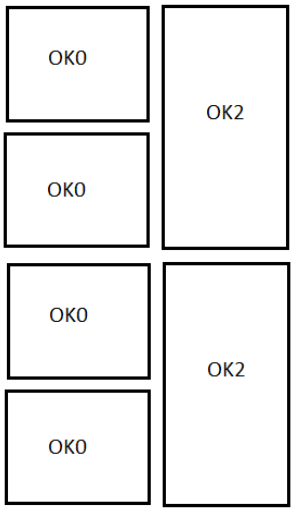
1 θ.μ.	10 bits
x	160 KB

$x = 160 \times 2^{10} \times \mathbf{2^3} /10 = 2^{17}$ θ.μ.

Οριζόντια επανάληψη: **2** (8 bits + 2 bits)

Κατακόρυφη επανάληψη για ΟΚ2: $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα ΟΚΜ}} = \frac{2^{17}\ \text{θ.μ.}}{2^{16}\ \text{θ.μ.}} = 2$ επαναλήψεις

Κατακόρυφη επανάληψη για ΟΚ0: $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα ΟΚΜ}} = \frac{2^{17}\ \text{θ.μ.}}{2^{15}\ \text{θ.μ.}} = 4$ επαναλήψεις



Επομένως χρειαζόμαστε 4 ΟΚ0 + 2 ΟΚ2 → 04+02.

Επαλήθευση: Έχουμε συνολική χωρητικότητα 4 * 32 Kbytes + 2 * 16 Kbytes = 160 Kbytes

1 byte	8 bits
x;	256 Kbits

$x = 2^8\ 2^{10}/2^3 = 2^{15}$ bytes $= 2^5 \times 2^{10}$ bytes = 32 KBytes

Θέμα Ιουνίου 2021

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης ΟΚ0 με χωρητικότητα 512 Kbits και οργάνωση οκτώ δυαδικών ψηφίων ανά θέση μνήμης καθώς και ολοκληρωμένα κυκλώματα ΟΚ2 με χωρητικότητα 16 Kbytes και οργάνωση δύο δυαδικών ψηφίων ανά θέση. Πόσα ολοκληρωμένα κυκλώματα τύπου ΟΚ0 και πόσα τύπου ΟΚ2 χρειάζεστε για να σχεδιάσετε ένα σύστημα μνήμης με χωρητικότητα 224 Kbytes και 14 bits ανά θέση: Να δώσετε το αποτέλεσμα ως δύο διψήφιους αριθμούς που χωρίζονται με το σύμβολο + πχ. 01+12 το οποίο σημαίνει 4 τύπου ΟΚ0 και 12 τύπου ΟΚ2. .

Λύση

ΟΚ0

1 θ.μ.	8 bits
x;	512 Kbits

$x = 2^9 \times 2^{10}/2^3 = 2^{16}$ θ.μ.

ΟΚ2

1 θ.μ.	2 bits
x	16 KB

$x = 2^4 \times 2^{10} \times \mathbf{2^3}/2^1 = 2^{17}/2^1 = 2^{16}$ θ.μ.

Κύρια μνήμη

1 θ.μ.	14 bits
x	224 KB

$x = 224 \times 2^{10} \times \mathbf{2^3}/14 = 16 \times 2^{13}$ θ.μ. $= 2^{17}$ θ.μ.

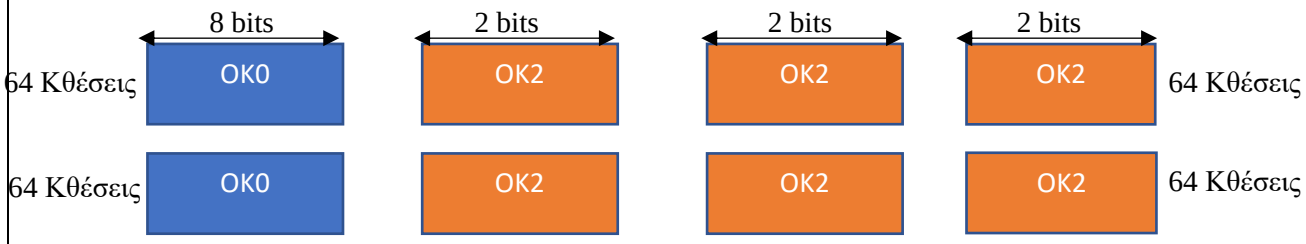
Οριζόντια επανάληψη: **4** (8 bits + 2 bits + 2 bits + 2 bits)

Κατακόρυφη επανάληψη: $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα ΟΚΜ}} = \frac{2^{17} \text{ θ.μ.}}{2^{16} \text{ θ.μ.}} = 2$ επαναλήψεις

Κάθε ΟΚ0 αφού περιέχει 2^{16} θ.μ., έχει $2^{16} = 2^6 2^{10} = 64$ Κθέσεις του ενός byte η καθεμία.

Κάθε ΟΚ1 αφού περιέχει 2^{16} θ.μ., έχει $2^{16} = 2^6 2^{10} = 64$ Κθέσεις των 2 bits η καθεμία.

Η **κύρια μνήμη** που θα σχεδιάσουμε έχει 2^{17} θ.μ. $= 2^7 2^{10} = 128$ Κθέσεις των 14 bits η καθεμία



Επομένως χρειαζόμαστε 2 ΟΚ0 και 6 ΟΚ2 → 02+06

Επαλήθευση: 2 * 512 Kbits + 6 * 16 KB = 2 * 64 KB + 96 KB = 128 KB + 96 KB = 224 KB

1 byte	8 bits
x;	512 Kbits

$x = 2^9\ 2^{10}/2^3 = 2^{16}$ bytes $= 2^6 \times 2^{10}$ bytes = 64 KBytes

Θέμα Ιουνίου 2020

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης ΟΚ1 με χωρητικότητα 32 Kbytes και οργάνωση τεσσάρων δυαδικών ψηφίων ανά θέση μνήμης καθώς και ολοκληρωμένα κυκλώματα ΟΚ2 με χωρητικότητα 16Kbytes και οργάνωση δύο δυαδικών ψηφίων ανά θέση. Πόσα ολοκληρωμένα κυκλώματα τύπου ΟΚ1 και πόσα τύπου ΟΚ2 χρειάζεστε για να σχεδιάσετε ένα σύστημα μνήμης με χωρητικότητα 160Kbytes και 10 bits ανά θέση; Να δώσετε το αποτέλεσμα ως δύο διψήφιους αριθμούς που χωρίζονται με το σύμβολο + πχ. 01+12 το οποίο σημαίνει 4 τύπου ΟΚ1 και 12 τύπου ΟΚ2. .

Λύση

ΟΚ1

1 θ.μ.	4 bits
x;	32KB

$x = (2^5 * 2^{10} * 2^3)/\ 2^2 = 2^{18}/2^2 = 2^{16}$ θ.μ.

ΟΚ2

1 θ.μ.	2 bits
x	16KB

$x = 2^4 \times 2^{10} \times 2^3/2^1 = 2^{17}/2^1 = 2^{16}$ θ.μ.

Κύρια μνήμη

1 θ.μ.	10 bits
x	160KB

$x = 160 \times 2^{10} \times 2^3\ /10 = 16 \times 2^{13}$ θ.μ. $= 2^4 \times 2^{13}$ θ.μ. $= 2^{17}$ θ.μ.

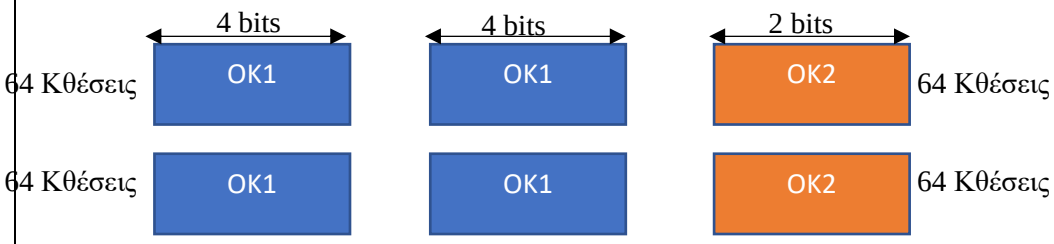
Οριζόντια επανάληψη: **3** (4 bits + 4 bits + 2 bits)

Κατακόρυφη επανάληψη: $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα ΟΚΜ}} = \frac{2^{17}\ \text{θ.μ.}}{2^{16}\ \text{θ.μ.}} = 2$ επαναλήψεις

Κάθε ΟΚ0 αφού περιέχει 2^{16} θ.μ., έχει $2^{16} = 2^6\, 2^{10} = 64$ Κθέσεις των 4 bits η καθεμία.

Κάθε ΟΚ1 αφού περιέχει 2^{20} θ.μ., έχει $2^{16} = 2^6\, 2^{10} = 64$ Κθέσεις των 2 bits η καθεμία

Η **κύρια μνήμη** που θα σχεδιάσουμε έχει 2^{17} θ.μ. = $2^7\, 2^{10} = 128$ Κθέσεις των 10 bits η καθεμία



Επομένως χρειαζόμαστε 4 ΟΚ1 και 2 ΟΚ2 → 04+02

Επαλήθευση: 4 * 32 KB + 2 * 16 KB = 128 KB + 32 KB = 160 KB

Θέμα Σεπτεμβρίου 2021

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης ΟΚ0 με χωρητικότητα 16 Mbits και οργάνωση οκτώ δυαδικών ψηφίων ανά θέση μνήμης καθώς και ολοκληρωμένα κυκλώματα ΟΚ1 με χωρητικότητα 0.5 MB και οργάνωση τεσσάρων δυαδικών ψηφίων ανά θέση. Πόσα ολοκληρωμένα κυκλώματα τύπου ΟΚ0 και πόσα τύπου ΟΚ1 χρειάζεστε για να σχεδιάσετε ένα σύστημα μνήμης με χωρητικότητα 60 Mbits και 12 bits ανά θέση; Να δώσετε το αποτέλεσμα ως δύο διψήφιους αριθμούς που χωρίζονται με το σύμβολο + πχ. 01+12 το οποίο σημαίνει 4 τύπου ΟΚ0 και 12 τύπου ΟΚ1.

Λύση

ΟΚ0

1 θ.μ. 8 bits
x; 16 Mbits

x = $(2^4 * 2^{20})/ 2^3 = 2^{21}$ =**2 x 2^{20} θ.μ. του ενός byte η καθεμία** = 2Μ θέσεις των 8 bits η καθεμία

ΟΚ1

1 θ.μ. 4 bits
x 0.5 MB

x = 2^{-1} x 2^{20} x $2^3/2^2$ = **2^{20} θ.μ. των 4 bits η καθεμία** = 1Μθέση των 4 bits η καθεμία

Κύρια μνήμη

1 θ.μ. 12 bits
x 60 Mbits

x = 60 x 2^{20} /12 = **5 x 2^{20} θ.μ. των 4 bits η καθεμία** = 5Μθέσεις των 4 bits η καθεμία

Οριζόντια επανάληψη: **2** (8 bits + 4 bits)

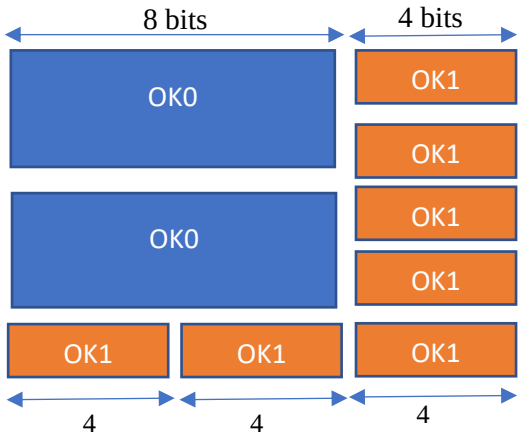
Κατακόρυφη επανάληψη: $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα ΟΚΜ}} = \frac{5 \text{ x } 2^{20}\ \text{θ.μ.}}{2^{21}\ \text{θ.μ.}} = 5 \text{ x } 2^{-1} = 2.5$. **Δεν επιστρέφεται ακέραιο πηλίκο**

Κατακόρυφη επανάληψη: $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα ΟΚΜ}} = \frac{5 \text{ x } 2^{20}\ \text{θ.μ}}{2^{20}\ \text{θ.μ.}} = 5$. **Επιστρέφεται ακέραιο πηλίκο**

Κάθε ΟΚ0 αφού περιέχει 2^{21} θ.μ., έχει $2^{21} = 2^1\, 2^{20} = 2$ Μθέσεις του ενός byte η καθεμία.

Κάθε ΟΚ1 αφού περιέχει 2^{20} θ.μ., έχει 1Μθέση των 4 bits η καθεμία.

Για να σχεδιάσουμε το επιθυμητό μέγεθος της κύριας μνήμης, θα πρέπει να επαναλάβουμε κατακόρυφα 5 ΟΚ1 (έχουμε ακέραιο πηλίκο) και όσον αφορά τα ΟΚ0, θα πρέπει να επαναλάβουμε κατακόρυφα 2 ΟΚ0 (στρογγυλοποίηση προς το μικρότερο ακέραιο) και το δεκαδικό μέρος που περισσεύει, θα το δημιουργήσουμε από τα ΟΚ1, όπως φαίνεται παρακάτω:



Επαλήθευση: Υπολογίσαμε 2 ΟΚ0 και 7 ΟΚ1 = 2 x 16 Mbits + 7 x 0.5 MB = 32 Mbits + 7 x 2^{-1} x 2^{20} x 2^3 bits = 32 Mbits + 7 x 2^{-1} x 2^3 Mbits = 32 Mbits + 28 Mbits = 60 Mbits.

Θέμα προόδου Ιανουαρίου 2022

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης ΟΚ0 με χωρητικότητα 3Mbyte και οργάνωση 12 δυαδικών ψηφίων ανά θέση μνήμης καθώς και ολοκληρωμένα κυκλώματα ΟΚ1 με χωρητικότητα 12 Mbits και οργάνωση 12 δυαδικών ψηφίων ανά θέση. Πόσα ολοκληρωμένα κυκλώματα τύπου ΟΚ0 και πόσα τύπου ΟΚ1 χρειάζεστε για να σχεδιάσετε ένα σύστημα μνήμης με χωρητικότητα 15 Mbytes και 24 bits ανά θέση ώστε το συνολικό πλήθος των ολοκληρωμένων που χρησιμοποιήσατε να είναι το μικρότερο; Να δώσετε το αποτέλεσμα ως δύο διψήφιους αριθμούς που χωρίζονται με το σύμβολο + πχ. 01+12 το οποίο σημαίνει 4 τύπου ΟΚ0 και 12 τύπου ΟΚ1.

Λύση

ΟΚ0

1 θ.μ. 12 bits
x; 3 Mbyte

x = (3 * 2^3 * 2^{20})/ 12 = 2^{21} = **2 x 2^{20} θ.μ.** = 2Μθέσεις των 8 bits η καθεμία

ΟΚ1

1 θ.μ. 12 bits
x 12 Mbits

x = 12 x 2^{20}/12 = **2^{20}** = 1Μθέση των 12 bits η καθεμία

Κύρια μνήμη

1 θ.μ. 24 bits
x 15 Mbytes

x = 15 x 2^3 x 2^{20} /24 = **5 x 2^{20}** = 5Μθέσεις των 24 bits η καθεμία

Οριζόντια επανάληψη: **2** (12 bits + 12 bits)

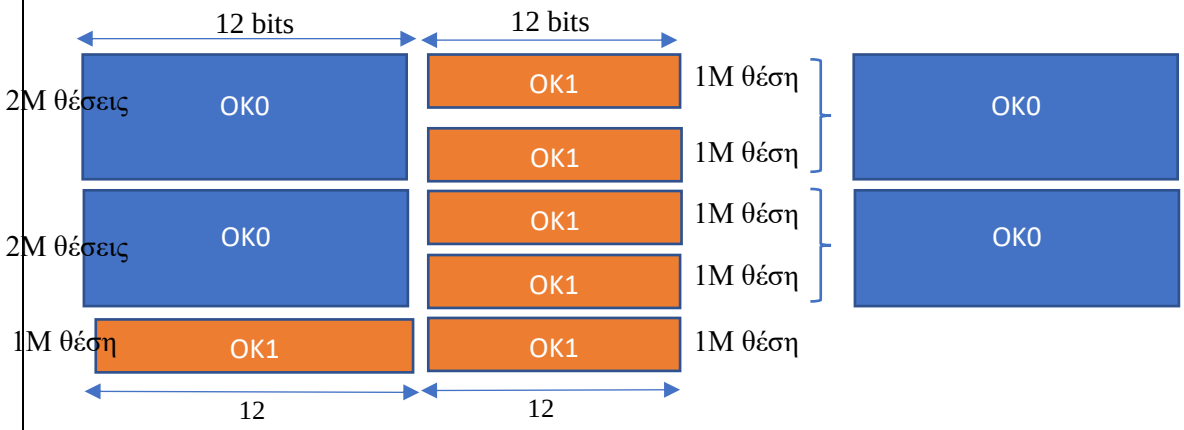
Κατακόρυφη επανάληψη: $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα ΟΚΜ}} = \frac{5 \text{ x } 2^{20} \text{ θ.μ.}}{2 \text{ x } 2^{20} \text{ θ.μ.}}$ = 2.5. **Δεν επιστρέφεται ακέραιο πηλίκο**

Κατακόρυφη επανάληψη: $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα ΟΚΜ}} = \frac{5 \text{ x } 2^{20} \text{ θ.μ}}{2^{20} \text{ θ.μ.}}$ = 5. **Επιστρέφεται ακέραιο πηλίκο**

Κάθε ΟΚ0 αφού περιέχει 2^{21} θ.μ., έχει $2^{21} = 2^1 2^{20} = 2$ Μθέσεις των 12 bits η καθεμία.

Κάθε ΟΚ1 αφού περιέχει 2^{20} θ.μ., έχει 1Μθέση των 12 bits η καθεμία.

Για να σχεδιάσουμε το επιθυμητό μέγεθος της κύριας μνήμης, θα πρέπει να επαναλάβουμε κατακόρυφα 5 ΟΚ1 (έχουμε ακέραιο πηλίκο) και όσον αφορά τα ΟΚ0, θα πρέπει να επαναλάβουμε κατακόρυφα 2 ΟΚ0 (στρογγυλοποίηση προς το μικρότερο ακέραιο) και το δεκαδικό μέρος που περισσεύει, θα το δημιουργήσουμε από τα ΟΚ1, όπως φαίνεται παρακάτω:



Επαλήθευση: Υπολογίσαμε 2 ΟΚ0 και 6 ΟΚ1 = 2 x 3 Mbyte + 6 x 12 Mbits = 6 Mbyte + 9 Mbyte = 15 Mbyte.

Σημείωση: Επειδή η άσκηση ζητά το **μικρότερο** δυνατό πλήθος, η σωστή απάντηση είναι **4+2**, αφού αντικαθιστούμε στη δεξιά στήλη ανά **δύο** τα ΟΚΜ των 1Μθέσεων με ένα ΟΚΜ των 2Μθέσεων.

Θέμα προόδου Ιανουαρίου 2022

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης ΟΚ0 με χωρητικότητα 80 Mbits και οργάνωση 10 δυαδικών ψηφίων ανά θέση μνήμης καθώς και ολοκληρωμένα κυκλώματα ΟΚ1 με χωρητικότητα 2.5 Mbyte με χωρητικότητα και οργάνωση 10 δυαδικών ψηφίων ανά θέση. Πόσα ολοκληρωμένα κυκλώματα τύπου ΟΚ0 και πόσα τύπου ΟΚ1 χρειάζεστε για να σχεδιάσετε ένα σύστημα μνήμης με χωρητικότητα 400 Mbits και 20 δυαδικά ψηφία ανά θέση ώστε το συνολικό πλήθος των ολοκληρωμένων που χρησιμοποιήσατε να είναι το μικρότερο; Να δώσετε το αποτέλεσμα ως δύο διψήφιους αριθμούς που χωρίζονται με το σύμβολο + πχ. 01+12 το οποίο σημαίνει 4 τύπου ΟΚ0 και 12 τύπου ΟΚ1.

Λύση

ΟΚ0

1 θ.μ. 10 bits
x; 80 Mbits

x = (80 * 2^{20})/ 10 = 8 * 2^{20} = 2^3 x 2^{20} θ.μ. = 8 Μθέσεις των 10 bits η καθεμία

ΟΚ1

1 θ.μ. 10 bits
x 2.5 Mbyte

x = (2.5 x 2^{20} x 2^3)/10 = 2 * 2^{20} = 2 Μθέσεις των 10 bits η καθεμία

Κύρια μνήμη

1 θ.μ. 20 bits
x 400 Mbits

x = (400 x 2^{20}) /20 = 20 x 2^{20} = 20 Μθέσεις των 20 bits η καθεμία

Οριζόντια επανάληψη: **2** (10 bits + 10 bits)

Κατακόρυφη επανάληψη: $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα ΟΚΜ}} = \frac{20 \text{ x } 2^{20} \text{ θ.μ.}}{8 \text{ x } 2^{20} \text{ θ.μ.}}$ = 2.5. **Δεν επιστρέφεται ακέραιο πηλίκο**

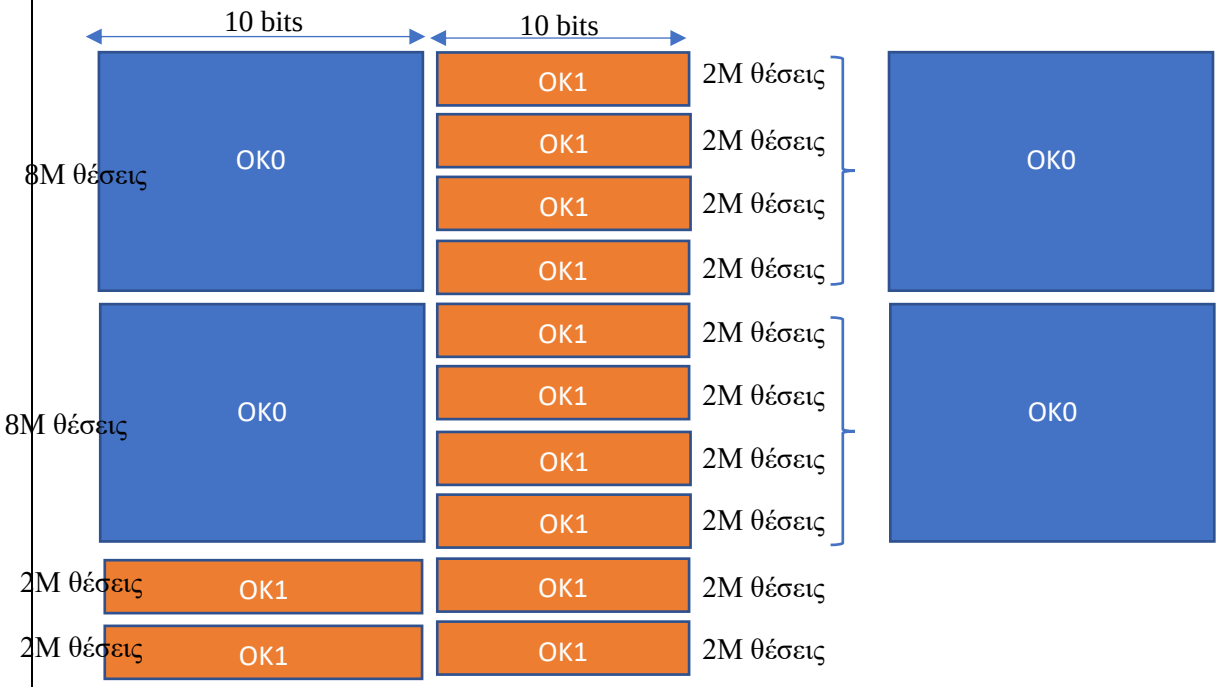
Κατακόρυφη επανάληψη: $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα ΟΚΜ}} = \frac{20 \text{ x } 2^{20} \text{ θ.μ}}{2 \text{ x } 2^{20} \text{ θ.μ.}}$ = 10. **Επιστρέφεται ακέραιο πηλίκο**

Κάθε ΟΚ0 αφού περιέχει 2^{23} θ.μ., έχει $2^{23} = 2^3 2^{20} = 8$ Μθέσεις των 10 bits η καθεμία.

Κάθε ΟΚ1 αφού περιέχει 2^{21} θ.μ., έχει 2 Μθέσεις των 10 bits η καθεμία.

Η κύρια μνήμη που θα σχεδιάσουμε περιέχει 20 Μθέσεις των 20 bits η καθεμία

Για να σχεδιάσουμε το επιθυμητό μέγεθος της κύριας μνήμης, θα πρέπει να επαναλάβουμε κατακόρυφα 10 ΟΚ1 (έχουμε ακέραιο πηλίκο) και όσον αφορά τα ΟΚ0, θα πρέπει να επαναλάβουμε κατακόρυφα 2 ΟΚ0 (στρογγυλοποίηση προς το μικρότερο ακέραιο) και το δεκαδικό μέρος που περισσεύει, θα το δημιουργήσουμε από τα ΟΚ1, όπως φαίνεται παρακάτω:



Επαλήθευση: Υπολογίσαμε 2 ΟΚ0 και 12 ΟΚ1 = 2 x 80 Mbits + 12 x 2.5 Mbyte = 160 Mbits + 30 Mbyte = 160 Mbits + 240 Mbits = 400 Mbits.

Σημείωση: Επειδή η άσκηση ζητά το **μικρότερο** δυνατό πλήθος, η σωστή απάντηση είναι **4+4**, αφού αντικαθιστούμε στη δεξιά στήλη ανά **τέσσερα** τα ΟΚΜ των 2Μθέσεων με ένα ΟΚΜ των 8Μθέσεων.

Θέμα προόδου Ιανουαρίου 2022

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης ΟΚ0 με χωρητικότητα 32 Mbits και οργάνωση 8 δυαδικών ψηφίων ανά θέση μνήμης καθώς και ολοκληρωμένα κυκλώματα ΟΚ1 με χωρητικότητα 1 Mbyte με χωρητικότητα και οργάνωση 8 δυαδικών ψηφίων ανά θέση. Πόσα ολοκληρωμένα κυκλώματα τύπου ΟΚ0 και πόσα τύπου ΟΚ1 χρειάζεστε για να σχεδιάσετε ένα σύστημα μνήμης με χωρητικότητα 176 Mbits και 16 δυαδικά ψηφία ανά θέση ώστε το συνολικό πλήθος των ολοκληρωμένων που χρησιμοποιήσατε να είναι το μικρότερο; Να δώσετε το αποτέλεσμα ως δύο διψήφιους αριθμούς που χωρίζονται με το σύμβολο + πχ. 01+12 το οποίο σημαίνει 4 τύπου ΟΚ0 και 12 τύπου ΟΚ1. .

Λύση

ΟΚ0

1 θ.μ.	8 bits
x;	32 Mbits

$x = (32 * 2^{20})/ 8 = 4 * 2^{20} = 2^2 \text{ x } 2^{20}$ θ.μ. = 4 Μθέσεις των 8 bits η καθεμία

ΟΚ1

1 θ.μ.	8 bits
x	1 Mbyte

$x = (1 \text{ x } 2^{20} \text{ x } 2^3)/8 = 2^{20} = 1$ Μθέσεις των 8 bits η καθεμία

Κύρια μνήμη

1 θ.μ.	16 bits
x	176 Mbits

x = (176 x 2^{20}) /16 = 11 x 2^{20} = 11 Μθέσεις των 16 bits η καθεμία

Οριζόντια επανάληψη: **2** (8 bits + 8 bits)

Κατακόρυφη επανάληψη: $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα ΟΚΜ}} = \frac{11 \text{ x } 2^{20} \text{ θ.μ.}}{4 \text{ x } 2^{20} \text{ θ.μ.}} = 2.75$. **Δεν επιστρέφεται ακέραιο πηλίκο**

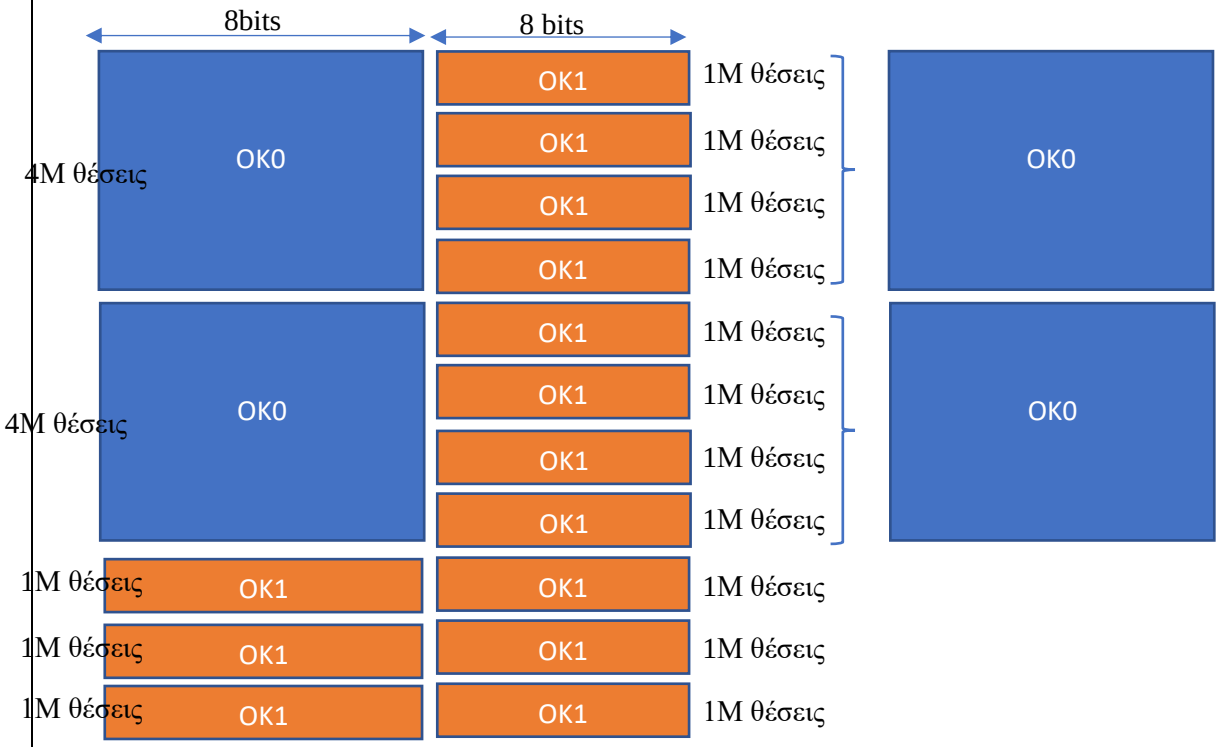
Κατακόρυφη επανάληψη: $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα ΟΚΜ}} = \frac{11 \text{ x } 2^{20} \text{ θ.μ}}{2^{20} \text{ θ.μ.}} = 11$. **Επιστρέφεται ακέραιο πηλίκο**

Κάθε ΟΚ0 αφού περιέχει 2^{22} θ.μ., έχει $2^{22} = 2^2\, 2^{20} = 4$ Μθέσεις των 8 bits η καθεμία.

Κάθε ΟΚ1 αφού περιέχει 2^{20} θ.μ., έχει 1 Μθέσεις των 8 bits η καθεμία.

Η κύρια μνήμη που θα σχεδιάσουμε περιέχει 11 Μθέσεις των 16 bits η καθεμία

Για να σχεδιάσουμε το επιθυμητό μέγεθος της κύριας μνήμης, θα πρέπει να επαναλάβουμε κατακόρυφα 11 ΟΚ1 (έχουμε ακέραιο πηλίκο) και όσον αφορά τα ΟΚ0, θα πρέπει να επαναλάβουμε κατακόρυφα 2 ΟΚ0 (στρογγυλοποίηση προς το μικρότερο ακέραιο) και το δεκαδικό μέρος που περισσεύει, θα το δημιουργήσουμε από τα ΟΚ1, όπως φαίνεται παρακάτω:



Επαλήθευση: Υπολογίσαμε 2 ΟΚ0 και 14 ΟΚ1 = 2 x 32 Mbits + 14 x 1 Mbyte = 64 Mbits + 14 Mbyte = 64 Mbits + 112 Mbits = 176 Mbits.

Σημείωση: Επειδή η άσκηση ζητά το **μικρότερο** δυνατό πλήθος, η σωστή απάντηση είναι **4+6**, αφού αντικαθιστούμε στη δεξιά στήλη ανά **τέσσερα** τα ΟΚΜ των 1Μθέσεων με ένα ΟΚΜ των 4Μθέσεων.

Παράδειγμα 5.8 με χαμηλής τάξης διαφύλλωση

Θεωρείστε ιεραρχική μνήμη τριών επιπέδων αποτελούμενη από ένα επίπεδο κρυφής μνήμης, κύρια μνήμη και βοηθητική μνήμη. Υποθέστε ότι η κύρια μνήμη έχει χωρητικότητα 1G ψηφιολέξεις, οργάνωση μια ψηφιολέξη ανά θέση μνήμης και κάθε πλαίσιο της κρυφής μνήμης είναι των τεσσάρων ψηφιολέξεων. Υποθέστε επίσης ότι η κύρια μνήμη έχει οργάνωση N-

δρόμων χαμηλής τάξης διαφύλλωση μνήμης στην οποία η πρώτη προσπέλαση διαρκεί 6 χρονικές περιόδους ενώ οι επόμενες Ν – 1 διαρκούν μια χρονική περίοδο η κάθε μία.

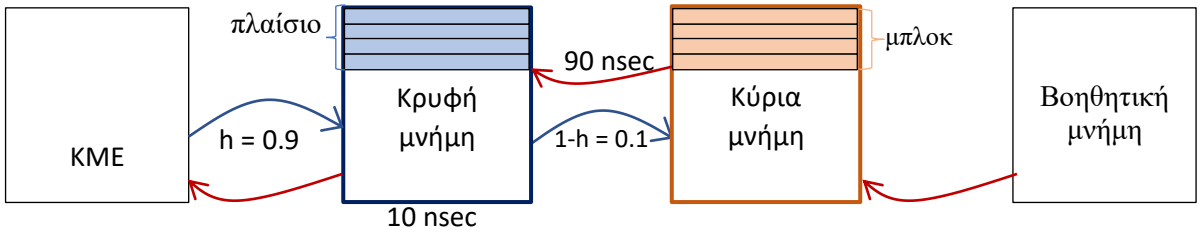
α. Ποια είναι η μικρότερη τιμή του Ν που μπορεί να μας εξασφαλίσει τη γρηγορότερη μεταφορά ενός μπλοκ πληροφορίας από την κύρια μνήμη στην κρυφή.

β. Πόσες χρονικές περίοδοι απαιτούνται για τη μεταφορά ενός μπλοκ πληροφορίας από την κύρια μνήμη στην κρυφή;

Γ. Αν ο χρόνος προσπέλασης της κρυφής μνήμης είναι μια χρονική περίοδος, μια χρονική περίοδος ισούται με 10 nsec και ο λόγος επιτυχίας της κρυφής μνήμης είναι 0.9 να υπολογίσετε το μέσο χρόνο προσπέλασης του συστήματος μνήμης, θεωρώντας ότι η κύρια μνήμη περιέχει πάντα τη ζητούμενη πληροφορία. Θεωρείστε ότι η ΚΜΕ μπορεί να διαβάσει πληροφορία μόνο από την κύρια μνήμη.

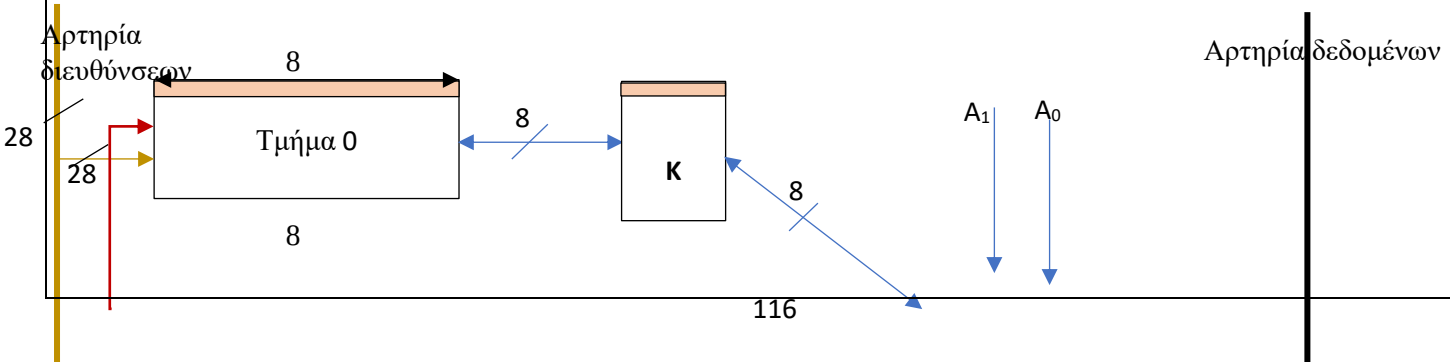
δ. Σχεδιάστε την κύρια μνήμη με οργάνωση Ν – δρόμων χαμηλής τάξης διαφύλλωση για την τιμή του Ν του α' ερωτήματος

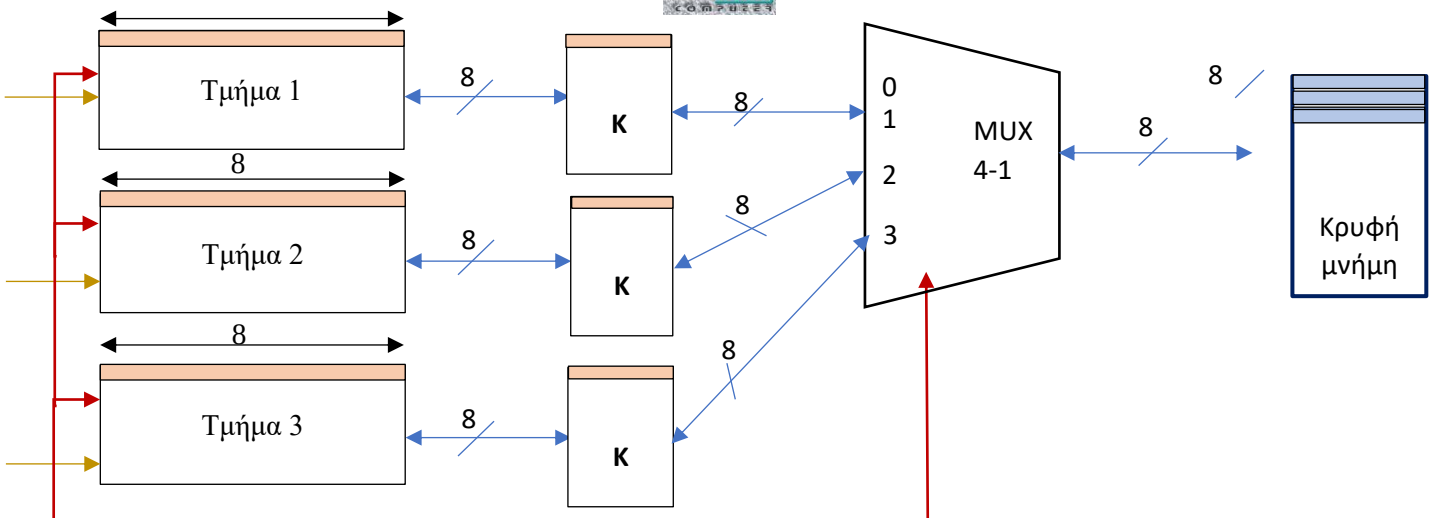
Λύση



α. Ισχύει ότι ένα μπλοκ της κύριας μνήμης → πλαίσιο της κρυφής μνήμης. **Ένα μπλοκ έχει πάντα ίδιο μέγεθος με το πλαίσιο** και είναι -όπως και το πλαίσιο- μια συλλογή γειτονικών (συνεχόμενων) θ.μ. Θέλουμε να μεταφέρουμε ένα μπλοκ (συλλογή 4 θ.μ. στην προκειμένη περίπτωση) της κύριας μνήμης σε ένα πλαίσιο (συλλογή 4 θ.μ.) της κρυφής μνήμης. Αυτό σημαίνει ότι πρέπει να μεταφερθούν ταυτόχρονα (με μια προσπέλαση) 4 θ.μ. από την κύρια στην κρυφή. Ζητάμε τη **μικρότερη τιμή του Ν** (όπου Ν = πλήθος δρόμων ή τρόπων της κύριας μνήμης που έχει οργάνωση χαμηλής τάξης διαφύλλωση) για να πετύχουμε τη **γρηγορότερη μεταφορά**. Προφανώς το Ν = 4, αφού με 4 – δρόμους μπορούμε να μεταφέρουμε το περιεχόμενο 4 θ.μ. με μια προσπέλαση από την κύρια στην κρυφή μνήμη. Αν το Ν = 2, θα χρειαζόντουσαν δύο προσπελάσεις μνήμης και θα υπήρχε μεγαλύτερη καθυστέρηση στην περίπτωση αυτή. Στη συνέχεια προχωράμε με τη σχεδίαση της Ν – δρόμων χαμηλής τάξης διαφύλλωση μνήμης, με Ν = 4. Το χαρακτηριστικό σε αυτή τη σχεδίαση είναι ότι δεν υπάρχει αποκωδικοποιητής για να επιλέγει τα τμήματα μνήμης ή τα ΟΚΜ και υπάρχει πολυπλέκτης 2^{μ} –1, ο οποίος επιλέγει κάθε φορά -με βάση τις γραμμές επιλογής του- μια από τις εισόδους του (δηλ. το περιεχόμενο ενός καταχωρητή Κ) και την οδηγεί στην έξοδο. Ως γραμμές επιλογής του MUX χρησιμοποιούνται τα λιγότερο σημαντικά bits της αρτηρίας διευθύνσεων.

δ. Η χωρητικότητα της Κ.Μ. = 1 GB = 2^{30} bytes = 2^{30} θ.μ. (λόγω της οργάνωσης 1 ψηφιολέξη ανά θ.μ.) Επειδή από την εκφώνηση δεν προσδιορίζεται ξεχωριστά το μέγεθος του address bus, θα θεωρήσουμε ότι αυτό είναι ίσο με 30 bits. Από αυτά τα δύο λιγότερο σημαντικά δυαδικά ψηφία (A_1 – A_0) χρησιμοποιούνται ως γραμμές επιλογής του MUX. Επομένως, περισσεύουν 28 δυαδικά ψηφία ποιυ χρησιμοποιούνται για τη διευθυνσιοδότηση των τμημάτων. Εκτός από την αρτηρία δεδομένων και διευθύνσεων και τον MUX, σχεδιάζεται επιπλέον η γραμμή **Ανάγνωση/Εγγραφή**.





β. Οι χρονικές περίοδοι που απαιτούνται για **τη μεταφορά ενός μπλοκ πληροφορίας από την κύρια μνήμη στην κρυφή** είναι 6 χρ. περίοδοι + (N – 1) x 1 χρ. περίοδοι = 6 χρ. περίοδοι + (4 – 1) x 1 χρ. περίοδοι = 9 χρ. περίοδοι = 9 x 10 nsec = 90 nsec.

γ. Ο **μέσος χρόνος ανάγνωσης (προσπέλασης) από ιεραρχικό σύστημα μνήμης** δίνεται από τον τύπο $T = \sum_{i=1}^{n=3}(E_i - E_{i-1}) \times T_i$. Κανονικά έχουμε τρία επίπεδα, επειδή όμως αναφέρει **η εκφώνηση ότι η κύρια μνήμη περιέχει πάντα τη ζητούμενη πληροφορία, δεν** προσπελαύνεται η βοηθητική μνήμη. Επομένως το n = 2 και ο προηγούμενος τύπος διαμορφώνεται ως εξής: $T = \sum_{i=1}^{n=2}(E_i - E_{i-1}) \times T_i = (E_1 - E_0) \times T_1 + (E_2 - E_1) \times T_2 = (0.9 - 0) \times 10$ nsec + (1 – 0.9) x (90 + 10) nsec = 0.9 x 10nsec + 0.1 x 100 nsec = 19 nsec.

Θέμα Φεβρουαρίου 2016 με χαμηλής τάξης διαφύλλωση

Θεωρήστε υπολογιστή στον οποίο σε κάθε ψηφιολέξη (byte) της κύριας μνήμης αντιστοιχεί μία διεύθυνση. Ποια πρέπει να είναι η οργάνωση της κύριας μνήμης και το εύρος της αρτηρίας δεδομένων μεταξύ της κύριας μνήμης και της κρυφής μνήμης ώστε ένα μπλοκ πληροφορίας των 64 ψηφιολέξεων να μεταφέρεται από την κύρια μνήμη στην κρυφή μνήμη σε 320 κύκλους ρολογιού εάν κάθε προσπέλαση της κύριας μνήμης μαζί με τη μεταφορά της αντίστοιχης πληροφορίας στην κρυφή μνήμη απαιτεί 20 κύκλους ρολογιού; Να αιτιολογήσετε την απάντησή σας.

Λύση

Συνολικά έχουμε 320 κ.ρ. και κάθε προσπέλαση της Κ.Μ. μαζί με τη μεταφορά στην κρυφή μνήμη κοστίζει 20 κ.ρ.

Επομένως, συνολικά έχουμε $\frac{320\ \text{κ.ρ.}}{\frac{20\ \text{κ.ρ.}}{\text{προσπέλαση}}} = 16$ προσπελάσεις, αφού ισχύει ότι:

1 πρ.	20 κ.ρ.
x;	320 κ.ρ.

$x = \frac{320}{20} = 16$

Επειδή μεταφέρουμε πληροφορία των 64 bytes και αυτή πρέπει να μεταφερθεί με 16 προσπελάσεις μνήμης, αυτό σημαίνει ότι με κάθε προσπέλαση πρέπει να μεταφέρονται $\frac{64\ \text{bytes}}{16\ \text{προσπ.}} = 4$ ψηφιολέξεις/προσπ. = 4 θ.μ./προσπ. **Άρα, θα πρέπει να έχουμε μια οργάνωση χαμηλής τάξης διαφύλλωσης μνήμης των 4 – τρόπων (δρόμων)**, αφού με τη συγκεκριμένη οργάνωση μπορούμε κάθε φορά να μεταφέρουμε 4 ψηφιολέξεις = 4 θ.μ. (λόγω της οργάνωσης 1 ψηφιολέξη ανά θέση μνήμης). Το εύρος της αρτηρίας δεδομένων θα είναι 4 bytes x 8 bits = 32 bits. Αυτό πρακτικά **σημαίνει ότι δεν θα υπάρχει MUX και προφανώς ούτε καταχωρητές**. Αν υπήρχε πολυπλέκτης, η εκφώνηση θα έδινε και μια επιπλέον πληροφορία του είδους: κάθε πέρασμα μέσω της αρτηρίας δεδομένων κοστίζει π.χ. 1 κ.ρ. Δηλαδή θα ήταν διαφορετικής διάρκειας η προσπέλαση μνήμης από τη μεταφορά στην κρυφή μνήμη και αυτοί οι δύο χρόνοι θα δινόντουσαν από την εκφώνηση ως δύο ξεχωριστά μεγέθη. Τώρα όμως αυτά τα δύο μεγέθη δίνονται **μαζί**, με συνολική διάρκεια 20 κ.ρ.

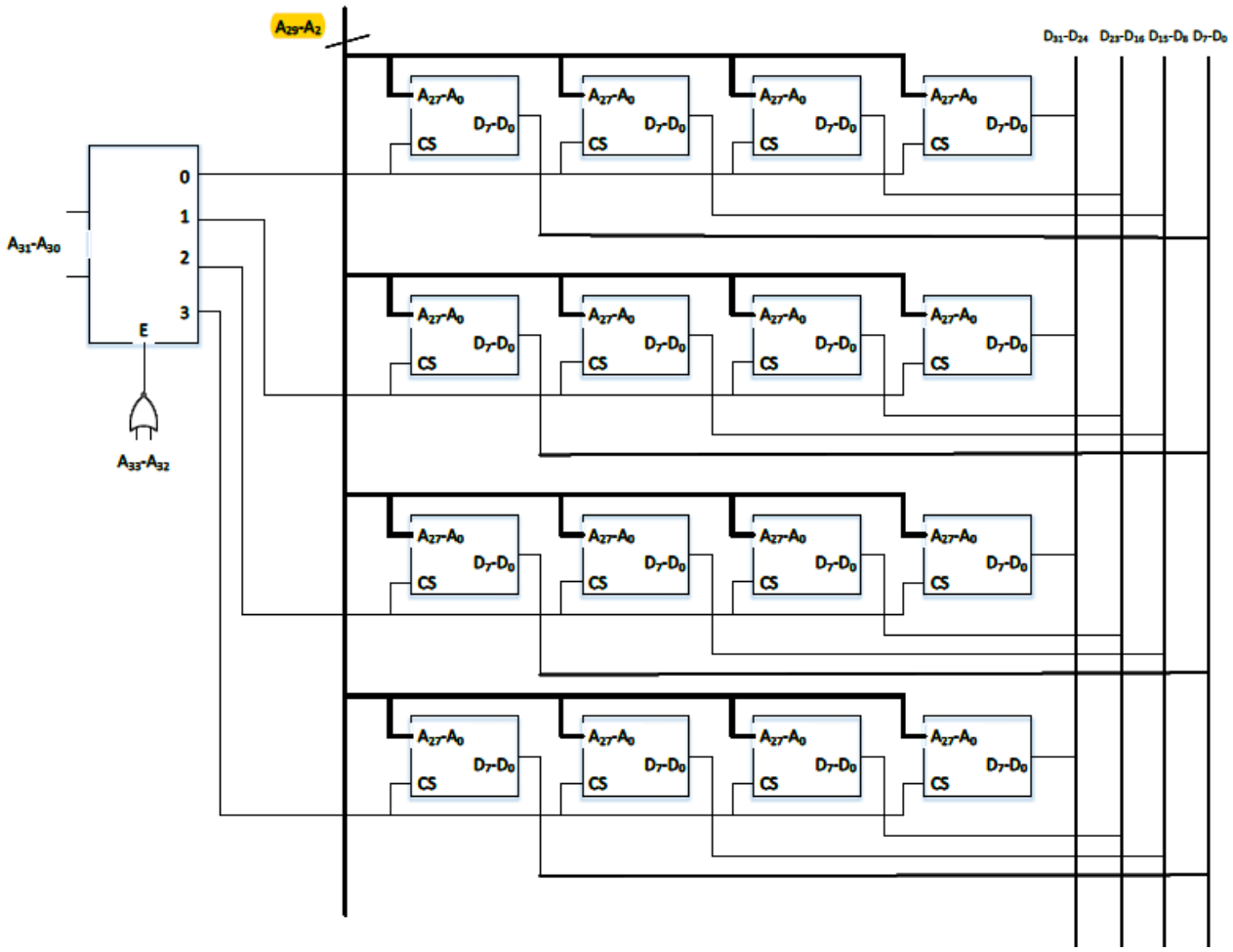
Θέμα Ιουνίου 2016 με σχεδιασμό κύριας μνήμης χαμηλής τάξης διαφύλλωσης

Έχετε στην διάθεσή σας ολοκληρωμένα κυκλώματα με χωρητικότητα 256 Mbytes (M ψηφιολέξεις) και οργάνωση 1 Byte (ψηφιολέξη) ανά θέση μνήμης.

α) Να σχεδιάσετε σύστημα μνήμης με χωρητικότητα 4 Gbytes έτσι ώστε σε κάθε byte να αντιστοιχεί μια διεύθυνση και να προσπελαύνονται ταυτόχρονα 4 bytes. Η αρτηρία δεδομένων είναι των 32 δυαδικών ψηφίων, ενώ η αρτηρία διευθύνσεων είναι των 34 δυαδικών ψηφίων. Ένα σχήμα στο οποίο να δίνεται όλη η απαραίτητη πληροφορία είναι απαιτούμενο και αρκετό.

β) Όταν ο επεξεργαστής στέλνει στο σύστημα μνήμης που σχεδιάσατε στο ερώτημα α τη διεύθυνση 38A46B432 να δώσετε τις διευθύνσεις των bytes που προσπελαύνονται ταυτόχρονα.

Λύση



Θέμα Φεβρουαρίου 2012 με χαμηλής τάξης διαφύλλωση

Δίνεται σύστημα μνήμης με οργάνωση 4-τρόπων χαμηλής τάξης διαφύλλωσης με μια ψηφιολέξη (byte) ανά θέση μνήμης (μια διεύθυνση αντιστοιχεί σε κάθε ψηφιολέξη). Η αρτηρία δεδομένων μεταξύ του συστήματος μνήμης και της κεντρικής μονάδας επεξεργασίας (ΚΜΕ) είναι

α. των 16 δυαδικών ψηφίων

β. των 32 δυαδικών ψηφίων

γ. των 8 δυαδικών ψηφίων

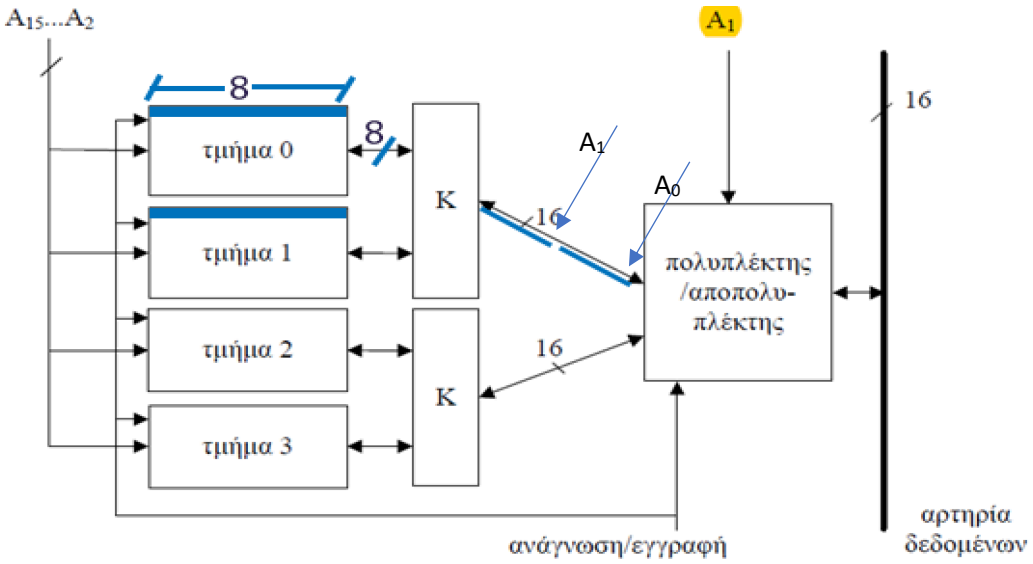
1. Να σχεδιάζεται το σύστημα μνήμης σε μπλοκ διάγραμμα για καθεμία των ανωτέρω περιπτώσεων. Να θεωρήσετε ότι η αρτηρία διευθύνσεων είναι των 16 δυαδικών ψηφίων. Όλη η απαιτούμενη πληροφορία να φαίνεται πάνω στο σχήμα.

2. Να υπολογίσετε το ρυθμό μεταφοράς πληροφορίας (εντολών και δεδομένων) από το σύστημα της μνήμης προς την κεντρική μονάδα επεξεργασίας σε ψηφιολέξεις ανά δευτερόλεπτο λαμβάνοντας υπ' όψιν ότι ο χρόνος προσπέλασης μιας θέσης του συστήματος μνήμης ισούται με 30 κύκλους ρολογιού και κάθε μεταφορά πληροφορίας μέσω της αρτηρίας διαρκεί ένα κύκλο ρολογιού. Θεωρήστε ότι οι διευθύνσεις τόσο των εντολών όσο και των δεδομένων στη μνήμη είναι ευθυγραμμισμένες, τόσο οι εντολές όσο και τα δεδομένα είναι των 4 ψηφιολέξεων και ένας κύκλος ρολογιού ισούται με 2 nsec.

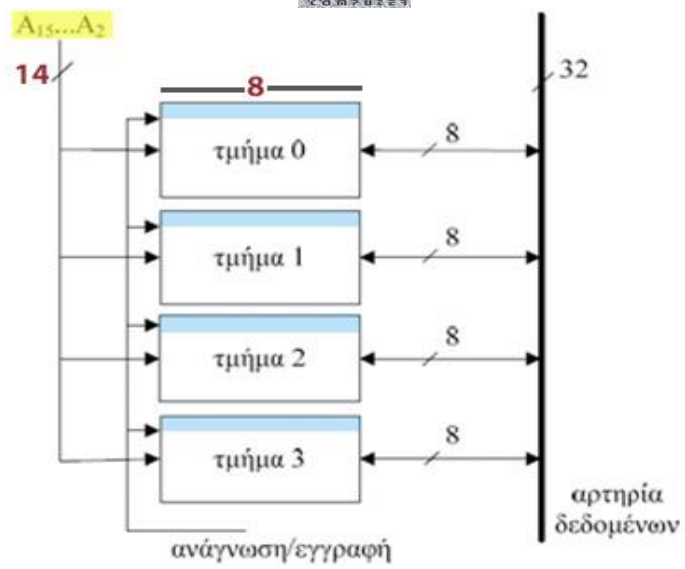
Λύση

Σε όλες τις περιπτώσεις που ακολουθούν, το μέγεθος του address bus είναι 16 bits.

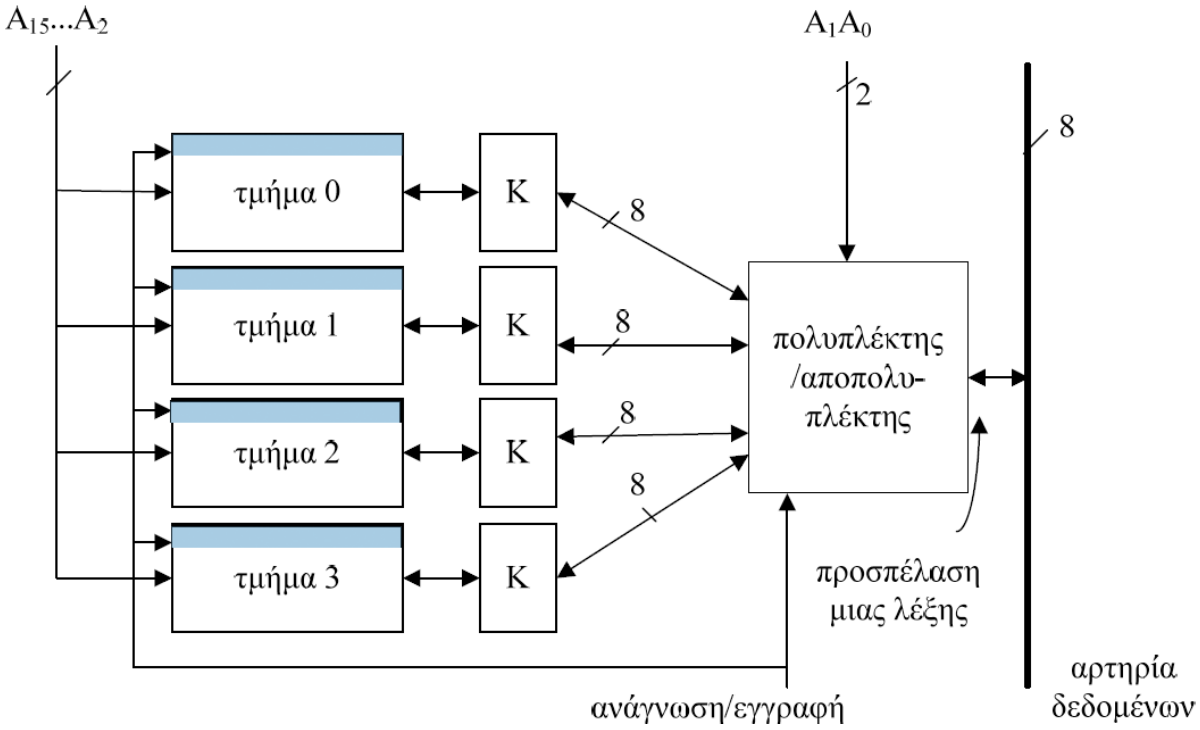
1a. Θα πρέπει να συνδυάσουμε ανά δύο τα ΟΚΜ των 4 – τρόπων προκειμένου να πετύχουμε μια αρτηρία δεδομένων των 16 bits. Αυτός ο συνδυασμός επιτυγχάνεται με τους καταχωρητές, οι οποίοι επεκτείνονται και γίνονται των 16 – bits. Επειδή, κάθε τέτοιος καταχωρητής οδηγεί δύο θ.μ., **θα πρέπει να χρησιμοποιηθεί το bit A_1 από το address bus ως γραμμή επιλογής του MUX 2 – 1, προκειμένου να επιλέγονται και οι δύο θ.μ. Αν χρησιμοποιούσαμε ως γραμμή επιλογής του MUX 2 – 1 το bit A_0, τότε θα μπορούσαμε από τις δύο λέξεις, να επιλέγουμε μόνο τη δεξιότερη**. Απαιτείται πολυπλέκτης 2 -1 και όχι 4 – 1, όπως θα ήταν κανονικά, αν η αρτηρία δεδομένων ήταν των 8 bits. 2^{μ} → 1. Επειδή το address bus = 16 bits, χρησιμοποιούνται τα 14 πιο σημαντικά (A15 – A2) ως είσοδοι διευθυνσιοδότησης σε κάθε τμήμα.



1b. Θα πρέπει να συνδέσουμε **απευθείας** τα 4 τμήματα με την αρτηρία δεδομένων, προκειμένου να μεγαλώσουμε το εύρος της και συγκεκριμένα να το τετραπλασιάσουμε. Στην περίπτωση αυτή **δεν** απαιτείται πολυπλέκτης, επειδή δεν χρειάζεται να επιλέγουμε μια λέξη για να την οδηγήσουμε στην αρτηρία δεδομένων. Από τη στιγμή που δεν χρειάζεται MUX, δεν χρειάζονται και καταχωρητές (για να αποθηκεύουν προσωρινά τα δεδομένα των λέξεων, πριν αυτά περάσουν στην αρτηρία δεδομένων). Θα πρέπει να παρατηρηθεί ότι παρόλο που δεν χρησιμοποιήθηκε πολυπλέκτης, τα bits A_1 και A_0 του address bus παραμένουν δεσμευμένα, και στη διευθυνσιοδότηση των τμημάτων χρησιμοποιούνται τα υπόλοιπα bits από το address bus, δηλ. τα bits $A_{15} – A_2$. Από αυτό συμπεραίνουμε ότι κάθε τμήμα έχει 2^{14} θ.μ. x 8 bits/θ.μ. = 2^{14} θ.μ. x 1 byte/θ.μ. = 2^{14} bytes 16 KB.



1c. Στην περίπτωση αυτή δεν μεγαλώνει το εύρος της αρτηρίας δεδομένων, επομένως τώρα θα πρέπει να χρησιμοποιηθεί ένας MUX 4 – 1, προκειμένου κάθε φορά να επιλέγει μια από τις 4 λέξεις του ενός byte που εισέρχονται, για να την οδηγήσει στην αρτηρία δεδομένων.



2a. Ο χρόνος προσπέλασης μιας θέσης του συστήματος **που ισούται με 30 κ.ρ.** αναφέρεται στο χρόνο που απαιτείται για να προσπελαστούν οι 4 θέσεις μνήμης και να μεταφερθούν τα περιεχόμενά τους στους καταχωρητές. Από εκεί και πέρα, κάθε πέρασμα μέσω του MUX απαιτεί 1 κ.ρ. Συνολικά έχουμε δύο τέτοια περάσματα, επομένως 2 κ.ρ. Ο ρυθμός μεταφοράς πληροφορίας (εντολών/δεδομένων) $= \frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{4 \text{ bytes}}{30 \text{ κ.ρ.} + 2 \text{ κ.ρ.}} = \frac{4 \text{ bytes}}{32 \text{ κ.ρ.}} = \frac{4 \text{ bytes}}{64 \text{ nsec.}}$

2b. Ο χρόνος προσπέλασης μιας θέσης του συστήματος που ισούται με 30 κ.ρ. αναφέρεται και πάλι στο χρόνο που απαιτείται για να προσπελαστούν οι 4 θέσεις μνήμης και να μεταφερθούν στους καταχωρητές. Από εκεί και πέρα, δεν υπάρχει τώρα MUX, όμως για την (ταυτόχρονη) μεταφορά των 4 θ.μ. στην αρτηρία δεδομένων απαιτείται ένας κ.ρ., οπότε ο ρυθμός μεταφοράς πληροφορίας (εντολών/δεδομένων) $= \frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{4 \text{ bytes}}{30 \text{ κ.ρ.} + 1 \text{ κ.ρ.}} = \frac{4 \text{ bytes}}{62 \text{ nsec.}}$

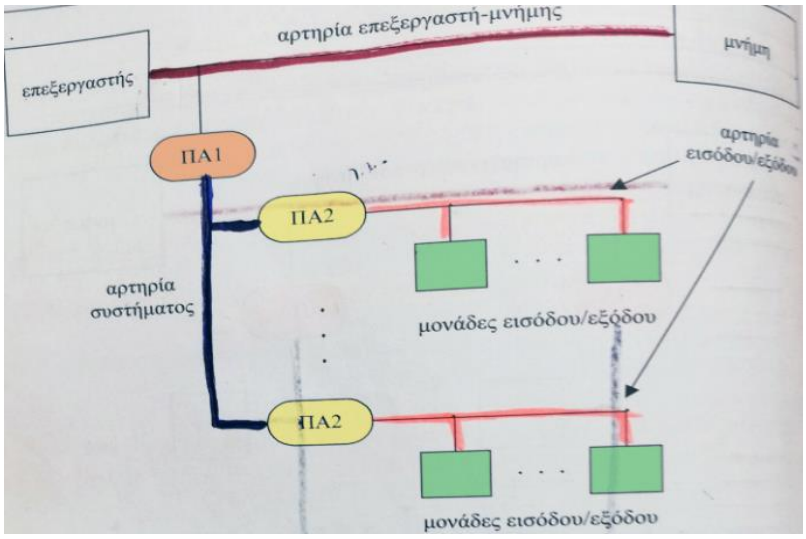
2c. Ο χρόνος προσπέλασης μιας θέσης του συστήματος που ισούται με 30 κ.ρ. αναφέρεται στο χρόνο που απαιτείται για να προσπελαστούν οι 4 θέσεις μνήμης και να μεταφερθούν στους καταχωρητές. Από εκεί και πέρα, κάθε πέρασμα μέσω του MUX απαιτεί 1 κ.ρ. Συνολικά έχουμε τέσσερα τέτοια περάσματα, επομένως απαιτούνται 4 κ.ρ. Ο ρυθμός μεταφοράς πληροφορίας (εντολών/δεδομένων) $= \frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{4 \text{ bytes}}{30 \text{ κ.ρ.} + 4 \text{ κ.ρ.}} = \frac{4 \text{ bytes}}{34 \text{ κ.ρ.}} = \frac{4 \text{ bytes}}{68 \text{ nsec.}}$

Κεφάλαιο 6 – Αρτηρίες

Βασικά σημεία θεωρίας

Το πλήθος των bits που μπορούν να μεταφερθούν ταυτόχρονα από μια παράλληλη αρτηρία καθορίζεται από το πλήθος των γραμμών δεδομένων (μέγεθος του data bus). Όταν υπάρχουν παράλληλες αρτηρίες τότε αυτές διακρίνονται σε αυτές που χρησιμοποιούν **πολυπλεγμένες** γραμμές διευθύνσεων και δεδομένων (σε αυτή την περίπτωση η αρτηρία χρησιμοποιείται για κάποιες χρονικές περιόδους προκειμένου να μεταφέρει διευθύνσεις και για άλλες χρονικές περιόδους προκειμένου να μεταφέρει δεδομένα) και σε αυτές που χρησιμοποιούν **διαφορετικές** γραμμές διευθύνσεων και δεδομένων. Οι παράλληλες έχουν υψηλό ρυθμό μεταφοράς δεδομένων, αλλά πρέπει να έχουν μικρό μήκος (διαφορετικά υπάρχουν χρονικές καθυστερήσεις και χρονικές αποκλίσεις) και επίσης έχουν υψηλό κόστος. Οι σειριακές αρτηρίες ενδείκνυνται για τη σύνδεσης μονάδων που έχουν μεγάλη απόσταση μεταξύ τους, έχουν μικρότερο κόστος (αφού περιέχουν λιγότερες γραμμές διασύνδεσης) και μπορούν να συνδέσουν περισσότερες μονάδες μεταξύ τους από ότι οι παράλληλες αρτηρίες.

Μια αρτηρία αποκλειστικής χρήσης είναι αυτή που συνδέει επεξεργαστή με κύρια μνήμη. Έχει υψηλό ρυθμό μεταφοράς δεδομένων και υψηλό κόστος. Μια βλάβη σε αυτή την αρτηρία δεν βγάζει εκτός λειτουργίας όλο το σύστημα. Σε μια αρτηρία κοινής χρήσης το κόστος της είναι χαμηλότερο, επιτρέπει την εύκολη προσθήκη νέων μονάδων, όμως έχει το μειονέκτημα ότι είναι αργή (αν π.χ. χρησιμοποιείται ήδη από δύο μονάδες δεν μπορεί να χρησιμοποιηθεί από άλλες) και επιπλέον μια βλάβη μπορεί να θέσει εκτός λειτουργίας τον Η/Υ. **Η αρτηρία εισόδου – εξόδου είναι μια αρτηρία κοινής χρήσης**, επειδή σε αυτή συνδέονται διαφορετικές μονάδες εισόδου – εξόδου. Όταν πρέπει να συνδεθούν διαφορετικά είδη αρτηριών μεταξύ τους, τότε ανάμεσά τους τοποθετούνται προσαρμοστές αρτηρίας (ΠΑ).

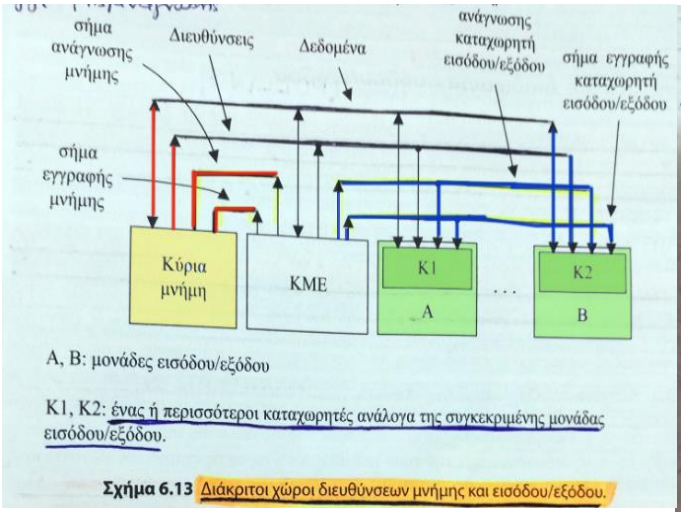


Μια κλασσική αρτηρία συστήματος είναι η αρτηρία PCI. Αρτηρίες εισόδου – εξόδου είναι οι ISA, EISA, SCSI. Σε μια αρτηρία σύγχρονη ένα από τα σήματα που μεταδίδονται είναι ένα σήμα χρονισμού (ρολόι) και όλες οι λειτουργίες γίνονται βάσει αυτού. Είναι γρήγορες, αλλά το πλήθος των μονάδων που μπορούν να συνδεθούν σε αυτές είναι μικρό. Προφανώς, όλες αυτές οι μονάδες πρέπει να λειτουργούν με το ίδιο σήμα χρονισμού. Παράδειγμα **σύγχρονης** αρτηρίας είναι η αρτηρία επεξεργαστή – μνήμης. Είναι πιο γρήγορες από τις ασύγχρονες, οι οποίες δεν έχουν σήμα χρονισμού και χρησιμοποιούν το πρωτόκολλο χειραψίας (handshaking protocol), που περιέχει τα σήματα Request και Acknowledge, για να μεταφέρουν πληροφορίες. Οι ασύγχρονες αρτηρίες επιτρέπουν τη σύνδεση μιας ποικιλίας μονάδων και επιπλέον έχουν μεγάλο μήκος. Παράδειγμα **ασύγχρονης** αρτηρίας είναι οι αρτηρίες εισόδου – εξόδου. Για το συνδυασμό της ταχύτητας των σύγχρονων αρτηριών και της δυνατότητας σύνδεσης πολλών διαφορετικών συσκευών που προσφέρουν οι ασύγχρονες αρτηρίες, χρησιμοποιούνται σύγχρονες αρτηρίες με δυνατότητα εισαγωγής κύκλων καθυστέρησης (wait cycles). Για την αποφυγή της περίπτωσης όπου διαφορετικές κύριες μονάδες επιθυμούν να θέσουν την **ίδια** χρονική στιγμή υπό τον έλεγχό τους την αρτηρία,

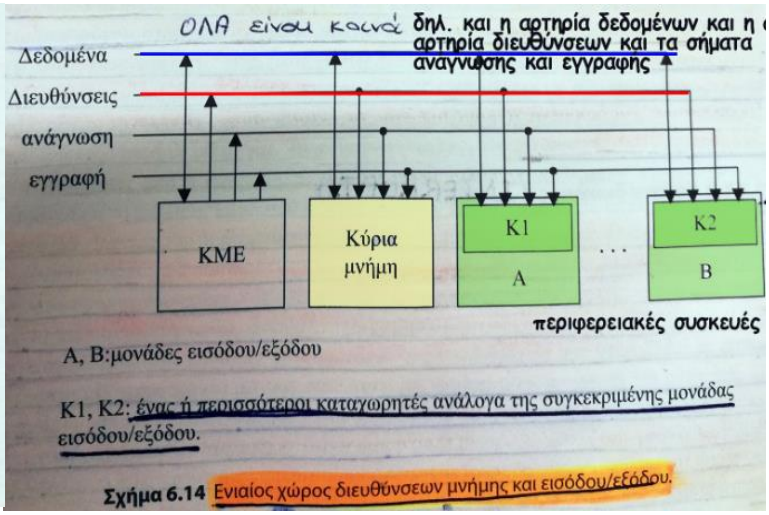
χρησιμοποιείται ο **μηχανισμός της διαιτησίας**. Υπάρχουν διαφορετικά σχήματα διαιτησίας, όπου το πιο σύνηθες είναι το σχήμα διαιτησίας με **χρήση αλυσίδας προτεραιότητας**. Το σχήμα αυτό έχει απλότητα υλοποίησης, αλλά παρουσιάζει έλλειψη δικαιοσύνης, καθώς μια αίτηση από μια μονάδα χαμηλότερης προτεραιότητας δεν ικανοποιείται, εφόσον υπάρχουν απαιτήσεις από μονάδες υψηλότερης προτεραιότητας, άσχετα με το χρόνο αναμονής. Τα κριτήρια επιλογής του κατάλληλου κάθε φορά σχήματος διαιτησίας είναι ο αριθμός των μονάδων που συνδέονται σε αυτή, το μήκος της αρτηρία, ο απαιτούμενος βαθμός δικαιοσύνης κ.λ.π.

Η **απόδοση μιας αρτηρίας εξαρτάται** από την καθυστέρηση (latency) που παρουσιάζει και το ρυθμό μεταφοράς της. Ο τελευταίος προσδιορίζεται από την ύπαρξη διακριτών γραμμών διευθύνσεων και δεδομένων, από τη δυνατότητα ομαδικής μεταφοράς δεδομένων και από το εύρος της αρτηρίας δεδομένων.

Αναφορικά με τους **τρόπους διακίνησης πληροφορίας ελέγχου μεταξύ ΚΜΕ και μονάδων Ε/Ε (Εισόδου/Εξόδου)**, είναι η ύπαρξη **διακριτών** χώρων διευθύνσεων μνήμης και Ε/Ε και η ύπαρξη **ενιαίου** χώρου διευθύνσεων μνήμης και Ε/Ε.



Σχήμα 6.13 Διάκριτοι χώροι διευθύνσεων μνήμης και εισόδου/εξόδου.

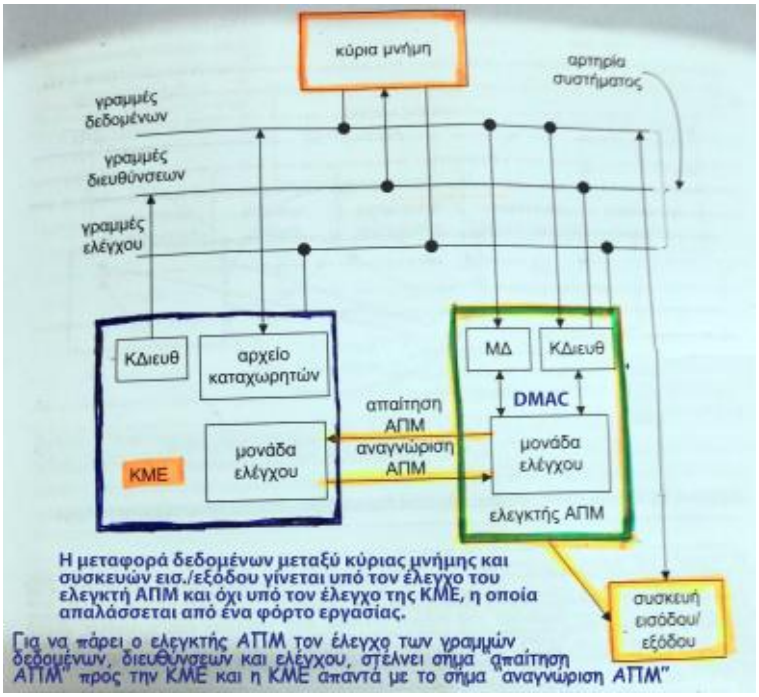


Σχήμα 6.14 Ενιαίος χώρος διευθύνσεων μνήμης και εισόδου/εξόδου.

Αναφορικά με τους **τρόπους διακίνησης πληροφορίας από κάποια συσκευή Ε/Ε προς την ΚΜΕ,** υπάρχουν δύο προσεγγίσεις: ο χρονοπρογραμματισμένος έλεγχος (polling) και η χρήση σημάτων διακοπής (interrupts). Αναφορικά με την πρώτη προσέγγιση, η ΚΜΕ διαβάζει περιοδικά (σε τακτά χρονικά διαστήματα) τον καταχωρητή κατάστασης του ελεγκτή μιας μονάδας Ε/Ε για να διαπιστώσει την κατάστασή της. Αυτό συμβαίνει στην περίπτωση που η μονάδα Ε/Ε είναι το πληκτρολόγιο. Το polling έχει απλή υλοποίηση, αλλά μεγάλη καθυστέρηση, ιδιαίτερα όταν το πλήθος των μονάδων Ε/Ε είναι πολύ μεγάλο. Ο πιο αποδοτικός τρόπος στην περίπτωση αυτή είναι η χρήση σημάτων διακοπής, όπου τότε μια μονάδα που θέλει εξυπηρέτηση, στέλνει σήμα διακοπής στην ΚΜΕ για να διακόψει τη λειτουργία της. Τότε η ΚΜΕ εκτελεί ένα υποπρόγραμμα που ονομάζεται ρουτίνα εξυπηρέτησης διακοπών (ISR = Interrupt Service Routine). Υπάρχουν πολλές διαφορετικές ρουτίνες εξυπηρέτησης διακοπών, ανάλογα με τη συσκευή που προκάλεσε τη διακοπή, που βρίσκονται σε προκαθορισμένες διευθύνσεις της κύριας μνήμης, από όπου και εντοπίζονται, για να εκτελεστούν. Τα σήματα διακοπής διαιρούνται σε δύο κατηγορίες: αυτά που μπορούν να απενεργοποιηθούν (maskable), δηλαδή αυτά που μπορούν να παρεμποδιστούν για κάποιο χρονικό διάστημα και αυτά που δεν μπορούν να απενεργοποιηθούν (non - maskable).

Αναφορικά με τη **συμμετοχή της ΚΜΕ στη διαδικασία εισόδου – εξόδου**, υπάρχουν τρεις περιπτώσεις: a) Η προγραμματισμένη Ε/Ε (η απόδοση του συστήματος υποβαθμίζεται σημαντικά), b) η προγραμματισμένη Ε/Ε με χρήση σημάτων διακοπής (σε αυτήν την περίπτωση αποδεσμεύεται η ΚΜΕ από το συνεχή έλεγχο της κατάστασης της συσκευής Ε/Ε), c) με χρήση άμεσης προσπέλασης μνήμης (ΑΠΜ), όπου ο ελεγκτής της συσκευής Ε/Ε καλείται σε αυτήν την περίπτωση ελεγκτής άμεσης προσπέλασης μνήμης (DMAC – DMA Controller). Ο ελεγκτής ΑΠΜ ζητά στην περίπτωση αυτή από την ΚΜΕ τον έλεγχο των διαύλων του συστήματος και όταν συμβεί κάτι τέτοιο, αναλαμβάνει τη μεταφορά πληροφοριών μεταξύ κύριας

μνήμης και περιφερειακών συσκευών χωρίς τη μεσολάβηση της ΚΜΕ, η οποία μπορεί να ασχοληθεί με δικές της διεργασίες. Όταν ολοκληρωθούν οι μεταφορές, ο έλεγχος των διαύλων επιστρέφει πίσω στην ΚΜΕ.



Άσκηση 6.1 βιβλίου με σύγχρονη/ασύγχρονη αρτηρία

Θεωρήστε μια αρτηρία που χρησιμοποιείται για τη μεταφορά πληροφορίας 4 ψηφιολέξεων κάθε φορά και ότι, ανάλογα με τις μονάδες που επικοινωνούν, η μεταφορά από τον υπηρέτη (slave) στον κύριο (master) διαρκεί στο 20% των περιπτώσεων 10 nsec, στο 30% των περιπτώσεων 20 nsec, στο 40% των περιπτώσεων 30 nsec και στο 10% των περιπτώσεων 60 nsec. Θεωρήστε επίσης ότι η διαιτησία βάζει καθυστέρηση 5 nsec. Υπολογίστε το **μέσο ρυθμό μεταφοράς πληροφορίας σε ψηφιολέξεις ανά δευτερόλεπτο** για κάθε μια από τις ακόλουθες περιπτώσεις.

α. Η αρτηρία είναι σύγχρονη με περίοδο του σήματος χρονισμού (ρολογιού) 5 nsec

i. χωρίς δυνατότητα εισαγωγής κύκλων ρολογιού καθυστέρησης (wait states)

ii. με δυνατότητα εισαγωγής κύκλων ρολογιού καθυστέρησης

β. Η αρτηρία είναι ασύγχρονη και απαιτεί 10 nsec επιπλέον για κάθε μεταφορά λόγω του πρωτοκόλλου χειραψίας

Λύση

Ο **μέσος ρυθμός μεταφοράς πληροφορίας** (εντολών/δεδομένων) $= \frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}}$

α. i) Όταν η αρτηρία είναι **σύγχρονη** και **δεν** χρησιμοποιούνται κύκλοι ρολογιού καθυστέρησης (wait states ή wait cycles)[1], τότε για τον υπολογισμό του **χρόνου μεταφοράς λαμβάνουμε υπόψη την πιο αργή συσκευή + την καθυστέρηση λόγω διατησίας**, δηλ. χρόνος μεταφοράς = 60 nsec + 5 nsec = 65 nsec. Επομένως ο **μέσος ρυθμός μεταφοράς πληροφορίας** (εντολών/δεδομένων) $= \frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{4 \text{ bytes}}{65 \text{ nsec}} = \frac{4 \text{ x } 10^{-6}\text{MB}}{65 \text{ x } 10^{-9}\text{sec}} = 0.0615 \text{ x } 10^3$ MB/sec = 61.5 MB/sec ή εναλλακτικά $\frac{4 \text{ bytes}}{65 \text{ nsec}} = \frac{4 \text{ x } 10^9\text{bytes}}{65 \text{ sec}} = 0.0615 \text{ x } 10^9$ bytes/sec = $0.0615 \text{ x } 10^3 \text{ x } 10^6$ bytes/sec = 61.5 MB/sec. Σε περίπτωση που η απάντηση ζητείται σε KB/sec θα έχουμε: $\frac{4 \text{ bytes}}{65 \text{ nsec}} = \frac{4 \text{ x } 10^9\text{bytes}}{65 \text{ sec}} = 0.0615 \text{ x } 10^6 \text{ x } 10^3$ bytes/sec = 61500 KB/sec.

Παρατήρηση: Όταν αναφερόμαστε **σε ρυθμό μεταφοράς πληροφορίας** χρησιμοποιούμε δυνάμεις του «10» και όχι δυνάμεις του «2» ως συνήθως.

[1] Τα wait states χρησιμοποιούνται όταν μια γρήγορη ΚΜΕ επικοινωνεί με αργές περιφερειακές συσκευές προκειμένου να επιβραδυνθεί η λειτουργία της

ii) Όταν η αρτηρία είναι **σύγχρονη** και χρησιμοποιούνται κύκλοι ρολογιού καθυστέρησης (wait states), τότε για τον υπολογισμό του χρόνου μεταφοράς λαμβάνουμε υπόψη την **καθυστέρηση κάθε συσκευής με το αντίστοιχο ποσοστό της + την καθυστέρηση λόγω διατησίας**, δηλ. χρόνος μεταφοράς = 0.20 x 10 nsec + 0.30 x 20 nsec + 0.40 x 30 nsec + 0.10 x 60 nsec + 5 nsec = (2 + 6 + 12 + 6 + 5) nsec = 31 nsec. Επομένως ο **μέσος ρυθμός μεταφοράς πληροφορίας** (εντολών/δεδομένων) = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{4 \text{ bytes}}{31 \text{ nsec}} = \frac{4 \text{ x } 10^{-6}\text{MB}}{31 \text{ x } 10^{-9}\text{sec}}$ = 0.129 x 10^3 MB/sec = 129 MB/sec. Παρατηρούμε ότι μέσος ρυθμός μεταφοράς πληροφορίας υπερδιπλασιάζεται σε σχέση με πριν.

Παρατήρηση: η πληροφορία της εκφώνησης για τη διάρκεια του κύκλου ρολογιού φαινομενικά δεν χρησιμοποιήθηκε πουθενά. Στην πραγματικότητα όμως χρησιμοποιείται, διότι πρέπει να ελέγξουμε όλους τους δοθέντες χρόνους (ακόμη και αυτόν της διαιτησίας) έτσι ώστε να είναι **ακέραια πολλαπλάσια της διάρκειας του κύκλου ρολογιού**. Αν δεν είναι, τότε πρέπει να τους στρογγυλοποιήσουμε προς τα πάνω για να γίνουν.

β) **Όταν είναι η αρτηρία είναι ασύγχρονη, τότε δεν χρησιμοποιείται ρολόι** και για την επικοινωνία των συσκευών Ε/Ε με την ΚΜΕ χρησιμοποιείται πρωτόκολλο χειραψίας με χρήση σημάτων χειραψίας (handshaking) που είναι τα σήματα της απαίτησης (request) και της αναγνώρισης (acknowledge) του Σχ. 6.10. Στην περίπτωση αυτή θα πρέπει, **εκτός από την καθυστέρηση των συσκευών και της διατησίας, να λάβουμε υπόψη και την καθυστέρηση λόγω του χρησιμοποιούμενου πρωτοκόλλου χειραψίας**. Ο νέος χρόνος μεταφοράς = 0.20 x (10 + 10 + 5)nsec + 0.30 x (20 +10 + 5)nsec + 0.40 x (30 + 10 + 5)nsec + 0.10 x (60 +10 + 5) nsec = 41 nsec Επομένως ο **μέσος ρυθμός μεταφοράς πληροφορίας** (εντολών/δεδομένων) = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{4 \text{ bytes}}{41 \text{ nsec}} = \frac{4 \text{ x } 10^{-6}\text{MB}}{41 \text{ x } 10^{-9}\text{sec}}$ = 0.09756 x 10^3 MB/sec = 97.56 MB/sec.

Άσκηση 6.2 βιβλίου με σύγχρονη/ασύγχρονη αρτηρία

Θεωρήστε μια αρτηρία που χρησιμοποιείται για τη μεταφορά πληροφορίας 8 ψηφιολέξεων κάθε φορά και ότι, ανάλογα με τις μονάδες που επικοινωνούν, η μεταφορά από τον υπηρέτη (slave) στον κύριο (master) διαρκεί στο 20% των περιπτώσεων 12 nsec, στο 30% των περιπτώσεων 19 nsec, στο 40% των περιπτώσεων 28 nsec και στο 10% των περιπτώσεων 62 nsec. Θεωρήστε επίσης ότι η διαιτησία βάζει καθυστέρηση 5 nsec. Υπολογίστε το μέσο ρυθμό μεταφοράς πληροφορίας ανά δευτερόλεπτο για κάθε μια από τις ακόλουθες περιπτώσεις.

α. Η αρτηρία είναι **σύγχρονη** με περίοδο του σήματος χρονισμού (ρολογιού) **5 nsec**

i. χωρίς δυνατότητα εισαγωγής κύκλων ρολογιού καθυστέρησης (wait states)

ii. με δυνατότητα εισαγωγής κύκλων ρολογιού καθυστέρησης

β. Η αρτηρία είναι **ασύγχρονη** για κάθε μεταφορά, λόγω του πρωτοκόλλου χειραψίας, απαιτεί 10 nsec επιπλέον.

Λύση

Ο **μέσος ρυθμός μεταφοράς πληροφορίας** (εντολών/δεδομένων) = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}}$

α. i) Όταν η αρτηρία είναι σύγχρονη και **δεν** χρησιμοποιούνται κύκλοι ρολογιού καθυστέρησης (wait states), τότε για τον υπολογισμό του χρόνου μεταφοράς λαμβάνουμε **υπόψη την πιο αργή συσκευή + την καθυστέρηση λόγω διατησίας**, δηλ. χρόνος μεταφοράς = 65 nsec + 5 nsec = 70 nsec. **Δηλαδή στρογγυλοποιούμε προς τα πάνω όλους τους χρόνους που δεν είναι ακέραια πολλαπλάσια της διάρκειας του κύκλου ρολογιού** για να γίνουν, και έτσι το **62 nsec → 65 nsec**. Επομένως ο **μέσος ρυθμός μεταφοράς πληροφορίας** (εντολών/δεδομένων) = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{8 \text{ bytes}}{70 \text{ nsec}}$.

ii) Όταν η αρτηρία είναι **σύγχρονη** και χρησιμοποιούνται **κύκλοι ρολογιού καθυστέρησης (wait states)**, τότε για τον υπολογισμό του χρόνου μεταφοράς λαμβάνουμε υπόψη **την καθυστέρηση κάθε συσκευής με το αντίστοιχο ποσοστό της + την καθυστέρηση λόγω διαιτησίας**, δηλ. χρόνος μεταφοράς = 0.20 x 15 nsec + 0.30 x 20 nsec + 0.40 x 30 nsec + 0.10 x

65 nsec + 5 nsec = 32.5 nsec. Παρατηρούμε ότι οι χρόνοι των συσκευών που δεν είναι ακέραια πολλαπλάσια της διάρκειας του κύκλου ρολογιού, στρογγυλοποιούνται προς τα πάνω, προκειμένου να γίνουν. Αυτό σημαίνει ότι τα 12 nsec → 15 nsec, τα 19 nsec → 20 nsec κ.ο.κ. Επομένως ο **μέσος ρυθμός μεταφοράς πληροφορίας** (εντολών/δεδομένων) = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{8 \text{ bytes}}{32.5 \text{ nsec}}$. Παρατηρούμε ότι ο μέσος ρυθμός μεταφοράς πληροφορίας υπερδιπλασιάζεται σε σχέση με πριν.

Παρατήρηση: η πληροφορία της εκφώνησης για τη διάρκεια του κύκλου ρολογιού **χρησιμοποιήθηκε**, διότι πρέπει να ελέγξουμε όλους τους δοθέντες χρόνους (ακόμη και αυτόν της διαιτησίας) ώστε να είναι όλοι ακέραια πολλαπλάσια της διάρκειας του κύκλου ρολογιού και **εφαρμόστηκε** σε όλους εκείνους τους χρόνους που δεν ήταν ακέραια.

β) Όταν είναι η αρτηρία είναι **ασύγχρονη**, τότε **δεν** χρησιμοποιείται ρολόι, -οπότε δεν πρέπει να γίνει στρογγυλοποίηση των χρόνων προς τα πάνω, διότι για την επικοινωνία των συσκευών εις/εξ με την ΚΜΕ χρησιμοποιείται το **πρωτόκολλο χειραψίας**, με τα σήματα χειραψίας (handshaking), που είναι τα σήματα της απαίτησης (request) και της αναγνώρισης (acknowledge). Στην περίπτωση θα πρέπει, εκτός από την καθυστέρηση των συσκευών και της διατησίας, να λάβουμε **υπόψη και την καθυστέρηση λόγω του πρωτοκόλλου χειραψίας**. Οι χρόνοι των συσκευών **δεν** στρογγυλοποιούνται προς τα πάνω, διότι δεν υπάρχει ρολόι. Ο χρόνος μεταφοράς = 0.20 x (12 + 10 + 5) nsec + 0.30 x (19 + 10 + 5)nsec + 0.40 x (28 + 10 + 5)nsec + 0.10 x (62 + 10 + 5) nsec = 40.5 nsec. Επομένως ο **μέσος ρυθμός μεταφοράς πληροφορίας** (εντολών/δεδομένων) = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{8 \text{ bytes}}{40.5 \text{ nsec}}$.

Θέμα προόδου Ιανουαρίου 2022

Θεωρήστε μια αρτηρία που χρησιμοποιείται για τη μεταφορά πληροφορίας 6 ψηφιολέξεων κάθε φορά από μια μονάδα εισόδου (υπηρέτης - slave) στον κύριο (master). Θεωρήστε ότι έχουμε τέσσερις διαφορετικές μονάδες εισόδου Α, Β, Γ και Δ συνδεδεμένες στην αρτηρία. Εάν στην αρτηρία ήταν συνδεδεμένη μόνο μια μονάδα εισόδου, η μονάδα Α ή Β ή Γ ή Δ τότε η μεταφορά πληροφορίας από τη μονάδα εισόδου στον κύριο θα απαιτούσε 37, 25, 27 και 55 nsec αντίστοιχα. Ο κύριος ζητάει δεδομένα στο 8% των περιπτώσεων από την μονάδα εισόδου Α, στο 30% των περιπτώσεων από την μονάδα Β, στο 17% των περιπτώσεων από την μονάδα Γ και στο 44% των περιπτώσεων από την μονάδα Δ. Θεωρήστε επίσης ότι η διαιτησία βάζει καθυστέρηση 5 nsec. Να υπολογίσετε το **μέσο ρυθμό μεταφοράς πληροφορίας σε Mbytes ανά δευτερόλεπτο** εάν η αρτηρία είναι **σύγχρονη**, **έχει τη δυνατότητα εισαγωγής κύκλων ρολογιού καθυστέρησης** (wait states) και η περίοδος του σήματος χρονισμού (ρολογιού) είναι 5nsec. Προσοχή, να δώσετε μόνο το ακέραιο μέρος του αποτελέσματος. π.χ. εάν το αποτέλεσμα ήταν 158,99 Mbytes ανά δευτερόλεπτο να απαντήσετε δίνοντας τον καθαρό αριθμό 158.

Λύση

Όταν η αρτηρία είναι **σύγχρονη** και χρησιμοποιούνται **κύκλοι ρολογιού καθυστέρησης (wait states)**, τότε για τον υπολογισμό του χρόνου μεταφοράς λαμβάνουμε υπόψη **την καθυστέρηση κάθε συσκευής με το αντίστοιχο ποσοστό της και την καθυστέρηση λόγω διαιτησίας**. Επομένως. χρόνος μεταφοράς = 0.08 x 40 nsec + 0.30 x 25 nsec + 0.17 x 30 nsec + 0.44 x 55 nsec + 5 nsec = 45 nsec. Οι χρόνοι των συσκευών που δεν είναι ακέραια πολλαπλάσια της διάρκειας του κύκλου ρολογιού, στρογγυλοποιούνται προς τα πάνω, προκειμένου να γίνουν. Ο **μέσος ρυθμός μεταφοράς πληροφορίας** (εντολών/δεδομένων) = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{6 \text{ bytes}}{45 \text{ nsec}}$ = 0.133 bytes/nsec = 0.133 bytes x 10^9 bytes/sec = 133.33 x 10^6 bytes/sec = 133.33 MB/sec → 133 MB/sec.

Θέμα προόδου Ιανουαρίου 2022

Θεωρήστε μια αρτηρία που χρησιμοποιείται για τη μεταφορά πληροφορίας 6 ψηφιολέξεων κάθε φορά από μια μονάδα εισόδου (υπηρέτης - slave) στον κύριο (master). Θεωρήστε ότι έχουμε τέσσερις διαφορετικές μονάδες εισόδου Α, Β, Γ και Δ συνδεδεμένες στην αρτηρία. Εάν στην αρτηρία ήταν συνδεδεμένη μόνο μια μονάδα εισόδου, η μονάδα Α ή Β ή Γ ή Δ τότε η μεταφορά πληροφορίας από τη μονάδα εισόδου στον κύριο θα απαιτούσε 37, 25, 27 και 55 nsec αντίστοιχα. Ο κύριος ζητάει δεδομένα στο 8% των περιπτώσεων από την μονάδα εισόδου Α, στο 30% των περιπτώσεων από την μονάδα Β, στο 17% των περιπτώσεων από την μονάδα Γ και στο 44% των περιπτώσεων από την μονάδα Δ. Θεωρήστε επίσης ότι η διαιτησία βάζει καθυστέρηση 5 nsec. Να υπολογίσετε το μέσο ρυθμό μεταφοράς πληροφορίας σε Mbytes ανά δευτερόλεπτο εάν η αρτηρία είναι **σύγχρονη**, **δεν έχει τη δυνατότητα εισαγωγής κύκλων ρολογιού καθυστέρησης** (wait states) και η περίοδος του σήματος χρονισμού (ρολογιού) είναι 5nsec. Προσοχή, να δώσετε μόνο το ακέραιο μέρος του αποτελέσματος. π.χ. εάν το αποτέλεσμα ήταν 158,99 Mbytes ανά δευτερόλεπτο να απαντήσετε δίνοντας τον καθαρό αριθμό 158.

Λύση

Η αρτηρία είναι **σύγχρονη** και **δεν** χρησιμοποιούνται κύκλοι ρολογιού καθυστέρησης (wait states ή wait cycles) οπότε για τον υπολογισμό του **χρόνου μεταφοράς λαμβάνουμε υπόψη την πιο αργή συσκευή και την καθυστέρηση λόγω διατησίας**, δηλ. χρόνος μεταφοράς = 55 nsec + 5 nsec = 60 nsec. Επομένως ο **μέσος ρυθμός μεταφοράς πληροφορίας** (εντολών/δεδομένων) = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{6 \text{ bytes}}{60 \text{ nsec}}$ = = 0.1 x 10^3 MB/sec = 100 MB/sec

Θέμα προόδου Ιανουαρίου 2022

Θεωρήστε μια αρτηρία που χρησιμοποιείται για τη μεταφορά πληροφορίας 7 ψηφιολέξεων κάθε φορά από μια μονάδα εισόδου (υπηρέτης - slave) στον κύριο (master). Θεωρήστε ότι έχουμε τέσσερις διαφορετικές μονάδες εισόδου Α, Β, Γ και Δ συνδεδεμένες στην αρτηρία. Εάν στην αρτηρία ήταν συνδεδεμένη μόνο μια μονάδα εισόδου, η μονάδα Α ή Β ή Γ ή Δ τότε η μεταφορά πληροφορίας από τη μονάδα εισόδου στον κύριο θα απαιτούσε 37, 18, 27 και 21 nsec αντίστοιχα. Ο κύριος ζητάει δεδομένα στο 20% των περιπτώσεων από την μονάδα εισόδου Α, στο 36% των περιπτώσεων από την μονάδα Β, στο 27% των περιπτώσεων από την μονάδα Γ και στο 17% των περιπτώσεων από την μονάδα Δ. Θεωρήστε επίσης ότι η διαιτησία βάζει καθυστέρηση 10 nsec. Να υπολογίσετε το μέσο ρυθμό μεταφοράς πληροφορίας σε Mbytes ανά δευτερόλεπτο εάν η αρτηρία είναι **σύγχρονη**, **δεν έχει τη δυνατότητα εισαγωγής κύκλων ρολογιού καθυστέρησης** (wait states) και η περίοδος του σήματος χρονισμού (ρολογιού) είναι 10nsec. Προσοχή, να δώσετε μόνο το ακέραιο μέρος του αποτελέσματος. π.χ. εάν το αποτέλεσμα ήταν 158,99 Mbytes ανά δευτερόλεπτο να απαντήσετε δίνοντας τον καθαρό αριθμό 158.

Λύση

Η αρτηρία είναι **σύγχρονη** και **δεν** χρησιμοποιούνται κύκλοι ρολογιού καθυστέρησης (wait states ή wait cycles) οπότε για τον υπολογισμό του **χρόνου μεταφοράς λαμβάνουμε υπόψη την πιο αργή συσκευή (που έχει καθυστέρηση 37 nsec και επειδή το μέγεθος αυτό δεν είναι ακέραιο πολλαπλάσιο της διάρκειας του κύκλου ρολογιού θα πρέπει με στρογγυλοποίηση προς τα πάνω να το κάνουμε) και την καθυστέρηση λόγω διατησίας**, δηλ. χρόνος μεταφοράς = 40 nsec + 10 nsec = 50 nsec. Επομένως ο **μέσος ρυθμός μεταφοράς πληροφορίας** (εντολών/δεδομένων) = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{7 \text{ bytes}}{50 \text{ nsec}}$ = = 0.14 x 10^3 MB/sec = 140 MB/sec

Άσκηση 6.3 βιβλίου με polling/interrupts

Θεωρήστε ένα υπολογιστή με επεξεργαστή συχνότητας λειτουργίας 1 GHz που, εκτός των άλλων, χρησιμοποιείται για να ελέγχει τη λειτουργία 1.000 μονάδων ενός εργοστασίου. Κατά μέσο όρο οι μονάδες εισ/εξ απαιτούν συνολικά εξυπηρέτηση

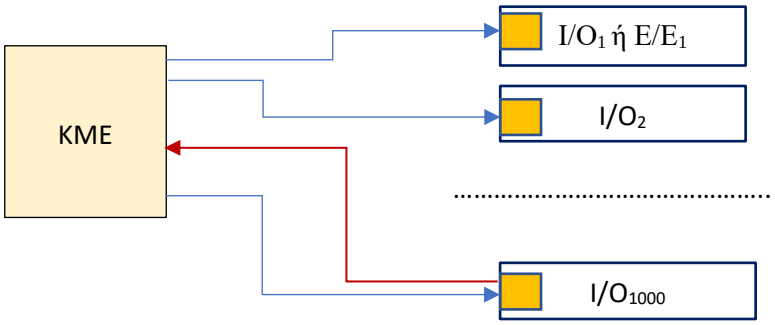
500 φορές το λεπτό. Να υπολογίσετε το ποσοστό του χρόνου του υπολογιστή που δαπανάται για τον έλεγχο των μονάδων εισ/εξ σε κάθε μια από τις ακόλουθες δύο περιπτώσεις.

α. Ο έλεγχος των μονάδων εισ/εξ βασίζεται στη χρονοπρογραμματισμένη διαδικασία (polling). Για να εξασφαλιστεί η ποιότητα των παραγόμενων προϊόντων θα πρέπει να ελέγχεται η κατάσταση κάθε μονάδας εισ/εξ 10 φορές το δευτερόλεπτο. Υποθέστε ότι για τον έλεγχο της κατάστασης μιας μονάδας εισ/εξ απαιτούνται 500 κύκλοι ρολογιού ενώ στις περιπτώσεις που απαιτείται εξυπηρέτηση, απαιτούνται και άλλοι 500 κύκλοι ρολογιού ανά εξυπηρέτηση.

β. Οι μονάδες εισ/εξ έχουν την δυνατότητα αποστολής και η ΚΜΕ τη δυνατότητα εξυπηρέτησης σημάτων διακοπής. Το κόστος διακοπής της λειτουργίας της ΚΜΕ για την εξυπηρέτηση ενός σήματος διακοπής και η εξυπηρέτηση της διακοπής είναι 1.300 κύκλοι ρολογιού.

γ. Να συγκρίνετε και να δικαιολογήσετε τα αποτελέσματα των περιπτώσεων α και β.

Λύση



Αρχικά θα υπολογίσουμε τη διάρκεια του κύκλου ρολογιού, δηλ. το tcycle = $\frac{1}{1\ \text{GHz}} = \frac{1}{10^9 \text{Hz}} = 10^{-9}$sec = 1 nsec.

α. Στην περίπτωση **της χρονοπρογραμματισμένης διαδικασίας (polling)**, θα πρέπει να λάβουμε υπόψη δύο χρόνους: έναν που απαιτείται για τον **έλεγχο** κάθε συσκευής (μονάδας) Ε/Ε και έναν που απαιτείται για την **εξυπηρέτηση** μόνο εκείνης της μονάδας που χρειάζεται εξυπηρέτηση. Στην Εικόνα, η ΚΜΕ ελέγχει περιοδικά τον καταχωρητή κατάστασης κάθε μονάδας Ε/Ε, για να διαπιστώσει ποια από αυτές θέλει εξυπηρέτηση, και εξυπηρετεί μια από αυτές. Επομένως στην περίπτωση αυτή απαιτείται, ο υπολογισμός δύο χρόνων.

Ο **πρώτος χρόνος** που απαιτείται για τον **έλεγχο κάθε μονάδας Ε/Ε**, υπολογίζεται από τον τύπο: 10 φορές x 500 κ.ρ. x 1 nsec x **1000** = 10 x 500 κ.ρ. x 10^{-9} sec x 10^3 = 10 $\frac{\text{έλεγχοι}}{\text{sec}}$ x 5 x 10^2 x 10^{-9} x 10^3 = 5 x 10^{-3} sec = 5 msec.

Ο **δεύτερος χρόνος** που απαιτείται για την **εξυπηρέτηση μόνο εκείνης** της μονάδας Ε/Ε που χρειάζεται εξυπηρέτηση, υπολογίζεται από τον τύπο: 500 $\frac{\text{φορές}}{\text{min}}$ x 500 κ.ρ. x 1 nsec x **1** = $\frac{250000}{60\ \text{sec}}$ x 10^{-9} sec = $\frac{25}{6}$ x 10^{-6} sec = 4.166 μsec. Επομένως, ο **συνολικός χρόνος που απαιτείται για το polling είναι το άθροισμα των δύο προηγούμενων χρόνων**, δηλ. 5 msec + 4.166 μsec = 5 x 10^{-3} sec + 4.166 x 10^{-6} sec = 0.005 sec + 0.000004166 sec = 0.005004166 sec → μετατροπή σε ποστό **0.5004166%. Επομένως, η ΚΜΕ δαπανά το 0.5% του χρόνου της για τον έλεγχο και την εξυπηρέτηση των μονάδων Ε/Ε.**

b. Στην περίπτωση χρήσης σημάτων διακοπής (Interrupts) έχουμε **μόνο χρόνο εξυπηρέτησης** 500 $\frac{\text{φορές}}{\text{min}}$ x 1300 κ.ρ. x 1 nsec x **1** = $\frac{500}{60\ \text{sec}}$ x 1300 κ.ρ x 10^{-9} sec = $\frac{65}{6}$ x 10^{-6} sec = 0.000018083 → μετατροπή σε ποστό **0.0018083%. Επομένως σε αυτή την περίπτωση η ΚΜΕ δαπανά το 0.001% του χρόνου της για την εξυπηρέτηση των μονάδων Ε/Ε.**

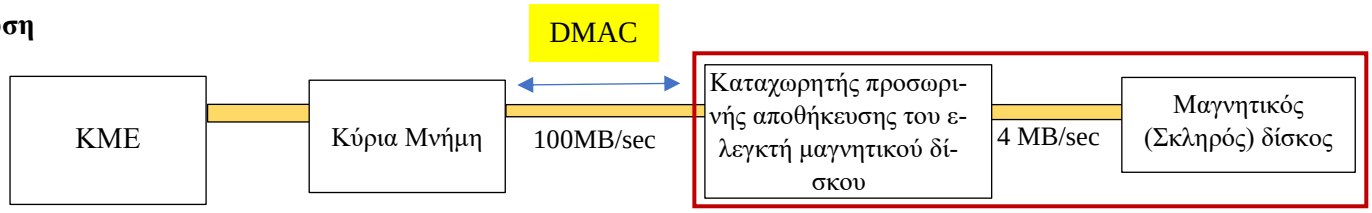
Παρατηρούμε ότι στη δεύτερη περίπτωση το ποσοστό απασχόλησης της ΚΜΕ είναι μικρότερο, γεγονός αναμενόμενο αφού στην περίπτωση επικοινωνίας της ΚΜΕ με συσκευές Ε/Ε το σημαντικό είναι ότι δεν χρειάζεται έλεγχος εκ' μέρους της ΚΜΕ για το ποια συσκευή θέλει εξυπηρέτηση.

Άσκηση 6.4 βιβλίου με ΑΠΜ (DMA)

Θεωρήστε ένα πρόγραμμα κατά την εκτέλεση του οποίου εκτελούνται 5.000.100 εντολές, οι 100 εκ των οποίων είναι εντολές εισόδου/εξόδου δεδομένων από το μαγνητικό δίσκο και είναι διάσπαρτες μέσα στο πρόγραμμα. Ο μέσος αριθμός κύκλων που απαιτούνται για την εκτέλεση κάθε εντολής που **δεν** είναι εντολή εισόδου/εξόδου είναι 6. Κατά την εκτέλεση μίας εντολής εισόδου/εξόδου μεταφέρονται από το μαγνητικό δίσκο στη κύρια μνήμη 4Κψηφιολέξεις (Kbytes). Έστω ότι το πρόγραμμα εκτελείται σε υπολογιστή που χρησιμοποιεί την τεχνική Άμεσης Προσπέλασης Μνήμης, ΑΠΜ, (Direct Memory Access, DMA). Η διαδικασία αρχικοποίησης του μηχανισμού Άμεσης Προσπέλασης Μνήμης και η ενημέρωση μετά την ολοκλήρωση της μεταφοράς κοστίζουν 2000 κύκλους ρολογιού. Η μεταφορά πληροφορίας μεταξύ της συσκευής του μαγνητικού δίσκου και του καταχωρητή προσωρινής αποθήκευσης του ελεγκτή της συσκευής μαγνητικού δίσκου γίνεται με ρυθμό 4Μψηφιολέξεις/δευτερόλεπτο (έχει συμπεριληφθεί και ο χρόνος προσπέλασης), ενώ η μεταφορά της πληροφορίας μεταξύ του καταχωρητή προσωρινής αποθήκευσης του ελεγκτή της συσκευής μαγνητικού δίσκου και της κύριας μνήμης γίνεται με ρυθμό 100Μψηφιολέξεις/δευτερόλεπτο και η συχνότητα λειτουργίας της ΚΜΕ είναι 1 GHz. Θεωρήστε ότι το πρόγραμμα εκτελείται μόνο του στον εν λόγω Η/Υ και υπολογίστε το μέγιστο πιθανό συνολικό χρόνο ολοκλήρωσής του.

Προσοχή: Η ποσότητα της πληροφορίας μετριέται σε δυνάμεις του 2 ενώ ο ρυθμός μεταφοράς της πληροφορίας σε δυνάμεις του 10.

Λύση



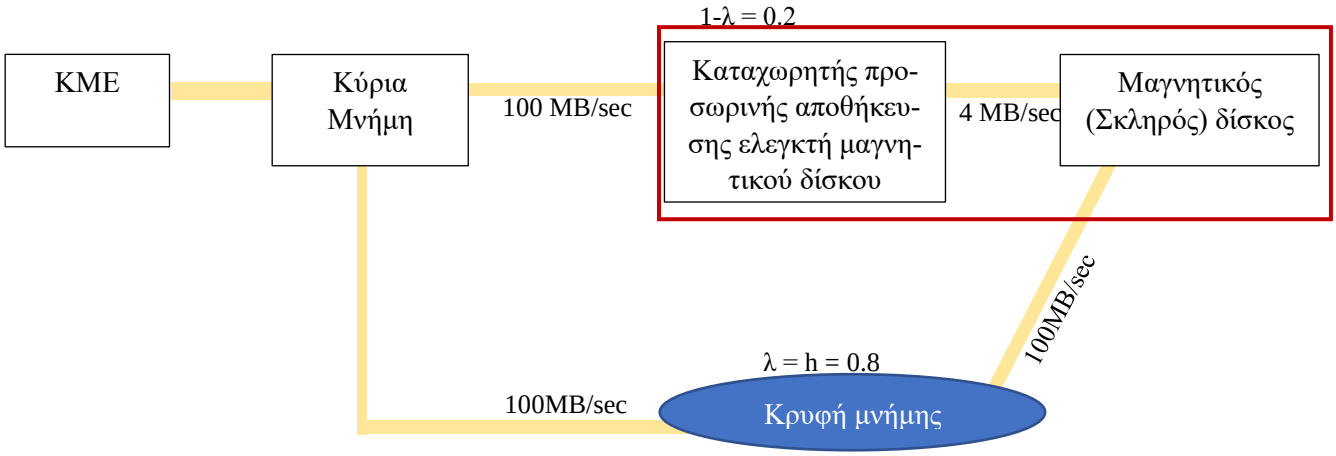
Από τη συχνότητα λειτουργίας του επεξεργαστή θα υπολογίσουμε τη διάρκεια του κύκλου ρολογιού tcycle = $\frac{1}{\text{συχνότητα λειτουργίας}} = \frac{1}{1\text{ GHz}}$ = 1 nsec. Έχουμε συνολικά **5.000.100 εντολές**. Από αυτές, τα 5.000.000 εντολές που **δεν** είναι εισόδου/εξόδου, απαιτούν κατά μέσο όρο 6 κ.ρ. για την εκτέλεσή τους. Άρα συνολικά έχουμε 5.000.000 εντολές x $\frac{6\text{ κ.ρ.}}{\text{εντολή}}$ x $1\frac{\text{nsec.}}{\text{κ.ρ.}}$ = 30 x 10^6 x 10^{-9} sec = 30 x 10^{-3} sec = 30 msec. Για τις **υπόλοιπες 100 εντολές που είναι εισόδου/εξόδου**, θα χρησιμοποιηθεί η τεχνική ΑΠΜ. Για **κάθε μια** από αυτές τις 100 εντολές, θα απαιτηθούν τρείς χρόνοι: t_1 + t_2 + t_3, όπου t_1 = χρόνος ΑΠΜ, t_2 = χρόνος μεταφοράς από το σκληρό δίσκο στον καταχωρητή προσωρινής αποθήκευσης του ελεγκτή του δίσκου και t_3 = χρόνος μεταφοράς από τον καταχωρητή προσωρινής αποθήκευσης του ελεγκτή του δίσκου στην κύρια μνήμη. Ο πρώτος χρόνος t_1 =2000 κ.ρ. x $1\frac{\text{nsec.}}{\text{κ.ρ.}}$ = 2 x 10^{-6} sec = 2 μsec. Αναφορικά με τον υπολογισμό του χρόνου t_2 έχουμε: ρυθμός μεταφοράς δεδομένων = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς δεδομένων}}$ και λύνουμε ως προς το **χρόνο μεταφοράς μεταξύ δίσκου και καταχωρητή προσωρινής αποθήκευσης και έχουμε: χρόνος μεταφοράς** = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{ρυθμός μεταφοράς δεδομένων}} = \frac{4\text{ KB}}{4\text{ MB/sec}}$ = $\frac{2^{10}\text{bytes}}{10^6\text{ bytes/sec}} = \frac{2^{10}\text{ sec}}{10^6}$ = 1024 x 10^{-6} sec = 1024 μsec. Αναφορικά με τον υπολογισμό του χρόνου t_3 έχουμε ότι: **χρόνος μεταφοράς μεταξύ καταχωρητή προσωρινής αποθήκευσης και κύριας μνήμης** = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{ρυθμός μεταφοράς δεδομένων}}$ = $\frac{4\text{ KB}}{100\text{ MB/sec}} = \frac{4\text{ x }2^{10}\text{bytes}}{100\text{ x }10^6\text{ bytes/sec}} = \frac{4\text{ x }2^{10}\text{ sec}}{10^8}$ = 0.00004096 sec = 40.96 μsec. Ο χρόνος για τις **100 εντολές εισόδου/εξόδου είναι:** (2 μsec + 1024 μsec + 40.96 μsec) x 100 = 1066.96 x 10^{-6} x 10^2 sec = 1066.96 x 10^{-4} sec = 106.696 msec. Επομένως, ο **συνολικός χρόνος που απαιτείται για τις 5.000.100 εντολές είναι: 30 msec + 106.696 msec = 136.696 msec.**

Παρατήρηση: διευκρινίζεται ότι στον υπολογισμό των χρόνων t_2 και t_3, στο ρυθμό μεταφοράς πληροφορίας χρησιμοποιούμε δυνάμεις του «10». Στην ποσότητα μεταφερόμενης πληροφορίας, χρησιμοποιούμε κανονικά δυνάμεις του «2».

Άσκηση 6.5 βιβλίου με DMA

Θεωρήστε ότι στο σύστημα που περιγράφεται στην άσκηση 6.4 η μονάδα μαγνητικού δίσκου αντικαταστάθηκε με άλλη πιο σύγχρονη που διαθέτει και κρυφή μνήμη δίσκου (τώρα η κρυφή μνήμη δίσκου παίζει το ρόλο του καταχωρητή προσωρινής αποθήκευσης του ελεγκτή της συσκευής μαγνητικού δίσκου). Εάν τα υπόλοιπα χαρακτηριστικά της μονάδας του μαγνητικού δίσκου που αναφέρονται στην άσκηση 6.4 παραμένουν τα ίδια και η πιθανότητα η ζητούμενη πληροφορία από τον μαγνητικό δίσκο να βρίσκεται στην κρυφή μνήμη του είναι 0.80 (ρυθμός επιτυχίας, hit ratio) να υπολογίσετε και πάλι το μέγιστο πιθανό χρόνο εκτέλεσης του προγράμματος της άσκησης 6.4.

Λύση



Θεωρούμε ότι η κρυφή μνήμη με πιθανότητα (λόγο) επιτυχίας 0.8 έχει το γρήγορο ρυθμό μεταφοράς δεδομένων των 100 MB/sec **σε όλο το εύρος της**, και το προηγούμενο μονοπάτι -χωρίς την κρυφή μνήμη- με πιθανότητα αποτυχίας 0.2 συνδυάζει τον αργό ρυθμό μεταφοράς δεδομένων που είναι των 4 MB/sec και το γρήγορο ρυθμό μεταφοράς δεδομένων των 100 MB/sec. Ισχύει ότι η διάρκεια του κύκλου ρολογιού tcycle = $\frac{1}{\text{συχνότητα λειτουργίας}} = \frac{1}{1\ \text{GHz}}$ = 1 nsec. Επίσης ισχύει ότι για τις 5.000.000 εντολές που δεν είναι εισόδου/εξόδου δεν αλλάζει ο χρόνος, οπότε έχουμε: 5000000 x $\frac{6\ \text{κ.ρ.}}{\text{εντολή}}$ x 1 $\frac{\text{nsec.}}{\text{κ.ρ.}}$ = 30 x 10^6 x 10^{-9} sec = 30 x 10^{-3} sec = 30 msec.

Για τις **υπόλοιπες 100 εντολές που είναι εισόδου/εξόδου**, θα χρησιμοποιηθεί η τεχνική ΑΠΜ και ο χρόνος που απαιτείται για την εκτέλεσή τους είναι: T = 100 * [**0.8** * $\frac{4\ \text{KB}}{100\ \text{MB/sec}}$ + **0.2** * ($\frac{4\ \text{KB}}{4\ \text{MB/sec}} + \frac{4\ \text{KB}}{100\ \text{MB/sec}}$)] = 100 * [0.8 * $\frac{4 \text{ x } 2^{10}\text{bytes}}{100 \text{ x } 10^6\ \text{bytes/sec}}$ + 0.2 * ($\frac{4 \text{ x } 2^{10}\text{bytes}}{4 \text{ x } 10^6\ \text{bytes/sec}} + \frac{4 \text{ x } 2^{10}\text{bytes}}{100 \text{ x } 10^6\ \text{bytes/sec}}$)] = 100 * [0.8 * 0.00004096 sec + 0.2 * (0.001024 sec + 0.00004096 sec)] = 100 * [0.000032768 + 0.00021299]sec =0.024576 sec = 24.576 msec. Επομένως, ο **συνολικός χρόνος που απαιτείται για τις 5.100.000 εντολές είναι: 30 msec + 24.576 msec = 54.576 msec.**

Παρατήρηση: διευκρινίζεται ότι στον υπολογισμό των χρόνων t_2 και t_3, μόνο στο ρυθμό μεταφοράς πληροφορίας που είναι στον παρανομαστή, χρησιμοποιούμε δυνάμεις του «10». Στον αριθμητή, που έχουμε ποσότητα μεταφερόμενης πληροφορίας, χρησιμοποιούμε κανονικά δυνάμεις του «2».

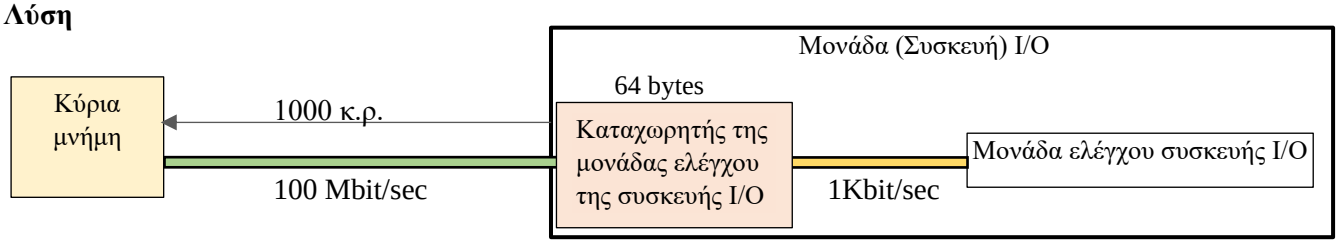
Άσκηση με προγραμματισμένη είσοδο – έξοδο με χρήση διακοπών

Μία συσκευή εισόδου/εξόδου (I/O) μεταφέρει δεδομένα στον καταχωρητή της μονάδας ελέγχου της με ρυθμό 1 Kbit/sec. Ο καταχωρητής είναι χωρητικότητας 64 ψηφιολέξεων (bytes). Όταν ο καταχωρητής γεμίσει, η μονάδα ελέγχου της συσκευής I/O στέλνει σήμα διακοπής (interrupt) για να γίνει η μεταφορά του περιεχομένου του καταχωρητή στην κύρια μνήμη (προγραμματισμένη διαδικασία I/O με χρήση σήματος διακοπής). Ο ρυθμός μεταφοράς από τον καταχωρητή του ελεγκτή της συσκευής I/O στην κύρια μνήμη είναι 100 Mbit/sec. Το κόστος έναρξης και τερματισμού της διαδικασίας εξυπηρέτησης του

σήματος διακοπής είναι 1000 κύκλοι ρολογιού. Θεωρήστε 1 κύκλος = 10 nsec. Τι ποσοστό του χρόνου ο επεξεργαστής χρησιμοποιείται για τη μεταφορά πληροφορίας από τον ελεγκτή της συσκευής I/O στην κύρια μνήμη;

Προσοχή: Στο ρυθμό μεταφοράς πληροφορίας χρησιμοποιούμε δυνάμεις του 10.

Λύση



Ισχύει ο τύπος: ρυθμός μεταφοράς δεδομένων = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}}$ $\Rightarrow$ χρόνος μεταφοράς = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{ρυθμός μεταφοράς δεδομένων}}$. Επειδή υπάρχουν **δύο διαφορετικοί ρυθμοί μεταφοράς δεδομένων**, θα χρησιμοποιήσουμε τον καθένα από αυτούς για να υπολογίσουμε τον αντίστοιχο χρόνο μεταφοράς, τη μία φορά μεταξύ μονάδας ελέγχου συσκευής Ε/Ε και του αντίστοιχου καταχωρητή της και την άλλη φορά μεταξύ της μονάδας ελέγχου συσκευής Ε/Ε και κύριας μνήμης.

i) **Χρόνος μεταφοράς μεταξύ μονάδας ελέγχου συσκευής Ε/Ε και του αντίστοιχου καταχωρητή της** = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{ρυθμός μεταφοράς δεδομένων}} = \frac{64 \text{ bytes}}{1 \text{ Kbit/sec}} = \frac{64 \text{ x } 8 \text{ bits}}{10^3 \text{ bits/sec}} = \frac{512 \text{ sec}}{1000}$ = 0.512 sec. Σε αυτό το χρονικό διάστημα γεμίζει ο καταχωρητής της μονάδας ελέγχου της συσκευής I/O.

ii) **Χρόνος μεταφοράς μεταξύ καταχωρητή της μονάδας ελέγχου της συσκευής Ε/Ε και κύριας μνήμης** = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{ρυθμός μεταφοράς δεδομένων}} = \frac{64 \text{ bytes}}{100 \text{ Mbit/sec}} = \frac{64 \text{ x } 8 \text{ bits}}{100 \text{ x } 10^6 \text{ bits/sec}} = \frac{512 \text{ sec}}{10^8}$ = 5.12 x 10^{-6} sec = 5.12 μsec. Στο χρόνο αυτό θα πρέπει **να συμπεριλάβουμε και τους 1000 κ.ρ. που απαιτούνται για την έναρξη και τον τερματισμό των σημάτων διακοπής**, διότι η μεταφορά των δεδομένων στην περίπτωση αυτή απαιτεί και τη χρήση σημάτων διακοπής, οπότε έχουμε 1000 κ.ρ. x 10 nsec/κ.ρ. = 10 x 10^{-6} sec = 10 μsec. Άρα **συνολικά** έχουμε 5.12 x 10^{-6} sec + 10 x 10^{-6} sec = 15.12 x 10^{-6} sec. Για να υπολογίσουμε το **ποσοστό απασχόλησης της ΚΜΕ** (επεξεργαστή) για τη μεταφορά πληροφορίας από τον ελεγκτή της συσκευής Ε/Ε στην κύρια μνήμη, θα χρησιμοποιήσουμε μια απλή μέθοδο των τριών, ως εξής:

Κάθε 0.512 sec που γεμίζει ο καταχωρητής	έχουμε απασχόληση της ΚΜΕ κατά 15.12 x 10^{-6} sec
100	x;

x = 15.12 x 10^{-6} sec x 100/0.512 = 2.953 x 10^{-3} = 0.002953%. Αυτό το μέγεθος είναι ήδη εκφρασμένο σε ποσοστό.

Άσκηση με διαιτησία αρτηρίας

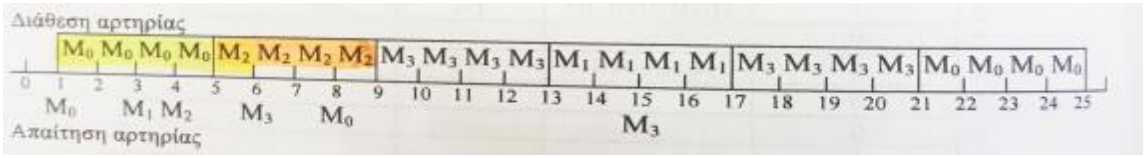
Θεωρείστε ότι σε σύγχρονη αρτηρία με συχνότητα 1GHz (περίοδο ρολογιού 1ns) είναι συνδεδεμένες 4 κύριες μονάδες (masters) M_0, M_1, M_2 και M_3. Θεωρήστε επίσης ότι χρησιμοποιείται η διαιτησία με χρήση αλυσίδας προτεραιότητας και ότι η μονάδα με μεγαλύτερο δείκτη έχει μεγαλύτερη προτεραιότητα από αυτή με μικρότερο δείκτη (π.χ. η M_2 έχει μεγαλύτερη προτεραιότητα από την M_1). Υποθέστε ότι όταν μία μονάδα παίρνει την κυριότητα της αρτηρίας την κρατάει για 4 περιόδους ρολογιού. Εάν η M_0 απαιτεί την αρτηρία τις χρονικές στιγμές 1ns και 8ns, η M_1 τη χρονική στιγμή 3ns, η M_2 τη χρονική στιγμή 4 ns και η M_3 τις χρονικές στιγμές 6ns και 15ns, να καταγράψετε τη σειρά με την οποία θα πάρει την κυριότητα της αρτηρίας η κάθε μονάδα. Να αιτιολογίσετε την απάντησή σας.

Λύση

Στη συνέχεια σχεδιάζουμε ένα διάγραμμα χρονισμού, στο οποίο φαίνεται η διάθεση της αρτηρίας στις μονάδες που τη ζητούν, όταν χρησιμοποιείται ως σχήμα διαιτησίας, η διαιτησία με **χρήση αλυσίδας προτεραιότητας**. Τη χρονική στιγμή

«1» η μόνη μονάδα που ζητά την αρτηρία είναι η M_0, οπότε και την παίρνει υπό την κυριότητά της και τη διατηρεί για 4 χρονικές περιόδους (κύκλους ρολογιού). Εν' τω μεταξύ τη χρονική στιγμή «3» ζητά την αρτηρία η μονάδα M_1 και τη χρονική στιγμή «4» ζητά την αρτηρία η μονάδα M_2. Φτάνοντας στη χρονική στιγμή «5», υπάρχουν σε εκκρεμότητα δύο απαιτήσεις διάθεσης αρτηρίας. Λόγω του υποστηριζόμενου μηχανισμού διαιτησίας, η μονάδα M_2 είναι αυτή που θα πάρει υπό την κυριοτητά της, την αρτηρία και θα τη διατηρήσει για 4 χρονικές περιόδους.

Εν' τω μεταξύ τη χρονική στιγμή «6» ζητά την αρτηρία η μονάδα M_3 και τη χρονική στιγμή «8» ζητά την αρτηρία η μονάδα M_0. Φτάνοντας στη χρονική στιγμή «9», υπάρχουν σε εκκρεμότητα τρείς απαιτήσεις διάθεσης αρτηρίας, από τις μονάδες M_1, M_3 και M_0. Λόγω του υποστηριζόμενου μηχανισμού διαιτησίας, η μονάδα M_3 που έχει τη μεγαλύτερη προτεραιότητα, είναι αυτή που θα πάρει υπό την κυριοτητά της την αρτηρία και θα τη διατηρήσει για 4 χρονικές περιόδους. Φτάνοντας στη χρονική στιγμή «13», υπάρχουν σε εκκρεμότητα δύο απαιτήσεις διάθεσης αρτηρίας, από τις μονάδες M_1 και M_0. Μεταξύ αυτών των δύο μονάδων, μεγαλύτερη προτεραιότητα έχει η M_1, η οποία θα πάρει υπό τον έλεγχό της την αρτηρία και θα τη διατηρήσει για 4 χρονικές περιόδους. Εν' τω μεταξύ τη χρονική στιγμή «15» ζητά την αρτηρία η μονάδα M_3. Φτάνοντας στη χρονική στιγμή «17», υπάρχουν σε εκκρεμότητα δύο απαιτήσεις διάθεσης αρτηρίας, από τις μονάδες M_0 και M_3. Λόγω του υποστηριζόμενου μηχανισμού διαιτησίας, η μονάδα M_3 που έχει τη μεγαλύτερη προτεραιότητα, είναι αυτή που θα πάρει υπό την κυριοτητά της την αρτηρία και θα τη διατηρήσει για 4 χρονικές περιόδους. Φτάνοντας στη χρονική στιγμή «21», υπάρχει σε εκκρεμότητα μόνο μια απαίτηση διάθεσης αρτηρίας, από τη μονάδα M_0. Αυτή θα πάρει υπό την κυριότητά της την αρτηρία και θα τη διατηρήσει για 4 χρονικές περιόδους, μέχρι τη χρονική στιγμή (περίοδο) «25».



Θέμα με σύγχρονη/ασύγχρονη αρτηρία - Ιανουάριος 2020

Θεωρείστε μια αρτηρία που χρησιμοποιείται για τη μεταφορά πληροφορίας 8 ψηφιολέξεων κάθε φορά από μια μονάδα εισόδου (υπηρέτης-slave) στον κύριο (master). Θεωρήστε ότι έχουμε τέσσερις διαφορετικές μονάδες εισόδου Α, Β, Γ και Δ. Η μεταφορά πληροφορίας στον κύριο από τη μονάδα εισόδου Α, Β, Γ και Δ απαιτεί 14, 18, 31 και 56 nsec αντίστοιχα. Θεωρήστε επίσης ότι η διαιτησία βάζει καθυστέρηση 5 nsec. Να υπολογίσετε το ρυθμό μεταφοράς πληροφορίας σε ψηφιολέξεις ανά δευτερόλεπτο **από τη μονάδα Β στον κύριο** για κάθε μια από τις ακόλουθες περιπτώσεις.

α. Η αρτηρία είναι σύγχρονη με περίοδο του σήματος χρονισμού (ρολογιού) 5 nsec

i. χωρίς δυνατότητα εισαγωγής κύκλων ρολογιού καθυστέρησης (wait states)

ii. με δυνατότητα εισαγωγής κύκλων ρολογιού καθυστέρησης

β. Η αρτηρία είναι ασύγχρονη και απαιτεί 8 nsec επιπλέον για κάθε μεταφορά λόγω του πρωτοκόλλου χειραψιάς.

Λύση

Το χαρακτηριστικό σε αυτήν την άσκηση είναι ότι λαμβάνουμε υπόψη μόνο τη μονάδα Β, γιατί αυτό αναφέρεται σχετικά στην εκφώνηση.

α) i) Ρυθμός μεταφοράς πληροφορίας = 8 bytes/(60 + 5) nsec = 8 bytes/65nsec = 8 bytes/65 x 10^{-9} sec = 0.123 x 10^9 bytes/sec

ii) Ρυθμός μεταφοράς πληροφορίας = 8 bytes/(20 + 5) nsec = 8 bytes /25 nsec = 0.32 x 10^9 bytes/sec, λαμβάνουμε υπόψη μόνο τη Β.

β) Ρυθμός μεταφοράς πληροφορίας = 8 bytes/(18 + 8 + 5) nsec = 8 bytes/31 nsec = 0.258 x 10^9 bytes/sec, λαμβάνουμε

Θέμα με σύγχρονη και ασύγχρονη αρτηρία

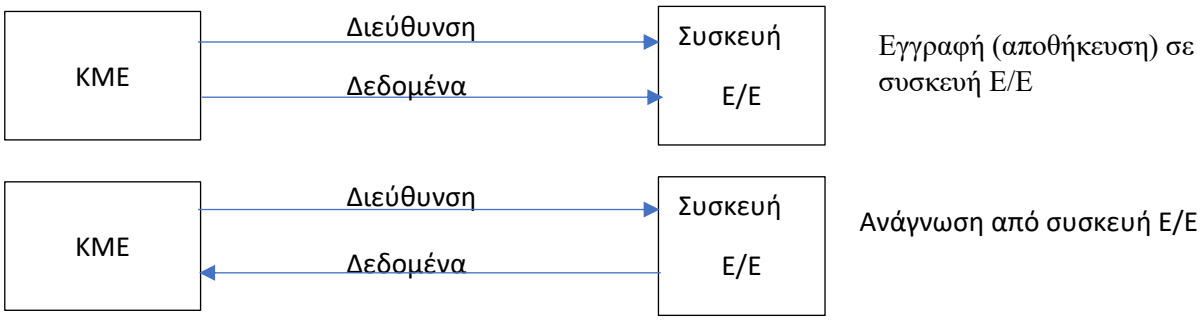
Υπολογίστε τον ρυθμό μεταφοράς δεδομένων μεταξύ κεντρικής μονάδας επεξεργασίας (ΚΜΕ) και μίας μονάδας εισόδου/εξόδου (εισ/εξ) κατά την ανάγνωση/εγγραφή δεδομένων από/στη μονάδα εισ/εξ σε κάθε ένα από τα παρακάτω συστήματα:

1) Σύστημα με σύγχρονη αρτηρία, που συνδέει τις δύο μονάδες, με κύκλο αρτηρίας 40 ns, καθυστέρηση ανάγνωσης/εγγραφής από/στη μονάδα εισ/εξ 200ns και ανάγνωση/εγγραφή μιας λέξης ανά πρόσβαση.
   α. Η αρτηρία διαθέτει διαφορετικές γραμμές δεδομένων και διευθύνσεων.
   β. Η αρτηρία διαθέτει πολυπλεγμένες γραμμές δεδομένων και διευθύνσεων.
2) Σύστημα με ασύγχρονη αρτηρία, που συνδέει τις δύο μονάδες, με χρόνο μεταφοράς σήματος κατά μήκος της αρτηρίας (άρα και χρόνο για την εκτέλεση κάθε βήματος του πρωτοκόλλου χειραψίας) 30ns, καθυστέρηση ανάγνωσης/εγγραφής από/στη μονάδα εισ/εξ 200ns και ανάγνωση/εγγραφή μιας λέξης ανά πρόσβαση.
   α. Η αρτηρία διαθέτει διαφορετικές γραμμές δεδομένων και διευθύνσεων.
   β. Η αρτηρία διαθέτει πολυπλεγμένες γραμμές δεδομένων και διευθύνσεων.

Θεωρήστε ότι το 50% των μεταφορών αφορούν ανάγνωση από τη μνήμη και το 50% εγγραφή σ'αυτή και 1 λέξη = 4 ψηφιολέξεις (bytes).

Λύση

1α. Αρτηρία σύγχρονη με ξεχωριστές γραμμές διευθύνσεων και δεδομένων

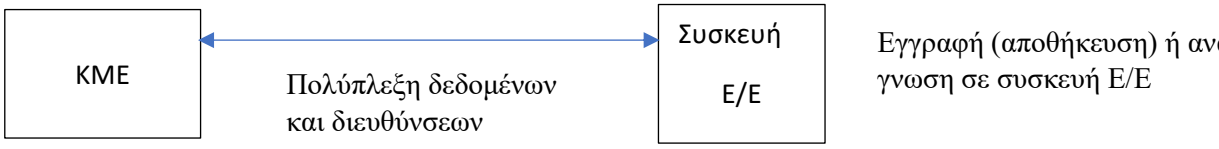


Ο ρυθμός μεταφοράς πληροφορίας (εντολών/δεδομένων) $= \frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} =$

$\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος εγγραφής} + \text{χρόνος ανάγνωσης}} = \frac{4 \text{ bytes}}{0.5 * (40 \text{ nsec} + 200 \text{ nsec}) + 0.5 * (40 \text{ nsec} + 200 \text{ nsec} + 40 \text{ nsec})} =$

Αναφορικά με το χρόνο εγγραφής, μεταφέρονται στον ίδιο χρόνο (40 nsec) τόσο η διεύθυνση της συσκευής Ε/Ε όπου θα γίνει η αποθήκευση, όσο και τα δεδομένα που θα αποθηκευτούν και η εγγραφή διαρκεί 200 nsec. Αναφορικά με το χρόνο ανάγνωσης, χρειάζονται τρεις χρόνοι: ένας για την αποστολή της διεύθυνσης της συσκευής Ε/Ε από όπου θα γίνει η ανάγνωση (40 nsec), ένας για την ανάγνωση (200 nsec) και ένας για την επιστροφή των δεδομένων που διαβάστηκαν πίσω στην ΚΜΕ (40 nsec).

1β. Πολυπλεγμένες γραμμές διευθύνσεων και δεδομένων



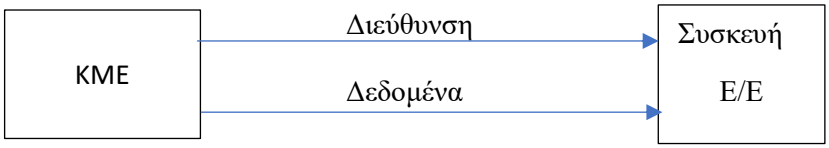
Στην περίπτωση χρησιμοποιείται **η ίδια η αρτηρία για να μεταφέρει τη μια φορά δεδομένα και την άλλη φορά διευθύνσεις**. Αυτό σημαίνει ότι διεύθυνση και δεδομένα **δεν** μπορούν τώρα να στέλνονται ταυτόχρονα. **Ο νέος ρυθμός**

μεταφοράς πληροφορίας (εντολών/δεδομένων) = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}}$ = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος εγγραφής+χρόνος ανάγνωσης}} = \frac{4 \text{ bytes}}{0.5*(40 \text{ nsec} + 40 \text{ nsec} + 200 \text{ nsec}) + 0.5*(40 \text{ nsec}+200 \text{ nsec}+40 \text{ nsec})} =$

Αναφορικά με το χρόνο εγγραφής, απαιτείται ένας χρόνος της τάξης των 40 nsec για να σταλθεί διεύθυνση της συσκευής Ε/Ε όπου θα γίνει η αποθήκευση, ένας δεύτερος χρόνος επίσης της τάξης των 40 nsec για την αποστολή των δεδομένων που θα αποθηκευτούν και ένας χρόνος της τάξης των 200 nsec που διαρκεί η εγγραφή. **Αναφορικά με το χρόνο ανάγνωσης, δεν αλλάζει κάτι σε σχέση με πριν**: δηλ. χρειάζονται τρεις χρόνοι: ένας για την αποστολή της διεύθυνσης της συσκευής Ε/Ε από όπου θα γίνει η ανάγνωση (40 nsec), ένας για την ανάγνωση (200 nsec) και ένας για την επιστροφή των δεδομένων που διαβάστηκαν πίσω στην ΚΜΕ (40 nsec).

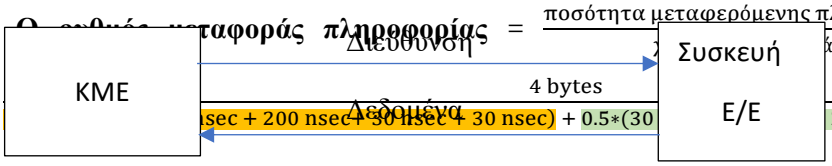
2α. Αρτηρία ασύγχρονη με ξεχωριστές γραμμές διευθύνσεων και δεδομένων

Τώρα **δεν** υπάρχει ρολόι και ισχύει το πρωτόκολλο με τα σήματα χειραψίας. Πιο συγκεκριμένα, χρειάζεται ένα ζεύγος σημάτων χειραψίας για το ξεκίνημα της διαδικασίας (είτε εγγραφής είτε ανάγνωσης) και επίσης ένα ζεύγος σημάτων χειραψίας για τον τερματισμό της διαδικασίας.



Εγγραφή (αποθήκευση) σε συσκευή Ε/Ε

Ο ρυθμός μεταφοράς πληροφορίας = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}}$ = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος εγγραφής+χρόνος ανάγνωσης}} =$



$\frac{4 \text{ bytes}}{\text{sec} + 200 \text{ nsec} + 30 \text{ nsec} + 30 \text{ nsec}) + 0.5*(30 \quad 200 \text{ nsec} + 30 \text{ nsec} + 30 \text{ nsec})} =$

Ανάγνωση συσκευή Ε/Ε

Στην περίπτωση της εγγραφής, έχουμε ένα ζεύγος χρόνων 30 nsec για τα σήματα χειραψίας που ξεκινούν τη διαδικασία και ένα αντίστοιχο ζεύγος χρόνων 30 nsec για τα σήματα χειραψίας που ολοκληρώνουν τη διαδικασία. Στην περίπτωση της ανάγνωσης, ισχύει ακριβώς ότι και στην εγγραφή.

2α. Αρτηρία ασύγχρονη με πολυπλεγμένες γραμμές διευθύνσεων και δεδομένων

Τώρα ισχύει και πάλι το πρωτόκολλο με τα σήματα χειραψίας και επίσης χρησιμοποιείται η ίδια η αρτηρία για να μεταφέρει τη μια φορά δεδομένα και την άλλη φορά διευθύνσεις. Αυτό σημαίνει ότι διεύθυνση και δεδομένα **δεν** μπορούν τώρα να στέλνονται ταυτόχρονα. Όπως και στην περίπτωση της σύγχρονης αρτηρίας, έτσι και εδώ, ο χρόνος ανάγνωσης **δεν** αλλάζει.

Ο ρυθμός μεταφοράς πληροφορίας = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}}$ = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος εγγραφής+χρόνος ανάγνωσης}} =$

$\frac{4 \text{ bytes}}{0.5*(30 \text{ nsec} + 30 \text{ nsec} + 30 \text{ nsec} + 30 \text{ nsec}+ 30 \text{ nsec} + 30 \text{ nsec}+ 200 \text{ nsec}+ 30 \text{ nsec} + 30 \text{ nsec}) + 0.5*(30 \text{ nsec} + 30 \text{ nsec} + 200 \text{ nsec} + 30 \text{ nsec} + 30 \text{ nsec})}$

Οι **πρώτες τέσσερις τιμές των 30 nsec** αφορούν ενεργοποίηση και απενεργοποίηση σημάτων χειραψίας για αποστολή της διεύθυνσης της συσκευής Ε/Ε όπου θα γίνει η αποθήκευση των δεδομένων και οι υπόλοιπες τιμές αφορούν ενεργοποίηση και απενεργοποίηση σημάτων χειραψίας για την αποθήκευση των δεδομένων.

Θέμα Φεβρουαρίου 2021

Υπολογιστικό σύστημα ειδικού σκοπού με καταχωρητές των 8 δυαδικών ψηφίων και αρτηρία δεδομένων των 8 δυαδικών ψηφίων χρησιμοποιεί την τεχνική του χρονοπρογραμματισμένου ελέγχου (polling) για να ελέγξει εάν θέλει εξυπηρέτηση κάποια από τις περιφερειακές συσκευές εισόδου και την τεχνική της προγραμματισμένης διαδικασίας εισόδου (programmed input/output) για τη μεταφορά του περιεχομένου των καταχωρητών δεδομένων του ελεγκτή της περιφερειακής συσκευής

εισόδου στην κύρια μνήμη. Ο ελεγκτής της περιφερειακής συσκευής έχει 2 καταχωρητές. Κάθε περιφερειακή συσκευή μεταφέρει σειριακά δεδομένα στους καταχωρητές του ελεγκτή της, με ρυθμό μεταφοράς ο οποίος δεν είναι σταθερός και η μέγιστη τιμή του είναι 4 δυαδικά ψηφία ανά sec. Θεωρήστε ότι όταν γεμίσουν οι καταχωρητές του ελεγκτή, θα πρέπει να αδειάσουν τελείως για να μπορέσει η περιφερειακή συσκευή να γράψει σ' αυτούς δεδομένα. Η μεταφορά της πληροφορίας από τους καταχωρητές του ελεγκτή της περιφερειακής συσκευής προς το υπολογιστικό σύστημα γίνεται μέσω αρτηρίας των 8 δυαδικών ψηφίων. Κάθε έλεγχος της κατάστασης του ελεγκτή της περιφερειακής συσκευής από την κεντρική μονάδα επεξεργασίας απαιτεί 0,006 sec, η εκτέλεση μιας εντολής ανάγνωσης ενός καταχωρητή δεδομένων του ελεγκτή της περιφερειακής συσκευής εισόδου/εξόδου απαιτεί 0,005 sec, η εκτέλεση μιας εντολής εγγραφής στην κύρια μνήμη απαιτεί 0,00009 sec. Τα ανωτέρω δεδομένα ισχύουν για όλες τις ερωτήσεις, ενώ τα στοιχεία που αναφέρονται σε κάθε ερώτηση ισχύουν μόνο σε αυτή την ερώτηση.

Α. Θεωρήστε ότι υπάρχουν 7 περιφερειακές μονάδες κάθε μια με τον ελεγκτή της και ότι στο 30% των ελέγχων απαιτείται εξυπηρέτηση (μεταφορά δεδομένων). Κάθε ελεγκτής ελέγχεται 5 φορές ανά sec. Να υπολογίσετε το ποσοστό του χρόνου που αφιερώνει η ΚΜΕ για τον έλεγχο και την εξυπηρέτηση των περιφερειακών συσκευών. Να γράψετε μόνο την τιμή με δύο ακέραια ψηφία, υποδιαστολή και δύο ψηφία δεξιά της υποδιαστολής, π.χ. 04.30. Για να περιορίσετε σε δύο το πλήθος των ψηφίων δεξιά της υποδιαστολής να εφαρμόσετε την τεχνική της περικοπής. [ ].

Β. Να υπολογίστε πόσες φορές ανά sec πρέπει να ελέγχεται η κατάσταση του ελεγκτή κάθε περιφερειακής συσκευής ώστε να μην χάνεται πληροφορία. Να γράψετε μόνο την τιμή με δύο ψηφία πχ. 04 ή 67. [ ].

Γ. Να υπολογίσετε το μέγιστο πλήθος περιφερειακών συσκευών (κάθε μία με τον ελεγκτή της) με τα ίδια ακριβώς χαρακτηριστικά λειτουργίας και χρήσης, που μπορούν να συνδεθούν στο υπολογιστικό σύστημα χωρίς να χάνεται πληροφορία. Να γράψετε διψήφιο αριθμό πχ. 02. [ ].

Λύση

α) **Αναφορικά με το polling έχουμε δύο χρόνους**: ελέγχου κάθε περιφερειακής συσκευής εξυπηρέτησης μόνο μίας. **Αναφορικά με το χρόνο ελέγχου, έχουμε: T1 = $\frac{5 \text{ φορές}}{\text{sec}}$ x 0.006 sec x 7** και αναφορικά με το χρόνο εξυπηρέτησης έχουμε: T2 = 0.3 x (0.005 sec + 0.0009 sec) x 1. Συνολικά έχουμε T1 + T2 = $\frac{5 \text{ φορές}}{\text{sec}}$ x 0.006 sec x 7 + 0.3 x (0.005 sec + 0.0009 sec) x 1 = 0.21177 sec. Αυτός είναι ο συνολικός χρόνος μεταφοράς των δεδομένων από τις 7 περιφερειακές συσκευές στην ΚΜΕ

Ισχύει ο τύπος: ρυθμός μεταφοράς δεδομένων από ελεγκτή περιφερειακής συσκευής στον καταχωρητή προσωρινής αποθήκευσης = $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}}$ $\Rightarrow$ **χρόνος μεταφοράς από ελεγκτή περιφερειακής συσκευής στον καταχωρητή προσωρινής αποθήκευσης** = $\frac{2 \text{ x } 8 \text{ bits}}{4 \text{ bits/sec}}$ = 4 sec.

Κάθε 4 sec που γεμίζει ο καταχωρητής έχουμε απασχόληση της ΚΜΕ 0.21177 sec

100 x;

5.29425% → 05.29

β) Έχουμε 7 περιφερειακές συσκευές και η καθεμία περιέχει τον ελεγκτή της, και έχουμε έλεγχο κάθε ελεγκτή 5 φορές/sec. Για να μην χάνεται πληροφορία θα πρέπει να πολλαπλασιάσουμε το ρυθμό ελέγχου με το πλήθος των ελεγκτών των περιφερειακών μονάδων, οπότε έχουμε ελεγκτή 5 φορές/sec x 7 = 35 φορές/sec.

γ)

Θέμα Ιουνίου 2020

Θεωρήστε ένα υπολογιστή με επεξεργαστή του οποίου η περίοδος του σήματος χρονισμού είναι 7 nsec, που, εκτός των άλλων, χρησιμοποιείται για να ελέγχει τη λειτουργία 1311 μονάδων ενός εργοστασίου. Για τον έλεγχο της κατάστασης μιας μονάδας εισ/εξ απαιτούνται 632 κύκλοι ρολογιού ενώ στις περιπτώσεις που απαιτείται εξυπηρέτηση, απαιτούνται και άλλοι 3160 κύκλοι ρολογιού ανά εξυπηρέτηση.

Οι μονάδες εισ/εξ έχουν μοιραστεί σε 19 ομάδες των 69 μονάδων εισ/εξ η κάθε μία. Οι μονάδες εισ/εξ έχουν τη δυνατότητα αποστολής και η ΚΜΕ τη δυνατότητα εξυπηρέτησης σημάτων διακοπής. Όλες οι μονάδες εισ/εξ μιας ομάδας στέλνουν σήμα διακοπής στην ΚΜΕ μέσω μιας κοινής γραμμής. Κάθε ομάδα χρησιμοποιεί διαφορετική γραμμή. Η ΚΜΕ προσδιορίζει την μονάδα εισ/εξ που απαιτεί εξυπηρέτηση διαβάζοντας την αντίστοιχη σημαία του καταχωρητή κατάστασης του ελεγκτή της μονάδας εισ/εξ.

Όλες οι ομάδες μονάδων εισ/εξ έχουν μεταξύ τους την ίδια προτεραιότητα, ενώ σε κάθε ομάδα οι μονάδες εισ/εξ είναι διατεταγμένες ανά προτεραιότητα. Η εξυπηρέτηση μιας μονάδας εισ/εξ δεν διακόπτεται για την εξυπηρέτηση μιας μονάδας εισ/εξ μεγαλύτερης προτεραιότητας.

Σημείωση: Το άθροισμα των ν πρώτων όρων μιας αριθμητικής προόδου, δίνεται από τη σχέση $\Sigma_v = [(\alpha_1+\alpha_v)v]/2$, όπου α_1 είναι ο πρώτος όρος και α_v ο τελευταίος.

Α. Να υπολογίσετε σε nsec το μικρότερο χρόνο που είναι δυνατόν να απαιτηθεί για την ολοκλήρωση της εξυπηρέτησης μιας μονάδας εισ/εξ. και να δώσετε μόνο την τιμή του χωρίς τη μονάδα μέτρησης και χωρίς τελείες ή κόμμα. . .

Β. Θεωρήστε ότι όταν ζήτησε εξυπηρέτηση μια μονάδα εισ/εξ δεν υπήρχε άλλη εξυπηρέτηση σε εξέλιξη.

Β.1. Να υπολογίσετε σε nsec το μεγαλύτερο χρονικό διάστημα που είναι δυνατόν να απαιτηθεί για την ολοκλήρωση της εξυπηρέτησης μιας μονάδας εισ/εξ. και να δώσετε μόνο την τιμή του χωρίς τη μονάδα μέτρησης και χωρίς τελείες ή κόμμα.

Β.2. Εάν όλες οι μονάδες εισ/εξ έχουν την ίδια πιθανότητα να ζητήσουν εξυπηρέτηση να υπολογίσετε σε nsec το μέσο χρόνο που απαιτείται για την ολοκλήρωση της εξυπηρέτησης μιας απαίτησης εισ/εξ. και να δώσετε μόνο την τιμή του χωρίς τη μονάδα μέτρησης και χωρίς τελείες ή κόμμα. .

Λύση

Α) 19 ομάδες x 7 nsec + 3160 κ.ρ. x 7 $\frac{nsec}{\kappa\rho}$ = 133 + 22120 = 22253 nsec. Αυτός είναι ο συνολικός χρόνος για την εξυπηρέτηση μιας μονάδας.

Β) 19 ομάδες x 7 nsec + 69 x 3160 κ.ρ. x 7$\frac{nsec}{\kappa\rho}$ = 1526413

Γ) $\Sigma_v = [(\alpha_1 + \alpha_v)v]/2 = [(1 + 19)19]/2 = 190$ → 190 x 69 x 7 = 91770 nsec

Γ) $\Sigma_v = [(\alpha_1 + \alpha_v)v]/2 = [(1 + 69)69]/2 = 2415$ → 2415 x 19 x 7 = 321195 nsec

Γ) $\Sigma_v = [(\alpha_1 + \alpha_v)v]/2 = [(1 + 1311)1311]/2 = 1720032$ → 1720032 x 7 = 6020112 nsec

Θέμα Σεπτεμβρίου 2020

Θεωρήστε ένα υπολογιστή με επεξεργαστή του οποίου η περίοδος του σήματος χρονισμού είναι 5 nsec, που, εκτός των άλλων, χρησιμοποιείται για να ελέγχει τη λειτουργία 864 μονάδων ενός εργοστασίου. Για τον έλεγχο της κατάστασης μιας μονάδας εισ/εξ απαιτούνται 612 κύκλοι ρολογιού ενώ στις περιπτώσεις που απαιτείται εξυπηρέτηση, απαιτούνται και άλλοι 1836 κύκλοι ρολογιού ανά εξυπηρέτηση.

Οι μονάδες εισ/εξ έχουν μοιραστεί σε 27 ομάδες των 32 μονάδων εισ/εξ η κάθε μία. Οι μονάδες εισ/εξ έχουν τη δυνατότητα αποστολής και η ΚΜΕ τη δυνατότητα εξυπηρέτησης σημάτων διακοπής. Όλες οι μονάδες εισ/εξ μιας ομάδας στέλνουν σήμα διακοπής στην ΚΜΕ μέσω μιας κοινής γραμμής. Κάθε ομάδα χρησιμοποιεί διαφορετική γραμμή. Η ΚΜΕ προσδιορίζει την μονάδα εισ/εξ που απαιτεί εξυπηρέτηση διαβάζοντας την αντίστοιχη σημαία του καταχωρητή κατάστασης του ελεγκτή της μονάδας εισ/εξ.

Όλες οι ομάδες μονάδων εισ/εξ έχουν μεταξύ τους την ίδια προτεραιότητα, ενώ σε κάθε ομάδα οι μονάδες εισ/εξ είναι διατεταγμένες ανά προτεραιότητα. Η εξυπηρέτηση μιας μονάδας εισ/εξ δεν διακόπτεται για την εξυπηρέτηση μιας μονάδας εισ/εξ μεγαλύτερης προτεραιότητας.

Σημείωση: Το άθροισμα των ν πρώτων όρων μιας αριθμητικής προόδου, δίνεται από τη σχέση $\Sigma_v = [(\alpha_1+\alpha_v)v]/2$, όπου α_1 είναι ο πρώτος όρος και α_v ο τελευταίος.

Α. Να υπολογίσετε σε nsec το μικρότερο χρόνο που είναι δυνατόν να απαιτηθεί για την ολοκλήρωση της εξυπηρέτησης μιας μονάδας εισ/εξ. και να δώσετε μόνο την τιμή του χωρίς τη μονάδα μέτρησης και χωρίς τελείες ή κόμμα. . .

Β. Θεωρήστε ότι όταν ζήτησε εξυπηρέτηση μια μονάδα εισ/εξ δεν υπήρχε άλλη εξυπηρέτηση σε εξέλιξη.

Β.1. Να υπολογίσετε σε nsec το μεγαλύτερο χρονικό διάστημα που είναι δυνατόν να απαιτηθεί για την ολοκλήρωση της εξυπηρέτησης μιας μονάδας εισ/εξ. και να δώσετε μόνο την τιμή του χωρίς τη μονάδα μέτρησης και χωρίς τελείες ή κόμμα.
Β.2. Εάν όλες οι μονάδες εισ/εξ έχουν την ίδια πιθανότητα να ζητήσουν εξυπηρέτηση να υπολογίσετε σε nsec το μέσο χρόνο που απαιτείται για την ολοκλήρωση της εξυπηρέτησης μιας απαίτησης εισ/εξ. και να δώσετε μόνο την τιμή του χωρίς τη μονάδα μέτρησης και χωρίς τελείες ή κόμμα. .

Λύση

Α. Να υπολογίσετε σε nsec το **μικρότερο χρόνο που είναι δυνατόν να απαιτηθεί για την ολοκλήρωση της εξυπηρέτησης μιας μονάδας εισ/εξ** και να δώσετε μόνο την τιμή του χωρίς τη μονάδα μέτρησης και χωρίς τελείες ή κόμμα

(612+1836)*5 nsec

Β1. Να υπολογίσετε σε nsec το μεγαλύτερο χρονικό διάστημα που είναι δυνατόν να απαιτηθεί για την ολοκλήρωση της εξυπηρέτησης μιας μονάδας εισ/εξ και να δώσετε μόνο την τιμή του χωρίς τη μονάδα μέτρησης χωρίς τελείες ή κόμμα

(31*612)*5nsec + (612+1836)*5nsec

Εναλλακτικά θα μπορούσαμε να το γράψουμε ως εξής: 32 * 612 κ.ρ. * 5 nsec + 1836 κ.ρ. * 5 nsec

Β2. Εάν όλες οι μονάδες εισ/εξ έχουν την ίδια πιθανότητα να ζητήσουν εξυπηρέτηση να υπολογίσετε σε nsec το μέσο χρόνο που απαιτείται για την ολοκλήρωση της εξυπηρέτησης μιας απαίτησης εισ/εξ και να δώσετε μόνο την τιμή του χωρίς τη μονάδα μέτρησης χωρίς τελείες ή κόμμα

B2)Ο Χρονος ειναι Σν=[(t1(Α ερωτημα)+t2(Β ερωτημα))*32]/2

Με επιφύλαξη:

A) Ο μικρότερος χρόνος που είναι δυνατόν να απαιτηθεί για εξυπηρέτηση μονάδας, είναι όταν εξυπηρετείται (αφού ελεγχθεί η κατάστασή της πρώτα) η πρώτη μονάδα. Ο χρόνος αυτός είναι:

t1 = (612κρ + 1836κρ) * 5nsec = 12.240nsec

B) Ο μεγαλύτερος χρόνος που είναι δυνατόν να απαιτηθεί για την εξυπηρέτηση μονάδας, είναι όταν εξυπηρετηθούν όλες οι μονάδες κατά σειρά. Οπότε είναι:

t2 = (612κρ + 1836κρ) * 5nsec * 864 μονάδες = 10.575.360 nsec

B2) Ο Χρονος ειναι Σν=[(t1(Α ερωτημα)+t2(Β ερωτημα))*32]/2

Και μετα Σν/32 για να βρουμε τον μεσο χρονο

Θέμα Ιουνίου 2016 με χρονοπρογραμματισμένη διαδικασία (polling)

Θεωρήστε ένα υπολογιστή με επεξεργαστή συχνότητας λειτουργίας 1 GHz που, εκτός των άλλων, χρησιμοποιείται για να ελέγχει τη λειτουργία 2.000 μονάδων ενός εργοστασίου. Κατά μέσο όρο όλες μαζί οι μονάδες εισ/εξ απαιτούν συνολικά εξυπηρέτηση 1.000 φορές το λεπτό. Να υπολογίσετε το ποσοστό του χρόνου του υπολογιστή που δαπανάται για τον έλεγχο των μονάδων εισ/εξ σε κάθε μια από τις ακόλουθες δύο περιπτώσεις.

α. Ο έλεγχος των μονάδων εισ/εξ βασίζεται στη χρονοπρογραμματισμένη διαδικασία (polling). Για να εξασφαλιστεί η ποιότητα των παραγόμενων προϊόντων θα πρέπει να ελέγχεται η κατάσταση κάθε μονάδας εισ/εξ 10 φορές το δευτερόλεπτο. Υποθέστε ότι για τον έλεγχο της κατάστασης μιας μονάδας εισ/εξ απαιτούνται 500 κύκλοι ρολογιού ενώ στις περιπτώσεις που απαιτείται εξυπηρέτηση, απαιτούνται και άλλοι 600 κύκλοι ρολογιού ανά εξυπηρέτηση.

β. Οι μονάδες εισ/εξ έχουν την δυνατότητα αποστολής και η ΚΜΕ τη δυνατότητα εξυπηρέτησης σημάτων διακοπής. Το κόστος διακοπής της λειτουργίας της ΚΜΕ για την εξυπηρέτηση ενός σήματος διακοπής και η εξυπηρέτηση της διακοπής είναι 1.200 κύκλοι ρολογιού.

Λύση

Αφού η συχνότητα είναι 1 GHz η διάρκεια του κύκλου ρολογιού είναι $1x10^{-9}$ δευτερόλεπτα = 1 nsec.

α. Αφού πρέπει να ελέγχεται η κατάσταση κάθε μονάδας εισ/εξ 10 φορές το δευτερόλεπτο και κάθε έλεγχος κοστίζει 500 κύκλους ρολογιού και έχουμε 2000 μονάδες εισόδου/εξόδου, η ΚΜΕ κάθε δευτερόλεπτο απασχολείται με τον έλεγχο των μονάδων εισ/εξ για χρόνο ίσο με: 10 x 500κ.ρ. x 10^{-9} sec x **2.000** = 10^{-2} sec.

Οι μονάδες εισ/εξ απαιτούν συνολικά εξυπηρέτηση 1000 φορές το λεπτό, δηλαδή 1000/60 φορές το δευτερόλεπτο. Επομένως οι μονάδες εισ/εξ απαιτούν για την εξυπηρέτησή τους επιπλέον $\frac{1000}{60}$ x 600 κ.ρ. x 10^{-9} sec x **1** = 10^{-5}sec. Επομένως, η ΚΜΕ κάθε δευτερόλεπτο απασχολείται με τον έλεγχο και την εξυπηρέτηση των μονάδων εισ/εξ: 10^{-2} sec + 10^{-5}sec = 0.01001 → μετατροπή σε ποσοστό: 1.001%. Σε αυτό το ποσοστό απασχολείται η ΚΜΕ δαπανά το του χρόνου της για τον έλεγχο και την εξυπηρέτηση των μονάδων εισ/εξ. όταν χρησιμοποιείται η χρονοπρογραμματισμένη διαδικασία (polling).

β. Αφού κατά μέσο όρο όλες οι μονάδες εισ/εξ στέλνουν συνολικά 500 σήματα διακοπής το λεπτό και η εξυπηρέτηση κάθε σήματος διακοπής απαιτεί 1.200 κύκλους ρολογιού, συνεπάγεται ότι στο ένα δευτερόλεπτο η ΚΜΕ αφιερώνει για την εξυπηρέτηση των μονάδων εισ/εξ: $\frac{1000}{60}$ x 1200 κ.ρ. x 10^{-9} sec x **1** = 0.002 x10^{-2} sec. Επομένως σε αυτή την περίπτωση η ΚΜΕ δαπανά το 0.002% του χρόνου της για την εξυπηρέτηση των μονάδων εισ/εξ.