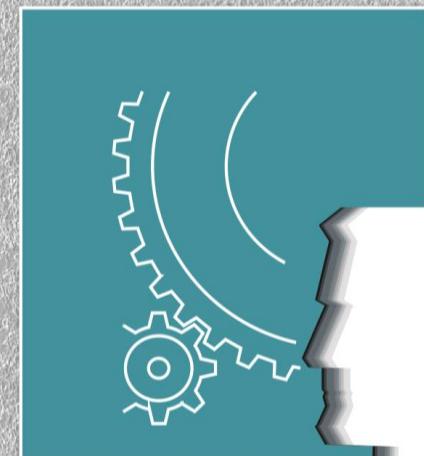


# **ΣΗΜΕΙΩΣΕΙΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ I**

## **(Βασικά Θέματα Αρχιτεκτονικής Υπολογιστών)**

### **Computer – Ανάλυση**



**ανάλιση**  
COMPUTER

ΕΡΓΑΣΤΗΡΙΟ ΕΛΕΥΘΕΡΟΝ ΣΠΟΥΔΩΝ  
ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

ΑΓ. ΑΝΔΡΕΟΥ 130-132 & ΓΟΥΝΑΡΗ, Τ.Κ. 26222, ΠΑΤΡΑ  
ΤΗΛ: 310-097, 324-900 - ΤΗΛ/FAX: 313-547  
E-MAIL: jianelisj@otenet.gr, jianelisk@otenet.gr

*...λυση  
σπουδων*

## Περιεχόμενα

Τυπολόγιο .....	1
Κεφάλαιο 1 – Απόδοση Η/Υ .....	1
Βασικά σημεία θεωρίας .....	1
Θέμα Σεπτεμβρίου 2021 - Υπολογισμός μέσου πλήθους κύκλων ρολογιού .....	2
Θέμα Προόδου 2021 - Υπολογισμός χρόνου εκτέλεσης .....	2
Παράδειγμα 1.1 - Υπολογισμός χρόνου εκτέλεσης - ΜΧΠΜ - μέσου χρόνου εκτέλεσης εντολής .....	4
Θέμα Φεβρουαρίου 2014 με ΜΚΡΕ και ΜΧΠΜ .....	5
Θέμα Ιουνίου 2021.....	6
Θέμα Ιουνίου 2021.....	7
Θέμα 2016 με υπολογισμό ΜΧΠΜ και ξεχωριστή κρυφή μνήμη εντολών/δεδομένων .....	7
Θέμα 2012 με υπολογισμό χρόνου εκτέλεσης και δύο επίπεδα κρυφής μνήμης .....	8
Παράδειγμα 1.2 - Χρήση μέτρου MIPS για την αξιολόγηση της απόδοσης Η/Υ .....	9
Άσκηση υπολογισμού χρόνου εκτέλεσης .....	10
Άσκηση με μονάδα πολλαπλασιασμού και διαίρεσης .....	11
Θέματα προόδου Νοεμβρίου 2022 .....	11
Κεφάλαιο 2 – Αρχιτεκτονικές Η/Υ – Μορφή εντολής – Είδη αριθμών .....	14
Επεξηγήσεις Κεφαλαίου 2.....	14
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 .....	20
Θέμα με αρχιτεκτονική καταχωρητή – καταχωρητή με 3 τελούμενα - Σεπτέμβριος 2015.....	21
Πρόοδος Νοεμβρίου 2016 .....	23
Θέμα με μεταβλητό μήκος εντολών – χώρο και προσπελάσεις προγράμματος – Ιανουάριος 2015 .....	23
Θέμα με μεταβλητό μήκος εντολών – χώρο και προσπελάσεις προγράμματος – Ιούνιος 2016.....	24
Θέμα με σταθερό μήκος εντολών και πλήθος θ.μ. που προσπελαύνονται - Νοέμβριος 2016 .....	25
Θέμα με σταθερή μορφή εντολών και πλήθος θ.μ. που προσπελαύνονται - Ιούνιος 2016 .....	26
Θέμα Φεβρουαρίου 2021 .....	26
Θέμα Ιουνίου 2020.....	28
Θέμα Σεπτεμβρίου 2020 .....	29
Θέμα Σεπτεμβρίου 2020 (άλλη ομάδα) .....	30
Θέμα Σεπτεμβρίου 2020 .....	31
Άσκηση 2.10 με ευθυγράμμιση μνήμης .....	31
Άσκηση με ευθυγράμμιση μνήμης .....	32
Άσκηση με μετατροπή αριθμού από κινητή σε σταθερή υποδιαστολή .....	33
Θέμα Ιανουαρίου 2017 με μορφή εντολών.....	34
Θέματα Σεπτεμβρίου 2020 - Φεβρουαρίου - Ιουνίου 2021 με εντολές αρχιτεκτονικής υπολογιστών .....	38
Άσκηση 2.6 από λυσάρι με αριθμούς κινητής υποδιαστολής.....	40
Άσκηση 2.7 από λυσάρι με αριθμούς κινητής υποδιαστολής.....	40
Θέμα προόδου με άθροισμα αριθμών σε μορφή IEEE 754 – Ιανουάριος 2022 .....	41

Λύση Θέματος 3 – Φεβρουάριος 2014 .....	42
Θέματα προόδου Νοεμβρίου 2022 .....	42
<b>Κεφάλαιο 3 – Κατηγορίες αριθμών και εντολές μεταφοράς από/προς μνήμη .....</b>	<b>45</b>
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 .....	50
Επεξηγήσεις για εντολές LOAD/STORE .....	52
Θέμα Φεβρουαρίου 2016 με πράξεις προσημασμένων/μη προσημασμένων αριθμών .....	53
Θέμα 1 με θύρες ανάγνωσης/εγγραφής .....	53
Θέμα 2 με υπερχείλιση/κρατούμενο .....	54
Θέμα 3 με καταχωρητές γενικού σκοπού .....	54
Θέματα Ιουνίου 2021 με πράξεις προσημασμένων/μη προσημασμένων αριθμών .....	54
Θέματα Ιουνίου 2021 με εντολές LOAD/STORE .....	55
Θέματα Σεπτεμβρίου 2020 με θύρες καταχωρητών και σταθερή/κινητή υποδιαστολή .....	56
Θέματα Ιουνίου 2020 με πράξεις προσημασμένων/μη προσημασμένων αριθμών .....	57
Άσκηση 3.3 βιβλίου για εντολές LOAD/STORE .....	59
Θέματα Φεβρουαρίου 2014 με εντολή LOAD και πράξεις αριθμών .....	60
Θέματα Σεπτεμβρίου 2016 με εντολές LOAD/STORE .....	61
Διαφορά little – endian και big – endian αποθήκευσης .....	64
Άσκηση με big – endian και little – endian .....	66
Θέμα Φεβρουαρίου 2021 με little – endian και big – endian .....	67
Θέμα Σεπτεμβρίου 2021 με little – endian και big – endian .....	68
Θέμα Σεπτεμβρίου 2020 με little – endian και big – endian .....	68
Θέμα προόδου 2021 με little – endian και big – endian .....	69
Θέμα Ιουνίου 2020 .....	69
Θέμα Σεπτεμβρίου 2020 .....	70
Θέμα Σεπτεμβρίου 2020 .....	70
Άσκηση με μετατροπή από σταθερή σε κινητή υποδιαστολή .....	71
1 <sup>η</sup> Άσκηση με τεχνική μικροπρογραμματισμού δύο επιπέδων .....	72
2 <sup>η</sup> Άσκηση με τεχνική μικροπρογραμματισμού δύο επιπέδων – Θέμα Σεπτέμβριος 2021 .....	73
3 <sup>η</sup> Άσκηση με τεχνική μικροπρογραμματισμού δύο επιπέδων από ΕΑΠ .....	73
4 <sup>η</sup> Άσκηση με τεχνική μικροπρογραμματισμού δύο επιπέδων από το μάθημα .....	74
5 <sup>η</sup> Άσκηση: Θέμα προόδου με τεχνική μικροπρογραμματισμού δύο επιπέδων – Ιανουάριος 2022 .....	75
<b>Κεφάλαιο 5 – Μνήμες .....</b>	<b>76</b>
Επεξηγήσεις κεφαλαίου 5 .....	76
1 <sup>ο</sup> Θέμα με ιεραρχία μνήμης .....	80
2 <sup>ο</sup> Θέμα με ιεραρχία μνήμης .....	80

Παράδειγμα 5.6 βιβλίου – Σχεδιασμός υψηλής τάξης διαφύλλωση μνήμης .....	81
Παράδειγμα 5.7 βιβλίου – Σχεδιασμός υψηλής τάξης διαφύλλωση μνήμης .....	83
Θέμα Φεβρουαρίου 2016 .....	86
Θέμα Ιουνίου 2016.....	87
Άσκηση 5.4 με τμήματα μνήμης .....	89
Θέμα Σεπτεμβρίου 2016 .....	90
Λύση Θέματος Ιανουαρίου 2018.....	92
Θέμα Ιουνίου 2014.....	95
Λύση Θέματος κύριας μνήμης – Ιανουάριος 2020 .....	96
Θέμα Φεβρουαρίου 2021 .....	97
Θέμα Ιουνίου 2021.....	98
Θέμα Ιουνίου 2020.....	99
Θέμα Σεπτεμβρίου 2020 .....	100
Θέμα Φεβρουαρίου 2021 .....	101
Θέμα Ιουνίου 2021.....	102
Θέμα Ιουνίου 2020.....	102
Θέμα Σεπτεμβρίου 2021 .....	103
Θέμα προόδου Ιανουαρίου 2022 .....	104
Θέμα προόδου Ιανουαρίου 2022 .....	105
Θέμα προόδου Ιανουαρίου 2022 .....	106
Παράδειγμα 5.8 με χαμηλής τάξης διαφύλλωση .....	107
Θέμα Φεβρουαρίου 2016 με χαμηλής τάξης διαφύλλωση.....	108
Θέμα Ιουνίου 2016 με σχεδιασμό κύριας μνήμης χαμηλής τάξης διαφύλλωσης .....	108
Θέμα Φεβρουαρίου 2012 με χαμηλής τάξης διαφύλλωση.....	109
Άσκηση 5.13 με οργάνωση χαμηλής τάξης διαφύλλωση μνήμης .....	112
Θέμα Σεπτεμβρίου 2015 με σχεδιασμό κύριας μνήμης χαμηλής τάξης διαφύλλωσης.....	114
Κεφάλαιο 6 – Αρτηρίες .....	116
Βασικά σημεία θεωρίας .....	116
Άσκηση 6.1 βιβλίου με σύγχρονη/ασύγχρονη αρτηρία .....	118
Άσκηση 6.2 βιβλίου με σύγχρονη/ασύγχρονη αρτηρία .....	119
Θέμα προόδου Ιανουαρίου 2022 .....	120
Θέμα προόδου Ιανουαρίου 2022 .....	120
Θέμα προόδου Ιανουαρίου 2022 .....	121
Άσκηση 6.3 βιβλίου με polling/interrupts .....	121
Θέμα Ιουνίου 2016 με χρονοπρογραμματισμένη διαδικασία (polling) .....	122
Άσκηση 6.4 βιβλίου με ΑΠΜ (DMA) .....	123
Άσκηση 6.5 βιβλίου με DMA.....	124
Άσκηση με προγραμματισμένη είσοδο – έξοδο με χρήση διακοπών .....	124
Άσκηση με διαιτησία αρτηρίας.....	125
Θέμα με σύγχρονη/ασύγχρονη αρτηρία - Ιανουάριος 2020 .....	126

Θέμα με σύγχρονη και ασύγχρονη αρτηρία .....	126
Θέμα Φεβρουαρίου 2021 .....	128
Θέμα Ιουνίου 2020.....	130
Θέμα Σεπτεμβρίου 2020 .....	131
Θέματα Ιανουαρίου 2022 .....	132
Θέματα Σεπτεμβρίου 2022 .....	138
Θέματα Ιανουαρίου 2022 .....	147
Θέματα Σεπτεμβρίου 2023 .....	150
Θέματα Ιανουαρίου 2023 .....	157

## Τυπολόγιο

### Μετατροπή από δεκαεξαδικό 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	1	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

$$2^0 = 1$$

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

$$2^1 = 2$$

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

$$2^2 = 4$$

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

$$2^3 = 8$$

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

$$2^4 = 16$$

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

$$2^5 = 32$$

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

$$2^6 = 64$$

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

$$2^7 = 128$$

$$2^8 = 256$$

$$2^9 = 512$$

## Μονάδες μέτρησης χωρητικότητας μνήμης

1 bit = 0 ή 1 (δυαδικό ψηφίο)

1 byte = 8 bits (ψηφιολέξη) = 1B

Πολλαπλάσια του byte

1 byte (ψηφιολέξη) = 8 bits

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

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

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

1 word (λέξη) = 32 bits = 4 bytes

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

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

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

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

## Κεφάλαιο 1 – Απόδοση Η/Υ

### Βασικά σημεία θεωρίας

**1. Μειονεκτήματα MIPS:** Η απόδοση του υπολογιστή έχει προφανώς σχέση με την ταχύτητα εκτέλεσης προγραμμάτων σε αυτόν. MIPS είναι τα αρχικά γράμματα του "Million Instructions Per Second" και χρησιμοποιείται για να δηλώσει πόσα εκατομμύρια εντολές εκτελούνται ανά δευτερόλεπτο. Αυτό που φαίνεται προφανές είναι ο υπολογιστής που εκτελεί περισσότερες εντολές ανά δευτερόλεπτο από έναν άλλο, θα εκτελεί ένα πρόγραμμα πιο γρήγορα, άρα θα έχει μεγαλύτερη απόδοση. Αυτό όμως δεν είναι αληθές για δύο λόγους: **Πρώτον** διότι ο χρόνος εκτέλεσης μιας εντολής εξαρτάται από την πολυπλοκότητά της. Επομένως οι πιο πολύπλοκες εντολές έχουν μεγαλύτερο χρόνο εκτέλεσης. **Δεύτερον** το πλήθος των εντολών, σε επίπεδο γλώσσας μηχανής, ενός προγράμματος μιας συγκεκριμένης εφαρμογής εξαρτάται από το σύνολο εντολών του υπολογιστή στο οποίο θα τρέξει το πρόγραμμα. Οι παραπάνω δύο λόγοι συνεπάγονται ότι τα συμπεράσματά μας δεν είναι καθόλου ασφαλή εάν χρησιμοποιήσουμε τα MIPS για να συγκρίνουμε υπολογιστές με διαφορετικά σύνολα εντολών. Επίσης ο πρώτος από τους παραπάνω λόγους συνεπάγεται ότι ακόμη και στον ίδιο υπολογιστή τρέχοντας διαφορετικά προγράμματα υπολογίζουμε διαφορετική τιμή για τα MIPS. **Τα MIPS μπορούν να χρησιμοποιηθούν με ασφάλεια για την σύγκριση επεξεργαστών που έχουν το ίδιο σύνολο εντολών αλλά διαφορετικές υλοποιήσεις.**

**2. Μειονεκτήματα MFLOPS:** MFLOPS είναι τα αρχικά γράμματα του "Million Floating-point Operations Per Second" και χρησιμοποιείται για να δηλώσει πόσα εκατομμύρια πράξεων αριθμών κινητής υποδιαστολής εκτελούνται ανά δευτερόλεπτο. Έχει προταθεί για να δηλώνει την απόδοση μόνο για πράξεις αριθμητικής κινητής υποδιαστολής. Σε αντίθεση με τα MIPS μετράει αριθμό πράξεων και όχι εντολών. Η τιμή του μέτρου αλλάζει όχι μόνο ανάλογα με το ποσοστό των πράξεων κινητής υποδιαστολής σε σχέση με τις πράξεις σταθερής υποδιαστολής αλλά και ανάλογα με το ποσοστό των αργών πράξεων κινητής υποδιαστολής σε σχέση με τις γρήγορες. Για παράδειγμα ένα πρόγραμμα με 100% προσθέσεις κινητής υποδιαστολής θα δίνει μεγαλύτερη τιμή του μέτρου MFLOPS από ότι ένα πρόγραμμα με 100% πολλαπλασιασμούς κινητής υποδιαστολής. Επίσης οι υπολογιστές περιλαμβάνουν στο σύνολο εντολών τους διαφορετικές εντολές κινητής υποδιαστολής με συνέπεια η επιλογή των πράξεων κινητής υποδιαστολής που θα ληφθεί υπόψη, να επηρεάζει τα αποτελέσματα.

### 3. Να αναφέρετε τους παράγοντες από τους οποίους εξαρτάται η απόδοση του υπολογιστή.

1. Χρόνος εκτέλεσης προγραμμάτων (XE<sub>προγραμμάτων</sub>)
2. Χρόνος αναμονής
3. Αριθμός εκτελούμενων προγραμμάτων ανά ώρα (throughput)
4. Το πλήθος εντολών που εκτελούνται
5. MKPE (Μέσος αριθμός Κύκλων Ρολογιού για την εκτέλεση Εντολής)
6. Διάρκεια κύκλου ρολογιού
7. Χρόνος προσπέλασης της κύριας μνήμης
8. Χρόνος προσπέλασης της κρυφής μνήμης
9. Λόγος επιτυχίας της κρυφής μνήμης
10. Χωρητικότητα της κύριας μνήμης
11. Ταχύτητα της βοηθητικής μνήμης
12. Το λειτουργικό σύστημα

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.**

**A' 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  $XE_{\text{εντολών προγρ.}} = \left\{ \sum_{i=1}^v [KPE_i(KME) + E * KPE_{i(\text{κρυφή μνήμη})} + (1 - E) * KPE_{i(\text{κύρια μνήμη})}] * \Pi_i \right\} * \text{tcycle}$ . Αν υπάρχει **ξεχωριστή** κρυφή μνήμη εντολών και δεδομένων, ο  $XE_{\text{εντολών προγρ.}} = \left\{ \sum_{i=1}^v [KPE_i(KME) + E_1 * KPE_{i(\text{κρυφή μνήμη εντολών})} + (1 - E_1) * KPE_{i(\text{κύρια μνήμη - προσπέλαση εντολών})} + E_2 * KPE_{i(\text{κρυφή μνήμη δεδομένων})} + (1 - E_2) * KPE_{i(\text{κύρια μνήμη - προσπέλαση δεδομένων})}] * \Pi_i \right\} * \text{tcycle}$

**B' 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  $XE_{\text{εντολών προγρ.}} = \left\{ \sum_{i=1}^v [KPE_i(KME) + KPE_{i(\text{κρυφή μνήμη})} + (1 - E) * KPE_{i(\text{κύρια μνήμη})}] * \Pi_i \right\} * \text{tcycle}$ . Αν υπάρχει **ξεχωριστή** κρυφή μνήμη εντολών και δεδομένων, ο  $XE_{\text{εντολών προγρ.}} = \left\{ \sum_{i=1}^v [KPE_i(KME) + KPE_{i(\text{κρυφή μνήμη εντολών})} + (1 - E_1) * KPE_{i(\text{κύρια μνήμη - προσπέλαση εντολών})} + KPE_{i(\text{κρυφή μνήμη δεδομένων})} + (1 - E_2) * KPE_{i(\text{κύρια μνήμη - προσπέλαση δεδομένων})}] * \Pi_i \right\} * \text{tcycle}$ . 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 KME προσπελάζει πληροφορία μόνο από το κοντινότερο επίπεδο της ιεραρχικής μνήμης.

A. 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 .

B. 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 .

### **Αύση**

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

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

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

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

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

### **Θέμα Προόδου 2021 - Υπολογισμός χρόνου εκτέλεσης**

1

Θεωρείστε επεξεργαστή με συχνότητα ρολογιού  $0.25 \text{ GHz}$  και **κύρια μνήμη** με χρόνο προσπέλασης  $228 \text{ nsec}$ . Οι εντολές του επεξεργαστή σε επίπεδο γλώσσας μηχανής ταξινομούνται σε τρία είδη A, B 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
A	1	1	0	276
B	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 KME 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  $t_{cycle} = \frac{1}{\text{συχνότητα λειτουργίας επεξεργαστή}} = \frac{1}{0.25 \text{ GHz}} = \frac{1}{0.25 \times 10^9 \text{ Hz}}$   $= 4 \times 10^{-9} \text{ sec} = 4 \text{ nsec}$ . 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  $228 \text{ nsec} / 4 \text{ nsec} = 57 \text{ κ.ρ.}$

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



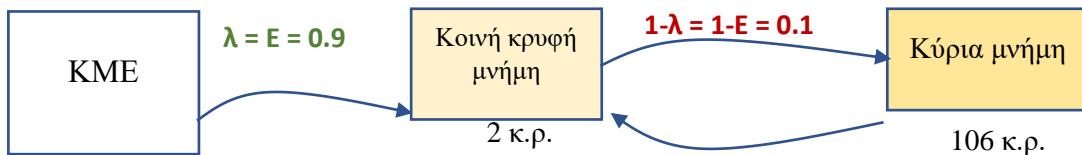
$$XEE_A = 1 \text{ κ.ρ.} + (1 + 0) * 57 \text{ κ.ρ.} = 58 \text{ κ.ρ.}$$

$$XEE_B = 4 \text{ κ.ρ.} + (4 + 2) * 57 \text{ κ.ρ.} = 346 \text{ κ.ρ.}$$

$$XEE_\Gamma = 6 \text{ κ.ρ.} + (4 + 3) * 57 \text{ κ.ρ.} = 405 \text{ κ.ρ.}$$

$XE_{\text{εντολών προγρ.}} = (XEE_A * \Pi_A + XEE_B * \Pi_B + XEE_\Gamma * \Pi_\Gamma) * t_{cycle} = (58 \text{ κ.ρ.} * 276 \text{ εντολές} + 346 \text{ κ.ρ.} * 523 \text{ εντολές} + 405 \text{ κ.ρ.} * 616 \text{ εντολές}) * 4 \text{ nsec} = 1785784 \text{ nsec}$ .

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  $8 \text{ nsec} / 4 \text{ nsec} = 2 \text{ κ.ρ.}$  και ο χρόνος προσπέλασης και μεταφοράς ενός μπλοκ εντολών/δεδομένων από την κύρια στην κρυφή της τάξης των 424 nsec, θα γίνει  $424 \text{ nsec} / 4 \text{ nsec} = 106 \text{ κ.ρ.}$  Επειδή η **KME** 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:



$$XE_{\text{εντολών προγρ.}} = \left\{ \sum_{i=1}^V [KPE_i(KME) + E * KPE_{i(\text{κρυφή μνήμη})} + (1 - E) * KPE_{i(\text{κύρια μνήμη})}] * \Pi_i \right\} * t_{cycle}.$$

$$XEE_A = 1 \text{ κ.ρ.} + 0.9 * (1 + 0) * 2 \text{ κ.ρ.} + 0.1 * (1 + 0) * 108 \text{ κ.ρ.} = 13.6 \text{ κ.ρ.}$$

$$XEE_B = 4 \text{ κ.ρ.} + 0.9 * (4+2) * 2 \text{ κ.ρ.} + 0.1 * (4+2) * 108 \text{ κ.ρ.} = 79.6 \text{ κ.ρ.}$$

$$XEE_G = 6 \text{ κ.ρ.} + 0.9 * (4+3) * 2 \text{ κ.ρ.} + 0.1 * (4+3) * 108 \text{ κ.ρ.} = 94.2 \text{ κ.ρ.}$$

$XE_{\text{εντολών προγρ.}} = (XEE_A * \Pi_A + XEE_B * \Pi_B + XEE_G * \Pi_G) * t_{\text{cycle}} = (13.6 \text{ κ.ρ.} * 276 \text{ εντολές} + 79.6 \text{ κ.ρ.} * 523 \text{ εντολές} + 94.2 \text{ κ.ρ.} * 616 \text{ εντολές}) * 4 \text{ nsec} = 413646.4 \text{ nsec}$ . 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:  $1785784 \text{ nsec} / 413646.4 = 4.3$ , επομένως έγινε 4.3 φορές μικρότερος ο νέος χρόνος. Ο χρόνος εκτέλεσης ενός προγράμματος ονομάζεται ισοδύναμα απόδοση.

### Παράδειγμα 1.1 - Υπολογισμός χρόνου εκτέλεσης - MXIM - μέσου χρόνου εκτέλεσης εντολής

Θεωρούμε έναν υπολογιστή που αποτελείται από την KME, 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 KME 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.

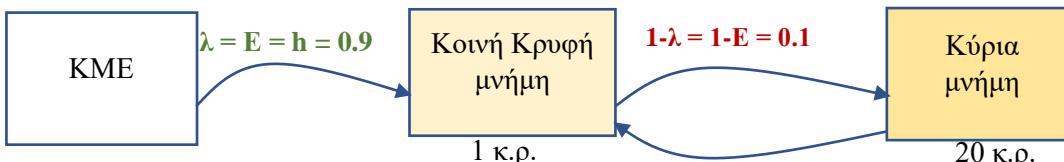
Πίνακας 1

Είδος ε-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
A	400
Γ	10
E	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.

$$XEE_A = 4 \text{ κ.ρ.} + 0.9 * (1+0) * 1 \text{ κ.ρ.} + 0.1 * (1+0) * 21 \text{ κ.ρ.} = 7 \text{ κ.ρ.}$$

$$XEE_G = 6 \text{ κ.ρ.} + 0.9 * (2+1) * 1 \text{ κ.ρ.} + 0.1 * (2+1) * 21 \text{ κ.ρ.} = 16 \text{ κ.ρ.}$$

$$XEE_E = 10 \text{ κ.ρ.} + 0.9 * (3+2) * 1 \text{ κ.ρ.} + 0.1 * (3+2) * 21 \text{ κ.ρ.} = 29 \text{ κ.ρ.}$$

$$XE_{\text{εντ. πρ.}} = (XEE_A * 400 + XEE_G * 10 + XEE_E * 30) * 4 \text{ nsec} = 14840 \text{ nsec}$$

β. Ζητάμε το **μέσο χρόνο προσπέλασης του συστήματος μνήμης (κρυφή – κύρια)**, δηλ. το **μέγεθος MXIM**. 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:  $\text{MXIM} = \frac{\text{Συνολικός χρόνος προσπέλασης συστήματος μνήμης}}{\text{Πλήθος προσπελάσεων μνήμης}}$ . 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, αντό προκύπτει αν αθροίσουμε για κάθε είδους εντολή, τις ψηφιολέξεις εντολών και δεδομένων και πολλαπλασιάσουμε αυτό το άθροισμα με το αντίστοιχο πλήθος εντολών. Επομένως,  $MXPM = \frac{\text{Συνολικός χρόνος προσπέλασης συστήματος μνήμης}}{\text{πλήθος προσπελάσεων μνήμης}} = \frac{[(7 \text{ κ.ρ.} - 4 \text{ κ.ρ.}) * 400 + (16 \text{ κ.ρ.} - 7 \text{ κ.ρ.}) * 10 + (29 \text{ κ.ρ.} - 10 \text{ κ.ρ.}) * 30] * 4 \text{ nsec}}{(1 + 0) * 400 + (2 + 1) * 10 + (3 + 2) * 30} = \dots$

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

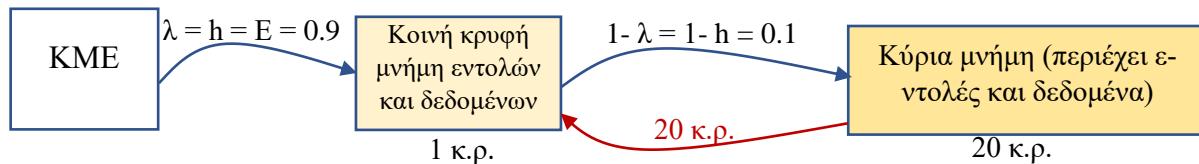
**Παρατήρηση:** Ο μέσος χρόνος εκτέλεσης εντολής είναι διαφορετικός από το μέσο αριθμό κύκλων ρολογιού ανά εντολή (MKPE), καθώς το MKPE =  $\frac{\sum_{i=1}^V KPE_i * \Pi_i}{\text{πλήθος εντολών}} = \frac{7 \text{ κ.ρ.} * 400 + 16 \text{ κ.ρ.} * 10 + 29 \text{ κ.ρ.} * 30}{440 \text{ εντολές}} = \dots \text{ κ.ρ./εντολή. Δηλαδή, λαμβάνουμε υπόψη τους κ.ρ. που υπολογίζαμε, χωρίς στο τέλος να κάνουμε τον πολλαπλασιασμό με το tcycle.}$

### Θέμα Φεβρουαρίου 2014 με MKPE και MXPM

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 MKPE και ο μέσος αριθμός προσπελάσεων του συστήματος μνήμης, δηλ. το MXPM.



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

=  $4 * 1.000.000 = 4.000.000 \text{ κ.ρ.}$ . 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 MXPM είναι:  $MXPM = \frac{\text{Συνολικός χρόνος προσπέλασης του συστήματος μνήμης}}{\text{πλήθος προσπελάσεων μνήμης}} = \frac{\text{Συνολικός χρόνος προσπέλασης του συστήματος μνήμης}}{1.000.000}$

$\Rightarrow \text{Συνολικός χρόνος προσπελάσεων του συστήματος μνήμης για αυτές τις εντολές είναι } 6.000.000 \text{.}$

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.  $6.000.000 \times 0.9 \times 1 \text{ κ.ρ.} + 6.000.000 \times 0.1 \times 21 \text{ κ.ρ.} = 5.400.000 \text{ κ.ρ.} + 12.600.000 \text{ κ.ρ.}$  Επομένως το συνολικό πλήθος των κ.ρ. που απαιτούνται είναι  $4.000.000 \text{ κ.ρ.} + 5.400.000 \text{ κ.ρ.} + 12.600.000 \text{ κ.ρ.} = 22.000.000 \text{ κ.ρ.}$

**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 Ιοννίου 2021

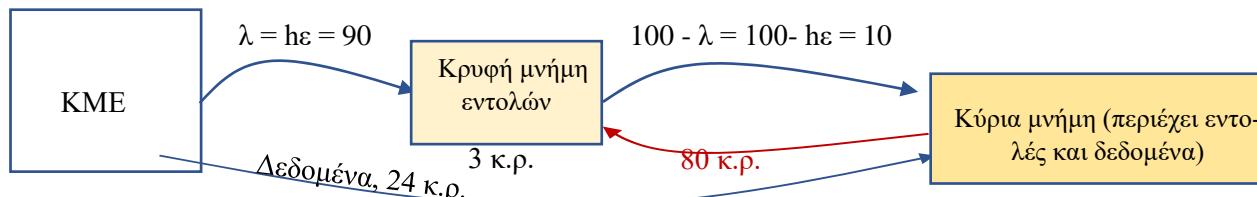
Ένας υπολογιστής ειδικού σκοπού διαθέτει μόνο κρυφή μνήμη εντολών με λόγο επιτυχίας  $he = 90$ . Η προσπέλαση του συστήματος μνήμης (κρυφή μνήμη ή κύρια μνήμη) από την KME γίνεται ανά ψηφιολέξη (byte). Ο χρόνος προσπέλασης της κρυφής μνήμης ισούται με 3 κύκλους ρολογιού, ενώ ο χρόνος προσπέλασης της 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 KME 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 A, B, Γ δίνεται πιο κάτω στα δεδομένα.

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

Υπολογίστε το **μέσο χρόνο προσπέλασης 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

#### Λύση

Παρότι από την εκφώνηση ζητείται **μόνο** το μέγεθος MXPI, θα υπολογίσουμε **επιπλέον** και το συνολικό χρόνο εκτέλεσης αυτού του προγράμματος. Επειδή υπάρχει **μόνο κρυφή μνήμη εντολών**, τα δεδομένα τα παίρνει η KME **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 εντολής χωριστά.



$$XEE_A = 7 \text{ κ.ρ.} + 0.9 * 3 \text{ ψηφ. εντολών * 3 κ.ρ.} + 0.1 * 3 \text{ ψηφ. εντολών * 83 κ.ρ.} + 0 \text{ ψηφιολέξεις δεδομένων * 24 κ.ρ.} = 40 \text{ κ.ρ.}$$

$$XEE_B = 5 \text{ κ.ρ.} + 0.9 * 1 \text{ ψηφ. εντολών * 3 κ.ρ.} + 0.1 * 1 \text{ ψηφ. εντολών * 83 κ.ρ.} + 2 \text{ ψηφιολέξεις δεδομένων * 24 κ.ρ.} = 64 \text{ κ.ρ.}$$

$$XEE_\Gamma = 6 \text{ κ.ρ.} + 0.9 * 2 \text{ ψηφ. εντολών * 3 κ.ρ.} + 0.1 * 2 \text{ ψηφ. εντολών * 83 κ.ρ.} + 0 \text{ ψηφιολέξεις δεδομένων * 24 κ.ρ.} = 28 \text{ κ.ρ.}$$

$$X_{\text{evt. pr.}} = (XEE_A * \Pi_A + XEE_B * \Pi_B + XEE_\Gamma * \Pi_\Gamma) * 3 \text{ nsec} = (40 \text{ κ.ρ.} * 121 + 64 \text{ κ.ρ.} * 213 + 28 \text{ κ.ρ.} * 169) * 3 \text{ nsec} = 69612 \text{ nsec.}$$

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** (κύρια μνήμη και κρυφή μνήμη) και όχι την KME. Το μέγεθος MXPI δίνεται από τον τύπο:

$$\text{MXPI} = \frac{\text{συνολικός χρόνος προσπέλασης συστήματος μνήμης}}{\text{πλήθος προσπελάσεων μνήμης}} = \frac{[(40-7)*121+(64-5)*213+(28-6)*169]*3 \text{ nsec}}{(3+0)*121+(1+2)*213+(2+0)*169} = \frac{60834 \text{ nsec}}{1340 \text{ πρ.}} = 45.3985 \text{ nsec/πρ.} \rightarrow 45$$

**Παρατήρηση 1:** Για τον υπολογισμό του μεγέθους MXPI λαμβάνουμε υπόψη μόνο το σύστημα μνήμης (κρυφή και κύρια μνήμη) με αποτέλεσμα να αφαιρούμε στον αριθμητή τους κύκλους ρολογιού που δαπανώνται στην KME. 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 Ιονίου 2021

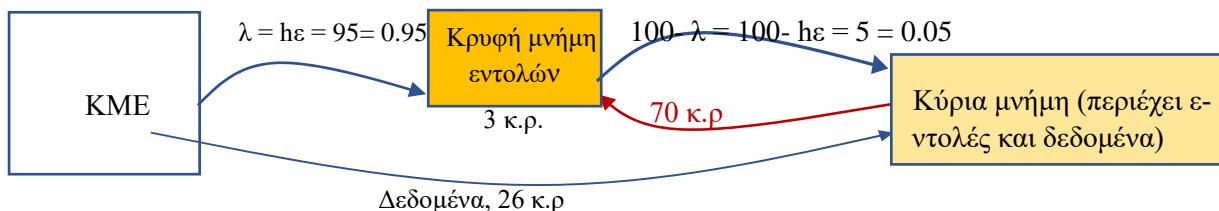
Ένας υπολογιστής ειδικού σκοπού διαθέτει μόνο κρυφή μνήμη εντολών με λόγο επιτυχίας  $h_e = 95$ . Η προσπέλαση του συστήματος μνήμης (κρυφή μνήμη ή κύρια μνήμη) από την KME γίνεται ανά ψηφιολέξη (byte). Ο χρόνος προσπέλασης της κρυφής μνήμης ισούται με 3 κύκλους ρολογιού, ενώ ο χρόνος προσπέλασης της 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 KME 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 των εκτελούμενων εντολών κάθε είδους A, B, Γ δίνεται πιο κάτω στα δεδομένα.

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

Υπολογίστε το μέσο χρόνο προσπέλασης του συστήματος μνήμης κατά την εκτέλεση του προγράμματος σε nsec.

### Λύση

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



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

$$[(0.95*1*3 \text{ κ.ρ.} + 0.05*1*73 \text{ κ.ρ.} + 0*26 \text{ κ.ρ.}) * 468 + (0.95*1*3 \text{ κ.ρ.} + 0.05*1*73 \text{ κ.ρ.} + 1*26 \text{ κ.ρ.}) * 56 + (0.95*3*3 \text{ κ.ρ.} + 0.05*3*73 \text{ κ.ρ.} + 3*26 \text{ κ.ρ.}) * 150] * 4 \text{nsec}$$

$$(1+0)*468 + (1+1)*56 + (3+3)*150$$

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

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

$$[(0.95*(1+0)*3 \text{ κ.ρ.} + 0.05*(1+0)*73 \text{ κ.ρ.}) * 468 + (0.95*(1+1)*3 \text{ κ.ρ.} + 0.05*(1+1)*73 \text{ κ.ρ.}) * 56 + (0.95*(3+3)*3 \text{ κ.ρ.} + 0.05*(3+3)*73 \text{ κ.ρ.}) * 150] * 4 \text{nsec}$$

$$(1+0)*468 + (1+1)*56 + (3+3)*150$$

## Θέμα 2016 με υπολογισμό MXΠΜ και ξεχωριστή κρυφή μνήμη εντολών/δεδομένων

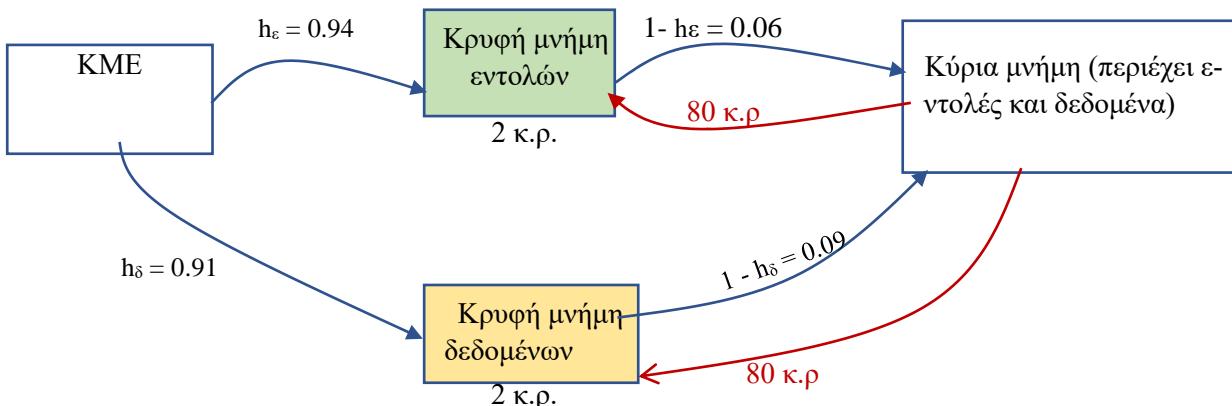
Σ' ένα υπολογιστή υπάρχει ξεχωριστή κρυφή μνήμη εντολών και δεδομένων με λόγο επιτυχίας αντίστοιχα  $h_e = 0.94$  και  $h_d = 0.91$ . Η προσπέλαση οποιασδήποτε από τις δύο κρυφές μνήμες βάζει καθυστέρηση δύο κύκλων ρολογιού. Σε κάθε περίπτωση αποτυχίας να βρεθεί η εντολή ή τα δεδομένα στην κρυφή μνήμη 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 KME 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 προσπέλαυνε 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 KME.



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

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

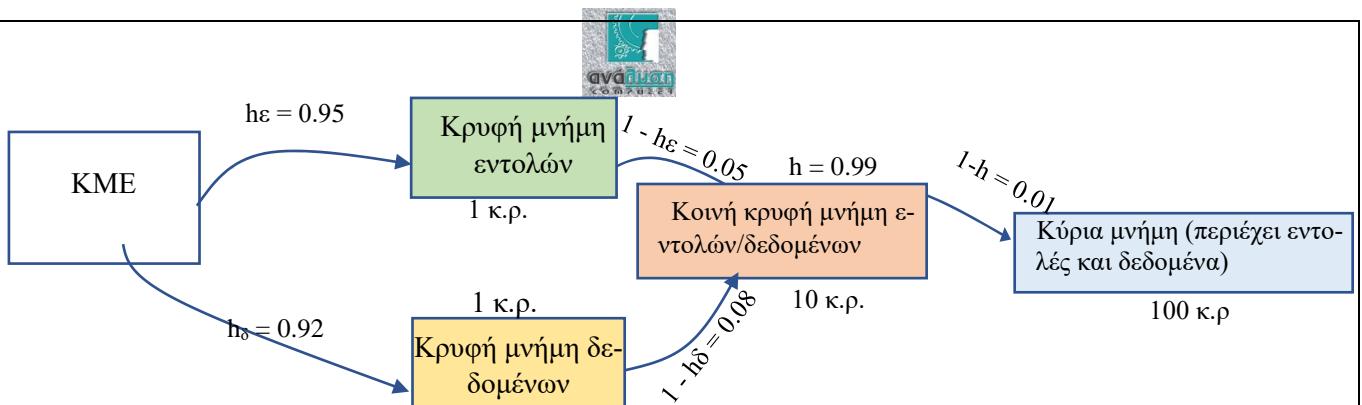
### Θαύμα 2012 με υπολογισμό χρόνου εκτέλεσης και δύο επίπεδα κρυφής μνήμης

Σ' ένα υπολογιστή υπάρχει ξεχωριστή κρυφή μνήμη εντολών και δεδομένων με λόγο επιτυχίας αντίστοιχα  $h_e = 0.95$  και  $h_\delta = 0.92$ . Η προσπέλαση οποιασδήποτε από τις δύο κρυφές μνήμες βάζει καθυστέρηση 1 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 δύο είδη εντολών, όπως φαίνεται στον Πίνακα που ακολουθεί. Υποθέστε ότι τόσο το εκτελούμενο πρόγραμμα όσο και τα απαιτούμενα δεδομένα βρίσκονται στην κύρια μνήμη. Η KME 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 προσπέλαυνε 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.

$$\text{XEE}_A = 4 \text{ k.p.} + 0.95 * 1 \text{ ψηφιολέξη εντολών * 1 k.p.} + 0.05 * [0.99 * 1 \text{ ψηφιολέξη εντολών * (1 + 10) k.p.} + (1 - 0.99) * 1 \text{ ψηφιολέξη εντολών * (1 + 10 + 100) k.p.}] + 0.92 * 0 \text{ ψηφιολέξεις δεδομένων * 1 k.p.} + 0.08 * [0.99 * 0 \text{ ψηφιολέξεις δεδομένων * (1 + 10) k.p.} + (1 - 0.99) * 0 \text{ ψηφιολέξεις δεδομένων * (1 + 10 + 100) k.p.}] = 5.5 \text{ k.p.}$$

$$\text{XEE}_B = 10 \text{ k.p.} + 0.95 * 3 \text{ ψηφιολέξεις εντολών * 1 k.p.} + 0.05 * [0.99 * 3 \text{ ψηφιολέξεις εντολών * (1 + 10) k.p.} + (1 - 0.99) * 3 \text{ ψηφιολέξεις εντολών * (1 + 10 + 100) k.p.}] + 0.92 * 2 \text{ ψηφιολέξεις δεδομένων * 1 k.p.} + 0.08 * [0.99 * 2 \text{ ψηφιολέξεις δεδομένων * (1 + 10) k.p.} + (1 - 0.99) * 2 \text{ ψηφιολέξεις δεδομένων * (1 + 10 + 100) k.p.}] = 18.41 \text{ k.p.}$$

$$\text{XEE}_{\text{evnt. προγρ.}} = (\text{XEE}_A * \Pi_A + \text{XEE}_B * \Pi_B) * t_{\text{cycle}} = (5.5 * 1000 + 18.41 * 2000) * 1 \text{ ns} = 42320 \text{ ns.}$$

$$\text{MXIM} = \frac{\text{Συνολικός χρόνος προσπέλασης συστήματος μνήμης}}{\text{Πλήθος προσπελάσεων μνήμης}} = \frac{[(5.5-4) * 1000 + (18.41-10) * 2000] * 1 \text{ ns}}{(1+0) * 1000 + (3+2) * 2000 \text{ προσπελάσεις μνήμης}} = 1.81 \text{ ns/προσπέλαση.}$$

### Παράδειγμα 1.2 - Χρήση μέτρου MIPS για την αξιολόγηση της απόδοσης H/Y

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 A, B 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 A και B.

Πίνακας 1.4

	Κύκλοι ρολογιού για την εκτέλεση μιας εντολής της ομάδας		
clock cycles	A	B	Γ
$Y_1$	1	2	3
$Y_2$	1	2	-

Πίνακας 1.5

	εκατομμύρια εντολών για κάθε ομάδα		
Πi	A	B	Γ
$Y_1$	5	1	1
$Y_2$	7	2	0

### Λύση

Το ζητούμενο στη συγκεκριμένη άσκηση είναι να εκτιμήσουμε την απόδοση δύο H/Y 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 ( $XE_{\text{εντολών προγράμματος}}$ ) θα χρησιμοποιήσουμε τον τύπο (2) της θεωρίας, που αναφέρει ότι  $XE_{\text{εντολών προγράμματος}} = [\sum_{i=1}^n (KPE_i * \Pi_i)] * t_{\text{cycle}}$  όπου το  $t_{\text{cycle}} = \frac{1}{\text{συχνότητα λειτουργίας}} = \frac{1}{2 \text{ GHz}} = \frac{1}{2 \times 10^9 \text{ Hz}} = 0.5 \times 10^{-9} \text{ sec} = 0.5 \text{ nsec.}$  Επομένως,  $XE_{\text{εντολών προγράμματος}} = [\sum_{i=1}^n (KPE_i * \Pi_i)] * 0.5 \text{ nsec.}$

Ο τύπος αυτός εφαρμόζεται για κάθε μια περίπτωση, δηλαδή για τον H/Y  $Y_1$  έχουμε:  $XE_{\text{εντολών προγράμματος}} Y_1 = [\sum_{i=1}^{n=3} (KPE_i * \Pi_i)] * 0.5 \text{ nsec} = (KPE_1 * \Pi_1 + KPE_2 * \Pi_2 + KPE_3 * \Pi_3) * 0.5 \text{ nsec} = (1 * 5 * 10^6 + 2 * 1 * 10^6 + 3 * 1 * 10^6) * 0.5 \text{ nsec} = 10 * 10^6 * 0.5 \text{ nsec} = 10^7 * 0.5 * 10^{-9} \text{ sec} = 0.5 * 10^{-2} \text{ sec} = 5 * 10^{-1} * 10^{-2} \text{ sec} = 5 * 10^{-3} \text{ sec} = 5 \text{ msec.}$



Για τον H/Y Y<sub>2</sub> έχουμε:  $XE_{\text{εντολών προγράμματος } Y_2} = [\sum_{i=1}^{n=2} (KPE_i * \Pi_i)] * 0.5 \text{ nsec} = (KPE_1 * \Pi_1 + KPE_2 * \Pi_2) * 0.5 \text{ nsec} = (1 * 7 * 10^6 + 2 * 2 * 10^6) * 0.5 \text{ nsec} = 11 * 10^6 * 0.5 \text{ nsec} = 11 * 10^6 * 0.5 * 10^{-9} \text{ sec} = 5.5 * 10^{-3} \text{ sec} = 5.5 \text{ msec}$ . Από τη σύγκριση των δύο χρόνων εκτέλεσης, παρατηρούμε ότι  $XE_{\text{εντολών προγράμματος } Y_1} < XE_{\text{εντολών προγράμματος } Y_2}$  κάτι που σημαίνει ότι ο H/Y Y<sub>1</sub> έχει μικρότερο χρόνο εκτέλεσης από τον Y<sub>2</sub>, με αποτέλεσμα να έχει καλύτερη απόδοση.

b) Για να συγκρίνουμε την απόδοση των δύο H/Y 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  $MIPS = \frac{\text{πλήθος εντολών}}{XE_{\text{εντολών προγράμματος}}}$ . Για τον υπολογιστή Y<sub>1</sub> έχουμε  $MIPS_{Y_1} = \frac{7 * 10^6 \text{ εντολές}}{5 \text{ msec}} = 1.4 * 10^6 * 10^3 \text{ εντολές/sec} = 1400 * 10^6 \text{ εντολές/sec}$ . Για τον υπολογιστή Y<sub>2</sub> έχουμε  $MIPS_{Y_2} = \frac{9 * 10^6 \text{ εντολές}}{5.5 \text{ msec}} = 1.63 * 10^6 * 10^3 \text{ εντολές/sec} = 1630 * 10^6 \text{ εντολές/sec}$ . Επομένως, καταλήγουμε στο λανθασμένο συμπέρασμα ότι ο Y<sub>2</sub> έχει καλύτερη απόδοση από τον Y<sub>1</sub>.

### ~~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 μνήμης (διάρκειας 20 κ.ρ.) γίνεται ανά ψηφιολέξη**, άρα ανά θέση μνήμης και διαρκεί 20 κ.ρ. Αντίθετα, στον υπολογιστή Z, **η προσπέλαση μνήμης (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

$$XEE_A = 4 \text{ κ.ρ.} + (1 + 0) * 20 \text{ κ.ρ.} = 24 \text{ κ.ρ.}$$

$$XEE_B = 7 \text{ κ.ρ.} + (2 + 1) * 20 \text{ κ.ρ.} = 67 \text{ κ.ρ.}$$

$$XEE_\Gamma = 10 \text{ κ.ρ.} + (3 + 2) * 20 \text{ κ.ρ.} = 110 \text{ κ.ρ.}$$

$$XE_{\text{εντ. προγράμματος}} = (XEE_A * \Pi_A + XEE_B * \Pi_B + XEE_\Gamma * \Pi_\Gamma) * t_{\text{cycle}} = (24 * 300 + 67 * 20 + 110 * 50) * 5 \text{ nsec} = 70200 \text{ nsec}$$

#### Υπολογιστής Z

$$XEE_A = 4 \text{ κ.ρ.} + (1 + 0) * 10 \text{ κ.ρ.} = 14 \text{ κ.ρ.}$$

$$XEE_B = 7 \text{ κ.ρ.} + (2 + 1) * 10 \text{ κ.ρ.} = 37 \text{ κ.ρ.}$$

$$XEE_{\Gamma} = 10 \text{ k.p.} + (3 + 2) * 10 \text{ k.p.} = 60 \text{ k.p.}$$

$$XE_{\text{evt.προγράμματος}} = (XEE_A * \Pi_A + XEE_B * \Pi_B + XEE_{\Gamma} * \Pi_{\Gamma}) * t_{\text{cycle}} = (14 * 300 + 37 * 20 + 60 * 50) * 5 \text{ nsec} = 39700 \text{ nsec}$$

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**

$$\frac{\text{Συνολικός αριθμός κύκλων ρολογιού}}{\text{πλήθος εντολών}} = \frac{24 \text{ k.p.} * 300 + 67 \text{ k.p.} * 20 + 110 \text{ k.p.} * 50}{370} = \frac{14040 \text{ k.p.}}{370}, \text{ ενώ στην περίπτωση του}$$

$$\text{υπολογιστή } Z \text{ είναι: } \frac{\text{Συνολικός αριθμός κύκλων ρολογιού}}{\text{πλήθος εντολών}} = \frac{14 \text{ k.p.} * 300 + 37 \text{ k.p.} * 20 + 60 \text{ k.p.} * 50}{370} = \frac{7940 \text{ k.p.}}{370}.$$

$$\gamma. W: \frac{\text{Συνολικός χρόνος προσπέλασης συστήματος μνήμης}}{\text{Πλήθος προσπελάσεων μνήμης}} = \frac{[(24 \text{ k.p.} - 4 \text{ k.p.}) * 300 + (67 \text{ k.p.} - 7 \text{ k.p.}) * 20 + (110 \text{ k.p.} - 10 \text{ k.p.}) * 50] * 5 \text{ nsec}}{(1 + 0) * 300 + (2 + 1) * 20 + (3 + 2) * 50} = 100 \text{ nsec. Ομοίως και για τον } Z.$$

### 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:  $T = T_{\text{πολλαπλασιασμών}} + T_{\text{διαιρέσεων}} + T_{\text{άλλων εντολών με}}$

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

$$T_{\text{διαιρέσεων}} = 0.1 * T$$

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

$$T_1 = 0.3 * T + 0.1 * T + 0.6 * T = 0.85 * T$$

$$T_2 = 0.3 * T + 0.25 * T + 0.6 * T = 0.925 * T$$

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

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

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

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

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

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

και

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

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. Αυτός είναι ο νόμος του Amdahl που αναφέρει ότι αν αντικαταστήσουμε εκείνη τη μονάδα που δαπανάται το μεγαλύτερο μέρος του χρόνου εκτέλεσης, θα έχουμε καλύτερη απόδοση.

### ~~Θέματα προόδου Νοεμβρίου 2022~~

1. Υπολογιστής έχει σύστημα μνήμης που καταλαμβάνει το 44% του χρόνου κατά την εκτέλεση ενός προγράμματος. Έπειτα από μια αλλαγή του συστήματος μνήμης, ο χρόνος εκτέλεσης του προγράμματος γίνεται 1.7 φορές μικρότερος από τον αρχικό. Να βρείτε πόσο **μικρότερος** είναι ο χρόνος προσπέλασης του καινούριου συστήματος μνήμης.

### Λύση

#### A' τρόπος

$$XE_1 = XE_{\text{εντολών}} + X\Pi_{\text{συστήματος μνήμης 1}} = 0.56 T + 0.44 T = T$$

$$\text{Θέλουμε } \frac{XE_1}{XE_2} = 1.7 \Rightarrow XE_2 = \frac{XE_1}{1.7} = \frac{T}{1.7} = 0.5882 T$$

$$XE_2 = XE_{\text{εντολών}} + X\Pi_{\text{συστήματος μνήμης 2}} = 0.56 T + X\Pi_{\text{συστήματος μνήμης 2}} \Rightarrow 0.5882 T = 0.56 T + X\Pi_{\text{συστήματος μνήμης 2}} \Rightarrow X\Pi_{\text{συστήματος μνήμης 2}} = (0.5882 - 0.56) * T = 0.0282 * T$$

Άρα  $\frac{\text{ΧΠ συστήματος μνήμης 1}}{\text{ΧΠ συστήματος μνήμης 2}} = \frac{0.44\text{T}}{0.0282\text{T}} = 15.58$  Άρα ο χρόνος προσπέλασης του νέου συστήματος μνήμης έγινε 15.58 φορές μικρότερος από αυτόν του αρχικού συστήματος μνήμης.

### B' τρόπος

- $1: 1.7 = 0.5882$  είναι ο νέος χρόνος εκτέλεσης → σε ποσοστό: 58.823%
- $58.823 - (100 - 44) = 58.82 - 56 = 2.82$  (αφαιρούμε από το νέο συνολικό χρόνο, το χρόνο που δαπανάται για τις εντολές  $(100 - 44)$ , δηλ. το XE<sub>εντολών</sub>, για να υπολογίσουμε το νέο χρόνο μνήμης μετά από την αλλαγή που κάνουμε)
- $\frac{\text{ΧΠ συστήματος μνήμης 1}}{\text{ΧΠ συστήματος μνήμης 2}} = \frac{44}{2.82} = 15.58$ . Εδώ διαιρούμε τους δύο χρόνους μνήμης, τον αρχικό προς τον νέο.

2. Η εκτέλεση ενός προγράμματος σε ένα υπολογιστή απαιτεί 207604 κύκλους. Εάν ο μέσος αριθμός κύκλων ανά εντολή είναι 34 ποιο είναι το πλήθος των εντολών του προγράμματος που εκτελέστηκαν;

Να δώσετε το αποτέλεσμα χωρίς χρήση τελειών πχ 54990

### Λύση

$$\text{MKPE} = \frac{\text{Συνολικοί κύκλοι ρολογιού}}{\text{Πλήθος εντολών}} = \frac{207604 \text{ κ.ρ.}}{\text{Πλήθος εντολών}} = 34 \Rightarrow \text{Πλήθος εντολών} = \frac{207604 \text{ κ.ρ.}}{34} = 6106 \text{ κ.ρ.}$$

3. Θεωρήστε επεξεργαστή με συχνότητα ρολογιού 0.25 GHz και κύρια μνήμη με χρόνο προσπέλασης 212 nsec. Οι εντολές του επεξεργαστή σε επίπεδο γλώσσας μηχανής ταξινομούνται σε τρία είδη A, B 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
A	1	1	0	325
B	5	4	1	493
Γ	3	3	2	733

Να υπολογίσετε το χρόνο εκτέλεσης του προγράμματος. Να δώσετε το αποτέλεσμα χωρίς χρήση τελειών πχ 67402

Ο σχεδιαστής του επόμενου μοντέλου αποφάσισε να προσθέσει στον υπολογιστή μία ενοποιημένη κρυφή μνήμη εντολών και δεδομένων με λόγο επιτυχίες 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 336 nsec. Σημειώστε ότι: 1. Η κεντρική μονάδα επεξεργασίας διαβάζει πληροφορία μόνο από την κρυφή μνήμη. 2. Η KME 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 κόμμα και τα δύο πιο σημαντικά ψηφία του δεκαδικού μέρους απορρίπτοντας τα υπόλοιπα δεκαδικά ψηφία (περικοπή) πχ. 3.41

### Λύση

Αρχικά θα υπολογίσουμε τη διάρκεια του κύκλου ρολογιού  $t_{cycle} = \frac{1}{\text{συχνότητα λειτουργίας επεξεργαστή}} = \frac{1}{0.25 \text{ GHz}} = \frac{1}{0.25 \times 10^9 \text{ Hz}} = 4 \times 10^{-9} \text{ sec} = 4 \text{ nsec}$ . 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 212 nsec, θα έχουμε  $212 \text{ nsec} / 4 \text{ nsec} = 53 \text{ κ.ρ.}$

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



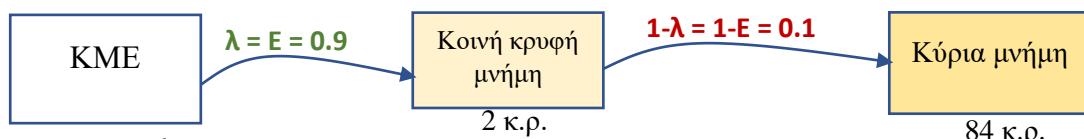
$$XEE_A = 1 \text{ κ.ρ.} + (1 + 0) * 53 \text{ κ.ρ.} = 54 \text{ κ.ρ.}$$

$$XEE_B = 5 \text{ κ.ρ.} + (4 + 1) * 53 \text{ κ.ρ.} = 270 \text{ κ.ρ.}$$

$$XEE_\Gamma = 3 \text{ κ.ρ.} + (3 + 2) * 53 \text{ κ.ρ.} = 268 \text{ κ.ρ.}$$

$$XE_{\text{εντολών προγρ.}} = (XEE_A * \Pi_A + XEE_B * \Pi_B + XEE_\Gamma * \Pi_\Gamma) * t_{\text{cycle}} = (54 \text{ κ.ρ.} * 325 \text{ εντολές} + 270 \text{ κ.ρ.} * 493 \text{ εντολές} + 268 \text{ κ.ρ.} * 733 \text{ εντολές}) * 4 \text{ nsec} = 1388416 \text{ nsec.}$$

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  $8 \text{ nsec}/4 \text{ nsec} = 2 \text{ κ.ρ.}$  και ο χρόνος προσπέλασης και μεταφοράς ενός μπλοκ εντολών/δεδομένων από την κύρια στην κρυφή της τάξης των 336 nsec, θα γίνει  $336 \text{ nsec}/4 \text{ nsec} = 84 \text{ κ.ρ.}$  Επειδή η KME 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:



$$XE_{\text{εντολών προγρ.}} = \left\{ \sum_{i=1}^v [KPE_i(KME) + E * KPE_{i(\text{κρυφή μνήμη})} + (1 - E) * KPE_{i(\text{κύρια μνήμη})}] * \Pi_i \right\} * t_{\text{cycle}}.$$

$$XEE_A = 1 \text{ κ.ρ.} + 0.9 * (1 + 0) * 2 \text{ κ.ρ.} + 0.1 * (1 + 0) * 86 \text{ κ.ρ.} = 11.4 \text{ κ.ρ.}$$

$$XEE_B = 5 \text{ κ.ρ.} + 0.9 * (4 + 1) * 2 \text{ κ.ρ.} + 0.1 * (4 + 1) * 86 \text{ κ.ρ.} = 57 \text{ κ.ρ.}$$

$$XEE_\Gamma = 3 \text{ κ.ρ.} + 0.9 * (3 + 2) * 2 \text{ κ.ρ.} + 0.1 * (3 + 2) * 86 \text{ κ.ρ.} = 55 \text{ κ.ρ.}$$

$XE_{\text{εντολών προγρ.}} = (XEE_A * \Pi_A + XEE_B * \Pi_B + XEE_\Gamma * \Pi_\Gamma) * t_{\text{cycle}} = (11.4 \text{ κ.ρ.} * 325 \text{ εντολές} + 57 \text{ κ.ρ.} * 493 \text{ εντολές} + 55 \text{ κ.ρ.} * 733 \text{ εντολές}) * 4 \text{ nsec} = 288484 \text{ nsec.}$  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:  $1388416 \text{ nsec} / 288484 \text{ nsec} = 4.81$ , επομένως έγινε 4.81 φορές μικρότερος ο νέος χρόνος. Ο χρόνος εκτέλεσης ενός προγράμματος ονομάζεται ισοδύναμα απόδοση.

## 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  $a_{v-1}a_v\dots\dots a_1a_0$ , οι τρεις τρόποι αναπαράστασής τους σχετίζονται με τη θέση της υποδιαστολής, η οποία μπορεί να είναι δεξιότερα του ψηφίου  $a_0$  με αποτέλεσμα όλοι οι αριθμοί να είναι ακέραιοι  $\geq 1$ , αριστερότερα του ψηφίου  $a_{v-1}$  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:  $a_{v-1}a_{v-2}\dots\dots a_1a_0 \xrightarrow{\text{metakínēsi}} \text{metakínēsi}$  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 αυτό έχει ως 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,  $\beta = 2$ ), **συμπλήρωμα ως προς μειωμένη βάση** (συμπλήρωμα ως προς 1, δεδομένου ότι μιλάμε για αριθμούς δυαδικούς,  $\beta = 2$ ) και **παράσταση προσημασμένου μεγέθους**. Στην περίπτωση αναπαράστασης προσημασμένου μεγέθους 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:  $Z = (-1)^\pi \times \Sigma_k \times B^E$ , όπου  $\pi$  = πρόσημο (0 ή 1), το  $\Sigma_k$  = 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 $\Sigma_k \neq 0$	Πρόσημο ( $\pi$ ) 1 - bit	Εκθέτης ( $E$ ) 8 - bit	συντελεστής ( $\Sigma_k$ ) 23 - bit
$N = (-1)^\pi \times \infty$ , αν $E = 255$ και $\Sigma_k = 0$			

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

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

$N = (-1)^\pi \times 0$ , αν  $E = 0$  και  $\Sigma_k = 0$

5. Όταν μετατρέπουμε έναν αριθμό από παράσταση σταθερής υποδιαστολής σε παράσταση κινητής υποδιαστολής, πρέπει πρώτα να **κανονικοποιήσουμε** τον αριθμό αυτό και να τον φέρουμε στη μορφή **1.xxxxx...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 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:

<b>ADD</b>	// παράδειγμα εντολής χωρίς τελούμενα (για την περίπτωση στοίβας)
<b>LOAD X</b>	// παράδειγμα εντολής με ένα τελούμενο (για την περίπτωση συσσωρευτή)
<b>ADD R1, #10</b>	// $R1 \leftarrow R1 + 10$ (παράδειγμα με δύο τελούμενα, καταχωρητή και αριθμό)
<b>ADD R1, \$200</b>	// $R1 \leftarrow R1 + \$200$ (παράδειγμα με δύο τελούμενα, καταχωρητή και διεύθυνση μνήμης)
<b>ADD R1, R2</b>	// $R1 \leftarrow R1 + R2$ (παράδειγμα με δύο τελούμενα, που είναι καταχωρητές)
<b>ADD R1, R2, R3</b>	// $R1 \leftarrow R2 + R3$ (παράδειγμα με τρία τελούμενα, που είναι καταχωρητές)
<b>ADD R1, R2, \$200</b>	// $R1 \leftarrow R2 + \$200$ (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  $2^x \geq \text{πλήθος εντολών}$ . 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.  $2^x \geq \text{πλήθος καταχωρητών}$  π.χ. για 32 καταχωρητές:  $2^x \geq 32 \Rightarrow x = 5$  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 ( ) αυτό δίνεται από τον τύπο:  $2^{\text{μήκος καταχωρητή}} \text{, ενώ αν έντελο είναι διεύθυνση μνήμης, τότε πλήθος των θέσεων μνήμης που μπορεί να διευθυνσιοδοτηθεί από την εντολή} = 2^{\text{μήκος εντολή}}$ .

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. Πιο συγκεκριμένα, ισχύει το εξής τυπολόγιο:

#### Τελούμενο

**Byte (ψηφιολέξη)**

**Halfword (μιστή λέξη)**

**Word (λέξη)**

**Double word (Διπλή λέξη)**

**Quad word (Τετραπλή λέξη)**

#### Τιμή 4 λιγότερο σημαντικών ψηφίων διεύθυνσης

xxxx (οποιαδήποτε διεύθυνση, δεν υπάρχει περιορισμός)

xxx0 (τέσσερα τελευταία δυαδικά ψηφία διεύθυνσης)

xx00 (τέσσερα τελευταία δυαδικά ψηφία διεύθυνσης)

x000 (τέσσερα τελευταία δυαδικά ψηφία διεύθυνσης)

0000 (τέσσερα τελευταία δυαδικά ψηφία διεύθυνσης)

- **Σε ασκήσεις με ευθυγράμμιση μνήμης**, έχει σημασία αν τα δεδομένα/εντολές που αποθηκεύονται στη μνήμη είναι ευθυγράμμισμένα ή όχι. Αν δεν είναι, 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 σε 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  $Y = A + B \times C + B \times E$  να γραφεί πρόγραμμα για H/Y 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 διεύθυνσιοδότησης του συστήματος μνήμης που χρησιμοποιείται 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. Κάθε εντολή καταλαμβάνει ακέραιο αριθμό ψηφιολέξεων.

γ) Θεωρήστε ότι τα δεδομένα είναι μιας ψηφιολέξης, η αρτηρία δεδομένων μεταξύ της Κεντρικής Μονάδας Επεξεργασίας (KME) 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 KME. 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:

$A + B \rightarrow AB+$

$A - B \rightarrow BA-$

$A \times B \rightarrow ABx$

$A/B \rightarrow BA/$

$$Y = A + B \times C + B \times E = A + BCx + BEx = ABCx+ + BEx = ABCx+BEx+$$

**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 A)

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 //Cx B, A (κάθε φορά η οποιαδήποτε πράξη γίνεται μεταξύ του κορυφαίου στοιχείου της στοίβας και του στοιχείου που είναι από κάτω).

ADD // CxB + A

**Κανόνας:** όταν υπολογίζουμε το μέγεθος κάθε εντολής, συμφέρει να το υπολογίζουμε

PUSH B //B, CxB + A αρχικά σε bits (δυαδικά ψηφία) και στη συνέχεια να μετατρέπουμε το μέγεθος αυτό

PUSH E //E, B, CxB + A σε bytes (αν χρειάζεται, πρέπει να κάνουμε στρογγυλοποίηση του μεγέθους αυτού προς τα πάνω, έτσι ώστε να είναι ακέραιο πολλαπλάσιο του «8», δηλ. της ψηφιολέξης).

MUL //ExB, CxB + A

ADD //ExB + CxB + A

POP Y //Y ← ExB + CxB + A

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 εντροπίας δεδομένων.

Δεδομένα: 8 bits	Αρτηρία δεδομένων (data bus): 8 bits	$\rightarrow 8/8 = 1$ επιπλέον προσπέλαση
Δεδομένα: 16 bits	Αρτηρία δεδομένων (data bus): 8 bits	$\rightarrow 16/8 = 2$ επιπλέον προσπελάσεις
Δεδομένα: 32 bits	Αρτηρία δεδομένων (data bus): 8 bits	$\rightarrow 32/8 = 4$ επιπλέον προσπελάσεις
Δεδομένα: 16 bits	Αρτηρία δεδομένων (data bus): 16 bits	$\rightarrow 16/16 = 1$ επιπλέον προσπέλαση
Δεδομένα: 32 bits	Αρτηρία δεδομένων (data bus): 16 bits	$\rightarrow 32/16 = 2$ 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: $Y = A + B \times C + B \times E$

Ο συσσωρευτής είναι ο βασικός καταχωρητής ενός H/Y 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, για την υπολογίσουμε και να την αποθηκεύσουμε.

8 bits 32 bits

LOAD B	$//\Sigma \leftarrow B$
MUL C	$//\Sigma \leftarrow B \times C$
ADD A	$//\Sigma \leftarrow B \times C + A$
STORE Y	$//Y \leftarrow B \times C + A$
LOAD B	$//\Sigma \leftarrow B$
MUL E	$//\Sigma \leftarrow B \times E$
ADD Y	$//\Sigma \leftarrow B \times E + B \times C + A$
STORE Y	$//Y \leftarrow B \times E + B \times C + A$

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: $Y = A + B \times C + B \times E$

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  $\leftarrow B$       44 bits  $\rightarrow$  48 bits  $\rightarrow$  6 bytes

LOAD R2, C // R2  $\leftarrow C$

8 bits 4 bits 4 bits

MUL R2, R1 // R2  $\leftarrow B \times C$       16 bits  $\rightarrow$  2 bytes

LOAD R3, E // R3  $\leftarrow E$

MUL R1, R3 // R1  $\leftarrow B \times E$

ADD R1, R2 // R1  $\leftarrow B \times E + B \times C$

LOAD R2, A // R2  $\leftarrow A$

ADD R1, R2 // R1  $\leftarrow B \times E + B \times C + A$

STORE Y, R1 //Y  $\leftarrow B \times E + B \times C + A$

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: $Y = A + B \times C + B \times E$

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.

<b>LOAD R1, B</b>	<b>8 bits</b>	<b>4 bits</b>	<b>32 bits</b>	<b>//R1 ← B</b>	<b>44 bits → 48 bits → 6 bytes</b>	O μηχανισμός καταχωρητή – καταχωρητή και καταχωρητή – μνήμης με δύο τελούμενα αναφέρεται στις πράξεις, οι οποίες τώρα γίνονται <b>τόσο</b> μεταξύ καταχωρητών <b>όσο</b> και μεταξύ καταχωρητών και θέσεων μνήμης. Σε αυτήν την περίπτωση το εκάστοτε αποτέλεσμα καταχωρείται στον <b>πρώτο</b> από τους δύο καταχωρητές.
<b>MUL R1, C</b>				<b>//R1 ← B x C</b>		
<b>ADD R1, A</b>				<b>//R1 ← B x C + A</b>		
<b>LOAD R2, B</b>				<b>// R2 ← B</b>		
<b>MUL R2, E</b>				<b>// R2 ← B x E</b>		
<b>ADD R1, R2</b>	<b>8 bits</b>	<b>4 bits</b>	<b>4 bits</b>	<b>//R1 ← B x C + A + B x E</b>	<b>16 bits → 2 bytes</b>	
<b>STORE Y, R1</b>						

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/τρία τελούμενα» αναφέρεται στο γεγονός ότι όταν κάνουμε **πράξεις** μεταξύ καταχωρητών, αλλά και μεταξύ καταχωρητών και θέσεων μνήμης, αυτοί θα είναι πάντα δύο/τρεις. 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 H/Y που να βασίζεται στη χρήση μηχανισμού στοίβας, συσσωρευτή, καταχωρητή – καταχωρητή με δύο τελούμενα και τέλος καταχωρητή – καταχωρητή – μνήμης με δύο τελούμενα.

#### Λύση

##### a) 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:

$$A + B \rightarrow AB+ \quad A - B \rightarrow BA- \quad A \times B \rightarrow ABx \quad A/B \rightarrow BA/$$

$$E = A \times (B - C \times D)/(C+G) = A \times (B - [CDx]) [CG+] = A \times [CDxB-] / [CG+] = [ACDxB-x] / [CG+] = [CG+ACDxB-x]$$

$$\text{PUSH C} \quad //C$$

$$\text{PUSH G} \quad //G, C$$

$$\text{ADD} \quad //G + C$$

$$\text{PUSH A} \quad //A, G + C$$

$$\text{PUSH C} \quad //C, A, G + C$$

$$\text{PUSH D} \quad //D, C, A, G + C$$

$$\text{MUL} \quad //DxC, A, G + C$$

$$\text{PUSH B} \quad //B, Dx C, A, G+C$$

$$\text{SUB} \quad //B - DxC, A, G + C$$

$$\text{MUL} \quad // (B - DxC)x A, G + C$$

$$\text{DIV} \quad // (B - DxC) x A / (G+C)$$

$$\text{POP E} \quad //E \leftarrow (B - DxC) x A / (G + C)$$

$$\beta) \text{ Μηχανισμός συσσωρευτή: } E = A \times (B - C \times D)/(C+G)$$

##### A' τρόπος (κακός):

$$\text{LOAD C} \quad //\Sigma \leftarrow C$$

$$\text{MUL D} \quad //\Sigma \leftarrow C \times D$$

$$\text{STORE X} \quad //X \leftarrow C \times D$$

LOAD B	// $\Sigma \leftarrow B$
SUB X	// $\Sigma \leftarrow B - X = B - C \times D$
MUL A	// $\Sigma \leftarrow (B - C \times D) \times A$
STORE X	// $X \leftarrow (B - C \times D) \times A$
LOAD C	// $\Sigma \leftarrow C$
ADD G	// $\Sigma \leftarrow C + G$
STORE Y	// $Y \leftarrow C + G$
LOAD X	// $\Sigma \leftarrow X = (B - C \times D) \times A$
DIV Y	// $\Sigma \leftarrow (B - C \times D) \times A / (C+G)$
STORE E	// $E \leftarrow (B - C \times D) \times A / (C+G)$

### 13 εντολές

B' τρόπος (καλός):  $E = A \times (B - C \times D) / (C+G)$

LOAD C	// $\Sigma \leftarrow C$
ADD G	// $\Sigma \leftarrow C + G$
STORE X	// $X \leftarrow C + G$
LOAD C	// $\Sigma \leftarrow C$
MUL D	// $\Sigma \leftarrow C \times D$
STORE Y	// $Y \leftarrow C \times D$
LOAD B	// $\Sigma \leftarrow B$
SUB Y	// $\Sigma \leftarrow B - Y = B - C \times D$
MUL A	// $\Sigma \leftarrow (B - C \times D) \times A$
DIV X	// $\Sigma \leftarrow (B - C \times D) \times A / (C+G)$
STORE E	// $E \leftarrow (B - C \times D) \times A / (C+G)$

### 11 εντολές

γ1) Μηχανισμός καταχωρητή – καταχωρητή με δύο τελούμενα:  $E = A \times (B - C \times D) / (C + G)$

LOAD R1, C	// $R1 \leftarrow C$
LOAD R2, G	// $R2 \leftarrow G$
ADD R2, R1	// $R2 \leftarrow C + G$
LOAD R3, D	// $R3 \leftarrow D$
MUL R1, R3	// $R1 \leftarrow C \times D$
LOAD R3, B	// $R3 \leftarrow B$
SUB R3, R1	// $R3 \leftarrow B - C \times D$
LOAD R1, A	// $R1 \leftarrow A$
MUL R1, R3	// $R1 \leftarrow A \times (B - C \times D)$
DIV R1, R2	// $R1 \leftarrow A \times (B - C \times D) / (C + G)$
STORE E, R1	// $E \leftarrow A \times (B - C \times D) / (C + G)$

γ2) Μηχανισμός καταχωρητή – καταχωρητή και καταχωρητή – μνήμης με δύο τελούμενα:  $E = Ax(B-C \times D)/(C+G)$

LOAD R1, C	// $R1 \leftarrow C$
ADD R1, G	// $R1 \leftarrow C + G$
LOAD R2, C	// $R2 \leftarrow C$
MUL R2, D	// $R2 \leftarrow C \times D$
LOAD R3, B	// $R3 \leftarrow B$
SUB R3, R2	// $R3 \leftarrow B - C \times D$
MUL R3, A	// $R3 \leftarrow A \times (B - C \times D)$
DIV R3, R1	// $R3 \leftarrow A \times (B - C \times D) / (C + G)$
STORE E, R3	// $E \leftarrow A \times (B - C \times D) / (C + G)$

### Αύση Θέματος 3 – Ιανουάριος 2020

Για τον υπολογισμό της έκφρασης  $Y = A + B \times C + D \times E$ , 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 (KME) 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 KME. Πόσες προσπελάσεις στη μνήμη απαιτεί η εκτέλεση κάθε εντολής.

### Αύση

(i) **Συσσωρευτής:  $Y = A + B \times C + D \times E$**

LOAD B	// $\Sigma \leftarrow B$
MUL C	// $\Sigma \leftarrow B \times C$
ADD A	// $\Sigma \leftarrow A + B \times C$
STORE Y	// $Y \leftarrow \Sigma$
LOAD D	// $\Sigma \leftarrow D$
MUL E	// $\Sigma \leftarrow D \times E$
ADD Y	// $\Sigma \leftarrow A + B \times C + D \times E$
STORE Y	// $Y \leftarrow A + B \times C + D \times E$

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  $2^x >= \text{πλήθος εντολών} = 200 \Rightarrow x = 8 \text{ bits}$ . Επομένως, επειδή όλες οι εντολές του συσσωρευτή είναι πανομοιότυπες, κάθε εντολή έχει μήκος  $8 + 24 = 32 \text{ bits} = 4 \text{ bytes}$ , 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 αρτηρία 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)**

$$Y = A + B \times C + D \times E = A + \boxed{BCx} + \boxed{DEx} = \boxed{ABCx+} + \boxed{DEx} = \boxed{ABCx+DEx+}$$

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 $\leftarrow E \times D + C \times B + A$

**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.**

$$Y = A + B \times C + D \times E$$

LOAD R1, B	// $R1 \leftarrow B$
LOAD R2, C	// $R2 \leftarrow C$
MUL R1, R2	// $R1 \leftarrow R1 \times R2 = B \times C$
LOAD R2, D	// $R2 \leftarrow D$
LOAD R3, E	// $R3 \leftarrow E$
MUL R2, R3	// $R2 \leftarrow R2 \times R3 = D \times E$
LOAD R3, A	// $R3 \leftarrow A$
ADD R1, R3	// $R1 \leftarrow R1 + R3$
ADD R1, R2	// $R1 \leftarrow R1 + R2$
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  $F = (A-B)x C - A/D + E$

- 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

#### a) Μηχανισμός στοίβας (σωρού)

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

$F = (A - B) \times C - A/D + E = BA - C \times DA + E = BA - Cx - DA + E = DA/BA - Cx + E = DA/BA - Cx - E +$
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: $F = (A - B) \times C - A/D + E$

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

LOAD A	//Σ ← A (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: $F = (A - B) \times C - A/D + E$

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: $F = (A - B) \times C - A/D + E$

**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, θεωρούμε ότι η πράξη ADD R1, R2, R3 έχει ως αποτέλεσμα  $R1 + R2 \rightarrow R3$  ή θεωρούμε ότι η ίδια πράξη θα έχει ως αποτέλεσμα  $R2 + R3 \rightarrow R1$ ). **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

### Πρόσδος 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 Α και Β 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/είδος** των εντέλων των αριθμητικών εντολών.

### Θέμα με μεταβλητό μήκος 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      // R1 ← #5
LDA R2, $90     // R2 ← $90
ADD R1, R2      // R1 ← R1 + R2
LDA R2, (R3)    // R2 ← (R3)
SUB R1, R2      // R1 ← R1 - R2
STORE R1, $200   // $200 ← R1
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  $2^x \geq \text{πλήθος εντολών}$ . 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:  $2^x \geq \text{πλήθος καταχωρητών} \Rightarrow 2^x \geq 32 \Rightarrow x = 5 \text{ bits}$ .

- 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

Μεταβλητό μήκος εντολής = 21 bits  $\rightarrow$  24 bits

opcode LDA	R1	#5	X
8 bits	5 bits	8 bits	3 bits αχρησιμοποίητο μέρος

**Χωρητικότητα**  
24 bits = 3 bytes

**Προσπελάσεις μνήμης**  
3 προσπελάσεις

opcode LDA	R2	\$90	X
8 bits	5 bits	16 bits	3 bits αχρησιμοποίητο μέρος

32 bits = 4 bytes

5 προσπελάσεις

Μεταβλητό μήκος εντολής = 18 bits  $\rightarrow$  24 bits

opcode ADD	R1	R2	X
8 bits	5 bits	5 bits	6 bits

24 bits = 3 bytes

3 προσπελάσεις

Μεταβλητό μήκος εντολής = 18 bits  $\rightarrow$  24 bits

opcode LDA	R2	R3	X
8 bits	5 bits	5 bits	6 bits

24 bits = 3 bytes

4 προσπελάσεις

Μεταβλητό μήκος εντολής = 18 bits  $\rightarrow$  24 bits

opcode SUB	R1	R2	X
8 bits	5 bits	5 bits	6 bits

24 bits = 3 bytes

3 προσπελάσεις

Μεταβλητό μήκος εντολής = 29 bits  $\rightarrow$  32 bits

opcode STORE	R1	\$200	X
8 bits	5 bits	16 bits	3 bits

32 bits = 4 bytes

5 προσπελάσεις

opcode END
8 bits

8 bits = 1 byte

1 προσπέλαση

24 προσπελάσεις

### Θέμα με μεταβλητό μήκος 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 διεύθυνσεις 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

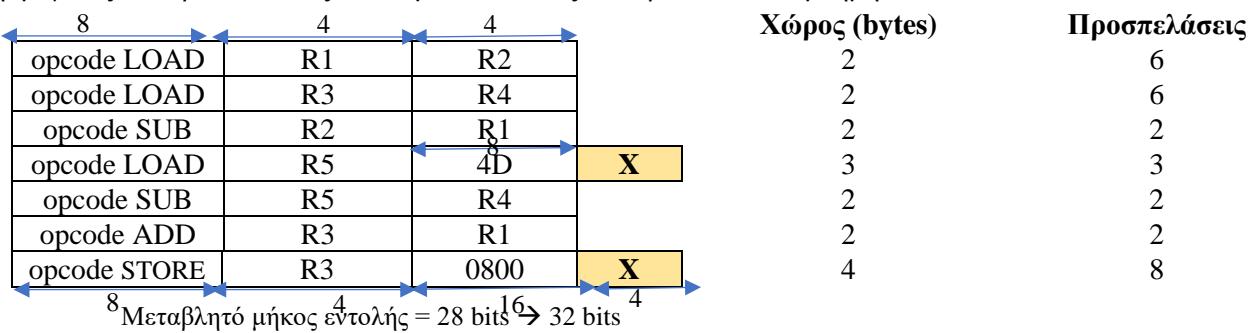
Χώρος που καταλαμβάνει η εντολή στην κύρια μνήμη σε bytes

Πλήθος προσπελάσεων κατά την εκτέλεση της εντολής

LOAD R1, (R2) /έμμεσος (indirect) τρόπος διεύθυνσιοδότησης με χρήση καταχωρητή			
LOAD R3, (R4)			
SUB R2, R1 /R2 – R1 → R2			
LOAD R5, #4D <sub>(16)</sub> /άμεσος (immediate) τρόπος διεύθυνσιοδότησης			
SUB R5, R4			
ADD R3, R1			
STORE R3, \$0800 κατ' ευθείαν (direct) τρόπος διεύθυνσιοδότησης			

### 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  $2^x \geq \text{πλήθος καταχωρητών} \Rightarrow 2^x \geq 16 \Rightarrow x = 4 \text{ bits}$ . Οι εντολές SUB, ADD και ADD έχουν παρόμοια μορφή, μέγεθος και προσπελάσεις και καμία από αυτές δεν προσπελαύνει μνήμη.



### Θέμα με σταθερό μήκος 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 διευθυνθούν** από κάθε μία των κάτωθι εντολών;

### Λύση

Τμήμα Προγράμματος Α	μορφή εντολής σε επίπεδο γλώσσας μηχανής	πλήθος θέσεων μνήμης που μπορεί να διευθυνθούν από την εντολή								
LOAD R, A // R ← A, όπου A διεύθυνση θέσης μνήμης (κατ' ευθείαν τρόπος διεύθυνσιοδότησης)	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>8 bits</td><td>4 bits</td><td>12 bits</td></tr> <tr><td>κωδικός λειτουργίας</td><td>R</td><td>διεύθυνση θέσης μνήμης</td></tr> </table>	8 bits	4 bits	12 bits	κωδικός λειτουργίας	R	διεύθυνση θέσης μνήμης	→ $2^{12}$ θέσεις		
8 bits	4 bits	12 bits								
κωδικός λειτουργίας	R	διεύθυνση θέσης μνήμης								
LOAD R1, (R2) // R1 ← M(R2), έμμεσος (indirect) τρόπος διεύθυνσιοδότησης με χρήση καταχωρητή	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>8 bits</td><td>4 bits</td><td>4 bits</td><td>8 bits</td></tr> <tr><td>κωδικός λειτουργίας</td><td>R1</td><td>R2</td><td>αχρησιμοποίητα</td></tr> </table>	8 bits	4 bits	4 bits	8 bits	κωδικός λειτουργίας	R1	R2	αχρησιμοποίητα	2 <sup>32</sup> θέσεις
8 bits	4 bits	4 bits	8 bits							
κωδικός λειτουργίας	R1	R2	αχρησιμοποίητα							
LOAD R1, d(R2) // R1 ← M[d + (R2)], έμμεσος (indirect) τρόπος διεύθυνσιοδότησης με χρήση καταχωρητή	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>8 bits</td><td>4 bits</td><td>4 bits</td><td>8 bits</td></tr> <tr><td>κωδικός λειτουργίας</td><td>R1</td><td>R2</td><td>d</td></tr> </table>	8 bits	4 bits	4 bits	8 bits	κωδικός λειτουργίας	R1	R2	d	2 <sup>32</sup> θέσεις
8 bits	4 bits	4 bits	8 bits							
κωδικός λειτουργίας	R1	R2	d							
LOAD R, #0AB <sub>(16)</sub> // άμεσος (immediate) τρόπος διεύθυνσιοδότησης	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>8 bits</td><td>4 bits</td><td>12 bits</td></tr> <tr><td>κωδικός λειτουργίας</td><td>R</td><td>δεδομένο</td></tr> </table>	8 bits	4 bits	12 bits	κωδικός λειτουργίας	R	δεδομένο	0		
8 bits	4 bits	12 bits								
κωδικός λειτουργίας	R	δεδομένο								
BRE d // σχετικός τρόπος διεύθυνσιοδότησης με χρήση του μετρητή προγράμματος, εάν η σημαία της ισότητας είναι 1, αλλαγή της ροής του προγράμματος	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>8 bits</td><td>16 bits</td></tr> <tr><td>κωδικός λειτουργίας</td><td>d</td></tr> </table>	8 bits	16 bits	κωδικός λειτουργίας	d	2 <sup>16</sup> θέσεις (με την έννοια ότι μπορεί να μεταφέρει τον έλεγχο σε 2 <sup>16</sup> διαφορετικές διεύθυνσεις)				
8 bits	16 bits									
κωδικός λειτουργίας	d									

## 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 32 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 διευθυνθούνται 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 A 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 <sub>(16)</sub> // άμεσος (immediate) τρόπος διευθυνσιοδότησης		

### Λύση

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

### 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 .

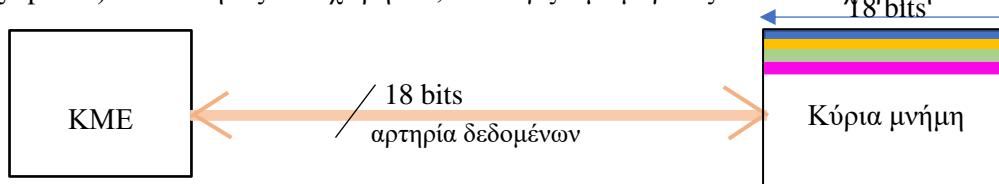
4. **LOAD1m R1, \$X** //κατευθείαν (direct) τρόπος διεύθυνσιο δότησης, το X είναι αριθμός των 16 δυαδικών ψηφίων και προσκόμιση περιεχομένου 1 θέσης μνήμης.

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 KME. 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 KME. Αν το μέγεθος της αρτηρίας δεδομένων ήταν 36 bits, αυτό σημαίνει ότι με κάθε προσπέλαση μνήμης θα μεταφερόντουσαν στην KME τα περιεχόμενα δύο θέσεων μνήμης, αν το μέγεθος της αρτηρίας δεδομένων ήταν 54 bits, αυτό σημαίνει ότι με κάθε προσπέλαση μνήμης θα μεταφερόντουσαν στην KME 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 KME 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 + 1)**.

### 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 (2 + 3)**.

### 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 (2 + 0)**.

### 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 (2 + 1)**.

### Θέμα Ιοννίου 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	R3	R1	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	R1	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  $\leftarrow$  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  $\rightarrow$  **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 γιατό 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 LOAD3m	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

Πλήθος προσπελάσεων που απαιτούνται για την προσκόμιση και την εκτέλεση της εντολής: 5

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 Y1, και Y2 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 Y1 όσο και στον Y2 οι εντολές είναι των 32 δυαδικών ψηφίων και είναι ευθυγραμμισμένες (aligned). Στον υπολογιστή Y1 **είναι και τα δεδομένα ευθυγραμμισμένα**, ενώ στον Y2 τα δεδομένα δεν είναι κατ' ανάγκη ευθυγραμμισμένα. Συγκεκριμένα, στον Y2 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 A, B, 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 Y1 και στον Y2.

### Πίνακας 2.10.1

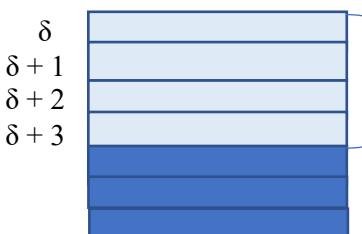
Είδος εντολής	Πλήθος ψηφιολέξεων δεδομένων που προσπελαύνει η εντολή κατά την εκτέλεσή της	Πλήθος εντολών που εκτελούνται
A	0	500
B	ένα δεδομένο της 1 ψηφιολέξης	1000
Γ	δύο δεδομένα των 2 ψηφιολέξεων το καθένα	600
Δ	ένα δεδομένο των 4 ψηφιολέξεων	2000

### Λύση

Έχουμε δύο υπολογιστές Y1 και Y2. 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 Y1, έχουμε πλήρη ευθυγράμμιση, για τον Y2 δεν έχουμε πλήρη ευθυγράμμιση για δεδομένα μεγέθους δύο και τεσσάρων ψηφιολέξεων. Σε αυτά τα δύο μεγέθη δεδομένων, έχουμε ευθυγράμμιση σε κάποιο ποσοστό.

$$T_1 \text{ (χρόνος εκτέλεσης προγράμματος στον Y1)} = [(1 \text{ εντ.} * 1 \text{ πρ.} + 0 \text{ δεδ.}) * 500 + (1 \text{ εντ.} * 1 \text{ πρ.} + 1 \text{ δεδ.} * 1 \text{ πρ.}) * 1000 + (1 \text{ εντ.} * 1 \text{ πρ.} + 2 \text{ δεδ.} * 1 \text{ πρ.}) * 600 + (1 \text{ εντ.} * 1 \text{ πρ.} + 1 \text{ δεδ.} * 1 \text{ πρ.}) * 2000] \text{ προσπελάσεις} * \frac{4 \text{ κ.ρ.}}{\text{προσπέλαση}} = 33200 \text{ κ.ρ.}$$

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



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

$$T_2 \text{ (χρόνος εκτέλεσης προγράμματος στον Y2)} = [(1 \text{ εντ.} * 1 \text{ πρ.} + 0 \text{ δεδ.}) * 500 + (1 \text{ εντ.} * 1 \text{ πρ.} + 1 \text{ δεδ.} * 1 \text{ πρ.}) * 1000 + (1 \text{ εντ.} * 1 \text{ πρ.} + 0.60 * 2 \text{ δεδ.} * 1 \text{ πρ.} + 0.40 * 2 \text{ δεδ.} * 2 \text{ πρ.}) * 600 + (1 \text{ εντ.} * 1 \text{ πρ.} + 0.30 * 1 \text{ δεδ.} * 1 \text{ πρ.} + 0.70 * 1 \text{ δεδ.} * 2 \text{ πρ.}) * 2000] \text{ προσπελάσεις} * \frac{4 \text{ κ.ρ.}}{\text{προσπέλαση}} = 40720 \text{ κ.ρ.}$$

**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 A, B, 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 ψηφιολέξεων δεδομένων που προσπελαύνει η εντολή κατά την εκτέλεσή της	Πλήθος εντολών που εκτελούνται
A	0	500
B	ένα δεδομένο των 2 ψηφιολέξεων	1100
Γ	ένα δεδομένο της 1 ψηφιολέξης και ένα δεδομένο των 2 ψηφιολέξεων	600
Δ	ένα δεδομένο των 4 ψηφιολέξεων	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

$$T = [(1\text{εντ.} \times 2 \text{ πρ.} + 0) \times 500 + (1\text{εντ.} \times 2 \text{ πρ.} + 0.75 \times 1 \text{ δεδ.} \times 1 \text{ πρ.} + 0.25 \times 1 \text{ δεδ.} \times 2 \text{ πρ.}) \times 1100 + (1 \text{ εντ.} \times 2 \text{ πρ.} + 1 \text{ δεδ.} \times 1 \text{ πρ.} + 0.75 \times 1 \text{ δεδ.} \times 1 \text{ πρ.} + 0.25 \times 1 \text{ δεδ.} \times 2 \text{ πρ.}) \times 600 + (1\text{εντ.} \times 2 \text{ πρ.} + (0.2 + 0.5) \times 2 \text{ πρ.} + 0.3 \times 3 \text{ πρ.}) \times 2000] \\ \text{προσπελάσεις} \times \frac{5 \text{ κ.ρ.}}{\text{προσπέλαση}}$$

Όταν έχουμε μη-ευθυγραμμισμένα δεδομένα ο αριθμός των προσπελάσεων θα είναι κατά "1" μεγαλύτερος από τον αριθμό των προσπελάσεων των ευθυγραμμισμένων δεδομένων:

**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 E δείχνει το εύρος του αριθμού, ενώ ο συντελεστής  $\Sigma_k$  δείχνει την ακρίβεια του αριθμού:  $\pi - \text{εκθέτης (E)} - \Sigma_k$ . Για να **μετατρέψουμε** έναν αριθμό από παράσταση κινητής υποδιαστολής σε σταθερή, πάντα ξεκινάμε από τον υπολογισμό του εκθέτη E, και ελέγχουμε σε ποιο διάστημα ανήκει, βάσει των τύπων (2.8) του βιβλίου.

### Παραδείγματα μετατροπής αριθμών από κινητή σε σταθερή υποδιαστολή

**0 10000111 10010000000000000000000000000000** Ζητούμενο είναι ο αντίστοιχος αριθμός σε σταθερή υποδιαστολή

### Λύση

Για τη μετατροπή του σε **σταθερή υποδιαστολή**, υπολογίζουμε αρχικά τον εκθέτη E, 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  $N = (-1)^{\pi} \times 2^{E-\text{πόλωση}} \times 1 \cdot \Sigma_k$ . Ο συντελεστής  $\Sigma_k$  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  $\Sigma_k = 10010000.....0 = 1 \times 2^{-1} + 1 \times 2^{-4} = 0.5 + 0.0625 = 0.5625$ . Επομένως, ο  $N = (-1)^\pi \times 2^{E-\text{πόλωση}} \times 1 \cdot \Sigma_k = (-1)^0 \times 2^{135-127} \times 1.5625 = 2^8 \times 1.5625 = 256 \times 1.5625 = 400$ .

**1 11001100 11000000000000000000000000000000** Ζητούμενο είναι ο αντίστοιχος αριθμός σε σταθερή υποδιαστολή

### Λύση

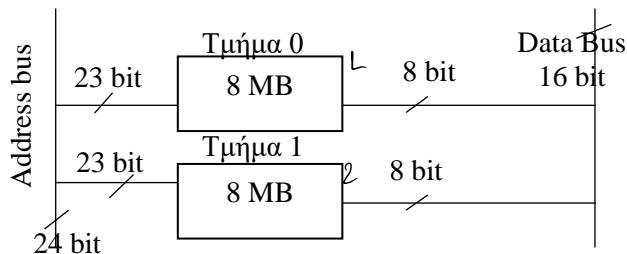
Για τη μετατροπή του σε **σταθερή υποδιαστολή**, υπολογίζουμε αρχικά τον εκθέτη  $E$ , οποίος είναι:  $E = 2^2 + 2^3 + 2^6 + 2^7 = 4 + 8 + 64 + 128 = 204 \rightarrow 0 < E = 204 < 255$ , οπότε για τη μετατροπή σε σταθερή υποδιαστολή ο τύπος είναι:  $N = (-1)^\pi \times 2^{E-\text{πόλωση}} \times 1 \cdot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

**0 10000010 11000000000000000000000000000000** Ζητούμενο είναι ο αντίστοιχος αριθμός σε σταθερή υποδιαστολή

### Λύση

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

## 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.

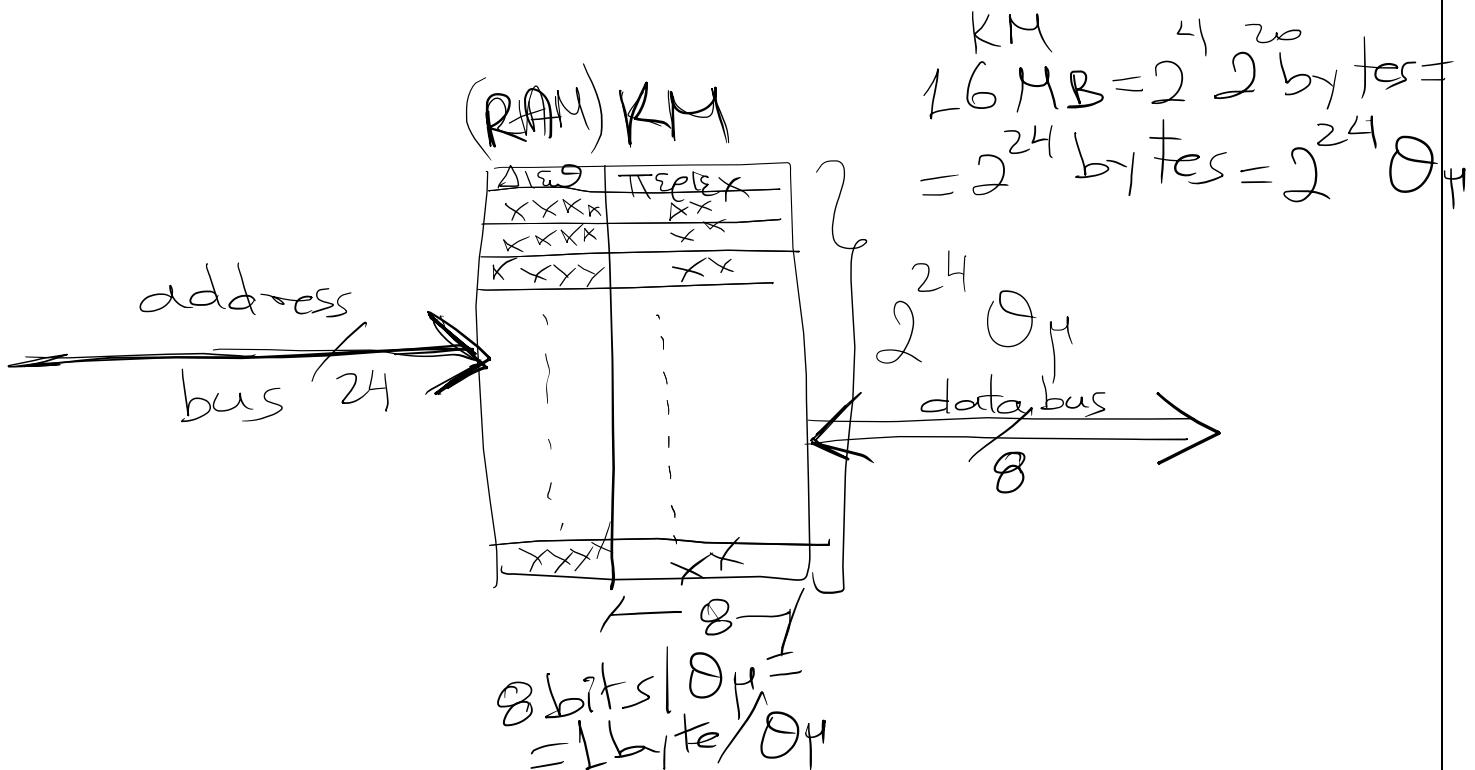
α. Για κάθε μία από τις εντολές σε συμβολική γλώσσα του Πίνακα 1 στην αντίστοιχη στήλη να σχεδιάσετε τη μορφή των εντολών σε επίπεδο γλώσσας μηχανής (δηλαδή τα πεδία από τα οποία αποτελείται κάθε εντολή σε επίπεδο γλώσσας μηχανής) και να δώσετε το εύρος κάθε πεδίου και τη σημασία του.

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.

**Σημείωση:** επειδή η χωρητικότητα της Κ.Μ. είναι 16 Mbytes και κάθε θέση μνήμης έχει χωρητικότητα 8 γραμμών (δυαδικών ψηφίων), έχουμε ότι η χωρητικότητα είναι  $16 \text{ MB} = 2^4 \times 2^{20} \text{ bytes} = 2^{24} \text{ bytes} = 2^{24} \text{ θ.μ.}$ , επομένως το address bus = 24 bits.





## 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 $\leftarrow$ A, όπου R καταχωρητής και A διεύθυνση θέση μνήμης (κατ' ευθείαν τρόπος διεύθυνσιοδότησης)	opcode 8 5 bytes	R A X 4 24 4	21 κ.ρ. x 3 πρ = 63 κ.ρ. διότι μεταφέρονται 5 bytes = 5 θ.μ. και για κάθε δύο bytes απαιτείται μια προσπέλαση μνήμης
LOAD-SB R1, (R2) /R1 $\leftarrow$ M(R2), R1, R2, καταχωρητές (έμμεσος (indirect) τρόπος διεύθυνσιοδότησης με χρήση καταχωρητή)	opcode 8 2 bytes	R <sub>1</sub> R <sub>2</sub> 4 4	21 κ.ρ. x 1 πρ = 21 κ.ρ. διότι μεταφέρονται 2 bytes = 2 θ.μ. και για αυτές χρειάζεται μόνο μια προσπέλαση μνήμης
STORE-LB R1, (A) /R1 $\leftarrow$ M(A), R1 καταχωρητής και A διεύθυνση θέσης μνήμης (έμμεσος (indirect) τρόπος διεύθυνσιοδότησης με χρήση της κύριας μνήμης)	opcode 8 5 bytes	R <sub>1</sub> A X 4 24	21 κ.ρ. x διότι μεταφέρονται 5 bytes = 5 θ.μ.
LOAD-UDB R, #0AB7 <sub>(16)</sub> /άμεσος (immediate) τρόπος διεύθυνσιοδότησης	opcode 8 4 bytes	R #0AB7 X 16 4	0 κ.ρ. δεν προσπελαύνεται κύρια μνήμη
LOAD R1, (R2) R1 $\leftarrow$ M(R2), R1, R2, καταχωρητές (έμμεσος (indirect) τρόπος διεύθυνσιοδότησης με χρήση καταχωρητή)	opcode 8 2 bytes	R <sub>1</sub> R <sub>2</sub> 4 4	21 κ.ρ. x 1 = 21 κ.ρ. διότι μεταφέρονται 2 bytes = 2 θ.μ. και για αυτές χρειάζεται μόνο μια προσπέλαση μνήμης
LOAD-SB R, #B7 <sub>(16)</sub> /άμεσος (immediate) τρόπος διεύθυνσιοδότησης	opcode 8 3 bytes	R #B7 X 8 4	0 κ.ρ. δεν προσπελαύνεται κύρια μνήμη
ADD R1, R2, R3 /R1 $\leftarrow$ R2+R3, πρόσθεσε τα περιεχόμενα των R2 και R3 και αποθήκευσε το αποτέλεσμα στον R1	opcode 8 3 bytes	R <sub>1</sub> R <sub>2</sub> R <sub>3</sub> X 4 4 4	0 κ.ρ. δεν προσπελαύνεται κύρια μνήμη

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 $V = 1, C_{out} = 1$	02 ← 51, και γίνεται λόγω επέκτασης προσήμου: 00000051, $V = 1, C_{out} = 1$
LOAD-UB 05, #FAB <sub>(16)</sub> Φορτώνεται στον καταχωρητή 05 ο αριθμός #FAB7	καταχωρητής 05: F0F0F0F0 $V = 1, C_{out} = 1$	05 ← FAB7, και γίνεται λόγω επέκτασης προσήμου: 0000FAB7, $V = 1, C_{out} = 1$
LOAD-UB 01, 300 <sub>(16)</sub> 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, $V = 1, C_{out} = 0$ .
LOAD-SB 08, #A3 <sub>(16)</sub>	καταχωρητής 08: F0F0F0FF $V = 1, C_{out} = 1$	08 ← A3, και γίνεται λόγω επέκτασης προσήμου: FFFFFFFA3, $V = 1, C_{out} = 1$
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, $V = 0, C_{out} = 1$
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 ← FFFFFFFC4, $V = 0, C_{out} = 1$
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)70000000 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
00000300	C2
00000450	51
00000451	05
00000551	F3
00000552	C0
00000570	12
	34
	78
	56
000700	C4
	08
000900	C4

$$\begin{array}{r}
 \text{E0000000} \quad 1110\ 0000\ 0000 \dots\dots 0000 \\
 + \\
 \text{20000000} \quad 0010\ 0000\ 0000 \dots\dots 0000 \\
 \hline
 \text{1)00000000} \quad 1)0000\ 0000\ 0000 \dots\dots 0000
 \end{array}$$

Από το παραπάνω άθροισμα (είτε απευθείας στο δεκαεξαεδικό ή στο δυαδικό σύστημα) προκύπτει  $K = C_{out} = 1$  και  $Y = V = \kappa_{v-1} \oplus \kappa_{v-2} = 1 \oplus 1 = 0$ . 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**.

- a) 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**.

- a) 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. Η συγκεκριμένη εντολή μπορεί να ανήκει σε αρχιτεκτονική που βασίζεται στη χρήση καταχωρητή – καταχωρητή με 2 ή 3 τελούμενα και καταχωρητή – μνήμης με 2 ή 3 τελούμενα.

#### 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**.

- a) **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**.

- a) 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. Η συγκεκριμένη εντολή μπορεί να ανήκει σε αρχιτεκτονική που βασίζεται στη χρήση καταχωρητή – καταχωρητή με 2 ή 3 τελούμενα και καταχωρητή – μνήμης με 2 ή 3 τελούμενα.

#### 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.

- a) 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 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 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 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

**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 ← A κ.ο.κ.

## Θέμα 9

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-καταχωρητή με τρία τελούμενα**.

α) Σωστό

β) Λάθος

### Λύση

**Λάθος**, ανήκει στην **αρχιτεκτονική συσσωρευτή**. Σε μια τέτοια αρχιτεκτονική που αναφέρεται στην εκφώνηση, η εντολή φόρτωσης 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.

### Άσκηση 2.6 από λυσάρι με αριθμούς κινητής υποδιαστολής

Δίνονται οι παρακάτω αριθμοί κινητής υποδιαστολής σύμφωνα με το πρότυπο IEEE 754:

α. 0 **10000010** 11000000000000000000000000000000

β. 0 **10000111** 10010000000000000000000000000000

γ. 1 **11001100** 11000000000000000000000000000000

Να υπολογίσετε ποιος είναι κάθε αριθμός στη δεκαδική μορφή που χρησιμοποιούμε στην καθημερινή μας ζωή.

## Λύση

α. Πάντα ξεκινάμε με τον υπολογισμό του εκθέτη, που έχει την τιμή  $E = 130 = 1 \times 2^7 + 1 \times 2^1$ , άρα δεν έχει ούτε τη μέγιστη τιμή, 255, αλλά ούτε και την ελάχιστη τιμή 0 και επομένως η συγκεκριμένη αναπαράσταση είναι η αναπαράσταση ενός κανονικοποιημένου αριθμού. Τότε η τιμή του δίνεται από τη σχέση  $N = (-1)^{\pi} \times 2^{E-127} \times (1. \Sigma_K)$ . Το κλασματικό μέρος του συντελεστή έχει την τιμή  $\Sigma_K = 11$ , επομένως  $1.\Sigma_K = 1.11_{(2)} = 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} = 1 + 0.5 + 0.25 = 1.75_{(10)}$  και  $N = (-1)^0 \times 2^{130-127} \times 1.75_{(10)} = 2^3 \times 1.75_{(10)} = 8 \times 1.75_{(10)} = 14_{(10)}$

β. Ο εκθέτης έχει την τιμή  $E = 135$ , άρα  $0 < 135 < 255$ . Επομένως η συγκεκριμένη αναπαράσταση είναι η αναπαράσταση ενός κανονικοποιημένου αριθμού και η τιμή του δίνεται από τη σχέση  $N = (-1)^{\pi} \times 2^{E-127} \times (1. \Sigma_K)$ . Το κλασματικό μέρος του συντελεστή έχει την τιμή  $\Sigma_K = 1001_{(2)} = 1 \times 2^{-1} + 1 \times 2^{-4} = 0.5 + 0.0625 = 0.5625$ , επομένως το  $1.\Sigma_K = 1.5625$  και  $N = (-1)^0 \times 2^{135-127} \times 1.1001_{(2)} = 2^8 \times 1.5625 = 400_{(10)}$ .

γ. Ο εκθέτης έχει την τιμή  $E = 204$ , άρα  $0 < 204 < 255$ . Επομένως η συγκεκριμένη αναπαράσταση είναι η αναπαράσταση ενός κανονικοποιημένου αριθμού και η τιμή του δίνεται από τη σχέση  $N = (-1)^{\pi} \times 2^{E-127} \times (1. \Sigma_K)$ . Το κλασματικό μέρος του συντελεστή έχει την τιμή  $\Sigma_K = 11$ , επομένως  $1.\Sigma_K = 1.11_{(2)} = 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} = 1 + 0.5 + 0.25 = 1.75_{(10)}$  και  $N = (-1)^1 \times 2^{204-127} \times 1.11_{(2)} = -2^{77} \times 1.75$ .

### Άσκηση 2.7 από λυσάρι με αριθμούς κινητής υποδιαστολής

Θεωρήστε το πρότυπο IEEE 754 για αναπαράσταση αριθμών κινητής υποδιαστολής απλής ακρίβειας. Ποιους αριθμούς αντιπροσωπεύουν οι ακόλουθες αναπαραστάσεις:

α. 0 **00000000** 00000000000000000000000000000000

β. 1 **00000000** 00000000000000000000000000000000

γ. 0 **11111111** 00000000000000000000000000000000

δ. 1 **11111111** 00000000000000000000000000000000

ε. 0 **11111111** 00111110000000000000000000000000

στ. 0 **10000000** 10000000000000000000000000000000

ζ. 0 **00000000** 01000000000000000000000000000000

**Υπενθύμιση.** Το πρότυπο απλής ακρίβειας της IEEE:

$N = NaN$  εάν  $E = 255$  και  $\Sigma_K \neq 0$ ,

$N = (-1)^{\pi} \infty$  εάν  $E = 255$  και  $\Sigma_K = 0$ ,

$N = (-1)^{\pi} \times 2^{E-127} \times (1. \Sigma_K)$  εάν  $0 < E < 255$  (κανονικοποιημένοι αριθμοί),

$N = (-1)^{\pi} \times 2^{126} \times (0. \Sigma_K)$  εάν  $E = 0$  και  $\Sigma_K \neq 0$  (μη κανονικοποιημένοι αριθμοί),

$N = (-1)^{\pi} 0$  εάν  $E = 0$  και  $\Sigma_K = 0$  (διπλή αναπαράσταση του μηδενός).

## Λύση

α. 0 00000000 00000000000000000000000000000000

β. 1 00000000 00000000000000000000000000000000

Παρατηρούμε ότι και στις δύο αναπαραστάσεις, τόσο ο εκθέτης  $E$  όσο και το κλασματικό μέρος του συντελεστή  $\Sigma_K$  έχουν τιμή 0, επομένως και οι δύο είναι αναπαραστάσεις του 0. Σύμφωνα με το πρότυπο IEEE 754 έχουμε δύο αναπαραστάσεις του μηδενός, όπου στην πρώτη, το πιο σημαντικό ψηφίο της αναπαράστασης, που δηλώνει το πρόσημο του αριθμού, έχει την τιμή μηδέν, ενώ στη δεύτερη έχει την τιμή 1. Πρόκειται για την τελευταία περίπτωση των τύπων (2.8), δηλαδή  $(-1)^{\pi} \times 0$ .

γ. 0 11111111 00000000000000000000000000000000

δ. 1 11111111 00000000000000000000000000000000

Παρατηρούμε ότι και στις δύο αναπαραστάσεις, ο εκθέτης  $E$  έχει τη μέγιστη τιμή 255 και το κλασματικό μέρος του συντελεστή  $\Sigma_K$  έχει τη μηδενική τιμή. Στην πρώτη περίπτωση το πιο σημαντικό ψηφίο της αναπαράστασης, που δηλώνει το



πρόσημο του αριθμού, έχει την τιμή μηδέν, ενώ στη δεύτερη έχει την τιμή 1. Επομένως, πρόκειται για τη δεύτερη περίπτωση των τύπων (2.8), δηλαδή  $(-1)^x \infty$  (αναπαράσταση του  $+\infty$  και του  $-\infty$ ).

$\varepsilon, 01111111\ 001111100000000000001000$

Παρατηρούμε ότι ο εκθέτης Ε έχει τη μέγιστη τιμή 255 ενώ το κλασματικό μέρος του συντελεστή Σκ έχει μη μηδενική τιμή. Επομένως η αναπαράσταση δεν αντιστοιχεί σε κάποιο αριθμό (NaN).

|στ. 0 10000000 10000000000000000000000000000

Ο εκθέτης έχει τιμή  $E=128$ , άρα  $0 < E < 255$  και πρόκειται για αναπαράσταση κανονικοποιημένου αριθμού. Τότε η τιμή του αριθμού είναι  $N = (-1)^{\Pi} \times 2^{E-127} \times (1.\Sigma_K)$ , όπου το κλασματικό μέρος του συντελεστή έχει την τιμή  $\Sigma_K = 1 = 1 \times 2^{-1} = 0.5$  και επομένως  $1.\Sigma_K = 1.1_{(2)} = 1.5_{(10)}$  και  $N = (-1)^0 \times 2^{128-127} \times 1.5_{(10)} = 2 \times 1.5_{(10)} = 3_{(10)}$ .

↪: 0 0000000 0100000000000000000000000000

Παρατηρούμε ότι ο εκθέτης έχει την ελάχιστη τιμή 0, ενώ το κλασματικό μέρος του συντελεστή  $\Sigma_K = 01 = 0 \times 2^{-1} + 1 \times 2^{-2} = 0.25$  και η αναπαράσταση αντιστοιχεί σε **μη κανονικοποιημένο αριθμό**. Τότε η τιμή του αριθμού δίνεται από τη σχέση:

$$N = (-1)^\pi \times 2^{-126} \times (0.\Sigma_K) \text{ or } N = (-1)^0 \times 2^{-126} \times 0.25_{(10)} = 2^{-126} \times 0.25_{(10)}$$

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

Λύση

Ένας **πρώτος** 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:  $N = (-1)^\pi * 2^{E-\text{πόλωση}} * 1.\Sigma_k$  οπότε για τον αριθμό A έχουμε ότι είναι:  $N = (-1)^0 * 2^{134-127} * 1.0625 = 2^7 * 1.0625 = 136$ , όπου  $\Sigma_k = 1/16 = 0.0625$  ενώ για τον αριθμό B έχουμε ότι είναι:  $N = (-1)^0 * 2^{133-127} * 1.1875 = 2^6 * 1.1875 = 76$ , όπου  $\Sigma_k = 1/8 + 1/16 = 0.1875$ . Το άθροισμά τους σε παράσταση σταθερής υποδιαστολής είναι **136 + 76 = 212**. Στη συνέχεια, πρέπει να μετατρέψουμε το άθροισμα αυτό πίσω σε αναπαράσταση κινητής υποδιαστολής, σύμφωνα με το πρότυπο IEEE 754. Θα έχουμε ότι  $212 = 1101\ 0100.0 \times 2^0 = 1.10101000 \times 2^7$ . Τώρα η πόλωση (127) πρέπει να **προστεθεί** στον εκθέτη, ο οποίος θα γίνει  $7 + 127 = 134 = 1000\ 0110$ . Ο συντελεστής  $\Sigma_k$  είναι το κλασματικό μέρος του αριθμού που συμπληρώνεται στο τέλος με «0» για να φτάσει τα 23 bits.

0 **1000 0110 101010000000000000000000**

**Σημείωση:** όταν ένας αριθμός μετατρέπεται από σταθερή σε κινητή υποδιαστολή, πρέπει πρώτα να **κανονικοποιηθεί**, δηλαδή να μετατραπεί στη μορφή **1.0000000**. Όσες θέσεις μετακινείται προς τα **αριστερά/δεξιά** η υποδιαστολή, τόσες τιμές αυξάνεται/μειώνεται αντίστοιχα ο εκθέτης.

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 δεξά 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  $A = 2^{134} \times 1.000100000000000000000000$  και ο  $B = 2^{133} \times 1.001100000000000000000000$ , οπότε ο  $B$  θα γίνει:  $B = 2^{134} \times 0.100110000000000000000000$  και στη συνέχεια οι αριθμοί μπορούν να αθροιιστούν αναφορικά με τους συντελεστές τους, αφού απέκτησαν **ίδιο** εκθέτη. Επομένως, από το άθροισμα των συντελεστών τους θα έχουμε:

**1.00010000000000000000000000**

0.10011000000000000000000000

1.1010100000000000000000000

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 1010**100000000000000000000000.

### Λύση Θέματος 3 – Φεβρουάριος 2014

Θεωρείστε επεξεργαστή απλού συνόλου εντολών (RISC) με 150 εντολές σε επίπεδο γλώσσας μηχανής **σταθερού** μήκους 32 δυαδικών ψηφίων και 64 καταχωρητές γενικού σκοπού των 32 δυαδικών ψηφίων ο καθένας.

a. Να γράψτε τη μορφή των κάτωθι εντολών και να βρείτε την περιοχή τιμών που μπορεί να λάβει το m.

ADD r1, r2, r3 //r1  $\leftarrow$  r2 + r3

LOAD r1, xxxx //r1  $\leftarrow$  M(xxxx)

LOAD r1, m(r2) //r1  $\leftarrow$  M(r2 + m)

b. Να υπολογίσετε το μέγεθος της μνήμης που μπορεί να διευθυνσιοδοτηθεί από κάθε μια εντολή LOAD.

#### Λύση

a. Από αριθμό εντολών υπολογίζουμε το μήκος του κωδικού λειτουργίας (opcode) και από το πλήθος των καταχωρητών γενικού σκοπού βρίσκουμε το μήκος κάθε καταχωρητή. Πιο συγκεκριμένα,  $2^x \geq 150 \Rightarrow x = 8$  bits και  $2^x \geq 64 \Rightarrow x = 6$  bits. Η πρώτη εντολή δεν προσπελαύνει μνήμη, γιατί δεν έχει καταχωρητές μέσα σε (). Όταν υπάρχει καταχωρητής μέσα σε (), όπως στην 3<sup>η</sup> εντολή, αυτό σημαίνει ότι το πλήθος των θέσεων μνήμης που μπορεί να διευθυνσιοδοτηθεί από την εντολή είναι  $2^{\text{μήκος καταχωρητή}} = 2^{32}$

Μήκος εντολής = 32 bits

opcode ADD	r1	r2	r3	X
8 bits	6 bits	6 bits	6 bits	6 bits αχρησιμοποίητο μέρος

Θέσεις μνήμης: 0

Μήκος εντολής = 32 bits

opcode LOAD	r1	xxxx
8 bits	6 bits	διεύθυνση 18 bits

Θέσεις μνήμης:  $2^{18}$

Μήκος εντολής = 32 bits

opcode LOAD	r1	r2	m
8 bits	6 bits	6 bits	12 bits

Θέσεις μνήμης:  $2^{32}$

Η περιοχή τιμών που μπορεί να λάβει το m προσδιορίζεται από τον τύπο: **ΜΠ –  $2^{m-1}$  έως ΜΠ +  $2^{m-1} - 1$** , η οποία στην προκειμένη περίπτωση διαμορφώνεται ως εξής: **ΜΠ –  $2^{12-1}$  έως ΜΠ +  $2^{12-1} - 1 = ΜΠ – 2^{11}$  έως ΜΠ +  $2^{11} - 1$** , όπου **ΜΠ = Μετρητής Προγράμματος**.

**Παρατήρηση:** Οι εντολές που δηλώνουν προσπέλαση μνήμης είναι τριών ειδών:

- Με το \$, π.χ. LOAD R1, \$0800
- Με () γύρω από καταχωρητές, π.χ. LOAD R1, (R2)
- Με εντολή του είδους LOAD R1, A, όπου A = διεύθυνση μνήμης.

#### Θέματα προόδου Νοεμβρίου 2022

1. 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 4 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  $\leftarrow$  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 / κατευθείαν (direct) τρόπος διεύθυνσιο δότησης, 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 πλήθος εντολών υπολογίζουμε το μήκος του κωδικού λειτουργίας της εντολής (opcode) που είναι  $2^x \geq 173 \Rightarrow x = 8$  bits. Επίσης, από πλήθος καταχωρητών, υπολογίζουμε το πλήθος των bits που χρειάζονται για την κωδικοποίηση του εντέλου που αφορά καταχωρητή και ισχύει ότι  $2^x \geq 33 \Rightarrow x = 6$  bits.

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

Opcode	R1	R2	Total
8	6	6	20 bits

Με δεδομένο ότι έχουμε 4 bits ανά θέση μνήμης, θα χρειαστούν 5 θέσεις μνήμης

Προσπελάσεις = 6 (5 για την προσκόμιση και 1 για την εκτέλεση).

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

Opcode	R1	R2	Total
8	6	6	20 bits

4 bits ανά θέση μνήμης άρα 5 θέσεις μνήμης και Προσπελάσεις = 8 (5 για την προσκόμιση και 3 για την εκτέλεση).

3. ADD R3, R1

Opcode	R3	R1	Total
8	6	6	20 bits

4 bit ανά θέση μνήμης, άρα 5 θέσεις μνήμης

Προσπελάσεις = 5 (5 για την προσκόμιση και 0 για εκτέλεση, αφού δεν υπάρχει προσπέλαση μνήμης).

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

Opcode	R1	X	Total
8	6	16	30 bits

4 bit ανά θέση μνήμης, άρα 8 θέσεις μνήμης

Προσπελάσεις = 9 (8 προσπελάσεις για την προσκόμιση και 1 για την εκτέλεση).

**Παρατήρηση:** σε περίπτωση που είχαμε κύρια μνήμη με 4 δυαδικά ψηφία ανά θέση μνήμης, και εύρος αρτηρίας δεδομένων 12 δυαδικών ψηφίων, τότε τα ερωτήματα θα είχαν τις εξής απαντήσεις:

- α. 5 θέσεις μνήμης και 3 προσπελάσεις (2 + 1)
- β. 5 θέσεις μνήμης και 3 προσπελάσεις (2 + 1)
- γ. 5 θέσεις μνήμης και 2 προσπελάσεις (2 + 0)
- δ. 8 θέσεις μνήμης και 4 προσπελάσεις (3 + 1)

2a. Η εντολή "ADD 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

2b. Η εντολή "ADD 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 μηχανισμού στοίβας.

- α) Σωστό
- β) Λάθος

### Λύση

Και τα δύο είναι λάθος

3. 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
102	71
103	E6
104	4B
105	90
106	54
107	F3
108	18
109	87
10A	29
10B	94
10C	3D
10D	87
10E	7C
10F	8A

111	F1
112	2B
113	F2
114	48
115	A7
116	79
117	92
118	79
119	F3
11A	5E
11B	93
11C	3C
11D	DF
11E	31
11F	EF

#### A. Απεικόνιση ανάλογη του μεγέθους (little endian)

LOADHW R1, 104 //Φόρτωσε μισή λέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διεύθυνσιοδότησης (direct addressing mode)

R1 = 904B

LOADSB R2, 104 //Φόρτωσε προσημασμένη ψηφιολέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διεύθυνσιοδότησης (direct addressing mode)

R2 = 004B

LOADUB R3, 104 //Φόρτωσε μη προσημασμένη ψηφιολέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διεύθυνσιοδότησης (direct addressing mode)

R3 = 004B

#### B. Απεικόνιση αντιστρόφως ανάλογη του μεγέθους (big endian)

LOADHW R4, 104 //Φόρτωσε μισή λέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διεύθυνσιοδότησης (direct addressing mode)

R4 = 4B90

LOADSB R5, 105 //Φόρτωσε προσημασμένη ψηφιολέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διεύθυνσιοδότησης (direct addressing mode)

R5 = FF90

LOADUB R6, 105 //Φόρτωσε μη προσημασμένη ψηφιολέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διεύθυνσιοδότησης (direct addressing mode)

R6 = 0090

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

#### Κατηγορίες αριθμών H/Y

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. **1101** → **1111** 1101 ή **01011010** → **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  $c_{v-1}, c_{v-2}, c_{out}, c_{out-1}, \kappa_{v-1}, \kappa_{v-2}$  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 **01011010** → **00000000** **01011010**

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.

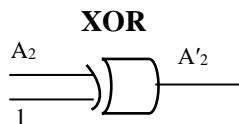
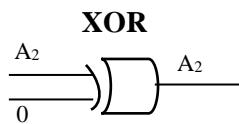
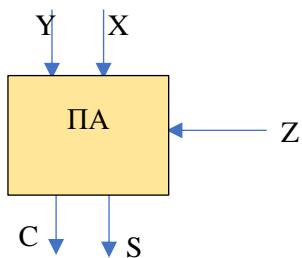
- Αν **K = 0**: **όχι** υπερχείλιση και το αποτέλεσμα είναι **σωστό**, διότι χωρά να αποθηκευτεί.

**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.

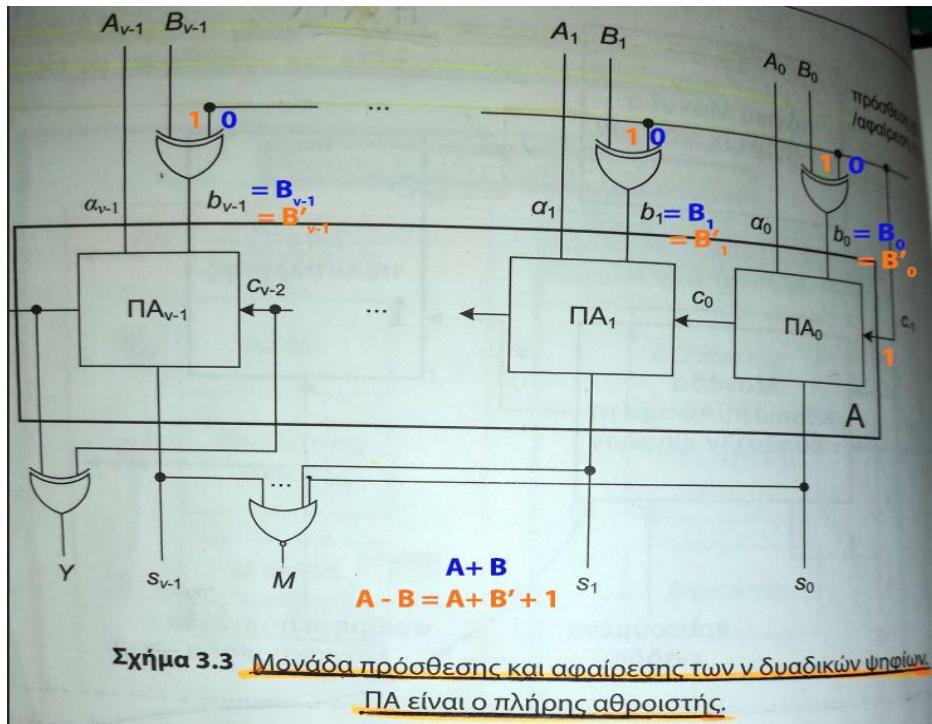
**Σημείωση 2:** Όταν χρειαστεί να κάνουμε **αφαίρεση μεταξύ προσημασμένων αριθμών**, η αφαίρεση αυτή θα γίνει μέσω πρόσθεσης, χρησιμοποιώντας το συμπλήρωμα ως προς «2». Πιο συγκεκριμένα,  $A - B = A + B' + 1$ . Η απλή αντιστροφή ενός αριθμού ( $B \rightarrow B'$ ) ονομάζεται συμπλήρωμα ως προς «1».

### Αθροισμα - 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:



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

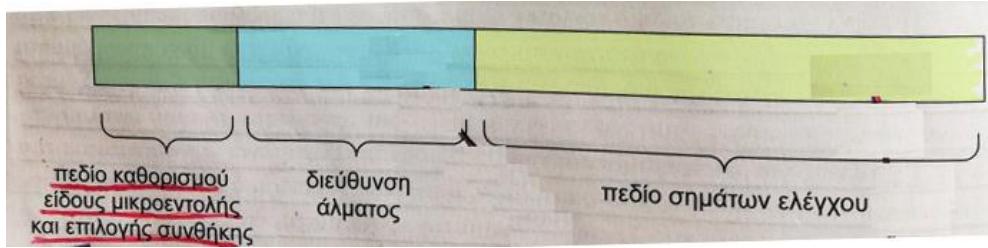
$$B = B_{v-1} B_{v-2} \dots B_1 B_0$$

**Σχήμα 3.3** Μονάδα πρόσθεσης και αφαίρεσης των ν δυαδικών ψηφίων.  
ΠΑ είναι ο πλήρης αθροιστής.

### Τεχνική μικροπρογραμματισμού – Τεχνικές μείωσης χωρητικότητας μνήμης ελέγχου

- Η μονάδα ελέγχου είναι ένα κύκλωμα που βρίσκεται μέσα στην KME και παράγει τα **σήματα ελέγχου**, ανάλογα με την εντολή που εκτελείται. Τα σήματα προσδιορίζουν τη λειτουργία συγκεκριμένων εξαρτημάτων της KME, όπως είναι η μνήμη, η ALM, 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**.
- Οταν 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 H/Y.
- 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 μικρο-εντολές) 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 ή 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 μικροπρόγραμμα.

### 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  $\alpha + \beta$ ,  $\gamma + \delta$ ,  $\alpha + \gamma$  και  $\beta + \delta$  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**

#### Αύση

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  $Y = \kappa_{v-1} \oplus \kappa_{v-2} = 1 \oplus 0 = 1 \rightarrow$  Το αποτέλεσμα είναι λάθος. Αυτό άλλωστε επαληθεύεται και ως εξής: αθροίσαμε δύο αριθμούς ομόσημους (αρνητικούς) και προέκυψε αποτέλεσμα ετερόσημο (θετικό)

- γ. 01111100

- δ. 01000001

101111101 Από την πράξη προκύπτει ότι το  $K = 0$ , το  $Y = \kappa_{v-1} \oplus \kappa_{v-2} = 0 \oplus 1 = 1 \rightarrow$  Το αποτέλεσμα είναι λάθος. Αυτό άλλωστε επαληθεύεται και ως εξής: αθροίσαμε δύο αριθμούς ομόσημους (θετικούς) και προέκυψε αποτέλεσμα ετερόσημο

- α. 10000110

- γ. 01111100

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

- β. 11000000

- δ. 01000001

1)00000001 Από την πράξη προκύπτει ότι το  $K = 1$ , το  $Y = \kappa_{v-1} \oplus \kappa_{v-2} = 1 \oplus 1 = 0 \rightarrow$  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. 111111110000110

- β. 111111111000000

- 1)1111111101000110

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

- γ. 000000001111100

- δ. 0000000001000001

- 0000000010111101

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

- α. 1111111110000110

- γ. 000000001111100

- 1)0000000000000010

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



β. 111111111000000

δ. 0000000001000001

1)0000000000000001

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

### Άσκηση 3.2 για άθροιση προσημασμένων αριθμών

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

Ενας 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  $n=32$  δυαδικών ψηφίων. Ο καταχωρητής κατάστασης μεταξύ των άλλων διαθέτει τις σημαίες K (κρατούμενο εξόδου), Y (υπερ-χείλιση,  $Y = \kappa_y \oplus \kappa_{y-1}$ ) 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       $\alpha = 1111111110000000$

$$\mu\alpha\varsigma\lambda\acute{e}\varsigma\eta\varsigma \quad \beta = 1111111111111110000000000000000$$

$$\delta = 011100$$

$$\varepsilon = 0001000$$

**Δεδομένα σε μορφή συμπληρώματος ως προς 2:**

μισής λέξης  $\zeta = 111111110000000$

μιας λέξης       $\eta = 1111111111111111000000000000000$

$$\theta = 1000$$

1<sup>η</sup> πράξη:  $\alpha + \beta$ , 2<sup>η</sup> πράξη:  $\beta + \gamma$ , 3<sup>η</sup> πράξη:  $\delta + \varepsilon$ , 4<sup>η</sup> πράξη  $\zeta + \eta$  και 5<sup>η</sup> πράξη:  $\eta + \theta$ .

Λύση

1η πράξη:  $\alpha + \beta$

Αφού οι αριθμοί είναι χωρίς πρόσημο η επέκταση προσήμου γίνεται πάντα με το «0» και ο α μετά την επέκταση θα γίνει:

$\alpha = 00000000000000001111111110000000$ , οπότε στη συνέχεια αθροίζεται με τον  $\beta$ :

$$\begin{array}{r}
 \alpha \quad 0000000000000000111111110000000 \\
 \beta \quad + 11111111111111110000000000000 \\
 \hline
 \alpha\pi \quad 1)0000000000000000111111110000000
 \end{array}$$

**K = 1, Y = κ<sub>v-1</sub> ⊕ κ<sub>v-2</sub> = 1 ⊕ 1 = 0, M = 0.** Υπολογίσαμε όλες τις σημαίες, όμως αφού οι αριθμοί **είναι χωρίς πρόσημο** η τιμή του κρατούμενου (K) είναι αυτή που πρέπει να ελεγχθεί καθώς δηλώνει την ύπαρξη υπερχείλισης. Επειδή **K = 1** έχουμε υπερχείλιση, επομένως το αποτέλεσμα **δεν γωρά** σε 32 δυαδικά ψηφία.

**2η πράξη:**  $\beta + \gamma$  (στην περίπτωση αυτή δεν χρειάζεται να γίνει επέκταση προσήμου, διότι οι αριθμοί έχουν ίδιο μέγεθος)

$$\begin{array}{ll} \beta & 11111111111111111000000000000000 \\ \gamma & +1000000000000000000000000000000 \\ \hline \alpha\pi & 1)011111111111111000000000000000 \end{array}$$

**K = 1, Y = κ<sub>v-1</sub> ⊕ κ<sub>v-2</sub> = 1 ⊕ 0 = 1, M = 0.** Υπολογίσαμε όλες τις σημαίες, όμως αφού οι αριθμοί **είναι χωρίς πρόσημο**, η τιμή του κρατούμενου είναι αυτή που πρέπει να εξεταστεί καθώς δηλώνει την ύπαρξη υπερχείλισης. Επειδή **K = 1** **έχουμε υπερχείλιση**, επομένως το αποτέλεσμα **δεν είναι σωστό**, αφού δεν χωρά σε 32 δυαδικά ψηφία.



### 3η πράξη: $\delta + \varepsilon$

$$\begin{array}{r} \delta \quad 01110000000000000000000000000000 \\ \varepsilon \quad + 00010000000000000000000000000000 \\ \hline \alpha \quad 10000000000000000000000000000000 \end{array}$$

**K = 0, Y = 1, M = 0** Υπολογίσαμε όλες τις σημαίες, όμως 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: $\zeta + \eta$

Αφού οι αριθμοί είναι σε **αναπαράσταση συμπληρώματος ως προς 2**, ο  $\zeta$  μετά την επέκταση προσήμου, που γίνεται τώρα με βάση το πιο σημαντικό ψηφίο (ΠΣΨ = MSB) θα γίνει:

$\zeta = 111111111111111111111111110000000$ , οπότε:

$$\begin{array}{r} \zeta \quad 111111111111111111111111110000000 \\ \eta \quad + 111111111111111111111111000000000 \\ \hline \alpha \quad 1) 11111111111111111111111101111111110000000 \end{array}$$

$K = 1, Y = \kappa_{v-1} \oplus \kappa_{v-2} = 1 \oplus 1 = 0, M = 0$ . 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: $\eta + \theta$

$$\begin{array}{r} \eta \quad 11111111111111111111111100000000000000 \\ \theta \quad + 100000000000000000000000000000000 \\ \hline \alpha \quad 1) 011111111111111111111111000000000000000 \end{array}$$

$K = 1, Y = \kappa_{v-1} \oplus \kappa_{v-2} = 1 \oplus 0 = 1, M = 0$ . 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  $\alpha + \beta$  και  $\alpha + \gamma$  με  $\alpha = 11100000, \beta = 00100000$  και  $\gamma = 10000000$  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  $\alpha, \beta$  και  $\gamma$  είναι αριθμοί χωρίς πρόσημο.

ii. Τα  $\alpha, \beta$  και  $\gamma$  είναι σε παράσταση συμπληρώματος ως προς 2.

### Λύση

$$\alpha = 11100000 \quad \textcolor{red}{K = 1, M = 1, Y = 0}$$

$$\beta = \underline{00100000} \quad + \quad \textcolor{blue}{1)00000000}$$

$$\alpha = 11100000 \quad \textcolor{red}{K = 1, M = 0, Y = 1}$$

$$\beta = \underline{00100000} \quad + \quad \textcolor{blue}{1)01100000}$$

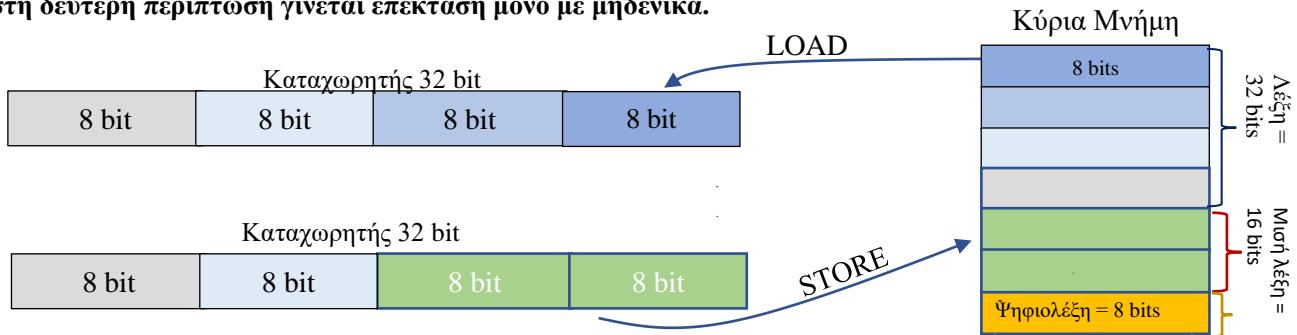


- i. Επειδή οι αριθμοί είναι **χωρίς πρόσημο**, κοιτάμε τη σημαία (flag) του Κ για να αποφανθούμε για την ορθότητα της πράξης. Αφού το  $K = 1$ , η πράξη είναι **λάθος** και στις δύο αθροίσεις, αφού δεν χωρά να αποθηκευτεί.
- ii. Επειδή οι αριθμοί είναι **με πρόσημο**, κοιτάμε τη σημαία (flag) του Y 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. Επίσης, αν φορτώσουμε ένα καταχωρητή των 16 bits με μια μισή λέξη, θα γεμίσει πλήρως και δεν έχει σημασία τότε αν η μισή λέξη είναι προσημασμένη ή μη – προσημασμένη (θετική), κάτι που σημαίνει ότι οι εντολές που θα χρησιμοποιηθούν για να γεμίσουν τον καταχωρητή θα είναι ίδιες και θα μπορεί η μία να αντικαταστήσει την άλλη, π.χ. οι εντολές LOADUH (ή LOADPH) και LOADSH είναι ισοδύναμες και ως εκ' τούτου αντικαθιστά η μία την άλλη, όταν χρησιμοποιούνται για να γεμίσουν ένα καταχωρητή των 16 bits. Αντίθετα όταν ο καταχωρητής αυτός γεμίζει με μια ψηφιολέξη (byte), επειδή ακριβώς δεν γεμίζει πλήρως και θα πρέπει να γίνει επέκταση προσήμου προκειμένου να γεμίσει πλήρως, οι εντολές LOADUB (ή LOADPB) και LOADSB δεν είναι ισοδύναμες, οπότε δεν μπορεί να αντικαταστήσει η μία την άλλη. **Επομένως, το κριτήριο στην εντολή LOAD του να έχουμε εντολές ισοδύναμες ή όχι είναι αν γεμίζει πλήρως ο καταχωρητής ή όχι.**

### 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 από κάποιον καταχωρητή προς την Κύρια Μνήμη, έχει σημασία το **μέγεθος των δεδομένων** που αποθηκεύονται και όχι το είδος τους. Επομένως, για δεδομένα ίδιου μεγέθους, οι εντολές είναι ισοδύναμες. Αυτό σημαίνει ότι οι εντολές STOREBP και STOREBS είναι ισοδύναμες, επίσης οι εντολές STOREHP και STOREHS είναι ισοδύναμες και τέλος οι εντολές STOREWP και STOREWS είναι ισοδύναμες.

**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»).

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

Οι εντολές **LOADBP** (ή **LOADBU**) και **LOADBS** όταν χρησιμοποιούνται για να γεμίσουν έναν καταχωρητή 16 – bit/32 – bit, **δεν** είναι **ισοδύναμες** (ίδιες), λόγω της αναμενόμενης επέκτασης προσήμου, που θα λάβει χώρα και η οποία γίνεται με διαφορετικό τρόπο σε κάθε περίπτωση.

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 και η οποία γίνεται με διαφορετικό τρόπο σε κάθε περίπτωση. Σε περίπτωση που ο καταχωρητής είναι **των 16 – bit**, τότε οι δύο εντολές θεωρούνται μεταξύ τους **ισοδύναμες**, γιατί τότε δεν γίνεται καμία επέκταση προσήμου.

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.

## 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 //R1 ← R1 + R2) είτε τρία τελούμενα (π.χ. ADD R1, R2, R3 //R1 ← R2 + R3) οι σύγχρονοι επεξεργαστές διαθέτουν καταχωρητές με **δύο θύρες ανάγνωσης και μία θύρα εγγραφής** και αυτό αιτιολογείται από τη **δυνατότητα** εκτέλεσης εντολών των προαναφερόμενων ειδών. Για παράδειγμα από τη πόρτα (θύρα) ανάγνωσης διαβάζουμε τον καταχωρητή R1, από τη δεύτερη πόρτα ανάγνωσης τον R2 και αποθηκεύουμε το αποτέλεσμα (άθροισμα) στον καταχωρητή R1 που είναι στην πόρτα (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  $C_{out} = 0$
- 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) **για 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 A και B που λαμβάνουν μέρος στην πράξη A – B είναι και οι δύο θετικοί. Επομένως, **είναι δυνατόν να προκύψει υπερχείλιση από την εκτέλεση της συγκεκριμένης πράξης**, χωρίς όμως αυτό να είναι βέβαιο.

#### 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 **A-B** 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 A, B που λαμβάνουν χώρα στην πράξη A – B 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 ίδιες εντολές.

#### 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 ανξήσουμε 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 ανξάνουμε το πλήθος των ακέραιων 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  $\leftrightarrow$  0111,10 = 7.5**

Αύξηση ακεραίου μέρους αριθμού σε παράσταση σταθερής υποδιαστολής  $\rightarrow$  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  $\rightarrow$  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 ανξήσουμε 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 ανξήσουμε 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**

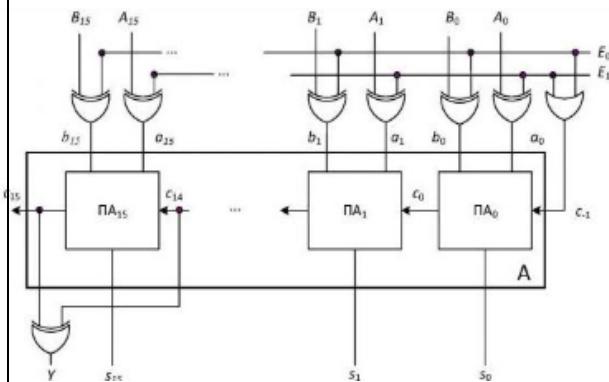
**Λύση**

**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  $\leftrightarrow$  0111,10  $\leftrightarrow$  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  $S_{15}, S_{14}, \dots, S_0$  του αθροιστή.



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:

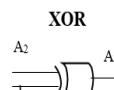
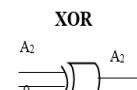
Υπενθυμίζουμε ότι στις πύλες XOR ισχύει ότι:

**E<sub>0</sub> = 0 και E<sub>1</sub> = 0  $\rightarrow$  A + B (ή B + A).**

**E<sub>0</sub> = 0 και E<sub>1</sub> = 1  $\rightarrow$  B - A = B + A' + 1**

**E<sub>0</sub> = 1 και E<sub>1</sub> = 0  $\rightarrow$  A - B = A + B' + 1**

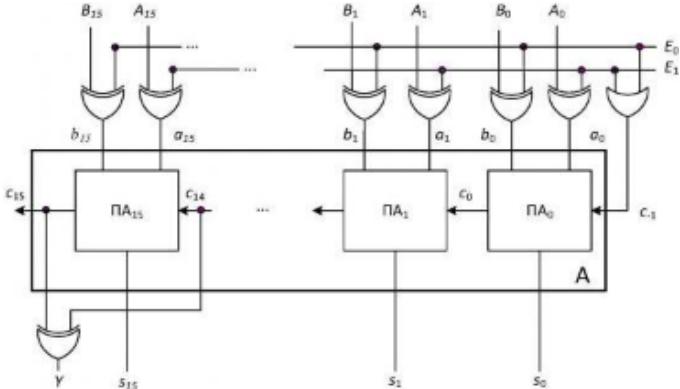
**E<sub>0</sub> = 1 και E<sub>1</sub> = 1  $\rightarrow$  -A - B (ή -B - A)**



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

### Θέμα 2

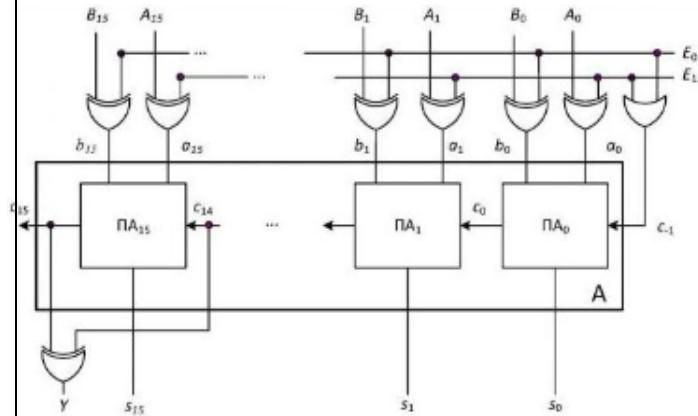
Η κεντρική μονάδα επεξεργασίας επεξεργαστή διαθέτει τον αθροιστή του επόμενου σχήματος. Να απαντήσετε εάν είναι σωστό το ακόλουθο ερώτημα: Για την πράξη  $B + A$  εάν οι αριθμοί είναι σε **αναπαράσταση συμπληρώματος ως προς 2** και  $Y = 1$  τότε το αποτέλεσμα της πράξης δεν ισούται με την τιμή στις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

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

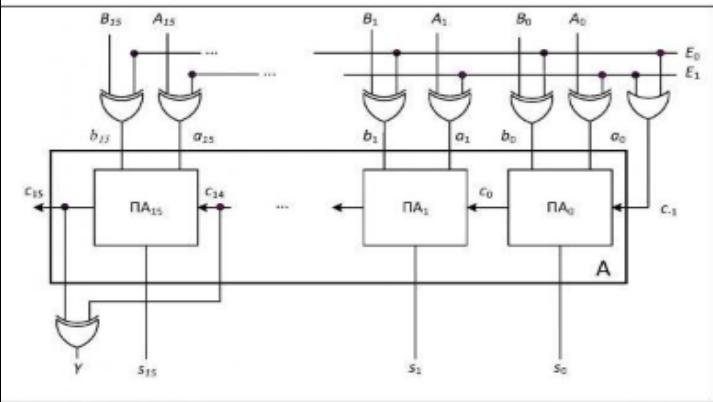
### Θέμα 5

Η κεντρική μονάδα επεξεργασίας επεξεργαστή διαθέτει τον αθροιστή του παραπάνω 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  $S_{15}, S_{14}, \dots, S_0$  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  $S_{15}, S_{14}, \dots, S_0$  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 προσημα- σμένα (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 κατα- χωρητές 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 πληροφο- ρία γράφεται στο λιγότερο σημαντικό τμήμα του καταχωρητή. Να απαντήσετε στην ακόλουθη ερώτηση: **Δεν μπορεί να χρησιμο- ποιεί την ίδια εντολή**, σε επίπεδο γλώσσας μηχανής, για τη μεταφορά πληροφορίας **από τη μνήμη σε κάποιο από τους καταχω- ρητές** 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 προσημα- σμένα (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 κατα- χωρητές 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 πληροφο- ρία 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  $n = 32$  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 μεταξύ 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 <sub>(16)</sub> , 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 <sub>(16)</sub> , 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<sub>↑</sub>UB r<sub>↑</sub>, \$8001", "LOAD-SB r<sub>2</sub>, \$8002", "LOAD-UB r<sub>3</sub>, \$8003" και "LOAD-SB r<sub>4</sub>, \$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 στους μη – προσημασμένους αριθμούς. Πιο συγκεκριμένα:

$r1 \leftarrow 0000006F$  (0000000.....000001101111),  $r2 \leftarrow FFFFFF80$  (1111111.....111110000000)

$r3 \leftarrow 000000C0$  (0000000.....000011000000),  $r4 \leftarrow 0000000E$  (0000000.....00000001110).

## 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

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  $v=32$  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 μισής λέξης, 2. προσημασμένα δεδομένα μεγέθους μισής 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.



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

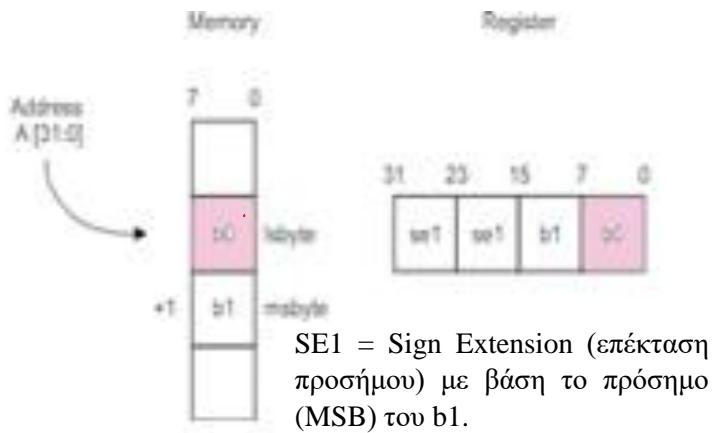
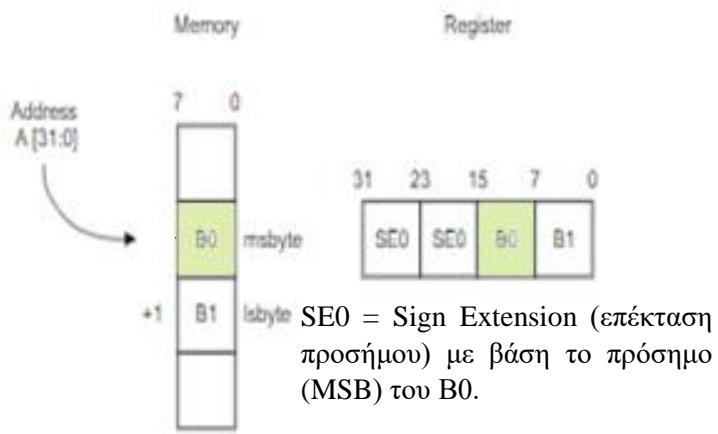
### Load word, big-endian



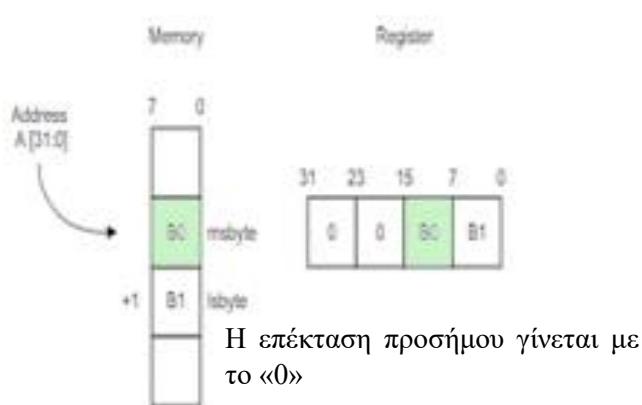
### Load word, little-endian



### Load signed halfword, big-endian



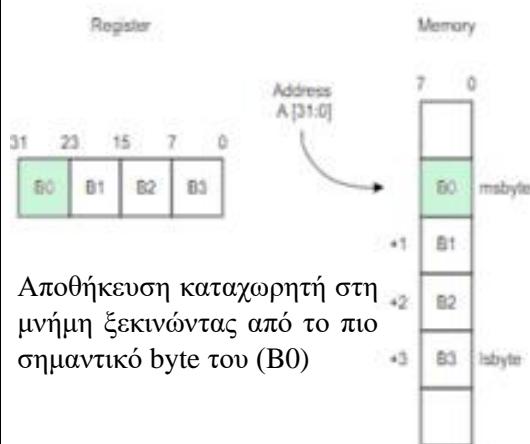
### Load unsigned halfword, big-endian



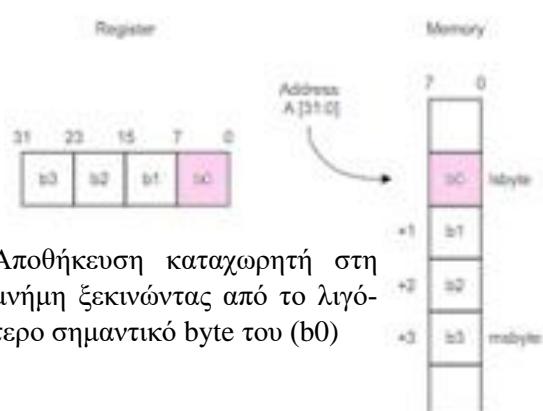
### Load unsigned halfword, little-endian



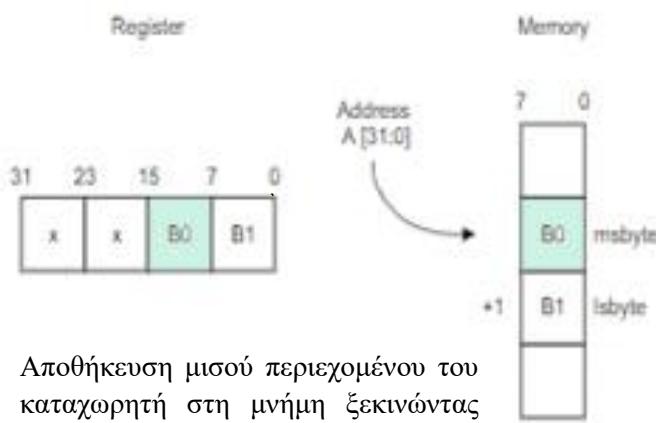
## Store word, big-endian



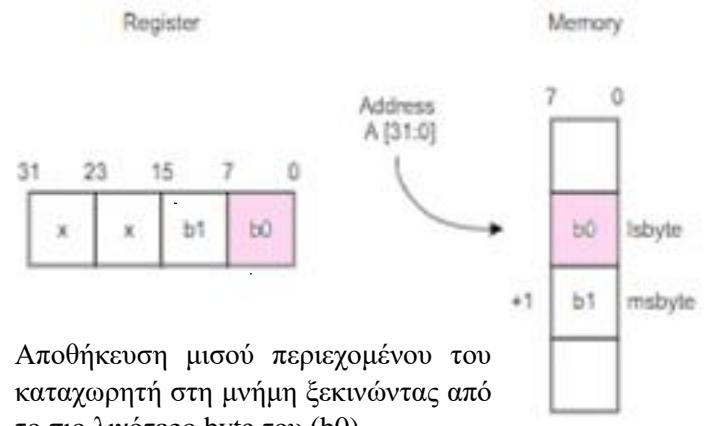
## Store word, little-endian



## Store halfword, big-endian



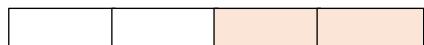
## Store halfword, little-endian



### 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 μισή λέξη
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 <b>πριν</b> την εκτέλεση των εντολών	Περιεχόμενο θέσης μνήμης στο δεκαεξαδικό <b>μετά</b> την εκτέλεση των εντολών
100	87	87
101	00	00
102	11	11
103	E3	E3
104	E4	E4
105	00	00
106	AB	AB
107	F0	F0
108	01	01
109	80	80
10A	07	07
10B	32	11
10C	AE	6A
10D	BC	23
10E	10	AE
10F	C4	00
110	89	24
111	1A	56
112	A0	A0

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

εντολή	Καταχωρητής	Περιεχόμενο του καταχωρητή στο δεκαεξαδικό <b>πριν</b> την εκτέλεση της εντολής	Περιεχόμενο του καταχωρητή στο δεκαεξαδικό <b>μετά</b> την εκτέλεση της εντολής
LOADSB R1, 10A <sub>(16)</sub>	R1=	0F 0F 0F 0F	00000007
LOADSHW R2, 103 <sub>(16)</sub>	R2=	0F 0F 0F 0F	FFFFE3E4
LOADW R3, 105 <sub>(16)</sub>	R3=	0F 0F 0F 0F	00ABF001
LOADUB R4, 106 <sub>(16)</sub>	R4=	0F 0F 0F 0F	000000AB
LOADUHW R5, 103 <sub>(16)</sub>	R5=	0F 0F 0F 0F	0000E3E4
STOREB 10B <sub>(16)</sub> , R6	R6=	00 FF 00 11	00 FF 00 11
STOREH 110 <sub>(16)</sub> , R7	R7=	FF 09 24 56	FF 09 24 56
STOREW 10C <sub>(16)</sub> , R8	R8=	6A 23 AE 00	6A 23 AE 00
LOADSB R9, 109 <sub>(16)</sub>	R9=	0F 0F 0F 0F	FFFFFFFFFF80

## 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.

κώδικα μνήμης		κώδικα μνήμης	
διεύθυνση θέσης μνήμης στο δεκαεξαδικό	παρεχόμενο θέσης μνήμης στο δεκαεξαδικό	διεύθυνση θέσης μνήμης στο δεκαεξαδικό	παρεχόμενο θέσης μνήμης στο δεκαεξαδικό
110	87	227	98
111	13	228	F8
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, 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

#### **A. 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 διεύθυνσιού της  
(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.

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

$R1 \leftarrow FFFF8713$ , διότι  $8 = 1000$  (αν είγαμε LOADUHW R1, 110 τότε  $R1 \leftarrow 00008713$ )

$R2 \leftarrow 000041A2$ , διότι 4 = 0100 (αν είγαμε LOADUHW R2, 22B τότε  $R2 \leftarrow 000041A2$ )

## 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  $\leftarrow$  F22B, επειδή οι καταχωρητές είναι των 16 bits δεν γίνεται καμία επέκταση προσήμου.

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

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

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

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

R6  $\leftarrow$  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		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

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

B. 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 = 1000 (αν είχαμε LOADUHW R1, 110 τότε R1 ← 00008713)

R2 ← 000041A2, διότι 4 = 0100 (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		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, 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 το περιεχόμενο του καταχωρητή μετά την εκτέλεση της εντολής στο δεκαεξαδικό. Προσοχή να μην παρεμβάλλονται κενά ή άλλα σύμβολα.

κίνηση μετάμματος		κίνηση μετάμματος	
διεύθυνση μετάμματος	παραγόμενο μέσης μετάμματος	διεύθυνση μετάμματος	παραγόμενο μέσης μετάμματος
110	87	227	98
111	13	228	F4
112	E1	229	F9
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	A1
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

A. Απεικόνιση αντιστρόφων ανάλογη του μεγέθους (big endian) LOADSHW R1, 128 //Φόρτωσε προσημασμένη μισή λέξη, χρησιμοποιείται ο κατ' ευθείαν τρόπος διευθυνσιοδότησης (direct addressing mode) R1 = \_\_\_\_\_.

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

### Λύση

FFFFFB332, διότι B = 1011  
000046AD, διότι 4 = 0100

### Ασκηση 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  $\beta + \delta$  στον υπολογιστή και να σχολιάσετε το αποτέλεσμα.
3. Να δώσετε την παράσταση  $\epsilon = BF0091_{(16)}$  σε παράσταση κινητής υποδιαστολής.

### Λύση

1.  $\alpha = +2A = 0010\ 1010$ , προσημασμένος αριθμός, είναι θετικός, αφού ξεκινά με “0”. Σωστή αναπαράσταση.  
 $\beta = -7E \rightarrow 7E = 0111\ 1110 \rightarrow 1000\ 0001 + 1 = 1000\ 0010$ , προσημασμένος, είναι αρνητικός, αφού ξεκινά με “1”. Σωστή αναπαράσταση

$\gamma = +F3 = 1111\ 0011$  δεν μπορεί να αναπαρασταθεί σωστά με 8 bits, διότι αφού είναι θετικός, θα έπρεπε να ξεκινά με “0”. Λάθος αναπαράσταση. Για τη σωστή αναπαράστασή του απαιτούνται περισσότερα από 8 bits.

$\delta = +1A9F$  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. Λάθος αναπαράσταση.

2. Επειδή ο αθροιστής είναι των 16 bits θα πρέπει οι αριθμοί που θα προστεθούν μέσω αυτού, να διαθέτουν αυτό το μέγεθος.  
 $\beta = -7E = 1000\ 0010 \rightarrow \underline{\underline{1111\ 1111}}\ 1000\ 0010$   
 $\delta = 1A9F \rightarrow \underline{\underline{0001\ 1010}}\ 1001\ 1111$   
 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 κάποιο υπολογισμό, 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**

$= 1011\ 1111\ 0000\ 0000\ 1001\ 0001 \times 2^0 = 1.011\ 1111\ 0000\ 0000\ 1001\ 0001 \times 2^{23}$  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  $1 \times 2^7 + 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^1 = 150$ . Η τελική απάντηση είναι:

1	10010110	011 1111 0000 0000 1001 0001
πρόσημο	εκθέτης	

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.

### 1<sup>η</sup> Ασκηση 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;

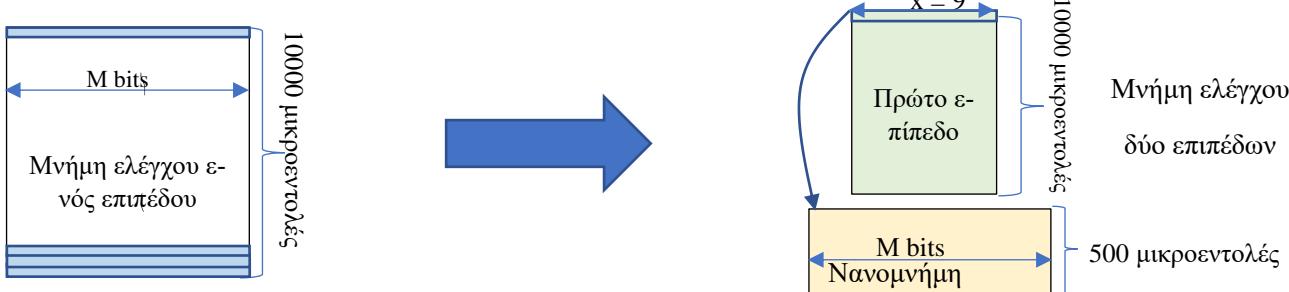
β) Καθορίστε τον αριθμό των δυαδικών ψηφίων που πρέπει να έχει η μικροεντολή προκειμένου να επιτυγχάνουμε μείωση του απαιτούμενου αποθηκευτικού χώρου με τη χρήση του 2<sup>ου</sup> επιπέδου κατά τουλάχιστον i) 50%, ii) 75% και iii) 90%.

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 συγχάρι 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 M. Τότε για την υλοποίηση της μνήμης ελέγχου ενός επιπέδου απαιτούνται **10.000 x M 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 έχει χωρητικότητα  $500 \times M$  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 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  $10.000 \times 9 + 500 \times M$

δυαδικά ψηφία. Ο λόγος μείωσης του απαιτούμενου αποθηκευτικού χώρου για τις δύο τεχνικές είναι:  $\Lambda = \frac{\text{αρχική χωρητικότητα} - \text{τελική χωρητικότητα}}{\text{αρχική χωρητικότητα}} = \frac{10000 \times M - (10.000 \times 9 + 500 \times M)}{10000 \times M}$ .

$$\text{i) Πρέπει } \Lambda \geq 50\% \Rightarrow \frac{10000 \times M - (10.000 \times 9 + 500 \times M)}{10000 \times M} \geq 0.5 \Rightarrow M \geq 20.$$

ii) Ομοίως για  $\Lambda \geq 75\%$ , πρέπει  $M \geq 45$  και iii) Ομοίως για  $\Lambda \geq 90\%$ , πρέπει  $M \geq 180$ .

## 2<sup>η</sup> Ασκηση με τεχνική μικροπρογραμματισμού δύο επιπέδων – Θέμα Σεπτέμβριος 2021

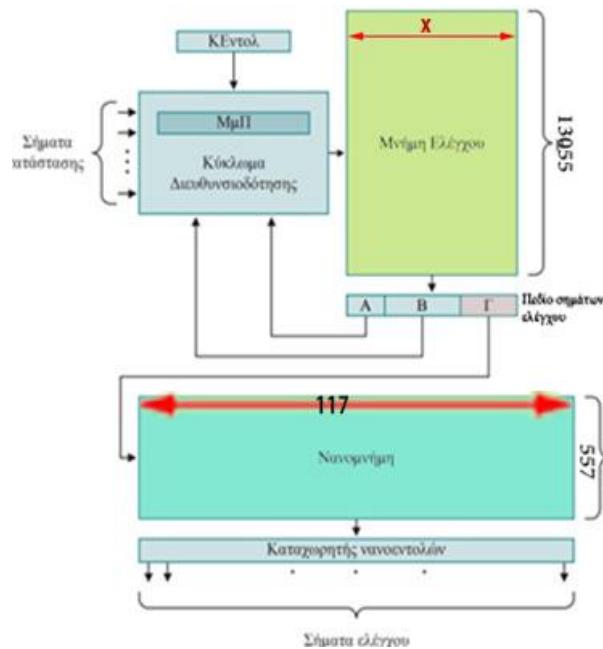
Θεωρήστε ένα μικροεπεξεργαστή με μονάδα ελέγχου που έχει σχεδιαστεί με την τεχνική του μικροπρογραμματισμού δύο επιπέδων (νανοπρογραμματισμός). Το σύνολο των μικροπρογραμμάτων αποτελείται από 13055 μικροεντολές και υπάρχουν 557 διαφορετικές μικροεντολές. Το **μήκος κάθε μικροεντολής είναι 119 δυαδικά ψηφία** εκ' των οπίων **τα 117 είναι σήματα ελέγχου**. Να υπολογίσετε την επί τοις εκάτο 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 τιμής 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 2<sup>o</sup> 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 1<sup>o</sup> επιπέδου + Χωρητικότητα 2<sup>o</sup> επιπέδου = 13055 \* (10 + 2) bits + 557 μικροεντολές \* 117 bits = 221829 bits**. Δηλαδή, στην περίπτωση χρήσης της νανομνήμης (δύο επιπέδων) για να υπολογίσουμε τη χωρητικότητα του 1<sup>o</sup> 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)) = 10 + (119 – 117) = 12 bits και για να υπολογίσουμε τη χωρητικότητα του 2<sup>o</sup> επιπέδου θα πρέπει να πολλαπλασιάσουμε το πλήθος των διαφορετικών μικροεντολών με τα bits (σήματα) ελέγχου, δηλ. 557 \* 117 bits.



Το ποσοστό μείωσης του απαιτούμενου αποθηκευτικού χώρου με τη χρήση της τεχνικής της νανομνήμης είναι:  $\Lambda = \frac{\text{αρχική χωρητικότητα} - \text{τελική χωρητικότητα}}{\text{αρχική χωρητικότητα}} = \frac{1553545 - 221829}{1553545} = 0.857 = 85.7\% \rightarrow 85\%$ .

**Σημείωση:** σε τέτοιου είδους ασκήσεις ισχύει ο τύπος: Για υλοποίηση με 1 επίπεδο:  $\alpha = \text{συνολικές εντολές} * \text{μέγεθος εντολής}$ , ενώ για 2 επίπεδα:  $\beta = (\text{μέγεθος εντολής} - \text{σήματα ελέγχου} + \text{ceiling}(\log_2(\text{διαφορετικές εντολές}))) * \text{συνολικές εντολές} + \text{διαφορετικές εντολές} * \text{σήματα ελέγχου}$ . Το πρώτο μέλος του αθροίσματος είναι το πρώτο επίπεδο, και αντίστοιχα για το δεύτερο. Άρα η μείωση είναι  $(\alpha - \beta)/\alpha$ .

## 3<sup>η</sup> Ασκηση με τεχνική μικροπρογραμματισμού δύο επιπέδων από ΕΑΠ

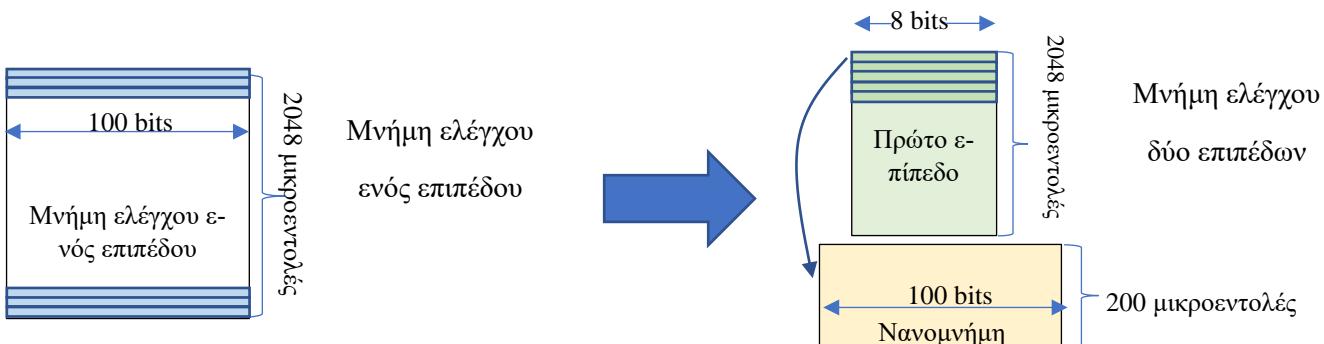
Θεωρήστε ότι σε ένα επεξεργαστή η μονάδα ελέγχου έχει σχεδιαστεί με την τεχνική του μικροπρογραμματισμού ενός επιπέδου. Θεωρήστε ότι **κάθε μικροεντολή είναι των 100 δυαδικών ψηφίων**, το σύνολο των μικροπρογραμμάτων καταλαμβάνει 2048

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 θέσεις μνήμης ελέγχου\*  $100 \frac{\text{δυαδικά ψηφία}}{\text{θέση μνήμης ελέγχου}} = 204800$

**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  $2048 * 8 \text{ bits} + 200 * 100 \text{ bits} = 36384$  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.



Κάθε θέση της μικρομνήμης αποθηκεύει και μια μικροεντολή. Επομένως, οι μικροεντολές είναι τόσες όσες και οι θέσεις της μνήμης ελέγχου, συνολικά 2048

### ✗ Ασκηση με τεχνική μικροπρογραμματισμού δύο επιπέδων από το μάθημα

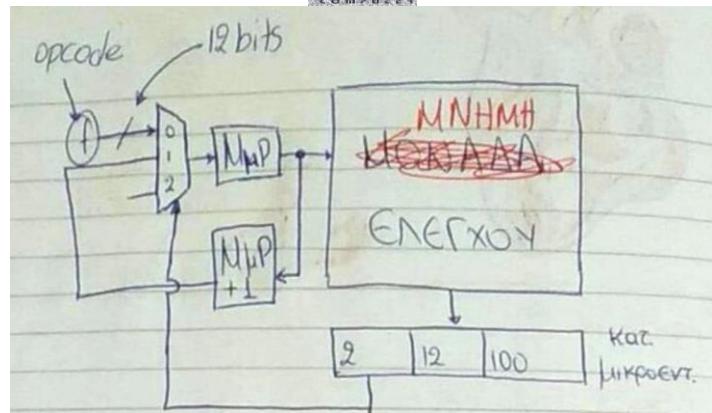
Να θεωρήσετε μονάδα ελέγχου με την τεχνική του μικροπρογραμματισμού **250** εντολών γλώσσας μηχανής. Κάθε εντολή καθορίζει τιμή **100** σημάτων ελέγχου. Το μικροπρόγραμμα **προσκόμισης εντολής** αποτελείται από 5 μικροεντολές. Κάθε μικροπρόγραμμα **εκτέλεσης εντολών** αποτελείται από 10 μικροεντολές. 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 =  $5 \text{ μικροεντολές} + 250 \text{ εντολές} * 10 \frac{\text{μικροεντολές}}{\text{εντολή}} = 2505 \text{ μικροεντολές}$ . 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 =  $2505 * 114 = 285.570$  bits



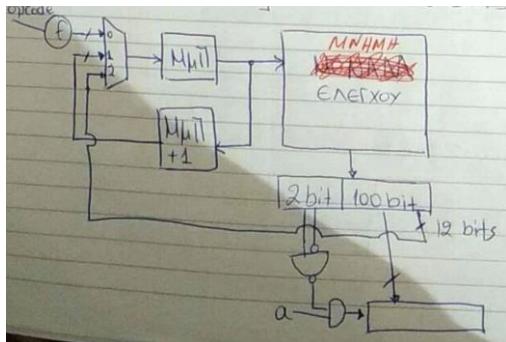
**f → ΜμΠ 00**

**ΜμΠ+1 → ΜμΠ 01**

**διεύθυνση → ΜμΠ 10**

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

$$XME = [5 + 250(10+1)] \cdot 102 = 281.570 \text{ bits}$$



### 5<sup>η</sup> Άσκηση: 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 τους 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 αποτέλεσμα 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   %

#### Αύση

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

Για τα δύο επίπεδα ισχύει ότι στο 2<sup>o</sup> επίπεδο υπάρχουν 1005 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 1<sup>o</sup> επιπέδου + Χωρητικότητα 2<sup>o</sup> επιπέδου = 45270 μικροεντολές \* (10 + 2) bits + 1005 \* 43 bits = 586455. Στο άθροισμα 10 + 2 bits, τα 10 bits χρειάζονται για τη διεύθυνσιο δότηση 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:  $\Lambda = \frac{\text{αρχική χωρητικότητα} - \text{τελική χωρητικότητα}}{\text{αρχική χωρητικότητα}} = \frac{2.037.150 - 586455}{2.037.150} = 0.711 = 71.1\%$

## Κεφάλαιο 5 – Μνήμες

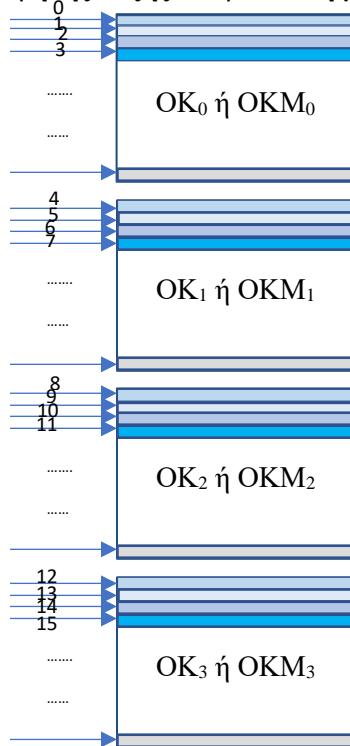
### Επεξηγήσεις κεφαλαίου 5

- 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  $t_i < t_{i+1}$  (λόγω τεχνολογίας κατασκευής) και χωρητικότητα  $X_i < X_{i+1}$  (διότι όσο απομακρυνόμαστε από την KME τόσο μεγαλώνει η χωρητικότητα του κάθε επιπέδου μνήμης).

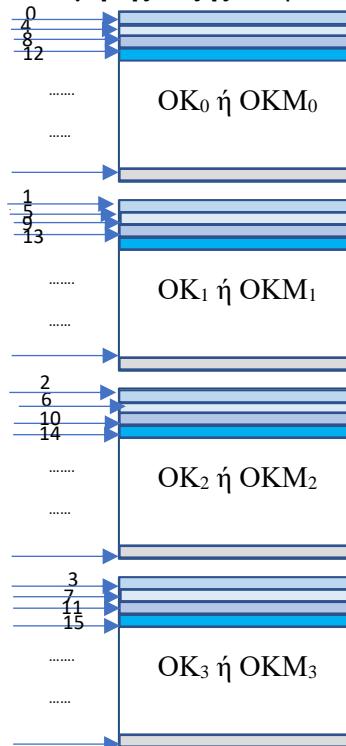
**- Όταν ζητείται ο μέσος χρόνος προσπέλασης (ανάγνωσης ή εγγραφής) από ιεραρχικό σύστημα μνήμης**, τότε χρησιμοποιήσουμε τον τύπο από το Κεφάλαιο 5.2, δηλ. τον τύπο:  $T = \sum_{i=1}^n (E_i - E_{i-1}) \times T_i$  και όχι τύπο από το Κεφάλαιο 1. Στον τύπο αυτό θεωρούμε ότι ο λόγος επιτυχίας του τελευταίου επιπέδου της ιεραρχικής μνήμης είναι πάντα «1», δηλ.  $E_n = 1$ , 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 **OKM** (Ολοκληρωμένα Κυκλώματα Μνήμης) ή απλώς **OK** (Ολοκληρωμένα Κυκλώματα ή chip), τα οποία συνδέουμε πάνω στην αρτηρία (διαύλο) 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 την 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** OK. 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 OK. 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 **ενόκλη επέκταση** του συστήματος μνήμης, προσθέτοντας ένα ή περισσότερα τμήματα μνήμης, όπου κάθε τμήμα αποτελείται από ένα ή περισσότερα

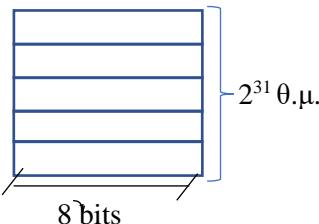
OK, δηλαδή κάθε τμήμα είναι μια συλλογή OKM. Το **μειονέκτημα** αυτής της οργάνωσης είναι μπορεί να δημιουργήσει **καθυστερήσεις** σε κάποιες περιπτώσεις, όπως π.χ. στην περίπτωση συστημάτων μερικώς επικαλυπτόμενων λειτουργιών.

- 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** διεύθυνσεις.

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

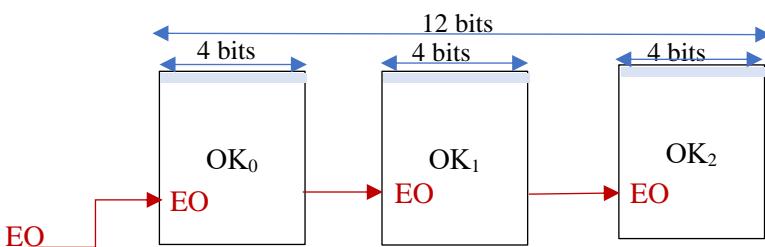
1 θ.μ.	1 ψηφιολέξη (byte)
x;	2 GB

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



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

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



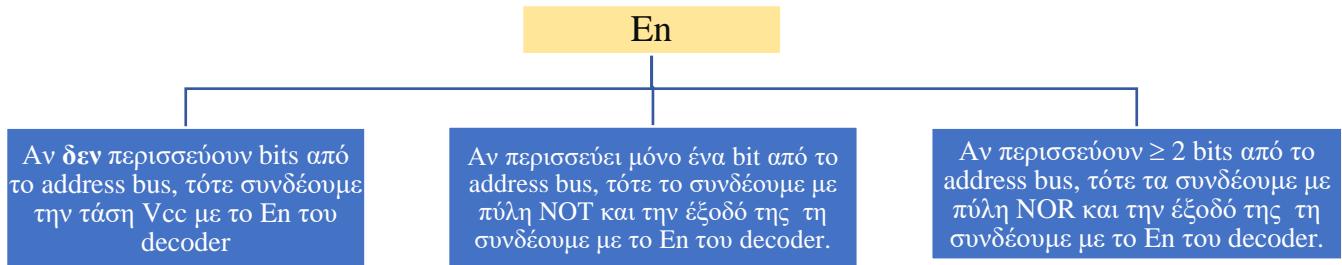
- Πιο συγκεκριμένα, **η κατακόρυφη επανάληψη OKM** έχει ως στόχο το **επιθυμητό πλήθος θ.μ.** και προκύπτει από το κλάσμα:  $\frac{\text{θ.μ. που θέλουμε να έχουμε στο τελικό σύστημα KM}}{\text{θ.μ. που ήδη έχουμε στα OKM που διαθέτουμε}}$ . Για το σωστό υπολογισμό της κατακόρυφης επανάληψης, απαιτείται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

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

1 θ.μ.	2 ψηφιολέξεις (bytes)	$x = \frac{2^{30} \text{ bytes}}{2 \text{ bytes/θ.μ.}} * = 2^{29} \text{ θ.μ.} \rightarrow \text{address bus} = 29 \text{ bits.}$
x;	1 GB	

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 OKM 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<sub>n</sub> (Enable) του decoder**, τότε συνδέουμε την τάση λειτουργίας V<sub>cc</sub> (που αντιστοιχεί σε λογικό «1») με το E<sub>n</sub> 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<sub>n</sub> του decoder. **Αν περισσεύουν περισσότερα από ένα bits από το address bus**, τα συνδέουμε σε πύλη NOR και στη συνέχεια στο E<sub>n</sub> 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  $2^8 \times 2^{20} \text{ θ.μ.} = 2^{28} \text{ θ.μ.}$ .

- 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:

$$1 \text{ θ.μ. } 8 \text{ bits} \\ x; \quad 2 \text{ MB} \\ x = \frac{2 \times 2^{20} \times 2^3 \text{ bits}}{2^3 \text{ bits}} = 2^{21} \text{ θ.μ.}$$

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

$$1 \text{ θ.μ. } 8 \text{ bits} \\ x; \quad 2 \text{ Mbit} \\ x = \frac{2 \times 2^{20} \text{ bits}}{2^3 \text{ bits}} = 2^{18} \text{ θ.μ.}$$

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

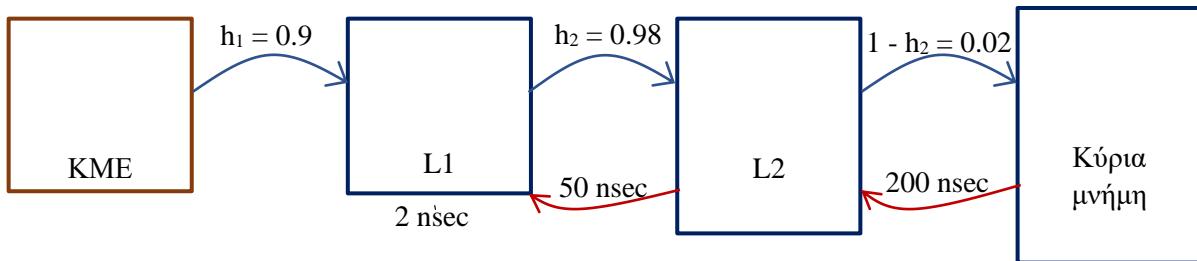
- Στο σχεδιασμό κύριας μνήμης με οργάνωση υψηλής τάξης διαφύλλωση, όταν επιλέγουμε **σειρές ΟΚ με διαφορετική χωρητικότητα**, θα πρέπει να εντοπίζουμε τη σειρά εκείνη με τη μικρότερη χωρητικότητα και για αυτήν θα πρέπει να χρησιμοποιούμε μια έξοδο του αποκωδικοποιητή, ενώ για τις υπόλοιπες σειρές, ανάλογα 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.
- **Γιατί η κρυφή μνήμη υλοποιείται με στατική μνήμη:** Οι στατικές μνήμες άμεσης προσπέλασης είναι γρήγορες αλλά έχουν σημαντικό κόστος διότι για την υλοποίηση κάθε κυψελίδας απαιτούνται αρκετά τρανζίστορ. Για αυτό το λόγο χρησιμοποιούνται κυρίως για την υλοποίηση της κρυφής μνήμης.
- **Γιατί η κύρια μνήμη υλοποιείται με δυναμική μνήμη:** Για την υλοποίηση της κύριας μνήμης, που οι απαιτήσεις σε χωρητικότητα είναι μεγάλες, απαιτείται η χρησιμοποίηση φτηνότερων ημιαγωγικών μνημών. Πιο φτηνές ημιαγωγικές μνήμες άμεσης προσπέλασης μπορούν να υλοποιηθούν χρησιμοποιώντας πιο απλή κυψελίδα. Τέτοιες μνήμες είναι οι δυναμικές μνήμες στις οποίες σήμερα η κυψελίδα αποτελείται από ένα τρανζίστορ και ένα πυκνωτή. Η λογική τιμή 1 ή 0 αποθηκεύεται στον πυκνωτή υπό μορφή φορτίου. Στις στατικές μνήμες τα δεδομένα αποθηκεύονται σε μια κυψελίδα τύπου flip-flop και διατηρούνται για όσο διάστημα υπάρχει τροφοδοσία ισχύος. Στις δυναμικές μνήμες λόγω των ρευμάτων διαρροής ο πυκνωτής μπορεί να αποφορτιστεί σε μερικά msec (milliseconds). Επομένως στις δυναμικές μνήμες απαιτείται αποκατάσταση του αποθηκευμένου φορτίου κάθε λίγα msec. Αυτή η διαδικασία λέγεται αναζωογόνηση (refreshing) της μνήμης.
- **Τοπικότητα των αναφορών:** Η δυνατότητα πρόβλεψης των διευθύνσεων μνήμης που θα αναφερθούν στο άμεσο μέλλον, η οποία είναι ουσιώδης για την επιτυχή λειτουργία της ιεραρχικής μνήμης, βασίζεται σε μία κοινή ιδιότητα των προγραμμάτων που καλείται τοπικότητα των αναφορών (locality of references). Σύμφωνα με την τοπικότητα των αναφορών, η πληροφορία (εντολές και δεδομένα) που χρησιμοποιήθηκε πρόσφατα είναι πιθανόν να ξαναχρησιμοποιηθεί στο άμεσο μέλλον και η πληροφορία που βρίσκεται κοντά στην πληροφορία που χρησιμοποιείται τώρα είναι πιθανόν να χρησιμοποιηθεί στο άμεσο μέλλον.
- **Αόγοι που συμβάλλουν στην τοπικότητα των αναφορών:** α) Ένας λόγος της τοπικότητας των αναφορών είναι ότι οι εντολές και σε μικρότερη έκταση τα δεδομένα, γράφονται και στη συνέχεια αποθηκεύονται στη μνήμη του υπολογιστή σχεδόν με τη σειρά με την οποία χρειάζονται κατά τη διάρκεια της εκτέλεσης του προγράμματος. Επίσης στις εφαρμογές η αποθήκευση των δεδομένων γίνεται με τέτοιο τρόπο ώστε να ενισχύεται η τοπικότητα των αναφορών. β) Ένας δεύτερος παράγοντας που συμβάλλει στην τοπικότητα των αναφορών είναι οι βρόχοι (loops) που εμφανίζονται μέσα στα προγράμματα. Η εκτέλεση των εντολών που περιέχονται μέσα σε ένα βρόχο επαναλαμβάνεται πολλές φορές με αποτέλεσμα να έχουμε μεγάλη συχνότητα εμφάνισης των διευθύνσεών τους. Όταν εκτελούνται εντολές που περιέχονται σε ένα βρόχο, είναι επιθυμητό να αποθηκεύσουμε όλες τις εντολές του βρόχου στο επίπεδο μνήμης M<sub>1</sub>. Που είναι το κοντινότερο στην KME Αυτό βέβαια δεν είναι πάντα δυνατό.

## 1<sup>o</sup> 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 KME μπορεί να διαβάσει πληροφορία μόνο από την L1 και ότι η κύρια μνήμη περιέχει πάντα τη ζητούμενη πληροφορία.

### Λύση



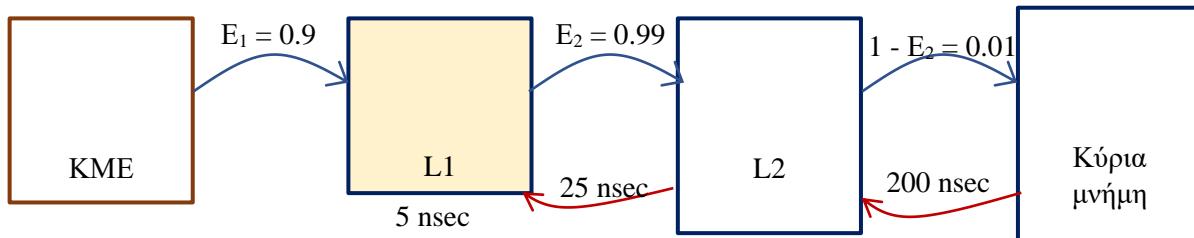
Το μοναδικό επίπεδο μιας ιεραρχικής μνήμης που είναι άμεσα προσπελάσιμο από την KME είναι πάντα το πρώτο επίπεδο κρυφής μνήμης L<sub>1</sub>. Αυτό σημαίνει ότι αν η πληροφορία βρεθεί σε κάποιο άλλο επίπεδο ιεραρχικής μνήμης (π.χ. στο L<sub>2</sub>) ή στην κύρια μνήμη, τότε θα πρέπει να μεταφερθεί στο επίπεδο L<sub>1</sub> προκειμένου να υποστεί επεξεργασία από την KME. Η διατύπωση ότι η κύρια μνήμη περιέχει πάντα τη ζητούμενη πληροφορία σημαίνει ότι ο λόγος επιτυχίας  $E_v = E_3 = 1$ .

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

## 2<sup>o</sup> Θέμα με ιεραρχία μνήμης

Δίνεται ιεραρχική μνήμη αποτελούμενη από τρία επίπεδα: 1<sup>o</sup> επίπεδο κρυφής μνήμης, 2<sup>o</sup> επίπεδο κρυφής μνήμης και κύρια μνήμη. Υποθέστε ότι η KME διαβάζει μόνο από το 1<sup>o</sup> 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 1<sup>o</sup> επιπέδου είναι  $t_1 = 5 \text{ nsec}$  και ο χρόνος μεταφοράς ενός μπλοκ πληροφορίας από το επίπεδο k + 1 στο επίπεδο k είναι  $t_B$  ( $k \leftarrow k + 1$ ), είναι  $t_B (1 \leftarrow 2) = 25 \text{ nsec}$  και  $t_B (2 \leftarrow 3) = 200 \text{ nsec}$

### Λύση



### Λύση

Ο μέσος χρόνος προσπέλασης της ιεραρχικής μνήμης είναι:

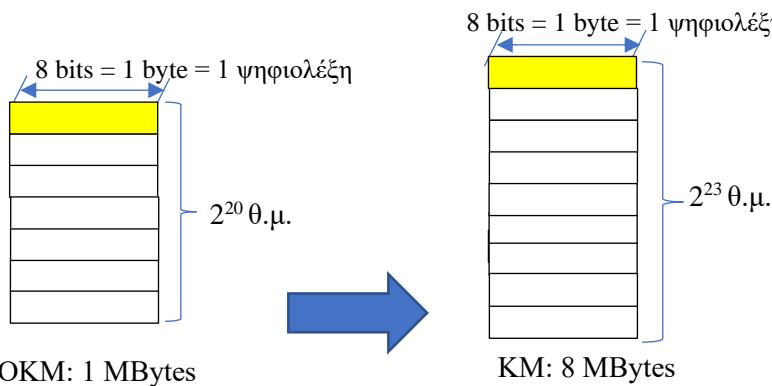
$$T = \sum_{i=1}^v (E_i - E_{i-1}) \cdot T_i = \sum_{i=1}^3 (E_i - E_{i-1}) \cdot T_i = (E_1 - E_0)T_1 + (E_2 - E_1)T_2 + (E_3 - E_2)T_3 = (0.9 - 0)T_1 + (0.99 - 0.9)T_2 + (1 - 0.99)T_3 = 0.9T_1 + 0.09T_2 + 0.01T_3 \text{ óπου } T_1 = 5 \text{nsec}, T_2 = 5 + 25 \text{ nsec} = 30 \text{ nsec} \text{ και } T_3 = 5 + 25 + 200 \text{ nsec} = 230 \text{ nsec. Άρα } T = 0.9 \times 5 + 0.09 \times 30 + 0.01 \times 230 \text{ nsec} = 4.5 + 2.7 + 2.3 = 9.5 \text{ nsec}$$

### Παράδειγμα 5.6 βιβλίου – Σχεδιασμός υψηλής τάξης διαφύλλωση μνήμης

Έχουμε στη διαθεσή μας OKM (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.)**.

**Άνση**

**Διαθέτουμε** OKM 1 Μψηφιολέξη x 1 ψηφιολέξη/θ.μ. → **Στόχος:** K.M. 8 Μψηφιολέξεων x 1 ψηφιολέξη/θ.μ.



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

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

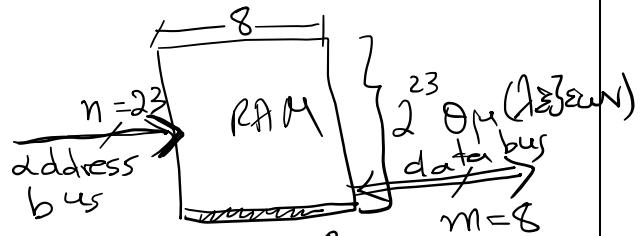
**Κατακόρυφη επανάληψη** (στόχος είναι το **επιθυμητό πλήθος θέσεων μνήμης**) και προκύπτει από το κλάσμα:   
θ.μ. που θέλουμε να έχουμε στο τελικό σύστημα KM  
θ.μ. που ήδη έχουμε στα OKM που διαθέτουμε

Για σωστό υπολογισμό αυτού του κλάσματος, θα πρέπει **πρώτα** με απλές μεθόδους των τριών, να μετατρέψουμε τις χωρητικότητες τόσο των OKM που διαθέτουμε όσο και της K.M. που θα σχεδιάσουμε, σε θέσεις μνήμης.

1 θ.μ.	1 byte	1 θ.μ.	1 byte
x;	1 MB	x;	8 MB

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

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



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

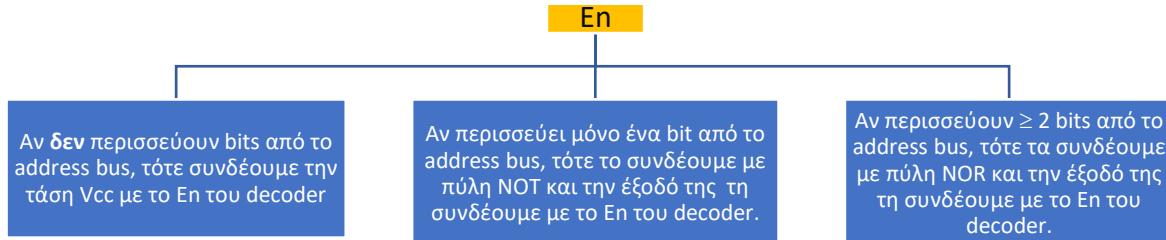
Στη συνέχεια προχωράμε στο σχεδιασμό της K.M. με βάση την οριζόντια και κατακόρυφη επανάληψη που υπολογίσαμε.

Στο σχεδιασμό αυτό, θα πρέπει να συνδέουμε τα OKM με την **αρτηρία δεδομένων** και την **αρτηρία διευθύνσεων**.

**Μεθοδολογία σχεδίασης:**

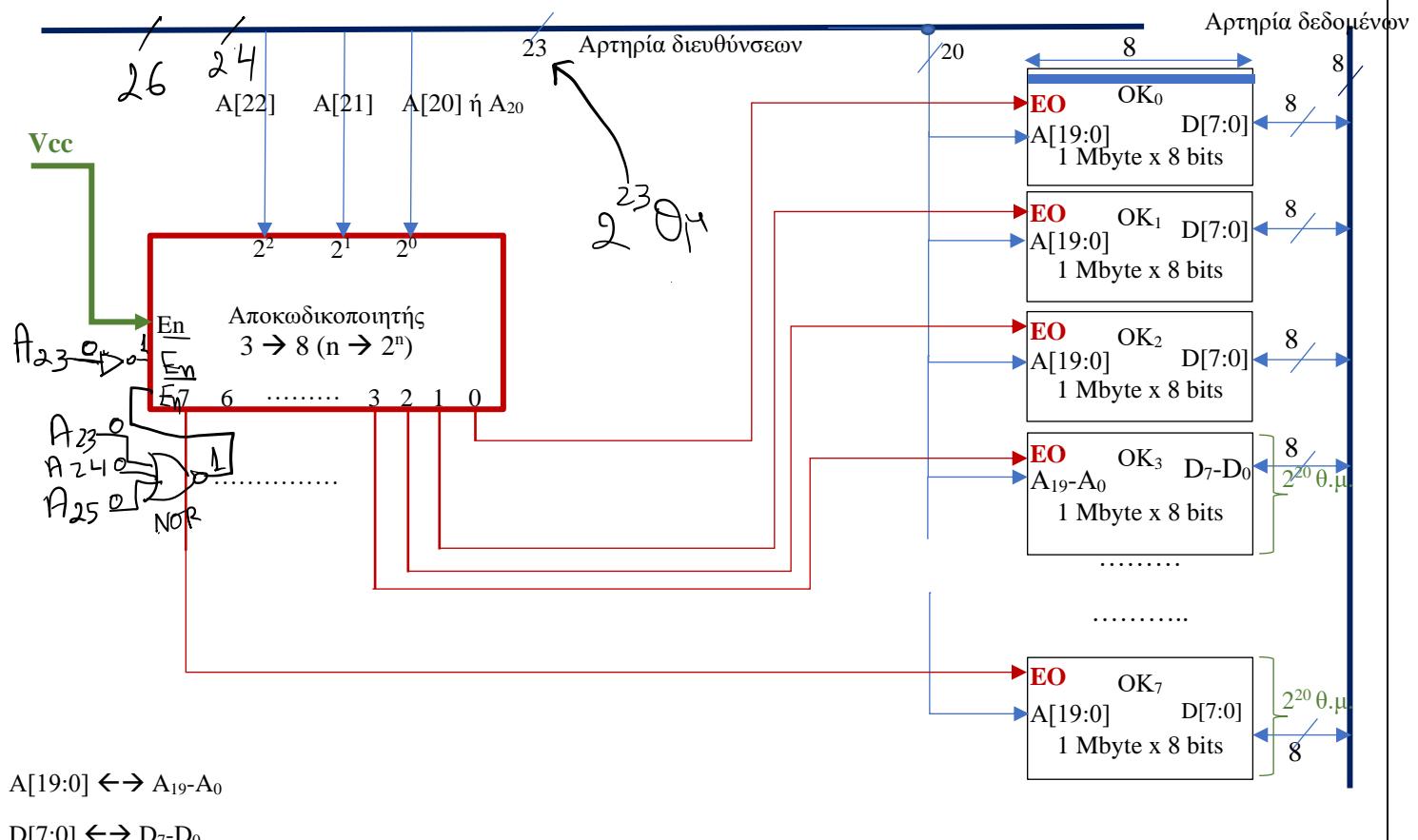
- Πρώτα** σχεδιάζουμε οριζόντια και κατακόρυφα τα OKM (ή OK) που διαθέτουμε.
- Στη συνέχεια, **συνδέουμε τα OKM με την αρτηρία δεδομένων** στα δεξιά με διπλά αμφίδρομα βέλη. Αν έχουμε οριζόντια επανάληψη των OKM, τότε το data bus επαναλαμβάνεται τόσες φορές όσο είναι η οριζόντια επανάληψη.
- Στη συνέχεια, **συνδέουμε τα OKM με την αρτηρία διευθύνσεων**, με βέλη που είναι μόνο είσοδοι στα OKM.
- Επιπλέον, είναι **απαραίτητη** η χρήση ενός **αποκωδικοποιητή** (συνδυαστικό κύκλωμα πι ιεσόδων και 2<sup>n</sup> εξόδων), το οποίο θα χρησιμοποιηθεί για να κάνει την επιλογή των OKM. Αυτό γίνεται διότι η κάθε σειρά με τα OK που χρησιμοποιούμε πρέπει να επιλέγεται ξεχωριστά από τις υπόλοιπες. **Επομένως, σχεδιάζουμε τον αποκωδικοποιητή και συνδέουμε κάθε**

έξοδό του με ένα ή περισσότερα OKM που είναι στην ίδια σειρά. Το μέγεθος του αποκωδικοποιητή προσδιορίζεται από τις εξόδους του και οι έξοδοι 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 K.M. από τα επιμέρους OKM, θα πρέπει να περιγράψουμε και το χάρτη διευθύνσεων (memory map) της K.M., ο οποίος δείχνει την περιοχή διευθύνσεων μνήμης που καταλαμβάνει κάθε OKM.

- Το μέγεθος της αρτηρίας δεδομένων και διευθύνσεων προκύπτει από την KM που θα σχεδιάσουμε. Στην προκειμένη περίπτωση η K.M. 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.

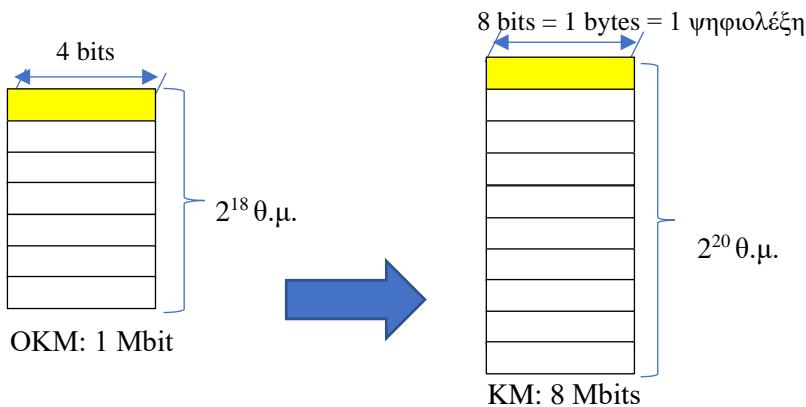


### Παράδειγμα 5.7 βιβλίου – Σχεδιασμός υψηλής τάξης διαφύλλωση μνήμης

Έχουμε στη διαθεσή μας OKM (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 (K.M.) των 8 Μδυαδικών ψηφίων και εσωτερική οργάνωση μιας ψηφιολέξης ανά θέση μνήμης (θ.μ.).

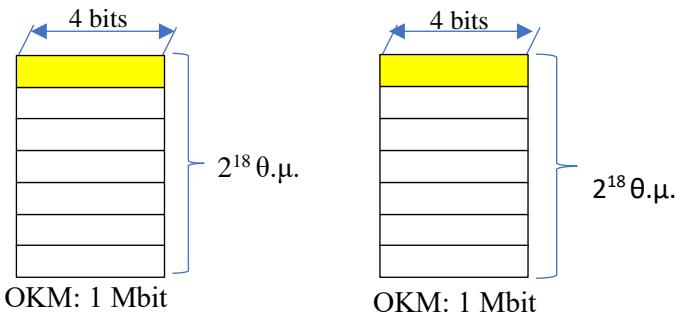
#### Λύση

**Διαθέτουμε** OKM 1 Mbit x 4 bits/θ.μ. → **Στόχος:** K.M. 8 Mbits x 1 ψηφιολέξη/θ.μ.



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

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



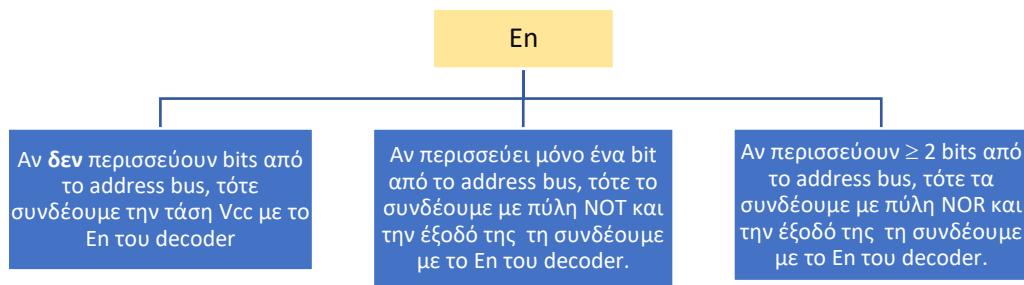
**Κατακόρυφη επανάληψη** (στόχος είναι το **επιθυμητό πλήθος θέσεων μνήμης**) και προκύπτει από το κλάσμα: **θ.μ. που θέλουμε να έχουμε στο τελικό σύστημα KM** / **θ.μ. που ήδη έχουμε στα OKM που διαθέτουμε**. Για σωστό υπολογισμό αυτού του κλάσματος, θα πρέπει **πρώτα** με απλές μεθόδους των τριών, να μετατρέψουμε τις χωρητικότητες τόσο των OKM που διαθέτουμε όσο και της K.M. που θα σχεδιάσουμε, σε θέσεις μνήμης.

1 θ.μ.	4 bits	1 θ.μ.	8 bits
x;		x;	
$\frac{2^{20} \text{ bits}}{4 \text{ bits/θ.μ.}} = 2^{18} \text{ θ.μ.}$		$\frac{2^3 2^{20} \text{ bits}}{2^3 \text{ bits/θ.μ.}} * = 2^{20} \text{ θ.μ}$	

$$\text{Επομένως } \frac{\theta.\mu. \text{ που θέλουμε να έχουμε στο τελικό σύστημα KM}}{\theta.\mu. \text{ που ήδη έχουμε στα OKM που διαθέτουμε}} = \frac{2^{20}}{2^{18}} = 4 \text{ επαναλήψεις}$$

Στη συνέχεια προχωράμε στο σχεδιασμό της K.M. με βάση την οριζόντια και την κατακόρυφη επανάληψη που υπολογίσαμε. Στο σχεδιασμό αυτό, θα πρέπει να συνδέσουμε τα OKM 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 OKM. 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 OKM (ή OK) που διαθέτουμε.

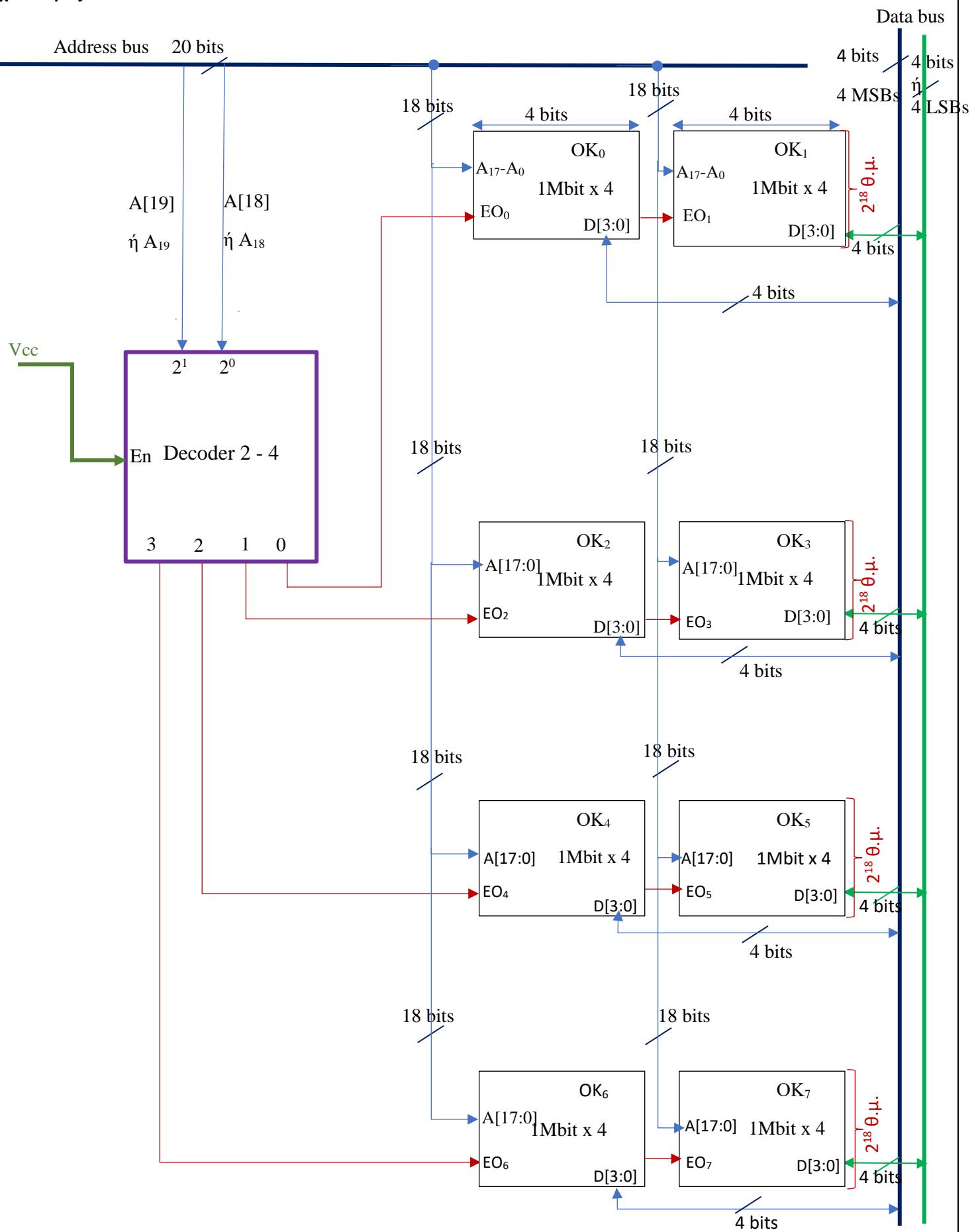
- **Στη συνέχεια, συνδέουμε τα OKM με την αρτηρία δεδομένων** στα δεξιά με διπλά αμφίδρομα βέλη. Αν έχουμε οριζόντια επανάληψη των OKM, τότε το data bus (αρτηρία δεδομένων) επαναλαμβάνεται τόσες φορές όσο είναι η οριζόντια επανάληψη. Στην προκειμένη περίπτωση το data bus επαναλαμβάνεται δύο φορές.
- **Στη συνέχεια συνδέουμε τα OKM με την αρτηρία διευθύνσεων**, με βέλη που είναι μόνο είσοδοι στα OKM.
- Το μέγεθος της αρτηρίας δεδομένων και διευθύνσεων προκύπτει από την KM που θα σχεδιάσουμε.
- **Τέλος, σχεδιάζουμε τον αποκωδικοποιητή και συνδέουμε κάθε έξοδό του με ένα ή περισσότερα OKM** (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 ούσα αναφέρθηκαν προηγουμένως.
- Αν χρειάζεται (αν ζητείται από την εκφώνηση) μετά το σχεδιασμό του συστήματος K.M. από τα επιμέρους OKM, θα πρέπει να περιγράψουμε και το χάρτη διευθύνσεων (memory map) της K.M., ο οποίος δείχνει την περιοχή διευθύνσεων μνήμης που καταλαμβάνει κάθε OKM.

### Χάρτης διευθύνσεων μνήμης (memory map)

Ολοκληρωμένο κύκλωμα μνήμης	Διευθύνσεις	
	A <sub>19</sub> A <sub>18</sub> A <sub>17</sub> ...	A <sub>1</sub> A <sub>0</sub>
OK <sub>0</sub> – OK <sub>1</sub>	από έως	00000000000000000000 00111111111111111111
OK <sub>2</sub> – OK <sub>3</sub>	από έως	01000000000000000000 01111111111111111111
OK <sub>4</sub> – OK <sub>5</sub>	από έως	10000000000000000000 10111111111111111111
OK <sub>6</sub> – OK <sub>7</sub>	από έως	11000000000000000000 11111111111111111111

Κάθε OKM διαθέτει συνολικά  $2^{18}$  θ.μ., για το λόγο αυτό απαιτούνται 18 bits από το address bus για τη διευθυνσιοδότηση, που είναι από A<sub>17</sub> – A<sub>0</sub>. 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

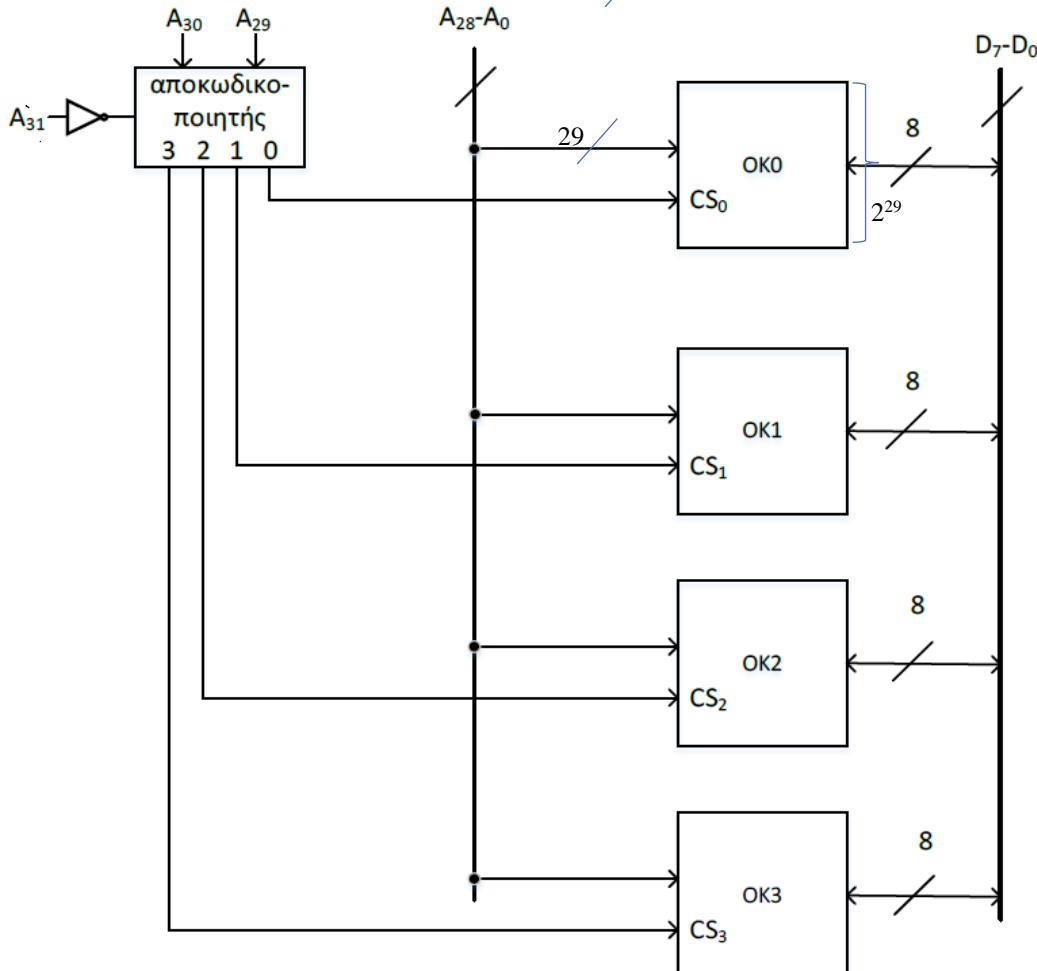
Έχετε στη διάθεσή σας ολοκληρωμένα κυκλώματα μνήμης με χωρητικότητα 512 MBytes και μία ψηφιολέξη (1 ψηφιολέξη = 1 byte) ανά θέση μνήμης (δηλ. σε κάθε ψηφιολέξη αντιστοιχεί μια διεύθυνση). Να σχεδιάσετε σύστημα κύριας μνήμης με χωρητικότητα 2 GBytes και μία ψηφιολέξη ανά θέση μνήμης. Η αρτηρία διευθύνσεων και η αρτηρία δεδομένων είναι των 32 και των 8 δυαδικών ψηφιών 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 OKM 512 MBytes x 1 ψηφιολέξη/θ.μ. → Στόχος: KM 2 GBytes x 1 ψηφιολέξη/θ.μ. Τα μεγέθη των δύο αρτηριών θα τα πάρουμε από την εκφόνηση της άσκησης. Αυτό σημαίνει ότι το data bus = 8 bits και το address bus = 32 bits. Επειδή έχουμε την ίδια εσωτερική οργάνωση μεταξύ των OK που διαθέτουμε και της K.M. που θα κατασκευάσουμε, τα OK επαναλαμβάνονται οριζόντια μόνο μια φορά. Όσον αφορά την κατακόρυφη επανάληψη, έχουμε θ.μ. που θέλουμε να έχουμε στο τελικό σύστημα KM  $\frac{2^{31}}{2^{29}} = 4$  επαναλήψεις. Τις θέσεις μνήμης τις έχουμε υπολογίσει με απλές μεθόδους των τριών:

$$\begin{array}{l} 1 \text{ θ.μ. } 1 \text{ byte} \\ \hline x; \quad 512\text{MB} \\ \hline x = 2^{29} \text{ θ.μ.} \end{array}$$

$$\begin{array}{l} 1 \text{ θ.μ. } 1 \text{ byte} \\ \hline x; \quad 2\text{GB} \\ \hline x = 2^{31} \text{ θ.μ.} \end{array}$$



## Θέμα Ιοννίου 2016

Σ' ένα υπολογιστή ο επεξεργαστής έχει αρτηρία διευθύνσεων των 34 δυαδικών ψηφίων. Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης OK1 με χωρητικότητα 2048 M δυαδικών ψηφίων (M bit) και οργάνωση μιας ψηφιολέξης ανά θέση μνήμης και ολοκληρωμένα κυκλώματα μνήμης OK2 με χωρητικότητα 256 M 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 M 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 OK1** 2048 Mbit x 8 bits/θ.μ., και **OK2** 256 Mbit x 1 bit/θ.μ. → **Στόχος:** K.M. 9216 Mbit x 9 bits/θ.μ.

Προϋπόθεση για να συνδέσουμε διαφορετικά OKM παράλληλα μεταξύ τους, είναι να έχουν **ίδιο πλήθος θ.μ.**

**Οριζόντια επανάληψη:** 1 OK1 + 1 OK2  $\leftrightarrow$  πάντα προσδιορίζεται με βάση την εσωτερική οργάνωση του τελικού συστήματος K.M. που θέλουμε να σχεδιάσουμε.

**Κατακόρυφη επανάληψη:** 4  $\leftrightarrow$  πάντα προσδιορίζεται με βάση το πλήθος των θ.μ. του τελικού συστήματος K.M. που θέλουμε να σχεδιάσουμε.

1 θ.μ.	8 bits	1 θ.μ.	1 bit	1 θ.μ.	9 bit
x;	2048 Mbit	x;	256 Mbit	x;	9216 Mbit

$$x = 1 \text{ θ.μ.} \times \frac{2^{11} \times 2^{20} \text{ bit}}{8 \text{ bit}} = 2^{28} \text{ θ.μ.} \quad x = 1 \text{ θ.μ.} \times \frac{2^8 \times 2^{20} \text{ bit}}{1 \text{ bit}} = 2^{28} \text{ θ.μ.} \quad x = 1 \text{ θ.μ.} \times \frac{9216 \times 2^{20} \text{ bit}}{9 \text{ bit}} = 1024 \times 2^{20} \text{ θ.μ.} = 2^{30}$$

$$\text{Επομένως } \frac{\text{θ.μ. που θέλουμε να έχουμε στο τελικό σύστημα KM}}{\text{θ.μ. που ήδη έχουμε στα OKM που διαθέτουμε}} = \frac{2^{30}}{2^{28}} = 4 \text{ επαναλήψεις}$$

α. Πλήθος εισόδων διευθύνσεων και εισόδων/εξόδων δεδομένων **OK1:** 28 bits (address bus) και 8 bits (data bus) αντίστοιχα.

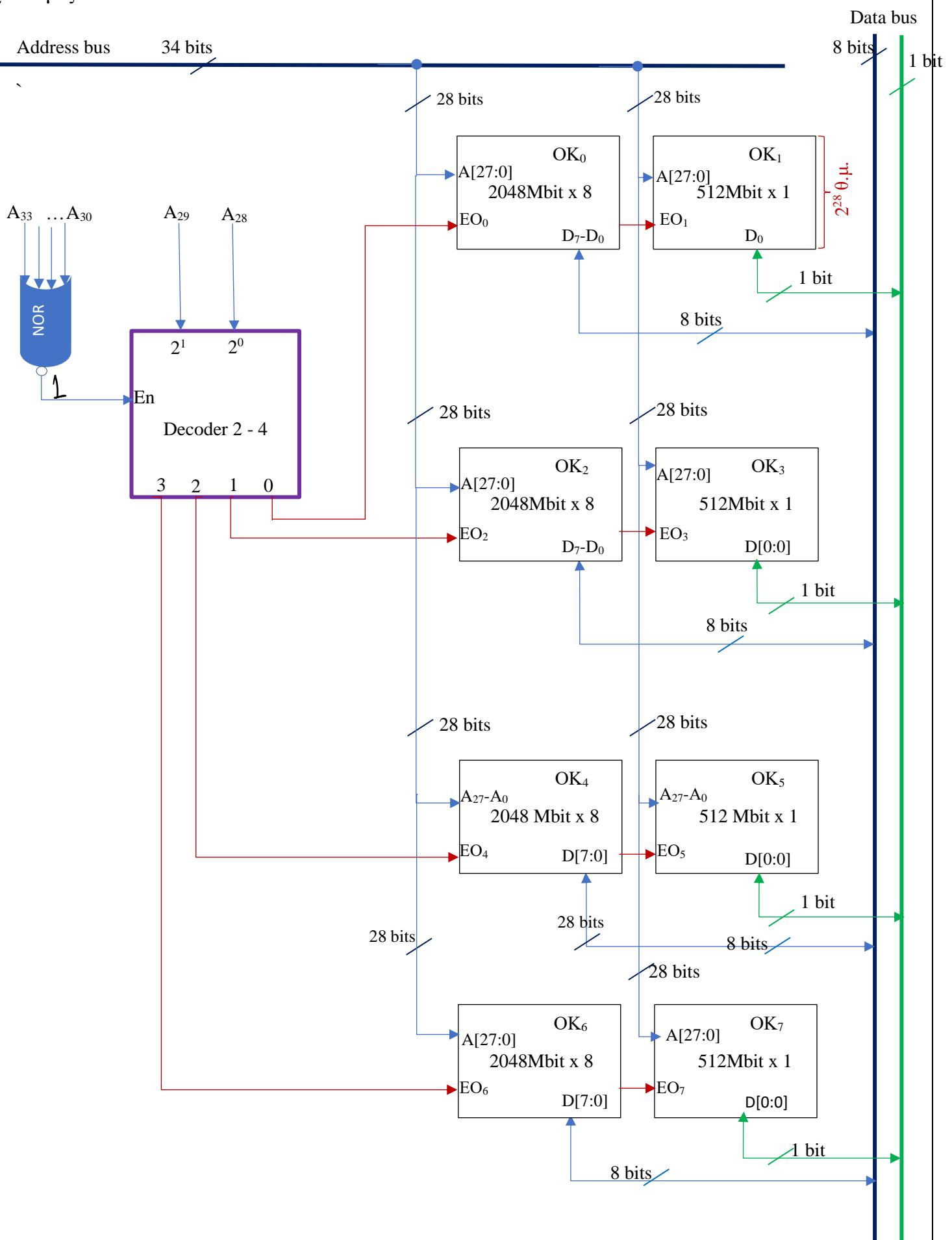
Πλήθος εισόδων διευθύνσεων και εισόδων/εξόδων δεδομένων **OK2:** 28 bits (address bus) και 1 bit (data bus) αντίστοιχα.

β. **Σχεδιασμός συστήματος K.M. με οργάνωση υψηλής τάξης διαφύλλωση.** Κατά σειρά σχεδιάζουμε τα OKM (OK) με την οριζόντια και κατακόρυφη επανάληψη που έχουμε υπολογίσει νωρίτερα (2 και 4 αντίστοιχα), στη συνέχεια τα συνδέομε 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 OKM.

γ. **Στο τέλος θα σχεδιάσουμε τον αποκωδικοποιητή,** το μέγεθος του οποίου προσδιορίζεται από τον αριθμό των σειρών με OKM που επιλέγουμε (στην προκειμένη περίπτωση έχουμε 4 σειρές OKM με δύο OKM ανά σειρά), και συνδέομε την κάθε έξοδο του αποκωδικοποιητή με την γραμμή ενεργοποίησης (EO ή CS) όλων των OKM που βρίσκονται στην ίδια σειρά.

**Σημείωση:** Επειδή από την εκφώνηση δίνεται ότι το εύρος του address bus είναι 34 bits (γραμμές), θα βάλουμε αυτό το μέγεθος στο σχήμα και ότι τα 30 bits που προκύπτουν από το στόχο μας, δηλ. από τις  $2^{30}$  θ.μ. της K.M. Τα bits από A<sub>30</sub>-A<sub>33</sub> που δεν χρησιμοποιούνται (περισσεύουν), θεωρούνται ότι έχουν τιμή «0». Για το λόγο αυτό εισέρχονται σε μια πόλη NOR, η οποία έχει την ιδιότητα ότι όταν όλες οι είσοδοι της είναι «0», τότε βγάζει «1» στην έξοδό της. Κάνει δηλ. την πράξη  $(0 + 0 + 0 + 0)' = 0' = 1$ .

## Σχεδιασμός Κ.Μ.



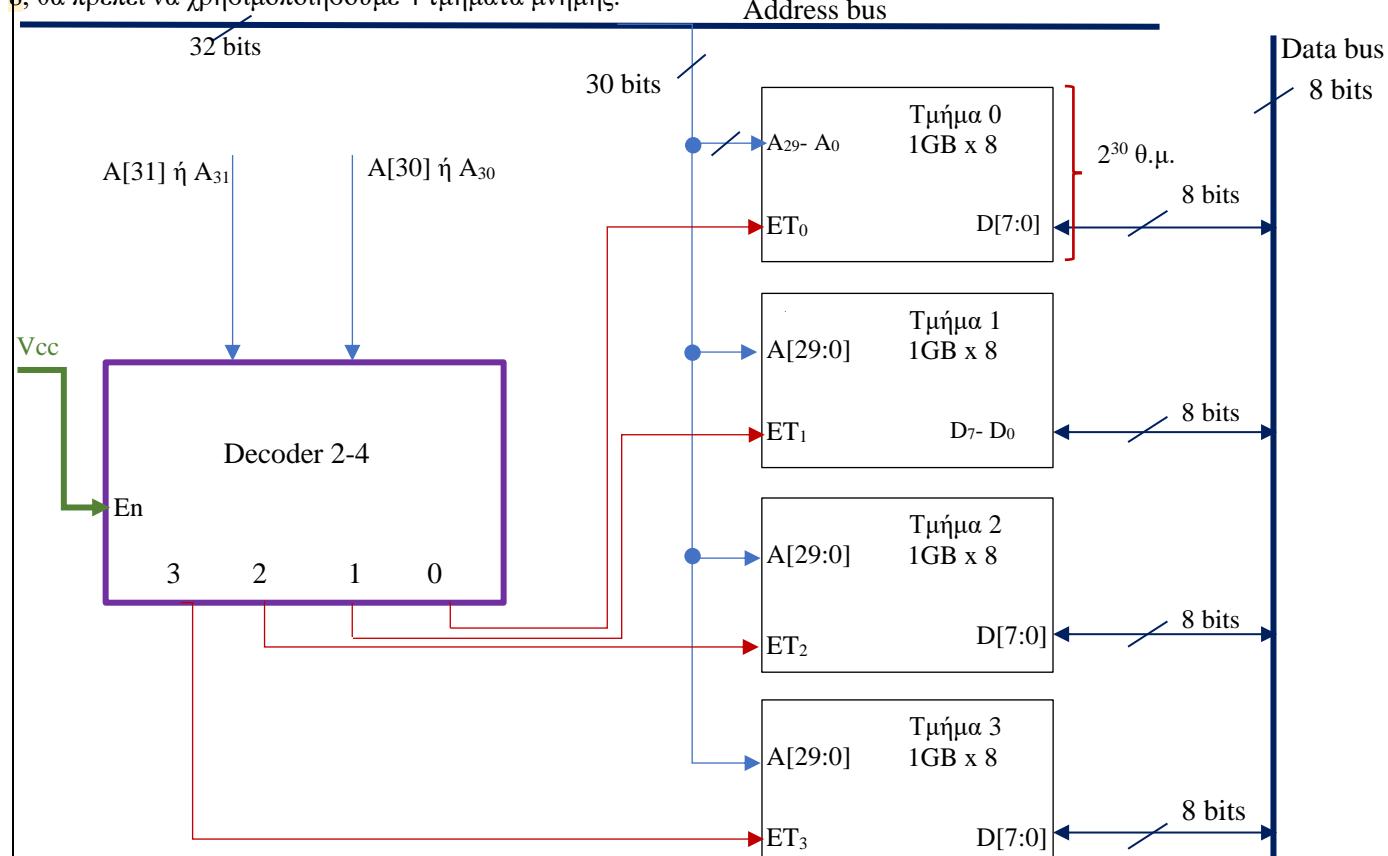
## Άσκηση 5.4 με τμήματα μνήμης

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 Γψηφιολέξεις** 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 OKM των 256MB x 8 και θέλουμε να σχεδιάσουμε σύστημα κύριας μνήμης (KM), 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 OKM και στην προκειμένη περίπτωση θα είναι μεγέθους 1 GB. Επομένως, ο λόγος για τον οποίο χρησιμοποιούμε τμήματα μνήμης σχετίζεται με την **επεκτασιμότητα της KM**.

Το μέγιστο της χωρητικότητας (με άλλα λόγια η μέγιστη χωρητικότητα) της K.M. που θα κατασκευάσουμε θα είναι  $2^{32}$  θ.μ., με δεδομένο ότι το **address bus = 32 bits**. Επειδή η οργάνωση της K.M. που θα σχεδιάσουμε θα είναι 1 ψηφιολέξη/θ.μ., αυτό σημαίνει ότι  $2^{32}$  θ.μ. =  $2^{32}$  bytes =  $2^{30} \times 2^2$  bytes = 4 GB. Επομένως, με δεδομένο ότι διαθέτουμε **τμήματα 1 GB x 8 → Στόχος είναι 4 GB x 8**, θα πρέπει να χρησιμοποιήσουμε 4 τμήματα μνήμης.

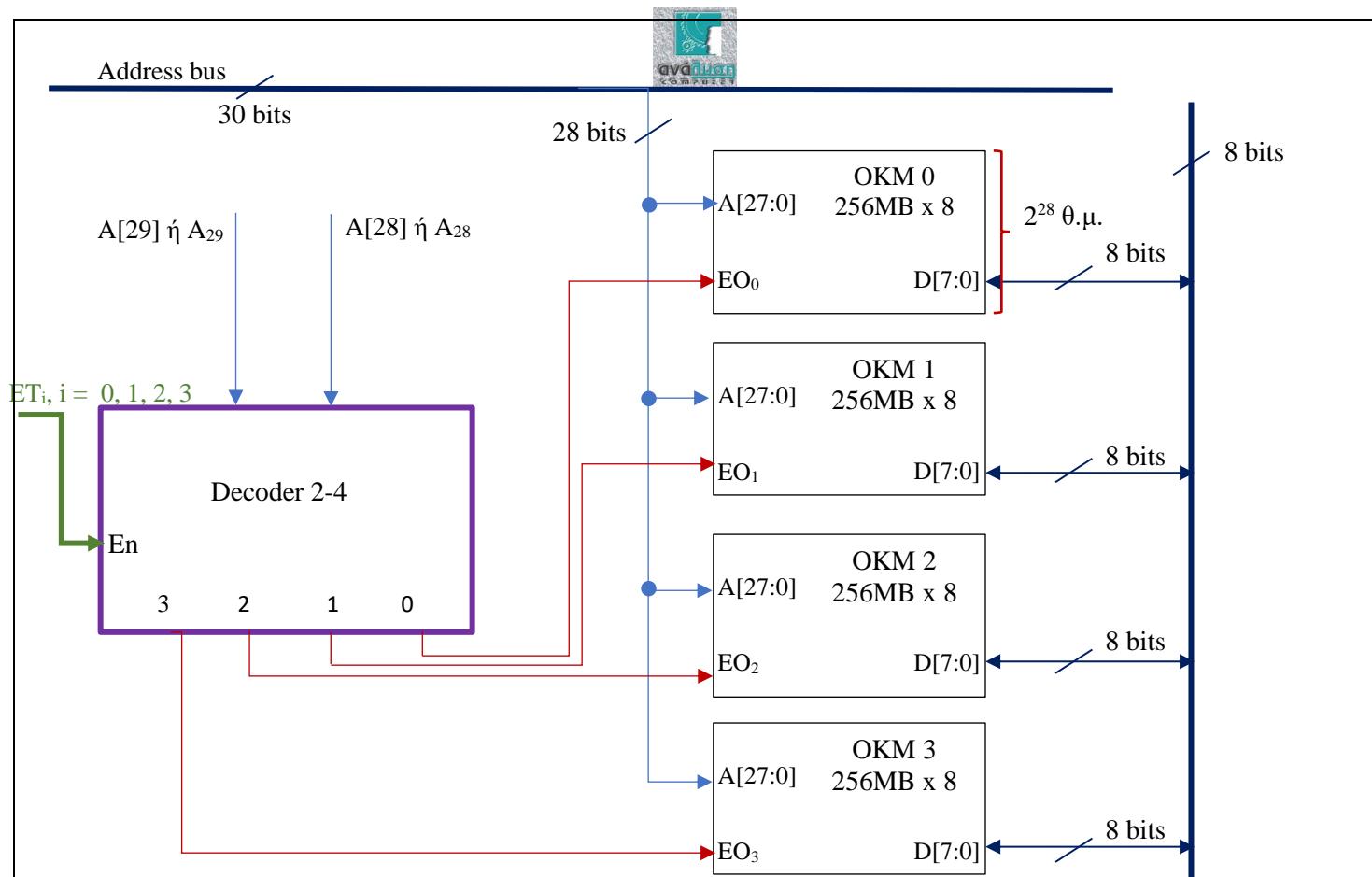


Ο χάρτης διευθύνσεων μνήμης για τα 4 τμήματα μνήμης που χρησιμοποιήσαμε είναι ο εξής:

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 <sub>0</sub>	30	Από 00 0 0 ... 0 0 έως 00 1 1 ... 1 1
T <sub>1</sub>	30	Από 01 0 0 ... 0 0 έως 01 1 1 ... 1 1
T <sub>2</sub>	30	Από 10 0 0 ... 0 0 έως 10 1 1 ... 1 1
T <sub>3</sub>	30	Από 11 0 0 ... 0 0 έως 11 1 1 ... 1 1

Στη συνέχεια, θα πρέπει να δείξουμε την κατασκευή του κάθε τμήματος μνήμης από τα επιμέρους OKM που διαθέτουμε. Πιο συγκεκριμένα: Διαθέτουμε **OKM 256MB x 8 → Τμήμα 1GB x 8**.



1 θ.μ. 1 byte  
x; 256MB  
 $x = 2^8 \times 2^{20}/1 = 2^{28} \text{ θ.μ.}$

1 θ.μ. 1 byte  
x; 1GB  
 $x = 2^{30}/1 = 2^{30} \text{ θ.μ.}$

$$\text{Επομένως } \frac{\text{θ.μ.που θέλουμε να έχουμε στο τμήμα}}{\text{θ.μ.που ήδη έχουμε στα OKM που διαθέτουμε}} = \frac{2^{30}}{2^{28}} = 4 \text{ επαναλήψεις}$$

Ο χάρτης διευθύνσεων μνήμης για τα 4 OKM μέσα σε κάθε τμήμα, είναι ο εξής:

Πίνακας 2: Χάρτης διευθύνσεων μνήμης για ένα τμήμα με τα OKM που περιέχει

Ολοκληρωμένο κύκλωμα μνήμης	Πλήθος εισόδων διευθύνσεων	Διευθύνσεις $A_{29}A_{28}A_{27}A_{26} \dots A_1A_0$
OK <sub>0</sub>	28	από <b>00</b> 0 0 ... 0 0 έως <b>00</b> 1 1 ... 1 1
OK <sub>1</sub>	28	από <b>01</b> 0 0 ... 0 0 έως <b>01</b> 1 1 ... 1 1
OK <sub>2</sub>	28	από <b>10</b> 0 0 ... 0 0 έως <b>10</b> 1 1 ... 1 1
OK <sub>3</sub>	28	από <b>11</b> 0 0 ... 0 0 έως <b>11</b> 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 KM 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.

### Λύση

**Διαθέτουμε** OK 512 Μψηφιολέξεις x 8 bits/θ.μ. → **Στόχος:** K.M. 2 GB x 8 bits/θ.μ.

Οριζόντια επανάληψη: 1 ↔ πάντα προσδιορίζεται με βάση την **εσωτερική οργάνωση της K.M.** που θα σχεδιάσουμε.

Κατακόρυφη επανάληψη: 4 ↔ πάντα προσδιορίζεται με βάση το **πλήθος των θ.μ. της K.M.** που θα σχεδιάσουμε.

1 θ.μ. 1 byte  
x; 512 MB

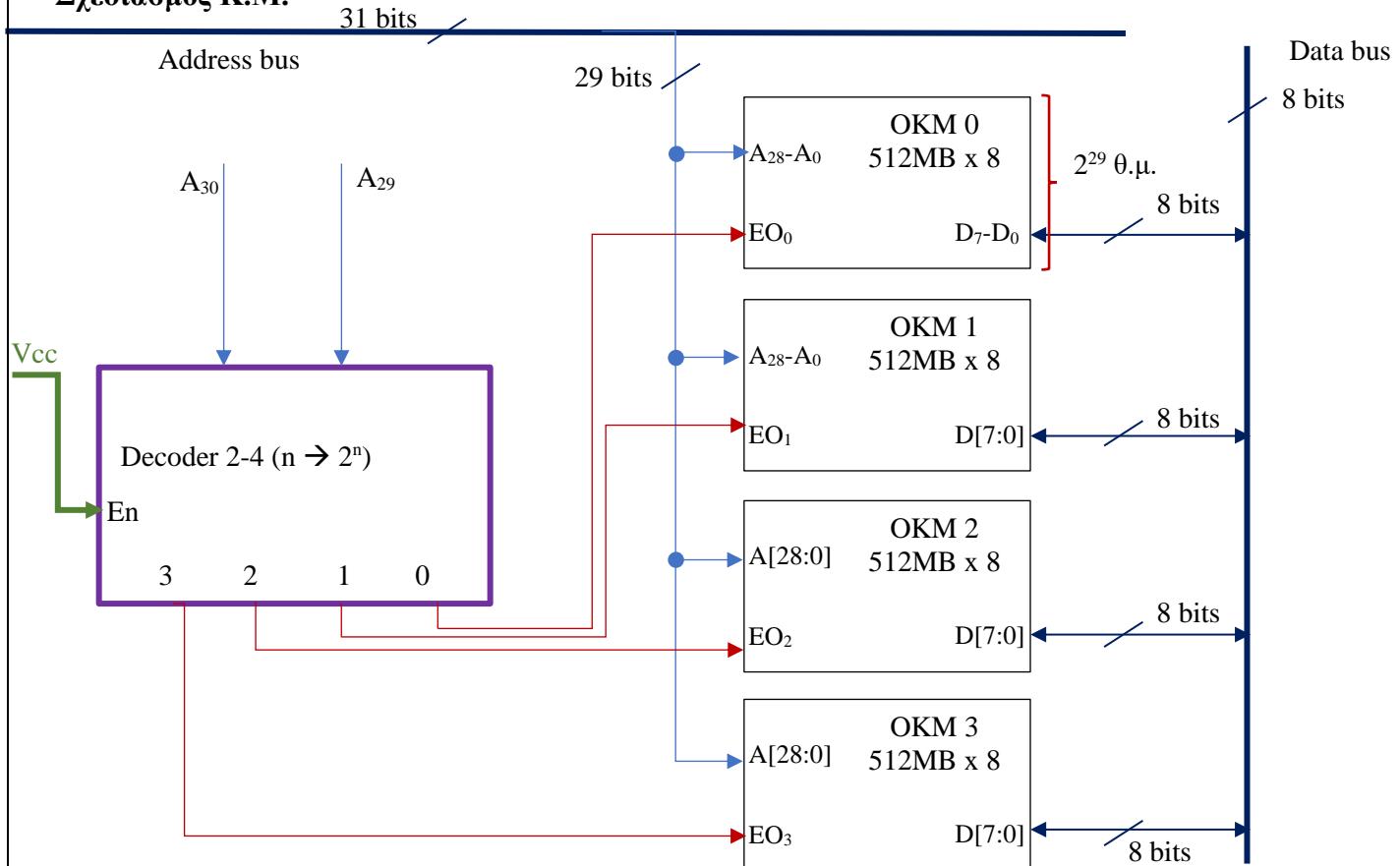
1 θ.μ. 1 byte  
x; 2 GB  
 $x = \frac{2 \times 2^{30} \text{ bytes}}{1 \text{ byte/θ.μ.}} = 2^{31} \text{ θ.μ.}$

$$x = \frac{2^9 \times 2^{20} \text{ bytes}}{1 \text{ byte}/\theta.\mu} = 2^{29} \theta.\mu.$$

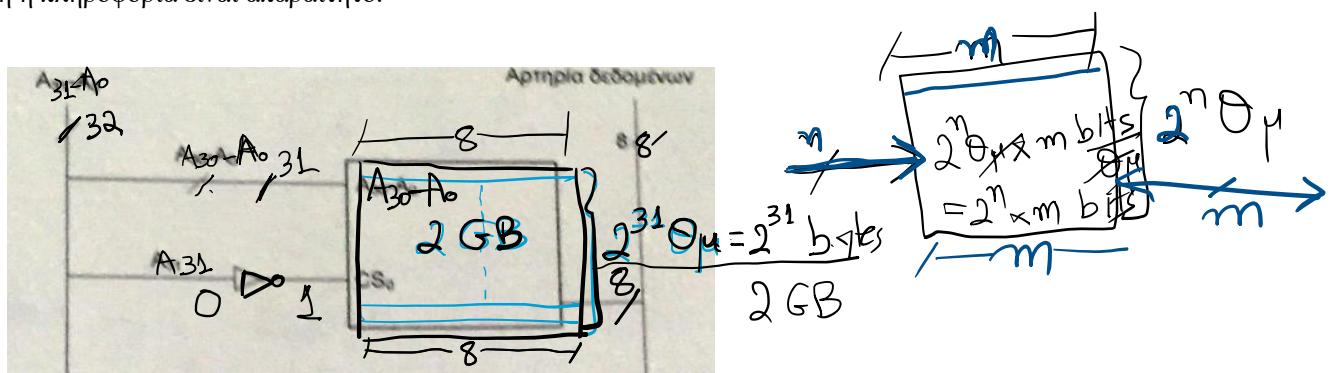
Επομένως  $\frac{\theta.\mu \text{ που θέλουμε να έχουμε στο τελικό σύστημα KM}}{\theta.\mu \text{ που ήδη έχουμε στα OKM που διαθέτουμε}} = \frac{2^{31}}{2^{29}} = 4$  επαναλήψεις

α. Στη συνέχεια προχωραμε στο σχεδιασμό της K.M. 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 K.M. που βρήκαμε προηγουμένως. Στην προκειμένη περίπτωση χρησιμοποιούμε όλα τα δυαδικά ψηφία από την αρτηρία διευθύνσεων προκειμένου να διευθυνθούμε τα OKM, με αποτέλεσμα να χρησιμοποιούμε την τάση λειτουργίας  $V_{cc}$ .

### Σχεδιασμός K.M.

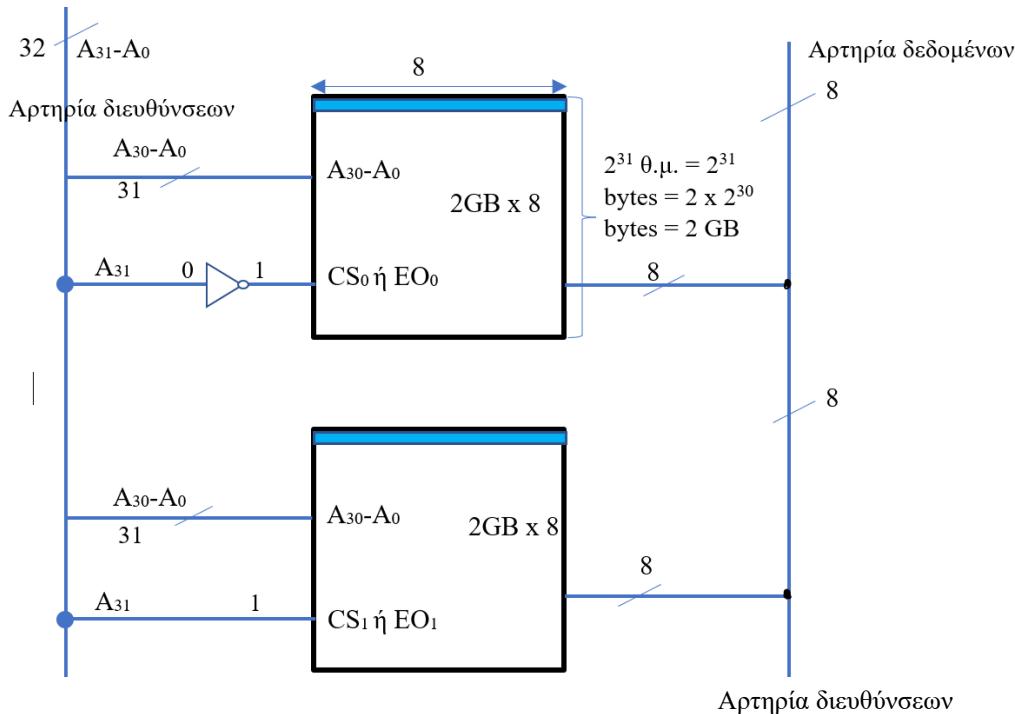


β. Στο σύστημα μνήμης που δίνεται στο επόμενο σχήμα να διασυνδέσετε κατάλληλα τόσα κυκλώματα μνήμης KM που σχεδιάσατε στο ερώτημα **α** ώστε το σύστημα της μνήμης να έχει συνολική χωρητικότητα 4 GBytes. Ένα σχήμα στο οποίο να φαίνεται όλη η πληροφορία είναι απαραίτητο.



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 =  $2 \times 2^{30}$  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 OKM (τόσο του αρχικού, όσο και αυτού που προσθέσαμε). Επομένως η επιλογή OKM 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 OKM, αν είναι «1», τότε επιλέγεται το κάτω OKM. Στη συνέχεια φαίνεται η επέκταση της υπάρχουσας KM κατά 2GB έτσι ώστε η συνολική χωρητικότητα να γίνει 4 GB.



### Άνση 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 γράψετε πάνω στο σχήμα τα δυαδικά ψηφία της αρτηρίας διευθύνσεων και της αρτηρίας δεδομένων που οδηγούν ή οδηγούνται από **κάθε τμήμα  $T_i$**  ( $i = 0, 1, 2, 3$ ) της μνήμης και του αποκωδικοποιητή.

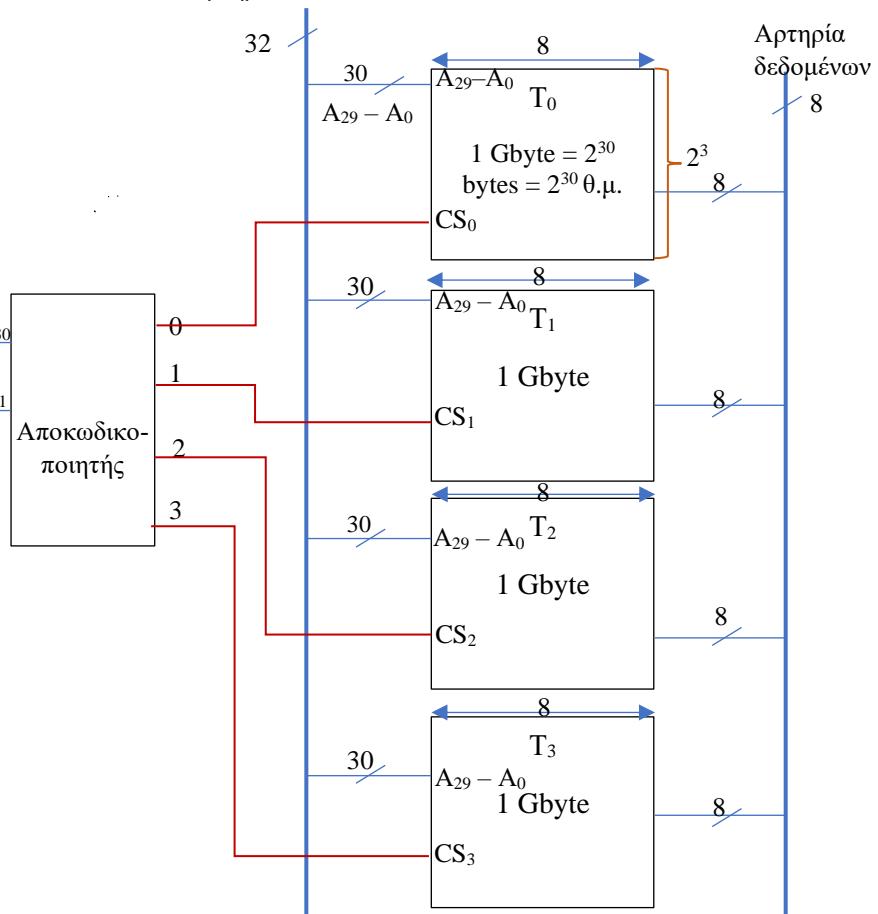
β. Να φτιάξετε το χάρτη διευθύνσεων της κύριας μνήμης στον οποίο να φαίνεται ποιες διευθύνσεις χρησιμοποιούνται από το κάθε τμήμα.

γ1. 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.

γ2. Να δώσετε τη δομή του τμήματος  $T_i$  της κύριας μνήμης. Ένα σχήμα στο οποίο να φαίνεται όλη η πληροφορία είναι απαιτούμενο και αρκετό.

### Δύση

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  $1 \text{ GB} = 2^{30} \text{ bytes} = 2^{30} \text{ θ.μ.}$  (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 OKM, 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 K.M. 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	
		$A_{31}A_{30}A_{29}A_{28} \dots A_1A_0$	
$T_0$	30	Από 00 0000000000.....00000 Έως 00 1111111111.....11111	
$T_1$	30	Από 01 0000000000.....00000 Έως 01 1111111111.....11111	
$T_2$	30	Από 10 0000000000.....00000 Έως 10 1111111111.....11111	
$T_3$	30	Από 11 0000000000.....00000 Έως 11 1111111111.....11111	

γ1) Κατασκευή τμήματος από επιμέρους OKM. Διαθέτουμε OKM 128 MB x 4  $\rightarrow$  Στόχος: 1 GB x 8 (Τμήμα)

**Οριζόντια επανάληψη:** 2 (διότι πρέπει να συνδέσουμε 2 OKM 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	1 θ.μ.	1 byte
x;	128 Mbyte	x;	1 GB
$x = 1 \theta.μ. \times \frac{2^7 \times 2^{20} \times 2^3 \text{ bits}}{2^2 \text{ bits}} = 2^{28} \theta.μ.$		$x = 1 \theta.μ. \times \frac{2^{30} \text{ bytes}}{1 \text{ byte}} = 2^{30} \theta.μ.$	

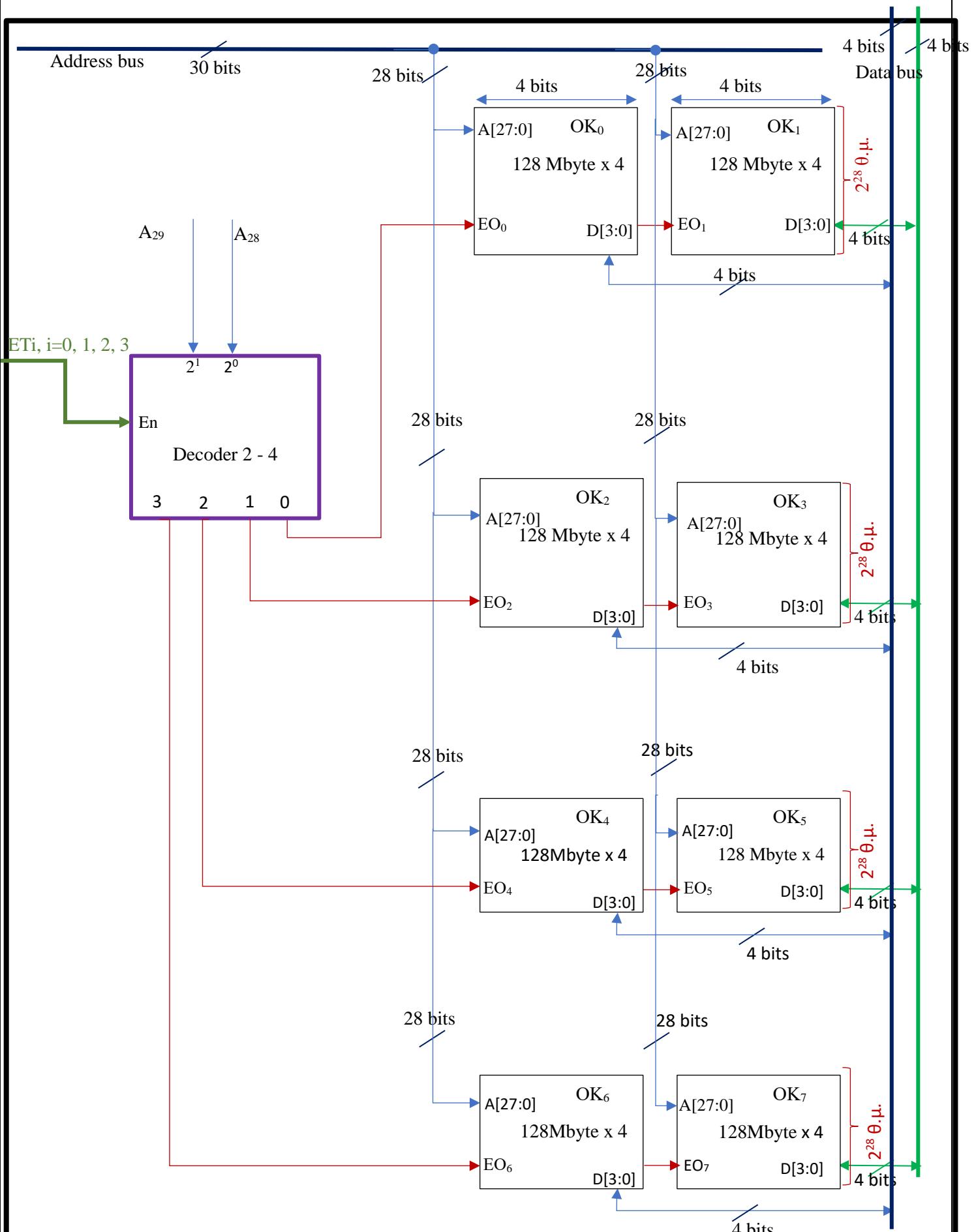
Επομένως  $\frac{\theta.μ. που θέλουμε να έχουμε στο τελικό σύστημα KM}{\theta.μ. που ήδη έχουμε στα OKM που διαθέτουμε} = \frac{2^{30}}{2^{28}} = 4$  επαναλήψεις. Στη συνέχεια ακολουθεί ο σχεδιασμός του τμάματος από τα επιμέρους OK που διαθέτουμε. Το περίγραμμα δείχνει τα όρια του κάθε τμήματος.

**Σημείωση:** Στο σχήμα της εκφώνησης, ο αποκωδικοποιητής που είχε δοθεί, **δεν** είχε ακροδέκτη ενεργοποίησης (En). Αν είχε, τότε θα έπρεπε σε αυτόν -από τη στιγμή που δεν περισσεύει κάποιο bit από το address bus- να συνδέσουμε τον ακροδέκτη  $ET_i$ , για  $i = 0, 1, 2, 3$ .

γ2) Στη συνέχεια, φαίνεται ένας χάρτης διευθύνσεων του κάθε τμήματος μνήμης

*Πίνακας 2: Χάρτης διευθύνσεων μνήμης του κάθε τμήματος*

Τμήμα	Πλήθος εισόδων διευθύνσεων	Διευθύνσεις	
		$A_{29}A_{28}A_{27} \dots A_1A_0$	
$OK_0 - OK_1$	28	00 00000000.....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	



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.

Αύση

**Διαθέτουμε** 4 OKM1 των 4Mbits x 2 bits και 1 OKM2 των 32Mbits x 8 bits → **Στόχος:** KM 6MB x 8 bits.

**Οριζόντια επανάληψη** δεν μπορεί να υπολογιστεί διότι στο OKM1 έχουμε εσωτερική οργάνωση 2 bits/θ.μ. και στο OKM2 έχουμε εσωτερική οργάνωση 8 bits/θ.μ. και θέλουμε 8 bits/θ.μ.

**Κατακόρυφη επανάληψη** (στόχος είναι το **κατάλληλο πλήθος θέσεων μνήμης**) και προκύπτει από το κλάσμα: θ.μ. που θέλουμε να έχουμε στο τελικό σύστημα **KM**

Θ.μ. που ήδη έχουμε στα ΟΚΜ που διαθέτουμε

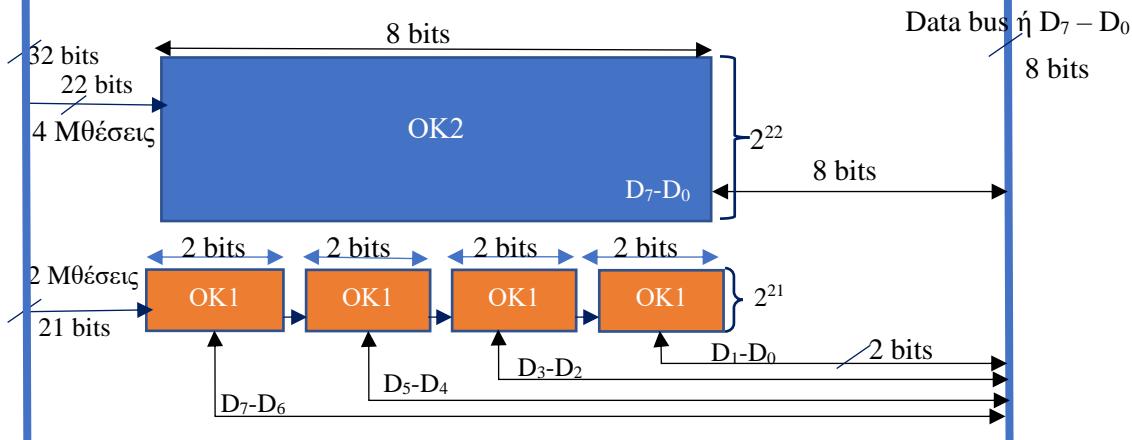
$$\begin{array}{ll} \text{Για σωστό υπολογισμού:} \\ \text{χωρητικότητες σε θέσεις:} \\ 1 \text{ θ.μ.} & 2 \text{ bits} \\ x; & 4 \text{ Mbi} \\ x = \frac{4 \times 2^{20} \text{ bits}}{2 \text{ bits/θ.μ.}} & = 2^{21} \text{ θ.μ.} \end{array}$$

$$x = \frac{2^5 2^{20} \text{ bits}}{2^3 \text{ bits}/\theta.\mu.} = 2^{22} \theta.\mu.$$

$1 \theta.\mu$	8 bits
$x;$	6 MB

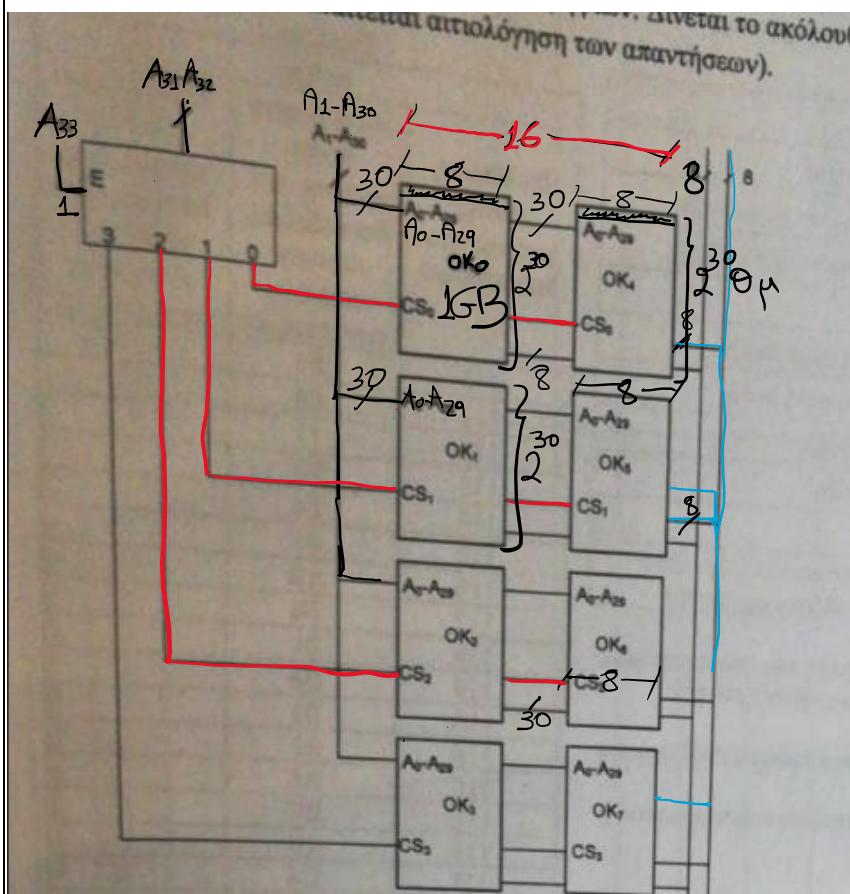
- Κάθε OK1 αφού περιέχει  $2^{21}$  θ.μ., έχει  $2^{21} = 2^1 \cdot 2^{20} = 2$  Μθέσεις των 2 bits η καθεμία.
  - Κάθε OK2 αφού περιέχει  $2^{22}$  θ.μ., έχει  $2^{22} = 2^2 \cdot 2^{20} = 4$  Μθέσεις των 8 bits η καθεμία.
  - **Η κύρια μνήμη** που θα συγεδιάσουμε έχει  $6 \times 2^{20}$  θ.μ. = 6 Μθέσεις των 8 bits η καθεμία

Address bus  $\{ A_{31} - A_0 \}$



## Λύση Θέματος κύριας μνήμης – Ιανουάριος 2020

Σε ένα υπολογιστή ο επεξεργαστής έχει αρτηρία διευθύνσεων των 34 δυαδικών ψηφίων και αρτηρία δεδομένων των 16 δυαδικών ψηφίων. Δίνεται το ακόλουθο σύστημα μνήμης. Να συμπληρώσετε τον ακόλουθο πίνακα (δεν απαιτείται αιτιολόγηση των απαντήσεων)



πλήθος θέσεων μνήμης κάθε ολοκληρωμένου κυκλώματος OK <sub>i</sub>	2 <sup>30</sup>
πλήθος δυαδικών ψηφίων <u>ανά θέση</u> μνήμης κάθε ολοκληρωμένου κυκλώματος OK <sub>i</sub>	8
πλήθος δυαδικών ψηφίων <u>ανά θέση</u> του συστήματος μνήμης	16
πλήθος ψηφιολέξεων που προσπελαύνεται ανά προσπέλαση του συστήματος μνήμης	2
Οργάνωση του συστήματος μνήμης (διαφύλλωση)	χαρητηριστικά
Χωρητικότητα του συστήματος μνήμης περιοχή διεύθυνσεων που καλύπτεται από το σύστημα μνήμης (χρώτη και τελευταία διεύθυνση στο δεκαεξαδικό)	8 GB

### Λύση

- α)  $2^{30}$  θ.μ. (λόγω της εσωτερικής οργάνωσης που έχουμε που είναι 1 byte/θ.μ.). Έχουμε  $2^{30}$  θ.μ. x 1byte/θ.μ.
- β) 8 bits (λόγω του μεγέθους του data bus που συνδέεται σε κάθε OKM).
- γ) 16 bits (προσδιορίζεται από το συνολικό μέγεθος του data bus), το οποίο είναι 16 bits.
- δ) 2 ψηφιολέξεις, διότι τα δύο OKM έχουν **κοινό σήμα ενεργοποίησης**, οπότε επιλέγονται ταυτόχρονα από κάθε OKM μια θέση μνήμης = 1 ψηφιολέξη.
- ε) Χαμηλής τάξης διαφύλλωση (επειδή από το address bus δεν χρησιμοποιείται το bit A<sub>0</sub>).
- ζ) Το κάθε OKM περιέχει  $2^{30}$  θ.μ. = 1 G θ.μ. με 8 bits/θ.μ. Επειδή διαθέτουμε 8 OKM, η συνολική χωρητικότητα του συστήματος μνήμης είναι 8 G θ.μ. x 8 bits/θ.μ. = 8 G θ.μ. x 1byte/θ.μ. = **8 Gbyte** (με 16 bits/θ.μ.). Εναλλακτικά, κάθε OKM έχει  $2^{30}$  θ.μ. =  $2^{30}$  bytes = 1 GB και συνολικά υπάρχουν 8 τέτοια OKM, άρα η συνολική χωρητικότητα είναι  $8 \times 1 \text{ GB} = 8 \text{ GB}$ .
- η) **Περιοχή διεύθυνσεων:** Αρχική διεύθυνση: **1xx.....x**, επειδή το bit A<sub>33</sub> συνδέεται απευθείας με τον αποκωδικοποιητή χωρίς αντιστροφέα και με δεδομένο ότι το E (Enable) του αποκωδικοποιητή είναι θετικής λογικής, πρέπει το A<sub>33</sub> = 1. Σε περίπτωση που σχεδιάσουμε τον πλήρη χάρτη μνήμης (memory map), αυτός φαίνεται παρακάτω:

A <sub>33</sub> A <sub>32</sub> A <sub>31</sub> A <sub>30</sub> A <sub>29</sub>	A <sub>1</sub> A <sub>0</sub>
1 00 000 0000 0000 0000 0000 000X	200000000
1 00 111 1111 1111 1111 1111 1111 111X	27FFFFFF
1 01 000 0000 0000 0000 0000 0000 000X	280000000
1 01 111 1111 1111 1111 1111 1111 111X	2FFFFFFF
1 10 000 0000 0000 0000 0000 0000 000X	300000000
1 10 111 1111 1111 1111 1111 1111 111X	37FFFFFF
1 11 000 0000 0000 0000 0000 0000 000X	380000000
1 11 111 1111 1111 1111 1111 1111 111X	3FFFFFFF

Επομένως, η αρχική διεύθυνση είναι η **200000000** και η τελική διεύθυνση είναι η **3FFFFFFF**.

## Θέμα Φεβρουαρίου 2021

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης OK0 με χωρητικότητα 32 Mbits και 8 δυαδικά ψηφία ανά θέση μνήμης καθώς και ολοκληρωμένα κυκλώματα μνήμης OK1 με χωρητικότητα 1 MBytes και 8 δυαδικά ψηφία ανά θέση μνήμης. Πόσα ολοκληρωμένα κυκλώματα τύπου OK0 και πόσα τύπου OK1 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 + π.χ. 4+12 το οποίο σημαίνει 4 τύπου OK0 και 12 τύπου OK1. Προσοχή μη βάλετε κενά μεταξύ των συμβόλων.

Αποτέλεσμα:  .

### Άνση

#### OK0

1 θ.μ. 8 bits  
x; 32 Mbits

$$= 2^5 \times 2^{20}/2^3 = 2^{22} \text{ θ.μ.}$$

#### OK1

1 θ.μ. 8 bits  
x; 1 MB

$$x = 2^{20} \times 2^3/2^3 = 2^{20} \text{ θ.μ.}$$

#### Κύρια μνήμη

1 θ.μ. 16 bits  
x; 176 Mbits

$$x = 176 \times 2^{20}/16 = 11 \times 2^{20} \text{ θ.μ.}$$

Οριζόντια επανάληψη: 2 (8 bits + 8 bits), πάντα δημιουργεί το επιθυμητό εύρος μνήμης

Κατακόρυφη επανάληψη:  $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{11 \times 2^{20} \text{ θ.μ.}}{2^{22} \text{ θ.μ.}} = 11 \times 2^{-2} = \frac{11}{4} = 2.75$ . Δεν επιστρέφεται ακέραιο πηλίκο

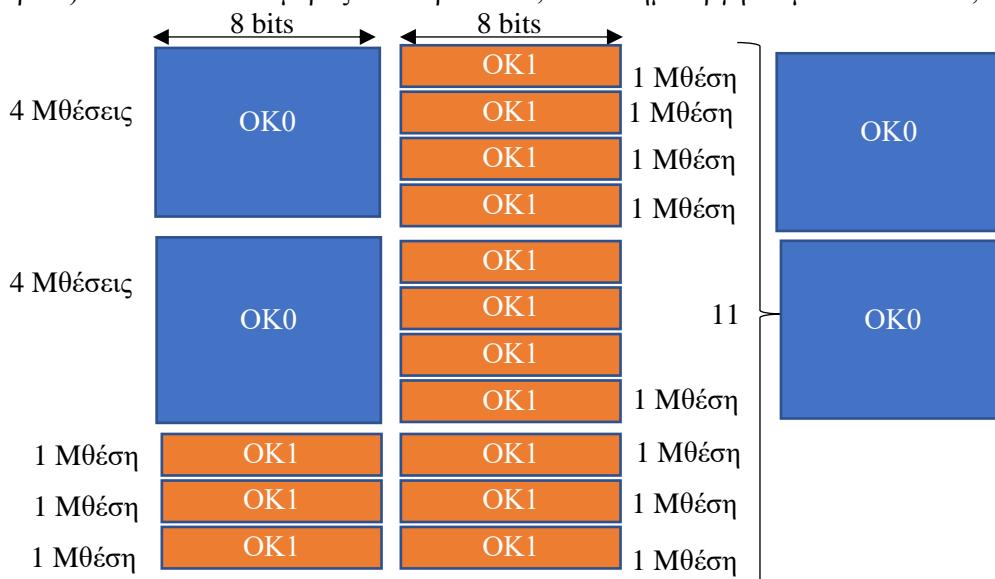
Κατακόρυφη επανάληψη:  $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{11 \times 2^{20} \text{ θ.μ.}}{2^{20} \text{ θ.μ.}} = 11$ . Επιστρέφεται ακέραιο πηλίκο

- Κάθε OK0 αφού περιέχει  $2^{22}$  θ.μ., έχει  $2^2 \times 2^{20} = 4$  Μθέσεις του ενός byte η καθεμία.

- Κάθε OK1 αφού περιέχει  $2^{20}$  θ.μ., έχει 1 Μθέσεις του ενός byte η καθεμία.

- Η κύρια μνήμη που θα σχεδιάσουμε έχει  $11 \times 2^{20}$  θ.μ. = 11 Μθέσεις των 16 bits η καθεμία

Για να σχεδιάσουμε το επιθυμητό μέγεθος της κύριας μνήμης, θα πρέπει να επαναλάβουμε κατακόρυφα 11 OK1 (έχουμε ακέραιο πηλίκο) και όσον αφορά τα OK0, θα πρέπει να επαναλάβουμε κατακόρυφα 2 OK0 (στρογγυλοποίηση προς το μικρότερο ακέραιο) και το δεκαδικό μέρος που περισσεύει, θα το δημιουργήσουμε από τα OK1, όπως φαίνεται:



Άρα, χρειαζόμαστε 2 OK0 + 14 OK1  $\rightarrow 2+14$

**Επαλήθευση:**  $2 * 32 \text{ Mbits} + 14 * 1 \text{ MB} = 64 \text{ Mbits} + 14 * 8 \text{ Mbits} = 64 \text{ Mbits} + 112 \text{ Mbits} = 176 \text{ Mbits}$

**Σημείωση:** Επειδή η εκφόνηση αναφέρει να χρησιμοποιήσουμε το ελάχιστο δυνατό πλήθος ολοκληρωμένων, μπορούμε στη δεξιά στήλη να αντικαταστήσουμε ανά 4 τα OK1 με ένα OK0, οπότε η τελική απάντηση είναι: 4 OK0 + 6 OK1.

Αυτό συμβαίνει διότι τα δύο είδη OKM έχουν την ίδια εσωτερική οργάνωση. Αν δεν την είχαν, κάτι τέτοιο δεν θα ήταν εφικτό.

## Θέμα Ιουνίου 2021

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης OK0 με χωρητικότητα 16 Mbits και 8 δυαδικά ψηφία ανά θέση μνήμης καθώς και ολοκληρωμένα κυκλώματα μνήμης OK1 με χωρητικότητα 0.5 MBytes και 4 δυαδικά ψηφία ανά θέση μνήμης. Πόσα ολοκληρωμένα κυκλώματα τύπου OK0 και πόσα τύπου OK1 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 OK0 και 12 τύπου OK1. Προσοχή μη βάλετε κενά μεταξύ των συμβόλων.

Αποτέλεσμα:  .

### Άνση

#### OK0

1 θ.μ. 8 bits

x; 16 Mbits

$$x = 2^4 \times 2^{20}/2^3 = 2^{21} \text{ θ.μ.}$$

#### OK1

1 θ.μ. 4 bits

x; 1/2 MB

$$x = \frac{1}{2} * 2^{20} \times 2^3/2^2 = 2^{-1} \times 2^{21} \text{ θ.μ.} = 2^{20} \text{ θ.μ.}$$

#### Κύρια μνήμη

1 θ.μ. 12 bits

x; 60 Mbits

$$x = 60 \times 2^{20}/12 = 5 \times 2^{20} \text{ θ.μ.}$$

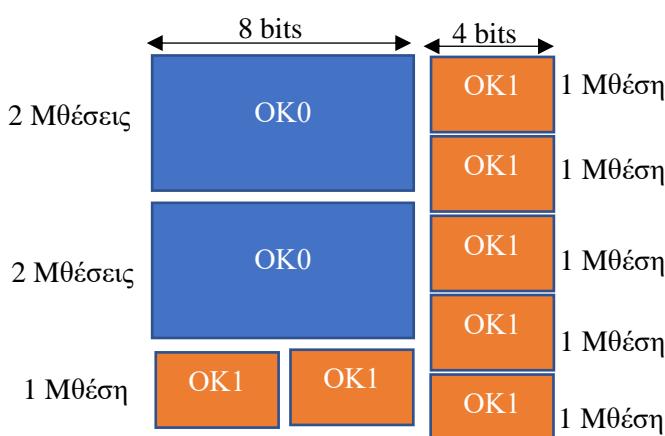
Οριζόντια επανάληψη: 2 (8 bits + 4 bits), πάντα δημιουργεί το επιθυμητό εύρος μνήμης

Κατακόρυφη επανάληψη:  $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{5 \times 2^{20} \text{ θ.μ.}}{2^{21} \text{ θ.μ.}} = 5 \times 2^{-1} = 2.5$ . Δεν επιστρέφεται ακέραιο πηλίκο

Κατακόρυφη επανάληψη:  $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{5 \times 2^{20} \text{ θ.μ.}}{2^{20} \text{ θ.μ.}} = 5$ . Επιστρέφεται ακέραιο πηλίκο

- Κάθε **OK0** αφού περιέχει  $2^{21}$  θ.μ., έχει  $2^{21} = 2^1 2^{20}$  θ.μ. = 2 Μθέσεις του ενός byte η καθεμία.
- Κάθε **OK1** αφού περιέχει  $2^{20}$  θ.μ., έχει 1 Μθέση των 4 bits η καθεμία.
- Η **κύρια μνήμη** που θα σχεδιάσουμε έχει  $5 \times 2^{20}$  θ.μ. = 5 Μθέσεις των 12 bits η καθεμία

Για να σχεδιάσουμε το επιθυμητό μέγεθος της κύριας μνήμης, θα πρέπει να επαναλάβουμε κατακόρυφα 5 OK1 (έχουμε ακέραιο πηλίκο) και όσον αφορά τα OK0, θα πρέπει να επαναλάβουμε κατακόρυφα 2 OK0 (στρογγυλοποίηση προς το μικρότερο ακέραιο) και το δεκαδικό μέρος που περισσεύει, θα το δημιουργήσουμε από τα OK1, όπως φαίνεται παρακάτω:



Άρα, χρειαζόμαστε 2 OK0 + 7 OK1 → 2+7

**Επαλήθευση:**  $2 * 16 \text{ Mbits} + 7 * 1/2 \text{ MB} = 32 \text{ Mbits} + 7 * 4 \text{ Mbits} = 32 \text{ Mbits} + 28 \text{ Mbits} = 60 \text{ Mbits}$

1 byte	8 bits
<u>1/2 MB</u>	<u>x;</u>

$$x = 2^3 \times 0.5 \times 2^{20} \text{ bits} = 2^3 \times 2^{-1} \times 2^{20} \text{ bits} = 2^2 \times 2^{20} \text{ bits} = 4 \text{ Mbits.}$$

**Σημείωση:** τώρα δεν μπορούμε να αντικαταστήσουμε το OK1 με OK0 (για να πετύχουμε το ελάχιστο πλήθος των ολοκληρωμένων, διότι αυτά δεν έχουν την ίδια εσωτερική οργάνωση).

## Θέμα Ιουνίου 2020

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης OK0 με χωρητικότητα 48Mbits και 6 δυαδικά ψηφία ανά θέση μνήμης καθώς και ολοκληρωμένα κυκλώματα μνήμης OK1 με χωρητικότητα 1.5 MBytes και 3 δυαδικά ψηφία ανά θέση μνήμης. Πόσα ολοκληρωμένα κυκλώματα τύπου OK0 και πόσα τύπου OK1 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 OK0 και 12 τύπου OK1. Προσοχή μη βάλετε κενά μεταξύ των συμβόλων.

Αποτέλεσμα:  .

### Άνση

OK0	OK1	Κύρια μνήμη
1 θ.μ. 6 bits x; 48 Mbits	1 θ.μ. 3 bits x 1.5 MB	1 θ.μ. 9 bits x 252 Mbits
$x = 48 \times 2^{20}/6 = 8 \times 2^{20} \text{ θ.μ.}$	$x = 1.5 \times 2^{20} \times 2^3/3 = 4 \times 2^{20} \text{ θ.μ.}$	$x = 252 \times 2^{20}/9 = 28 \times 2^{20} \text{ θ.μ.}$

Οριζόντια επανάληψη: 2 (6 bits + 3 bits), πάντα δημιουργεί το επιθυμητό εύρος μνήμης

Κατακόρυφη επανάληψη:  $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{28 \times 2^{20} \text{ θ.μ.}}{8 \times 2^{20} \text{ θ.μ.}} = \frac{28}{8} = 3.5$ . Δεν επιστρέφεται ακέραιο πηλίκο

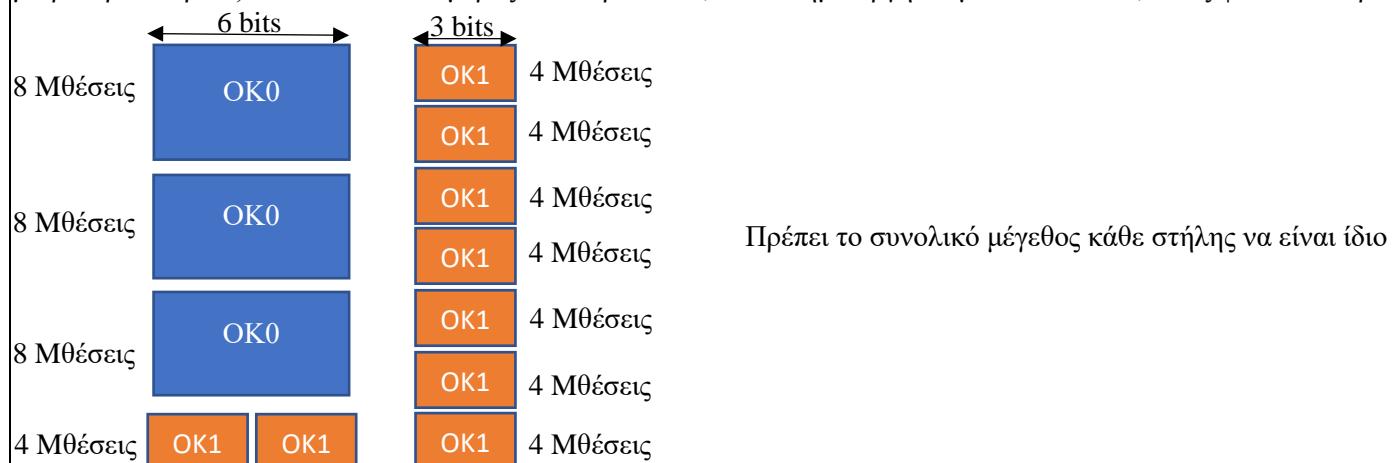
Κατακόρυφη επανάληψη:  $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{28 \times 2^{20} \text{ θ.μ.}}{4 \times 2^{20} \text{ θ.μ.}} = 7$ . Επιστρέφεται ακέραιο πηλίκο

Κάθε OK0 αφού περιέχει  $8 \times 2^{20} \text{ θ.μ.}$ , έχει  $8 \times 2^{20} = 8 \text{ Mθέσεις}$  των 6 bits η καθεμία.

Κάθε OK1 αφού περιέχει  $4 \times 2^{20} \text{ θ.μ.}$ , έχει  $4 \text{ Mθέσεις}$  των 3 bits η καθεμία.

Η κύρια μνήμη που θα σχεδιάσουμε έχει  $28 \times 2^{20} \text{ θ.μ.} = 28 \text{ Mθέσεις}$  των 9 bits η καθεμία

Για να σχεδιάσουμε το επιθυμητό μέγεθος της κύριας μνήμης, θα πρέπει να επαναλάβουμε κατακόρυφα 7 OK1 (έχουμε ακέραιο πηλίκο) και όσον αφορά τα OK0, θα πρέπει να επαναλάβουμε κατακόρυφα 3 OK0 (στρογγυλοποίηση προς το μικρότερο ακέραιο) και το δεκαδικό μέρος που περισσεύει, θα το δημιουργήσουμε από τα OK1, όπως φαίνεται παρακάτω:



Άρα, χρειαζόμαστε 3 OK0 + 9 OK1 → 3+9

Επαλήθευση:  $3 * 48 \text{ Mbits} + 9 * 1.5 \text{ MB} = 144 \text{ Mbits} + 9 * 12 \text{ Mbits} = 252 \text{ Mbits}$

1 byte	8 bits	
1.5 MB	x;	$x = 1.5 \times 2^{20} \times 2^3 = 12 \times 2^{20} \text{ bits} = 12 \text{ Mbits}$

Σημείωση: τώρα δεν μπορούμε να αντικαταστήσουμε OK1 με OK0 (για να πετύχουμε το ελάχιστο πλήθος των ολοκληρωμένων, διότι αυτά δεν έχουν την ίδια εσωτερική οργάνωση).

## Θέμα Σεπτεμβρίου 2020

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης OK0 με χωρητικότητα 256 Kbits και οργάνωση οκτώ δυαδικών ψηφίων ανά θέση μνήμης καθώς και ολοκληρωμένα κυκλώματα OK2 με χωρητικότητα 32 Kbytes και οργάνωση δύο δυαδικών ψηφίων ανά θέση. Πόσα ολοκληρωμένα κυκλώματα τύπου OK0 και πόσα τύπου OK2 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 OK0 και 12 τύπου OK2.  .

### Αύση

#### OK0

$$1 \text{ θ.μ.} \quad 8 \text{ bits}$$

$$x; \quad \underline{256 \text{ Kbits}}$$

$$x = 2^8 \times 2^{10}/2^3 = 2^{15} \text{ θ.μ.}$$

#### OK2

$$1 \text{ θ.μ.} \quad 2 \text{ bits}$$

$$x \quad \underline{32 \text{ KB}}$$

$$x = 2^5 \times 2^{10} \times 2^3/2^1 = 2^{18}/2^1 = 2^{17} \text{ θ.μ.}$$

#### Κύρια μνήμη

$$1 \text{ θ.μ.} \quad 10 \text{ bits}$$

$$x \quad \underline{160 \text{ KB}}$$

$$x = 160 \times 2^{10} \times 2^3 / 10 = 2^{17} \text{ θ.μ.}$$

Οριζόντια επανάληψη: 2 (8 bits + 2 bits)

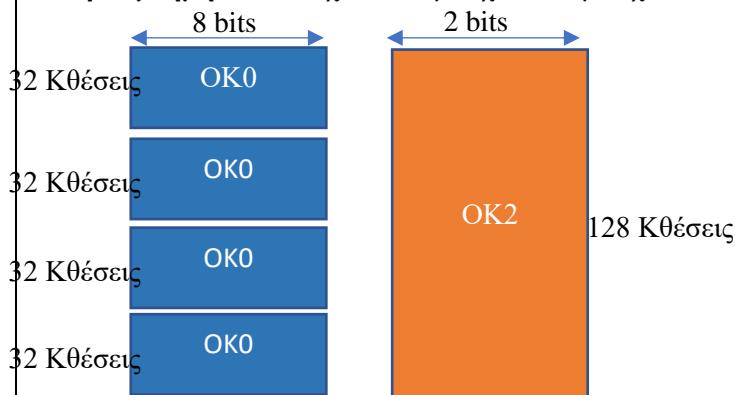
Κατακόρυφη επανάληψη για OK2:  $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{2^{17} \text{ θ.μ.}}{2^{17} \text{ θ.μ.}} = 1$  επανάληψη

Κατακόρυφη επανάληψη για OK0:  $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{2^{17} \text{ θ.μ.}}{2^{15} \text{ θ.μ.}} = 4$  επαναλήψεις

Κάθε OK0 αφού περιέχει  $2^{15}$  θ.μ., έχει  $2^5 \times 2^{10} = 32$  Κθέσεις των 8 bits η καθεμία.

Κάθε OK1 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 OK0 + 1 OK2 → 04+01.

**Επαλήθευση:** Έχουμε συνολική χωρητικότητα  $4 * 256 \text{ Kbits} + 1 * 32 \text{ Kbytes} = 4 * 32 \text{ Kbytes} + 1 * 32 \text{ Kbytes} = 160 \text{ Kbytes}$

$$1 \text{ byte} \quad 8 \text{ bits}$$

$$x; \quad \underline{256 \text{ Kbits}}$$

$$x = 2^8 \times 2^{10}/2^3 = 2^{15} \text{ bytes} = 2^5 \times 2^{10} \text{ bytes} = 32 \text{ KBytes}$$

## Θέμα Φεβρουαρίου 2021

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης OK0 με χωρητικότητα 256 Kbits και οργάνωση οκτώ δυαδικών ψηφίων ανά θέση μνήμης καθώς και ολοκληρωμένα κυκλώματα OK2 με χωρητικότητα 16 Kbytes και οργάνωση δύο δυαδικών ψηφίων ανά θέση. Πόσα ολοκληρωμένα κυκλώματα τύπου OK0 και πόσα τύπου OK2 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 OK0 και 12 τύπου OK2.  .

### Αύση

#### OK0

1 θ.μ. 8 bits

x; 256 Kbits

$$x = 2^8 \times 2^{10}/2^3 = 2^{15} \text{ θ.μ.}$$

#### OK2

1 θ.μ. 2 bits

x 16 KB

$$x = 2^4 \times 2^{10} \times 2^3/2^1 = 2^{17}/2^1 = 2^{16} \text{ θ.μ.}$$

#### Κύρια μνήμη

1 θ.μ. 10 bits

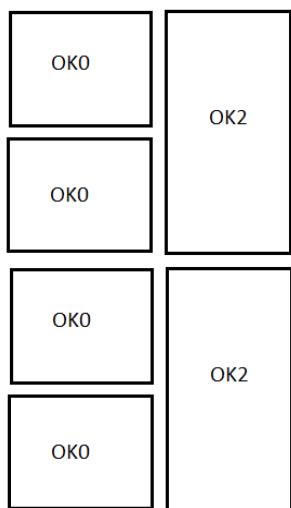
x 160 KB

$$x = 160 \times 2^{10} \times 2^3/10 = 2^{17} \text{ θ.μ.}$$

Οριζόντια επανάληψη: 2 (8 bits + 2 bits)

Κατακόρυφη επανάληψη για OK2:  $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{2^{17} \text{ θ.μ.}}{2^{16} \text{ θ.μ.}} = 2$  επαναλήψεις

Κατακόρυφη επανάληψη για OK0:  $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{2^{17} \text{ θ.μ.}}{2^{15} \text{ θ.μ.}} = 4$  επαναλήψεις



Επομένως χρειαζόμαστε 4 OK0 + 2 OK2 → 04+02.

**Επαλήθευση:** Έχουμε συνολική χωρητικότητα  $4 * 32 \text{ Kbytes} + 2 * 16 \text{ Kbytes} = 160 \text{ Kbytes}$

1 byte 8 bits

x; 256 Kbits

$$x = 2^8 \times 2^{10}/2^3 = 2^{15} \text{ bytes} = 2^5 \times 2^{10} \text{ bytes} = 32 \text{ KBytes}$$

## Θέμα Ιουνίου 2021

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης OK0 με χωρητικότητα 512 Kbits και οργάνωση οκτώ δυαδικών ψηφίων ανά θέση. Πόσα ολοκληρωμένα κυκλώματα τύπου OK0 και πόσα τύπου OK2 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 OK0 και 12 τύπου OK2.

**Άνση**

**OK0**

1 θ.μ. 8 bits

x; 512 Kbits

**OK2**

1 θ.μ. 2 bits

x 16 KB

$$x = 2^4 \times 2^{10} \times 2^3 / 2^1 = 2^{17} / 2^1 = 2^{16} \text{ θ.μ.}$$

$$x = 2^9 \times 2^{10} / 2^3 = 2^{16} \text{ θ.μ.}$$

**Κύρια μνήμη**

1 θ.μ. 14 bits

x 224 KB

$$x = 224 \times 2^{10} \times 2^3 / 14 = 16 \times 2^{13} \text{ θ.μ.} = 2^{17} \text{ θ.μ.}$$

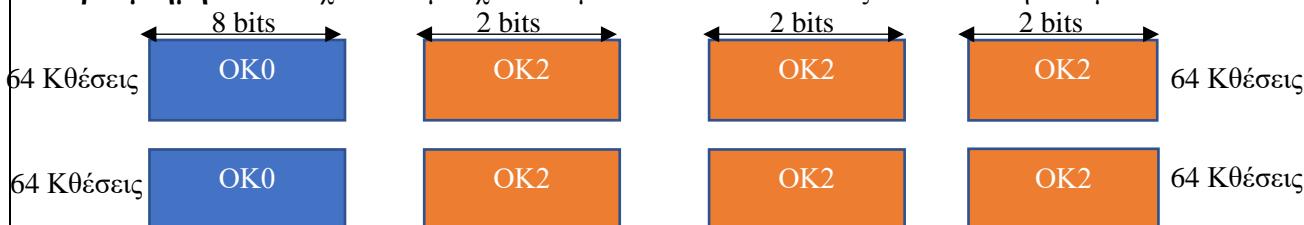
Οριζόντια επανάληψη: 4 (8 bits + 2 bits + 2 bits + 2 bits)

$$\text{Κατακόρυφη επανάληψη: } \frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{2^{17} \text{ θ.μ.}}{2^{16} \text{ θ.μ.}} = 2 \text{ επαναλήψεις}$$

Κάθε OK0 αφού περιέχει  $2^{16}$  θ.μ., έχει  $2^{16} = 2^6 \times 2^{10} = 64$  Κθέσεις του ενός byte η καθεμία.

Κάθε OK1 αφού περιέχει  $2^{16}$  θ.μ., έχει  $2^{16} = 2^6 \times 2^{10} = 64$  Κθέσεις των 2 bits η καθεμία.

Η κύρια μνήμη που θα σχεδιάσουμε έχει  $2^{17}$  θ.μ. =  $2^7 \times 2^{10} = 128$  Κθέσεις των 14 bits η καθεμία



Επομένως χρειαζόμαστε 2 OK0 και 6 OK2 → 02+06

**Επαλήθευση:**  $2 * 512 \text{ Kbits} + 6 * 16 \text{ KB} = 2 * 64 \text{ KB} + 96 \text{ KB} = 128 \text{ KB} + 96 \text{ KB} = 224 \text{ KB}$

1 byte 8 bits  
x; 512 Kbits

$$x = 2^9 \times 2^{10} / 2^3 = 2^{16} \text{ bytes} = 2^6 \times 2^{10} \text{ bytes} = 64 \text{ KBytes}$$

## Θέμα Ιουνίου 2020

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης OK1 με χωρητικότητα 32 Kbytes και οργάνωση τεσσάρων δυαδικών ψηφίων ανά θέση. Πόσα ολοκληρωμένα κυκλώματα τύπου OK1 και πόσα τύπου OK2 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 OK1 και 12 τύπου OK2.

**Άνση**

**OK1**

1 θ.μ. 4 bits

x; 32KB

**OK2**

1 θ.μ. 2 bits

x 16KB

$$x = (2^5 * 2^{10} * 2^3) / 2^2 = 2^{18} / 2^2 = 2^{16} \text{ θ.μ.}$$

$$x = 2^4 \times 2^{10} \times 2^3 / 2^1 = 2^{17} / 2^1 = 2^{16} \text{ θ.μ.}$$

**Κύρια μνήμη**

1 θ.μ. 10 bits

x 160KB

$$x = 160 \times 2^{10} \times 2^3 / 10 = 16 \times 2^{13} \text{ θ.μ.} = 2^4 \times 2^{13} \text{ θ.μ.} = 2^{17} \text{ θ.μ.}$$

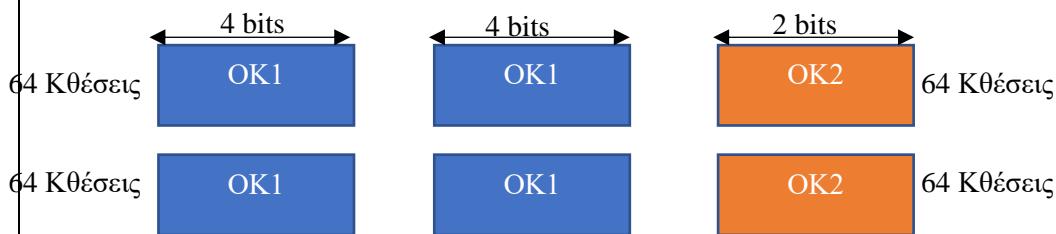
Οριζόντια επανάληψη: 3 (4 bits + 4 bits + 2 bits)

$$\text{Κατακόρυφη επανάληψη: } \frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{2^{17} \text{ θ.μ.}}{2^{16} \text{ θ.μ.}} = 2 \text{ επαναλήψεις}$$

Κάθε OK0 αφού περιέχει  $2^{16}$  θ.μ., έχει  $2^{16} = 2^6 \times 2^{10} = 64$  Κθέσεις των 4 bits η καθεμία.

Κάθε OK1 αφού περιέχει  $2^{16}$  θ.μ., έχει  $2^{16} = 2^6 \times 2^{10} = 64$  Κθέσεις των 2 bits η καθεμία

Η κύρια μνήμη που θα σχεδιάσουμε έχει  $2^{17}$  θ.μ. =  $2^7 \cdot 2^{10} = 128$  Κθέσεις των 10 bits η καθεμία



Επομένως χρειαζόμαστε 4 OK1 και 2 OK2  $\rightarrow 04+02$

**Επαλήθευση:**  $4 * 32 \text{ KB} + 2 * 16 \text{ KB} = 128 \text{ KB} + 32 \text{ KB} = 160 \text{ KB}$

### Θέμα Σεπτεμβρίου 2021

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης OK0 με χωρητικότητα 16 Mbits και οργάνωση οκτώ δυαδικών ψηφίων ανά θέση μνήμης καθώς και ολοκληρωμένα κυκλώματα OK1 με χωρητικότητα 0.5 MB και οργάνωση τεσσάρων δυαδικών ψηφίων ανά θέση. Πόσα ολοκληρωμένα κυκλώματα τύπου OK0 και πόσα τύπου OK1 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 OK0 και 12 τύπου OK1.

### Λύση

Διαθέτουμε **OK0**: 16 Mbits x **8 bits** και **OK1**: 0.5 MB x **4 bits**  $\rightarrow$  Στόχος: **KM** 60 Mbits x **12 bits**

### OK0

1 θ.μ. 8 bits  
 $x; 16 \text{ Mbits}$   $x = (2^4 * 2^{20}) / 2^3 = 2^{21} = 2 * 2^{20} \text{ θ.μ. των ενός byte η καθεμία} = 2\text{M} \text{ θέσεις των 8 bits η καθεμία}$

### OK1

1 θ.μ. 4 bits  
 $x 0.5 \text{ MB}$   $x = 2^{-1} * 2^{20} * 2^3 / 2^2 = 2^{20} \text{ θ.μ. των 4 bits η καθεμία} = 1\text{M} \text{ θέση των 4 bits η καθεμία}$

### Κύρια μνήμη

1 θ.μ. 12 bits  
 $x 60 \text{ Mbits}$   $x = 60 * 2^{20} / 12 = 5 * 2^{20} \text{ θ.μ. των 4 bits η καθεμία} = 5\text{M} \text{ θέσεις των 12 bits η καθεμία}$

Οριζόντια επανάληψη: 2 (8 bits + 4 bits)

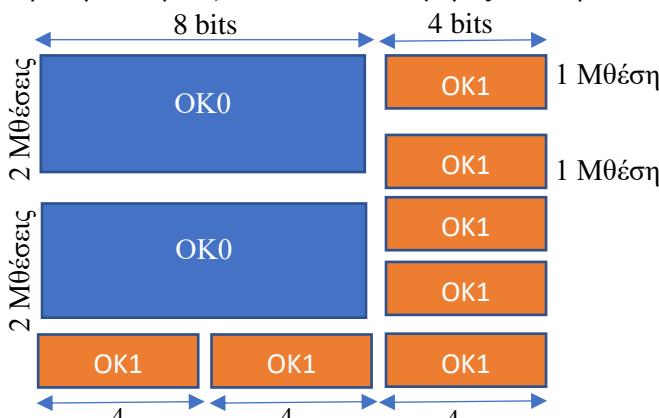
Κατακόρυφη επανάληψη:  $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{5 * 2^{20} \text{ θ.μ.}}{2^{21} \text{ θ.μ.}} = 5 * 2^{-1} = 2.5$ . Δεν επιστρέφεται ακέραιο πηλίκο

Κατακόρυφη επανάληψη:  $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{5 * 2^{20} \text{ θ.μ.}}{2^{20} \text{ θ.μ.}} = 5$ . Επιστρέφεται ακέραιο πηλίκο

Κάθε OK0 αφού περιέχει  $2^{21}$  θ.μ., έχει  $2^{21} = 2^1 \cdot 2^{20} = 2$  Mθέσεις του ενός byte η καθεμία.

Κάθε OK1 αφού περιέχει  $2^{20}$  θ.μ., έχει 1Mθέση των 4 bits η καθεμία.

Για να σχεδιάσουμε το επιθυμητό μέγεθος της κύριας μνήμης, θα πρέπει να επαναλάβουμε κατακόρυφα 5 OK1 (έχουμε ακέραιο πηλίκο) και όσον αφορά τα OK0, θα πρέπει να επαναλάβουμε κατακόρυφα 2 OK0 (στρογγυλοποίηση προς το μικρότερο ακέραιο) και το δεκαδικό μέρος που περισσεύει, θα το δημιουργήσουμε από τα OK1, όπως φαίνεται παρακάτω:



**Επαλήθευση:** Υπολογίσαμε  $2 \text{ OK0} + 7 \text{ OK1} = 2 \times 16 \text{ Mbits} + 7 \times 0.5 \text{ MB} = 32 \text{ Mbits} + 7 \times 2^{-1} \times 2^{20} \times 2^3 \text{ bits} = 32 \text{ Mbits} + 7 \times 2^{-1} \times 2^3 \text{ Mbits} = 32 \text{ Mbits} + 28 \text{ Mbits} = 60 \text{ Mbits}$ .

### Θέμα προόδου Ιανουαρίου 2022

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης OK0 με χωρητικότητα 3Mbyte και οργάνωση 12 δυαδικών ψηφίων ανά θέση μνήμης καθώς και ολοκληρωμένα κυκλώματα OK1 με χωρητικότητα 12 Mbits και οργάνωση 12 δυαδικών ψηφίων ανά θέση. Πόσα ολοκληρωμένα κυκλώματα τύπου OK0 και πόσα τύπου OK1 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 OK0 και 12 τύπου OK1.  

### Λύση

Διαθέτουμε **OK0**: 3 Mbyte x **12 bits** και **OK1**: 12 Mbits x **12 bits** → Στόχος: **KM** 15 Mbytes x **24 bits**

### OK0

1 θ.μ. 12 bits

$$x; \quad 3 \text{ Mbyte} \quad x = (3 \times 2^{20} \times 2^3)/12 = 2^{21} = 2 \times 2^{20} \text{ θ.μ.} = 2\text{M} \text{ θέσεις των 12 bits η καθεμία}$$

### OK1

1 θ.μ. 12 bits

$$x \quad 12 \text{ Mbits} \quad x = 12 \times 2^{20}/12 = 2^{20} = 1\text{M} \text{ θέση των 12 bits η καθεμία}$$

### Κύρια μνήμη

1 θ.μ. 24 bits

$$x \quad 15 \text{ Mbytes} \quad x = 15 \times 2^{20} \times 2^3/24 = 5 \times 2^{20} = 5\text{M} \text{ θέσεις των 24 bits η καθεμία}$$

Οριζόντια επανάληψη: **2** (12 bits + 12 bits)

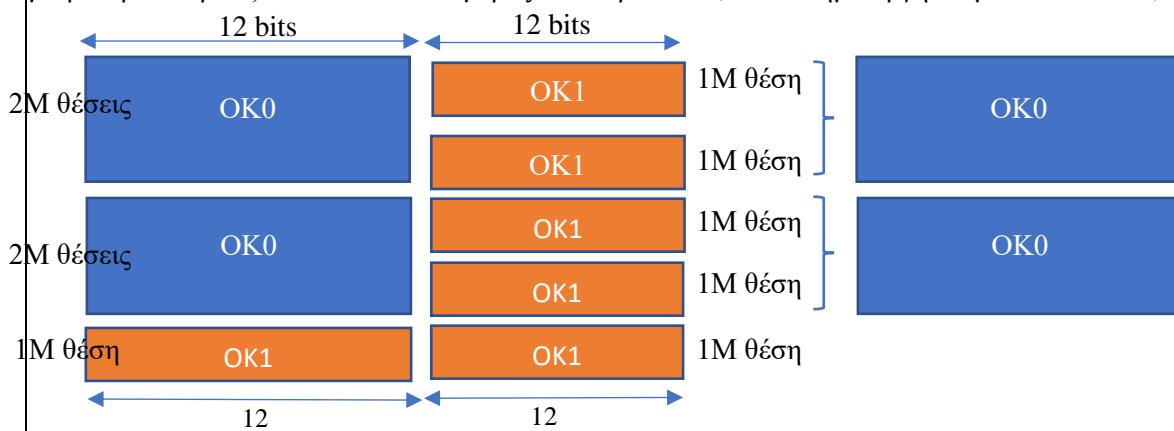
Κατακόρυφη επανάληψη:  $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{5 \times 2^{20} \text{ θ.μ.}}{2 \times 2^{20} \text{ θ.μ.}} = 2.5$ . **Δεν επιστρέφεται ακέραιο πηλίκο**

Κατακόρυφη επανάληψη:  $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{5 \times 2^{20} \text{ θ.μ.}}{2^{20} \text{ θ.μ.}} = 5$ . **Επιστρέφεται ακέραιο πηλίκο**

Κάθε OK0 αφού περιέχει  $2^{21}$  θ.μ., έχει  $2^{21} = 2^1 2^{20} = 2$  Mθέσεις των 12 bits η καθεμία.

Κάθε OK1 αφού περιέχει  $2^{20}$  θ.μ., έχει 1Mθέση των 12 bits η καθεμία.

Για να σχεδιάσουμε το επιθυμητό μέγεθος της κύριας μνήμης, θα πρέπει να επαναλάβουμε κατακόρυφα 5 OK1 (έχουμε ακέραιο πηλίκο) και όσον αφορά τα OK0, θα πρέπει να επαναλάβουμε κατακόρυφα 2 OK0 (στρογγυλοποίηση προς το μικρότερο ακέραιο) και το δεκαδικό μέρος που περισσεύει, θα το δημιουργήσουμε από τα OK1, όπως φαίνεται παρακάτω:



**Επαλήθευση:** Υπολογίσαμε  $2 \text{ OK0} + 6 \text{ OK1} = 2 \times 3 \text{ Mbyte} + 6 \times 12 \text{ Mbits} = 6 \text{ Mbyte} + 9 \text{ Mbyte} = 15 \text{ MB}$ . Εναλλακτικά  $10 \text{ M.θ.μ.} \times 12 \text{ bits/θ.μ.} = 10 \times 2^{20} \text{ θ.μ.} \times 12 \text{ bits/θ.μ.} = 10 \times 2^{20} \times 12 \text{ bits} = 120 \times 2^{20} \text{ bits} = 15 \times 2^{20} \text{ bytes} = 15 \text{ MB}$

**Σημείωση:** Επειδή ή άσκηση ζητά το μικρότερο δυνατό πλήθος, η σωστή απάντηση είναι **4 + 2**, αφού αντικαθιστούμε στη δεξιά στήλη ανά **δύο** τα OKM των 1Mθέσεων με ένα OKM των 2Mθέσεων. Αυτό είναι εφικτό, διότι τα διαφορετικά OK έχουν ίδια εσωτερική οργάνωση (12 bits/θ.μ.). Σε διαφορετική περίπτωση, κάτι τέτοιο δεν θα ήταν εφικτό.

## Θέμα προόδου Ιανουαρίου 2022

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης OK0 με χωρητικότητα 80 Mbits και οργάνωση 10 δυαδικών ψηφίων ανά θέση μνήμης καθώς και ολοκληρωμένα κυκλώματα OK1 με χωρητικότητα 2.5 Mbyte με χωρητικότητα και οργάνωση 10 δυαδικών ψηφίων ανά θέση. Πόσα ολοκληρωμένα κυκλώματα τύπου OK0 και πόσα τύπου OK1 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 OK0 και 12 τύπου OK1.  

### Λύση

Διαθέτουμε **OK0**: 80 Mbits x **10 bits** και **OK1**: 2.5 Mbyte x **10 bits** → Στόχος: **KM** 400 Mbits x **20 bits**

### OK0

1 θ.μ. 10 bits

$$x = (80 * 2^{20})/10 = 8 * 2^{20} = 2^3 * 2^{20} \text{ θ.μ.} = 2^{23} \text{ θ.μ.} = 8 \text{ Μθέσεις των 10 bits η καθεμία}$$

### OK1

1 θ.μ. 10 bits

$$x = (2.5 * 2^{20} * 2^3)/10 = 2 * 2^{20} = 2^{21} \text{ θ.μ.} = 2 \text{ Μθέσεις των 10 bits η καθεμία}$$

### Κύρια μνήμη

1 θ.μ. 20 bits

$$x = (400 * 2^{20})/20 = 20 * 2^{20} = 20 \text{ Μθέσεις των 20 bits η καθεμία}$$

x 400 Mbits

Οριζόντια επανάληψη: 2 (10 bits + 10 bits)

$$\text{Κατακόρυφη επανάληψη: } \frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{20 \times 2^{20} \text{ θ.μ.}}{8 \times 2^{20} \text{ θ.μ.}} = 2.5. \text{ Δεν επιστρέφεται ακέραιο πηλίκο}$$

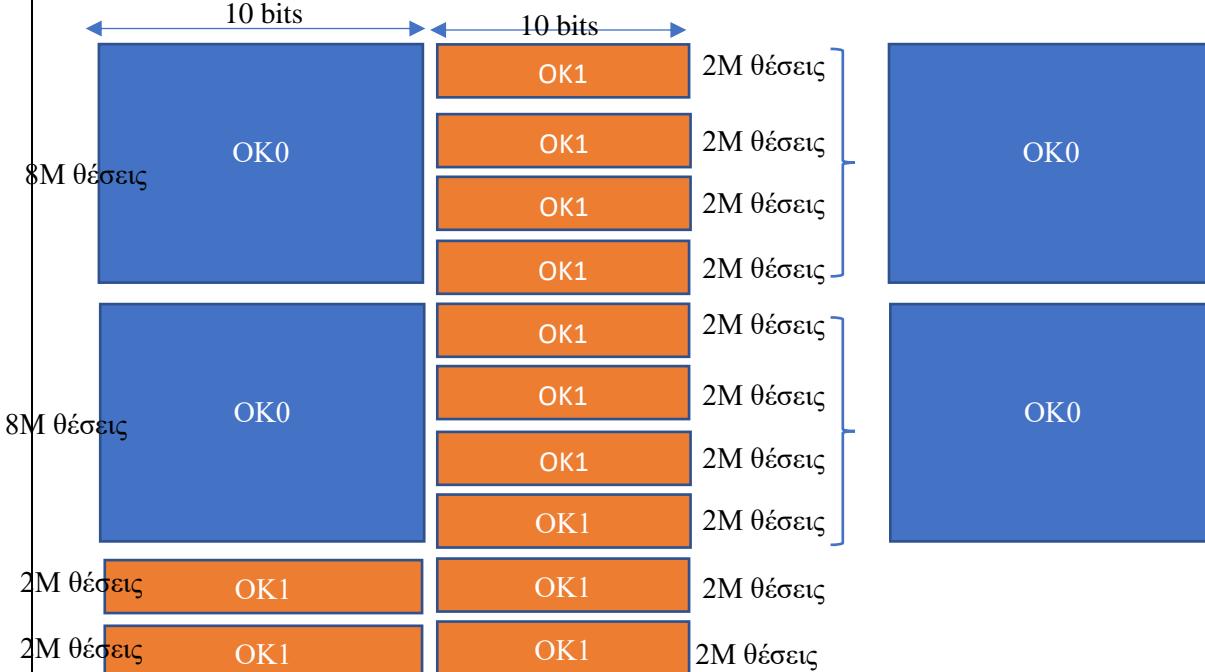
$$\text{Κατακόρυφη επανάληψη: } \frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{20 \times 2^{20} \text{ θ.μ.}}{2 \times 2^{20} \text{ θ.μ.}} = 10. \text{ Επιστρέφεται ακέραιο πηλίκο}$$

Κάθε OK0 αφού περιέχει  $2^{23}$  θ.μ., έχει  $2^{23} = 2^{20} = 8$  Μθέσεις των 10 bits η καθεμία.

Κάθε OK1 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 OK1 (έχουμε ακέραιο πηλίκο) και όσον αφορά τα OK0, θα πρέπει να επαναλάβουμε κατακόρυφα 2 OK0 (στρογγυλοποίηση προς το μικρότερο ακέραιο) και το δεκαδικό μέρος που περισσεύει, θα το δημιουργήσουμε από τα OK1, όπως φαίνεται παρακάτω:



**Επολήθευση:** Υπολογίσαμε  $2 \text{ OK0} + 12 \text{ OK1} = 2 \times 80 \text{ Mbits} + 12 \times 2.5 \text{ Mbyte} = 160 \text{ Mbits} + 30 \text{ Mbyte} = 160 \text{ Mbits} + 240 \text{ Mbits} = 400 \text{ Mbits.}$

**Σημείωση:** Επειδή η άσκηση ζητά το **μικρότερο** δυνατό πλήθος, η σωστή απάντηση είναι **4+4**, αφού αντικαθιστούμε στη δεξιά στήλη ανά **τέσσερα** τα OKM των 2Μθέσεων με ένα OKM των 8Μθέσεων.

### Θέμα προόδου Ιανουαρίου 2022

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης OK0 με χωρητικότητα 32 Mbits και οργάνωση 8 δυαδικών ψηφίων ανά θέση μνήμης καθώς και ολοκληρωμένα κυκλώματα OK1 με χωρητικότητα 1 Mbyte με χωρητικότητα και οργάνωση 8 δυαδικών ψηφίων ανά θέση. Πόσα ολοκληρωμένα κυκλώματα τύπου OK0 και πόσα τύπου OK1 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 OK0 και 12 τύπου OK1.  

#### Λύση

Διαθέτουμε **OK0**: 32 Mbits x **8 bits** και **OK1**: 1 Mbyte x **8 bits** → Στόχος: **KM** 176 Mbits x **16 bits**

#### OK0

1 θ.μ. 8 bits

x; 32 Mbits

#### OK1

1 θ.μ. 8 bits

x 1 Mbyte

$$x = (32 * 2^{20})/8 = 4 * 2^{20} = 2^{22} = 2^2 \times 2^{20} \text{ θ.μ.} = 4 \text{ Μθέσεις των 8 bits η καθεμία}$$

$$x = (1 * 2^{20} \times 2^3)/8 = 2^{20} = 1 \text{ Μθέσεις των 8 bits η καθεμία}$$

#### Κύρια μνήμη

1 θ.μ. 16 bits

x 176 Mbits

$$x = (176 * 2^{20})/16 = 11 * 2^{20} = 11 \text{ Μθέσεις των 16 bits η καθεμία}$$

Οριζόντια επανάληψη: 2 (8 bits + 8 bits)

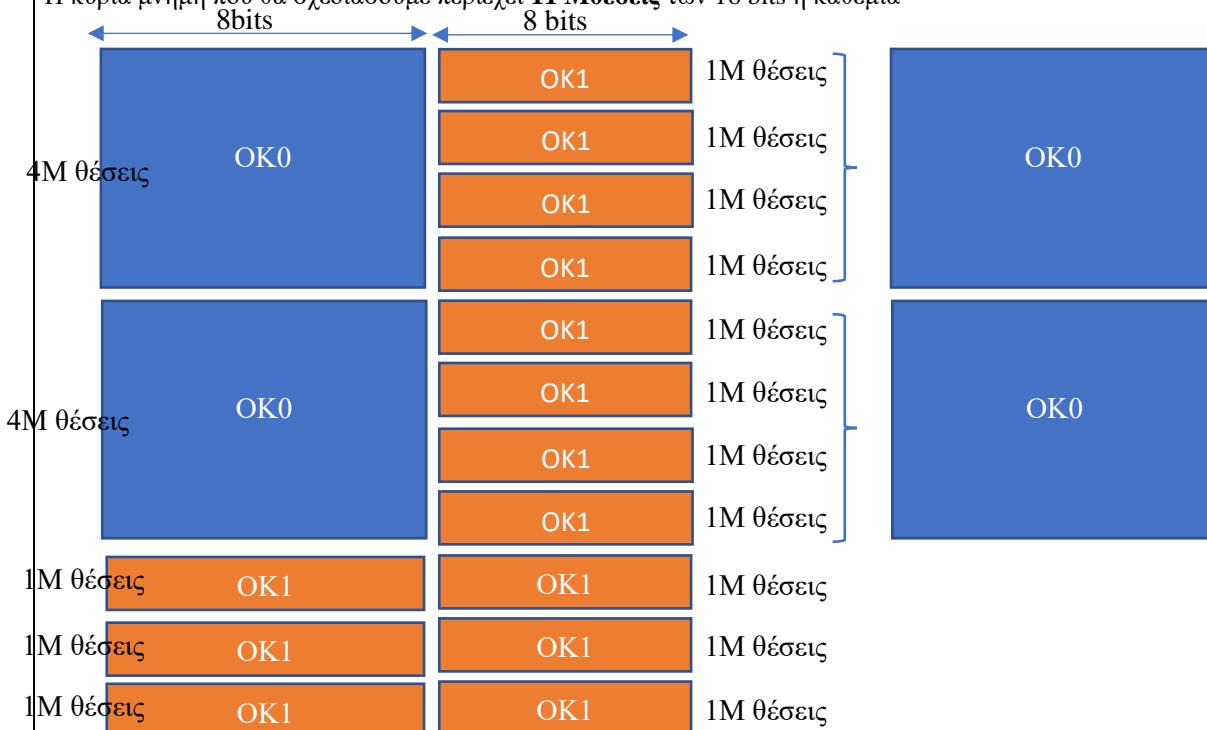
$$\text{Κατακόρυφη επανάληψη: } \frac{\theta.\mu. \text{ που θέλουμε να έχουμε}}{\theta.\mu. \text{ που διαθέτουμε από τα OKM}} = \frac{11 \times 2^{20} \text{ θ.μ.}}{4 \times 2^{20} \text{ θ.μ.}} = 2.75. \text{ Δεν επιστρέφεται ακέραιο πηλίκο}$$

$$\text{Κατακόρυφη επανάληψη: } \frac{\theta.\mu. \text{ που θέλουμε να έχουμε}}{\theta.\mu. \text{ που διαθέτουμε από τα OKM}} = \frac{11 \times 2^{20} \text{ θ.μ.}}{2^{20} \text{ θ.μ.}} = 11. \text{ Επιστρέφεται ακέραιο πηλίκο}$$

Κάθε OK0 αφού περιέχει  $2^{22}$  θ.μ., έχει  $2^{22} = 2^2 \times 2^{20} = 4$  Μθέσεις των 8 bits η καθεμία.

Κάθε OK1 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



**Επαλήθευση:** Υπολογίσαμε 2 OK0 και 14 OK1 =  $2 \times 32 \text{ Mbits} + 14 \times 1 \text{ Mbyte} = 64 \text{ Mbits} + 14 \text{ Mbyte} = 64 \text{ Mbits} + 112 \text{ Mbits} = 176 \text{ Mbits.}$

**Σημείωση:** Επειδή η άσκηση ζητά το **μικρότερο** δυνατό πλήθος, η σωστή απάντηση είναι **4+6**, αφού αντικαθιστούμε στη δεξιά στήλη ανά **τέσσερα** τα OKM των 1Μθέσεων με ένα OKM 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 N - 1 διαρκούν μια χρονική περίοδο η κάθε μία.

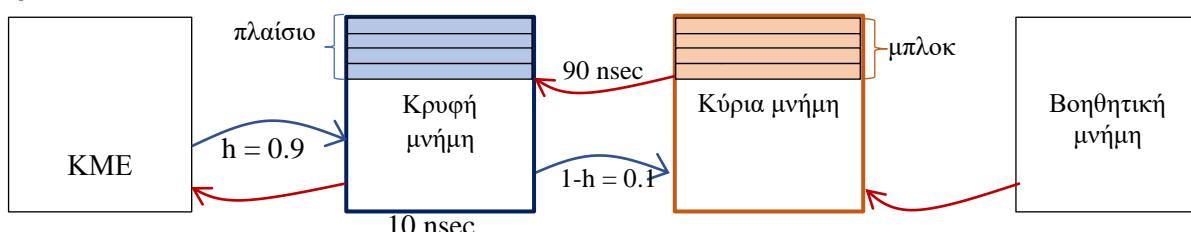
α. Ποια είναι η μικρότερη τιμή του N που μπορεί να μας εξασφαλίσει τη **γρηγορότερη** μεταφορά ενός μπλοκ πληροφορίας από την κύρια μνήμη στην κρυφή.

β. Πόσες χρονικές περιόδους 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 KME μπορεί να διαβάσει πληροφορία μόνο από την κρυφή μνήμη.

δ. Σχεδιάστε την κύρια μνήμη με οργάνωση N – δρόμων χαμηλής τάξης διαφύλλωση για την τιμή του N του α' ερωτήματος

### Άνση

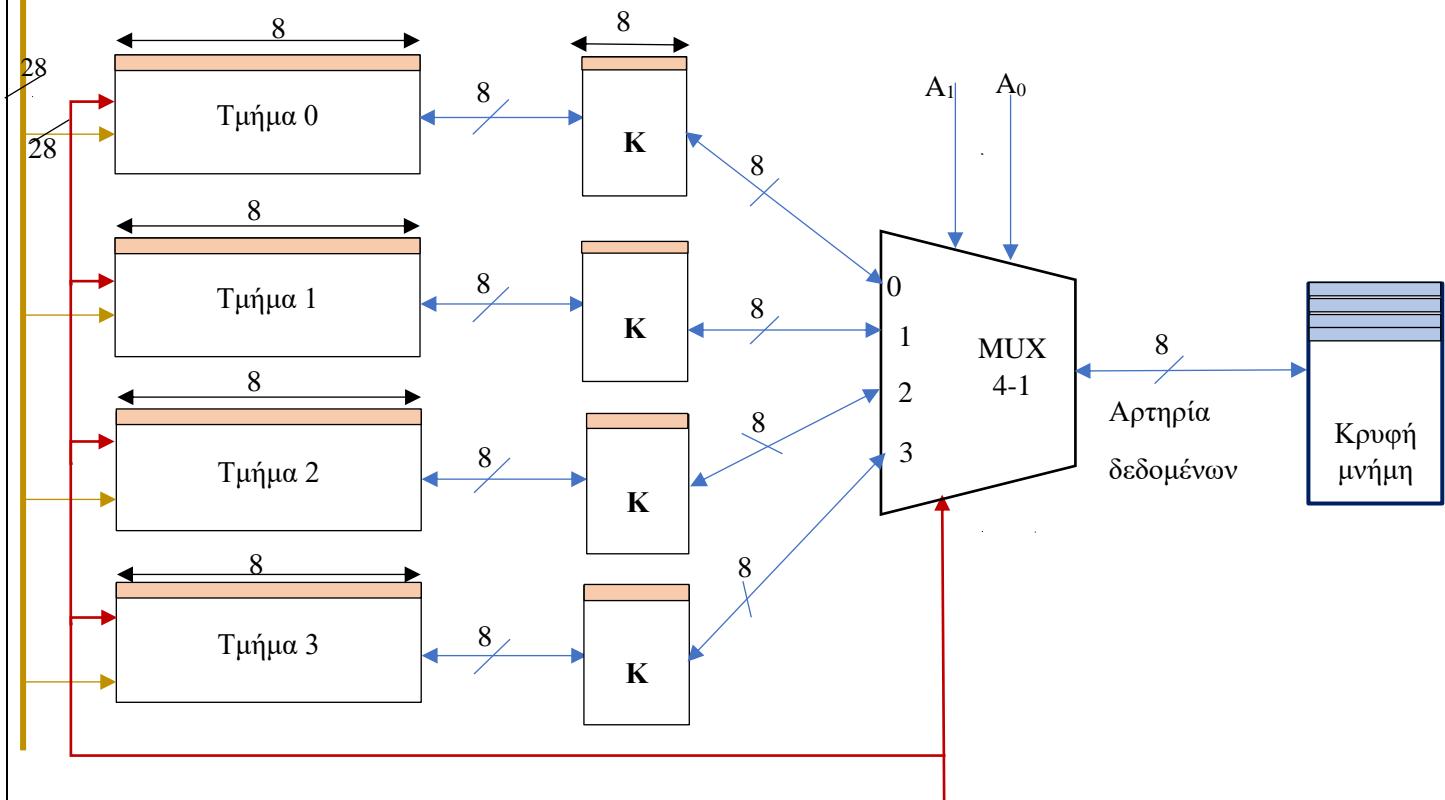


α. Ισχύει ότι ένα μπλοκ της κύριας μνήμης → πλαίσιο της κρυφής μνήμης. **Ένα μπλοκ έχει πάντα ίδιο μέγεθος με το πλαίσιο** και είναι -όπως και το πλαίσιο- μια συλλογή γειτονικών (συνεχόμενων) θ.μ. Θέλουμε να μεταφέρουμε ένα μπλοκ (συλλογή 4 θ.μ. στην προκειμένη περίπτωση) της κύριας μνήμης σε ένα πλαίσιο (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 N** (όπου N = πλήθος δρόμων ή τρόπων της κύριας μνήμης που έχει οργάνωση χαμηλής τάξης διαφύλλωση) για να πετύχουμε τη **γρηγορότερη μεταφορά**. Προφανώς το N = 4, αφού με 4 – δρόμους μπορούμε να μεταφέρουμε το περιεχόμενο 4 θ.μ. με μια προσπέλαση από την κύρια στην κρυφή μνήμη. Αν το N = 2, θα χρειαζόντουσαν δύο προσπελάσεις μνήμης και θα υπήρχε μεγαλύτερη καθυστέρηση στην περίπτωση αυτή. Στη συνέχεια προχωράμε με τη σχεδίαση της N – δρόμων χαμηλής τάξης διαφύλλωση μνήμης, με N = 4. Το χαρακτηριστικό σε αυτή τη σχεδίαση είναι ότι δεν υπάρχει αποκωδικοποιητής για να επιλέγει τα τμήματα μνήμης ή τα OKM και υπάρχει πολυπλέκτης  $2^{\mu} - 1$ , ο οποίος επιλέγει κάθε φορά -με βάση τις γραμμές επιλογής του- μια από τις εισόδους του (δηλ. το περιεχόμενο ενός καταχωρητή K) και την οδηγεί στην έξοδο. Ως γραμμές επιλογής του MUX χρησιμοποιούνται τα λιγότερο σημαντικά bits της αρτηρίας διευθύνσεων.

δ. Η χωρητικότητα της K.M. = 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 που χρησιμοποιούνται για τη διευθυνσιοδότηση των τμημάτων.

β. Οι χρονικές περίοδοι που απαιτούνται για τη **μεταφορά ενός μπλοκ πληροφορίας από την κύρια μνήμη στην κρυφή** είναι 6 χρ. περίοδοι +  $(N - 1) \times 1$  χρ. περ. = 6 χρ. περ. +  $(4 - 1) \times 1$  χρ. περίοδοι = 9 χρ. περίοδοι =  $9 \times 10$  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:  $T = \sum_{i=1}^{n=2} (E_i - E_{i-1}) \times T_i = (E_1 - E_0) \times T_1 + (E_2 - E_1) \times T_2 = (0.9 - 0) \times 10 \text{ nsec} + (1 - 0.9) \times (90 + 10) \text{ nsec} = 0.9 \times 10 \text{ nsec} + 0.1 \times 100 \text{ nsec} = 19 \text{ nsec}$ .



### 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  $\frac{320 \text{ κ.ρ.}}{20 \text{ κ.ρ.}} = 16$  προσπελάσεις. 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

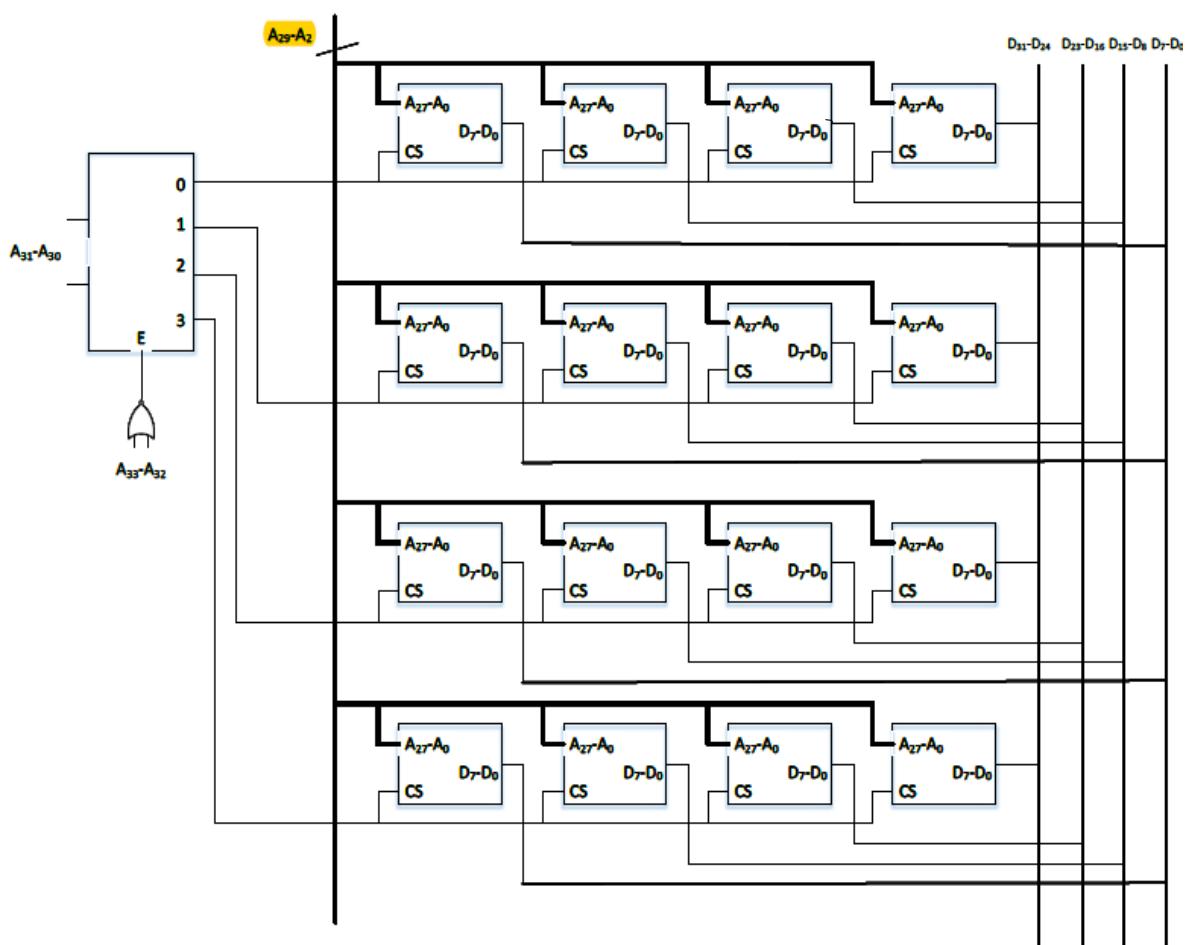
Έχετε στην διάθεσή σας ολοκληρωμένα κυκλώματα με χωρητικότητα 256 Mbytes (M ψηφιολέξεις) και οργάνωση 1 Byte (ψηφιολέξη) ανά θέση μνήμης.

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

α) Με βάση τη χωρητικότητα και την εσωτερική οργάνωση των OKM που διαθέτουμε, προκύπτει ότι κάθε OKM έχει 256 Mbytes = 256 Μθέσεις =  $2^8 \cdot 2^{20}$  θέσεις μνήμης =  $2^{28}$  θ.μ. Τα Gbytes = 4 Gθέσεις =  $4 \times 2^{30}$  θ.μ. =  $2^{32}$  θ.μ. Η διαίρεση  $2^{32}$  θ.μ./ $2^{28}$  θ.μ. = 16 επαναλήγεις, όμως το γεγονός ότι σε κάθε προσπέλαση μνήμης προσπελαύνονται ταυτόχρονα 4 bytes = 4 θ.μ. σημαίνει ότι έχουμε οργάνωση 4 – τρόπων χαμηλής τάξης διαφύλλωσης και το γεγονός ότι η **αρτηρία δεδομένων** είναι **των 32 δυαδικών ψηφίων** σημαίνει ότι θα πρέπει να τοποθετήσουμε σε κάθε δρόμο 4 OKM, τα οποία στη συνέχεια να συνδέσουμε απευθείας με την αρτηρία δεδομένων, προκειμένου να πολλαπλασιαστεί το μέγεθός της. Άρα δεν υπάρχει ανάγκη χρήση MUX και κατ' επέκταση καταχωρητών που συνδέονται απευθείας μαζί του. Παρά το ότι δεν χρησιμοποιείται MUX, **οι γραμμές A<sub>1</sub>-A<sub>0</sub> θεωρούνται δεσμευμένες**. Επειδή περισσεύουν bits, χρησιμοποιείται και αποκωδικοποιητής, για επιλογή της κάθε σειράς (δρόμου).



β) Η διεύθυνση 38A46B432 = 0011 1000 1010 0100 0110 1011 0100 0011 0010 δηλώνει ότι οι διευθύνσεις των bytes που προσπελαύνονται ταυτόχρονα είναι 38A46B430 - 38A46B433.

### 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 (KME) είναι:

- α. των 16 δυαδικών ψηφίων
- β. των 32 δυαδικών ψηφίων
- γ. των 8 δυαδικών ψηφίων

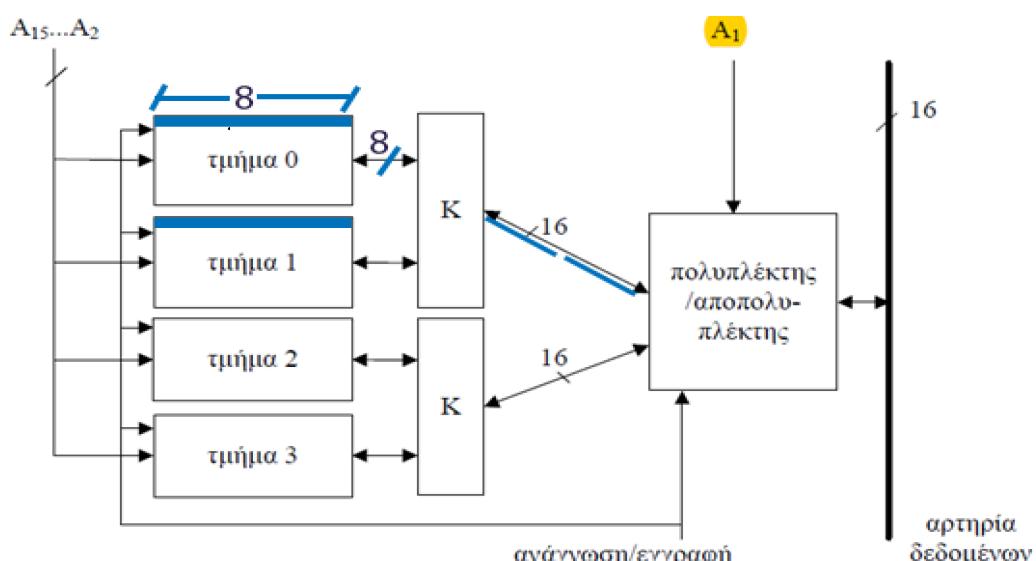
1. Να σχεδιάζεται το σύστημα μνήμης σε μπλοκ διάγραμμα για καθεμία των ανωτέρω περιπτώσεων. Να θεωρήσετε ότι η **αρτηρία διευθύνσεων** είναι **των 16 δυαδικών ψηφίων**. Όλη η απαιτούμενη πληροφορία να φαίνεται πάνω στο σχήμα.

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 υπόψιν 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.

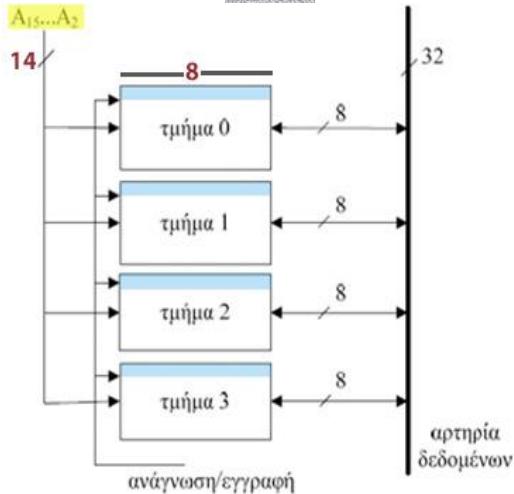
### Λύση

**Σε όλες τις περιπτώσεις που ακολουθούν, το μέγεθος του address bus είναι 16 bits.**

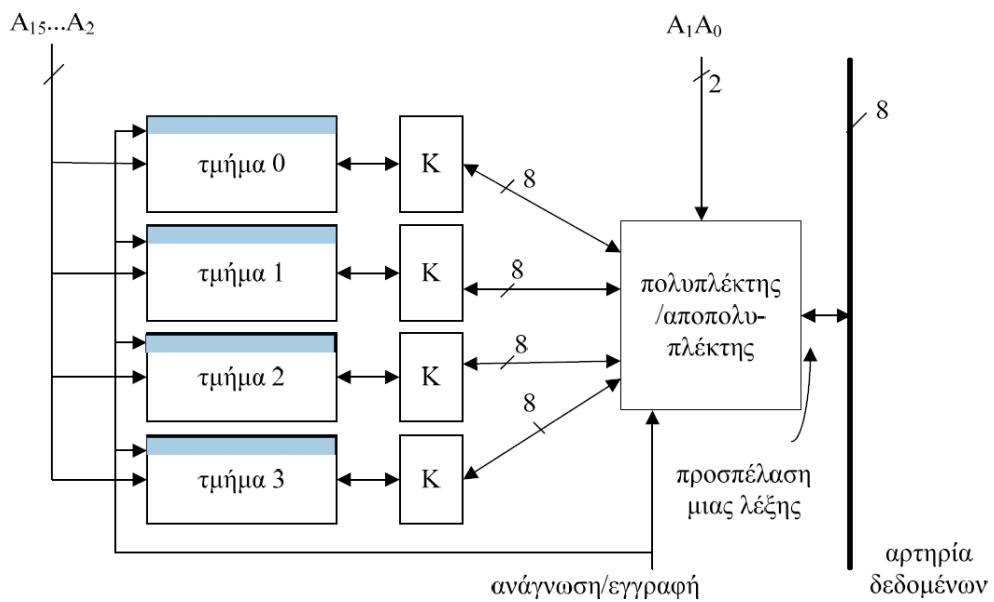
1a. Θα πρέπει να συνδυάσουμε ανά δύο τα OKM των 4 – τρόπων προκειμένου να πετύχουμε μια αρτηρία δεδομένων των 16 bits. Αυτός ο συνδυασμός επιτυγχάνεται με τους καταχωρητές, οι οποίοι επεκτείνονται και γίνονται των 16 – bits. Επειδή, κάθε τέτοιος καταχωρητής οδηγεί δύο θ.μ., **Θα πρέπει να χρησιμοποιηθεί το bit A<sub>1</sub> 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<sub>0</sub>, 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.  $2^4 \rightarrow 1$ . Επειδή το address bus = 16 bits, χρησιμοποιούνται τα 14 πιο σημαντικά (A<sub>15</sub> – A<sub>2</sub>) 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<sub>1</sub> και A<sub>0</sub> του address bus παραμένουν δεσμευμένα, και στη διευθυνσιοδότηση των τμημάτων χρησιμοποιούνται τα υπόλοιπα bits από το address bus, δηλ. τα bits A<sub>15</sub> – A<sub>2</sub>.** Από αυτό συμπεραίνουμε ότι κάθε τμήμα έχει  $2^{14}$  θ.μ.  $\times$  8 bits/θ.μ. =  $2^{14}$  θ.μ.  $\times$  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) =  $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{4 \text{ bytes}}{30 \text{ κ.ρ.} + 4 \text{ κ.ρ.}} = \frac{4 \text{ bytes}}{34 \text{ κ.ρ.}} = \frac{4 \text{ bytes}}{68 \text{ nsec}}$

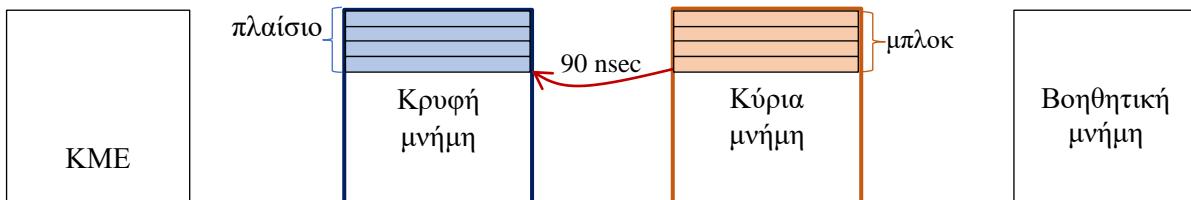
### Άσκηση 5.13 με οργάνωση χαμηλής τάξης διαφύλλωση μνήμης

Θεωρήστε σύστημα μνήμης αποτελούμενο από κρυφή μνήμη δεδομένων και ξεχωριστή κρυφή μνήμη εντολών με **4 λέξεις της μίας ψηφιολέξης (byte) ανά πλαίσιο**, κύρια μνήμη και βοηθητική μνήμη (το σύστημα δεν υποστηρίζει ιδεατή μνήμη). Κάθε θέση της κύριας μνήμης είναι των 8 δυαδικών ψηφίων. Η αρτηρία για την μεταφορά εντολών μεταξύ κρυφής μνήμης και κύριας μνήμης είναι των 8 δυαδικών ψηφίων. Η εγγραφή μίας ψηφιολέξης στην κρυφή μνήμη απαιτεί 1 κύκλο ρολογιού. Να σχεδιάσετε κύρια μνήμη με χωρητικότητα 64 Κψηφιολέξεις (Kbytes) **έτσι ώστε η μεταφορά ενός μπλοκ της κύριας μνήμης σε ένα πλαίσιο της κρυφής μνήμης εντολών να ολοκληρώνεται σε 16 κύκλους ρολογιού**. Έχετε στη διάθεσή σας ολοκληρωμένα κυκλώματα μνήμης DRAM με χωρητικότητα 8 Κψηφιολέξεις (Kbytes) με μία ψηφιολέξη ανά θέση μνήμης και χρόνο προσπέλασης ίσο με 12 κύκλους ρολογιού. Θεωρήστε ότι τα ολοκληρωμένα κυκλώματα μνήμης DRAM, τα οποία έχετε στη διάθεσή σας, δεν έχουν τη δυνατότητα γρήγορης προσπέλασης ομάδας θέσεων οι οποίες έχουν την ίδια διεύθυνση γραμμής, θεωρήστε ότι έχετε στη διάθεσή σας οποιαδήποτε άλλα κυκλώματα χρειάζεστε και ότι η αρτηρία διεύθυνσεων είναι των 16 δυαδικών ψηφίων.

#### Λύση

Διαθέτουμε OKM 8 Κψηφιολέξεις x 8 με 12 κ.ρ. → Στόχος K.M. 64 Κψηφιολέξεις x 8. με 16 κ.ρ.

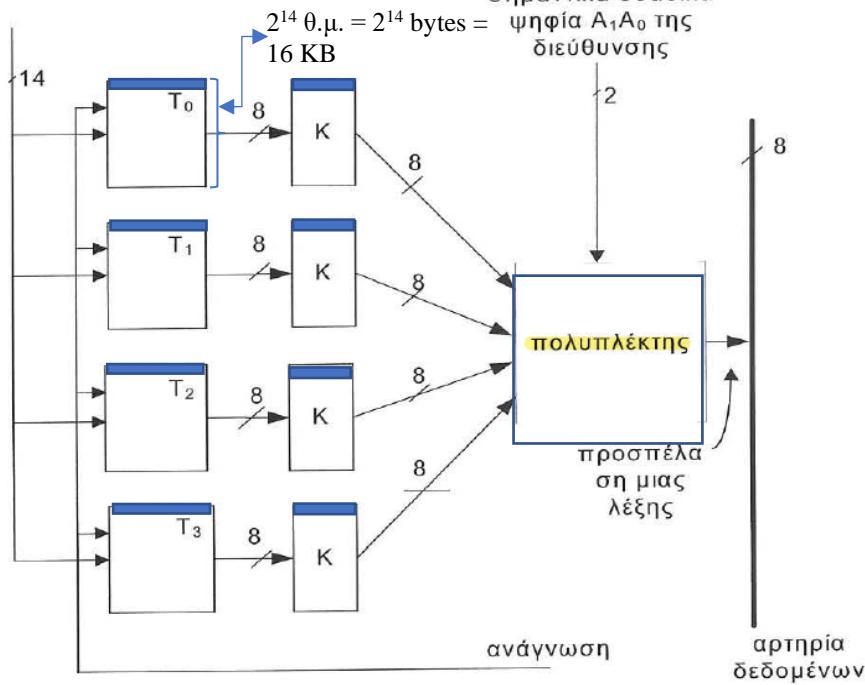
Φαινομενικά η άσκηση μοιάζει να χρειάζεται οργάνωση υψηλής τάξης διαφύλλωση μνήμης, όμως το γεγονός ότι τίθενται περιορισμοί στη μεταφορά ενός μπλοκ (αποτελούμενο από 4 θ.μ.) σε ένα πλαίσιο (αποτελούμενο από 4 θ.μ.) σε 16 κ.ρ. σημαίνει ότι πρέπει να χρησιμοποιηθεί οργάνωση χαμηλής τάξης διαφύλλωσης. Αυτό γιατί, κάθε προσπέλαση της κύριας μνήμης κοστίζει 12 κ.ρ. και αν **χρησιμοποιούσαμε οργάνωση υψηλής τάξης διαφύλλωση**, επειδή διαδοχικές διευθύνσεις προσπελαύνουν το ίδιο OKM, για τη μεταφορά ενός μπλοκ των 4 ψηφιολέξεων (4 θ.μ.) σε ένα αντίστοιχο πλαίσιο της κυφής, θα χρειαζόντουσαν 12 κ.ρ. x 4 προσπελάσεις = 48 κ.ρ. > 16 κ.ρ..



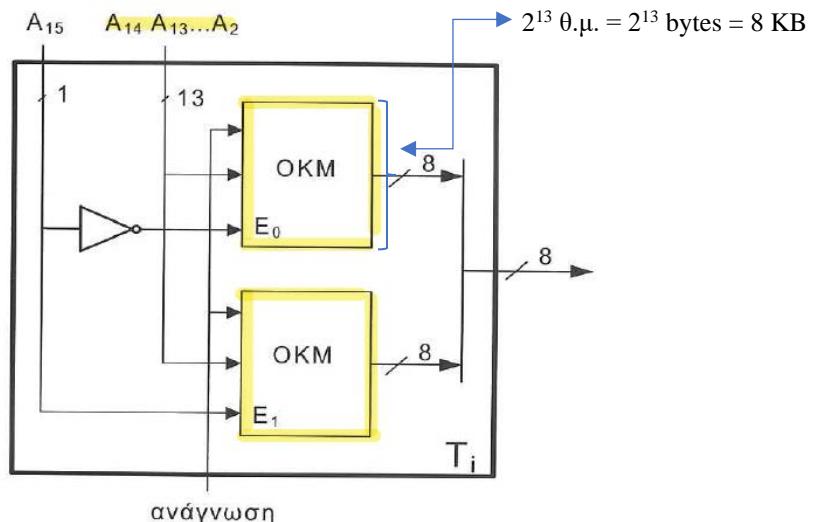
Άρα θα χρησιμοποιήσουμε οργάνωση χαμηλής τάξης διαφύλλωσης και μάλιστα 4 – τρόπων, καθώς θέλουμε να μεταφέρουμε με μια προσπέλαση ένα μπλοκ των 4 ψηφιολέξεων σε ένα πλαίσιο (με προφανώς ίδιο μέγεθος). Έτσι, με αυτή την οργάνωση μεταφέρουμε από κάθε δρόμο (τρόπο) 1 ψηφιολέξη έως τους καταχωρητές σε χρόνο ίσο με 12 κ.ρ. και στη συνέχεια για το πέρασμα αυτών των ψηφιολέξεων μέσω του MUX χρειάζονται 4 ακόμη κ.ρ. (ένας για κάθε ψηφιολέξη που περνά μέσω του MUX). Στη συνέχεια φαίνεται η 4 – τρόπων χαμηλής τάξης διαφύλλωση μνήμης: Υπενθυμίζεται ότι το μέγεθος του address bus = 16. Από αυτά, τα δύο λιγότερο σημαντικά bits ( $A_1 - A_0$ ) χρησιμοποιούνται ως γραμμές επιλογής του MUX και τα υπόλοιπα 14 bits χρησιμοποιούνται για τη διεύθυνσιο δότηση των τμημάτων. Αυτό σημαίνει ότι το κάθε τμήμα, έχοντας ως είσοδο αυτά τα 14 bits, περιέχει  $2^{14}$  θ.μ. =  $2^4 \times 2^{10}$  θ.μ. =  $2^4 \times 2^{10}$  bytes = 16 KB. Προσοχή, αυτά που διαθέτουμε, είναι OKM των 8 Κψηφιολέξεις =  $2^3 \times 2^{10}$  ψηφιολέξεις =  $2^3 \times 2^{10}$  θ.μ. =  $2^{13}$  θ.μ. Αυτό σημαίνει ότι το κάθε τμήμα αποτελείται από 2 επιμέρους τέτοια OKM που διαθέτουμε.

τα 14 αμέσως περισσότερο  
σημαντικά δυαδικά ψηφία  
 $A_{15} A_{14}...A_2$  της διεύθυνσης

τα 2 λιγότερο  
σημαντικά δυαδικά  
ψηφία  $A_1 A_0$  της  
διεύθυνσης



Στη συνέχεια φαίνεται το εσωτερικό του κάθε τμήματος, αποτελούμενο από 2 OKM. Από τα 14 bits του address bus που εισέρχονται στο κάθε τμήμα, τα 13 bits χρησιμοποιούνται για τη διεύθυνση διότι την ομάδα OKM που διαθέτουμε, καθώς κάθε τέτοιο έχει  $2^{13} \text{ θ.μ.}$  και το τελευταίο bit που περισσεύει ( $A_{15}$ ), χρησιμοποιείται για την επιλογή του κάθε OKM μέσα στο τμήμα. Όταν το  $A_{15}=0$  επιλέγεται το πάνω OKM και όταν το  $A_{15}=1$  επιλέγεται το κάτω OKM του κάθε τμήματος.



Τμήμα  $T_i$ , με  $i=0, 1, 2, 3$ .

## Θέμα Σεπτεμβρίου 2015 με σχεδιασμό κύριας μνήμης χαμηλής τάξης διαφύλλωσης

Θεωρήστε υπολογιστικό σύστημα με αρτηρία διευθύνσεων των 36 δυαδικών ψηφίων και αρτηρία δεδομένων των 16 δυαδικών ψηφίων. Να σχεδιάσετε 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 (OK) 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.

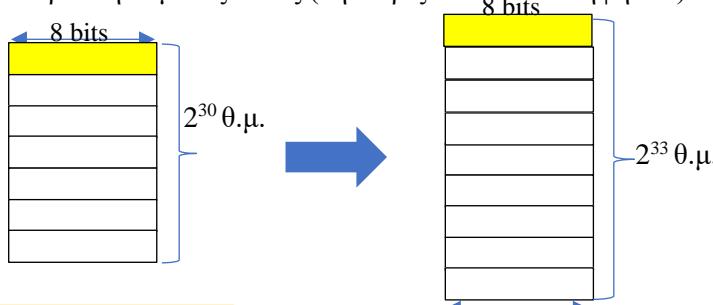
### Λύση

**Πρόκειται για οργάνωση χαμηλής τάξης διαφύλλωση μνήμης, αφού θέλουμε σε κάθε προσπέλαση να προσπελαύνεται το περιεχόμενο δύο διαδοχικών θέσεων μνήμης.**

**Διαθέτουμε OKM 1 GB x 8 bits/θ.μ. → Στόχος:** K.M. 8 GB x 8 bits/θ.μ.

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

**Οριζόντια επανάληψη** (στόχος είναι η δημιουργία του **κατάλληλου εύρους της θέσης μνήμης**). Παρότι έχουμε ίδια εσωτερική οργάνωση τόσο στα OKM που διαθέτουμε, όσο και στην K.M. 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 OKM παράλληλα μεταξύ τους (δηλ. οριζόντια επανάληψη = 2).



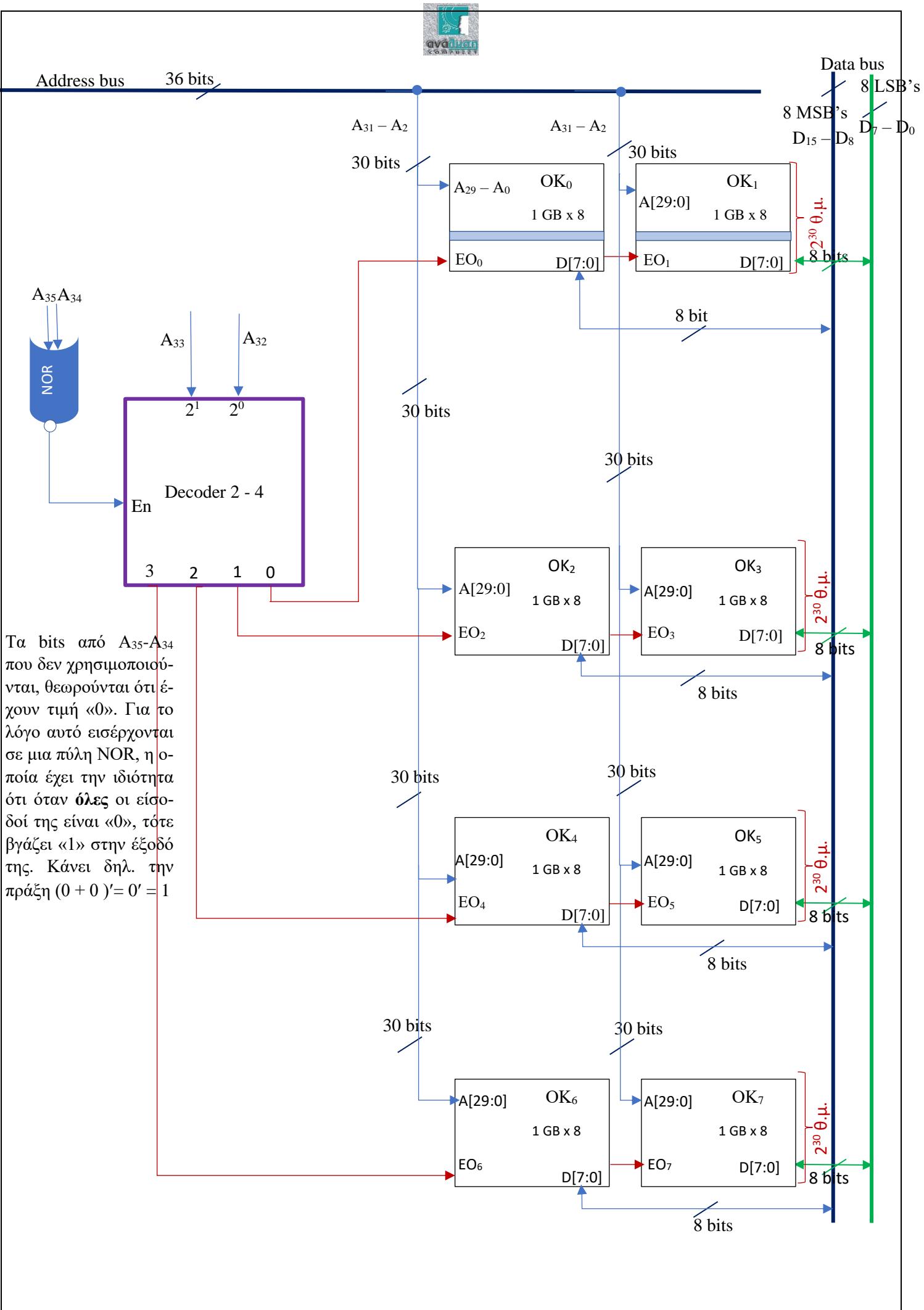
### Κατακόρυφη επανάληψη.

$1 \text{ θ.μ.}$ $x;$	$1 \text{ byte}$ $1 \text{ GB}$	$x = \frac{2^{30} \text{ bytes}}{1 \text{ byte/θ.μ.}} = 2^{30} \text{ θ.μ.}$	$1 \text{ θ.μ.}$ $x;$	$1 \text{ byte}$ $8 \text{ GB}$	$x = \frac{2^3 \times 2^{30} \text{ bytes}}{1 \text{ byte/θ.μ.}} = 2^{33} \text{ θ.μ.}$
--------------------------	------------------------------------	--	--------------------------	------------------------------------	---

$$\text{Επομένως } \frac{\text{θ.μ.που θέλουμε να έχουμε στο τελικό σύστημα KM}}{\text{θ.μ.που ήδη έχουμε στα OKM που διαθέτουμε}} = \frac{2^{33}}{2^{30}} = 8 \text{ επαναλήψεις}$$

Στο σχεδιασμό θα πρέπει να συνδέσουμε τα OKM 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 OKM σε μια γραμμή, έτσι ώστε να επιλέγονται ταυτόχρονα με την ίδια έξοδο του αποκωδικοποιητή

**Σημείωση:** πρέπει να δοθεί ιδιαίτερη προσοχή στο address bus που χρησιμοποιείται για τη διεύθυνση διοδότηση των OKM. Πιο συγκεκριμένα στο εξωτερικό κάθε OKM γράφουμε A<sub>29</sub> – A<sub>30</sub>, ενώ εξωτερικά εισέρχονται τα A<sub>32</sub> – A<sub>2</sub>.

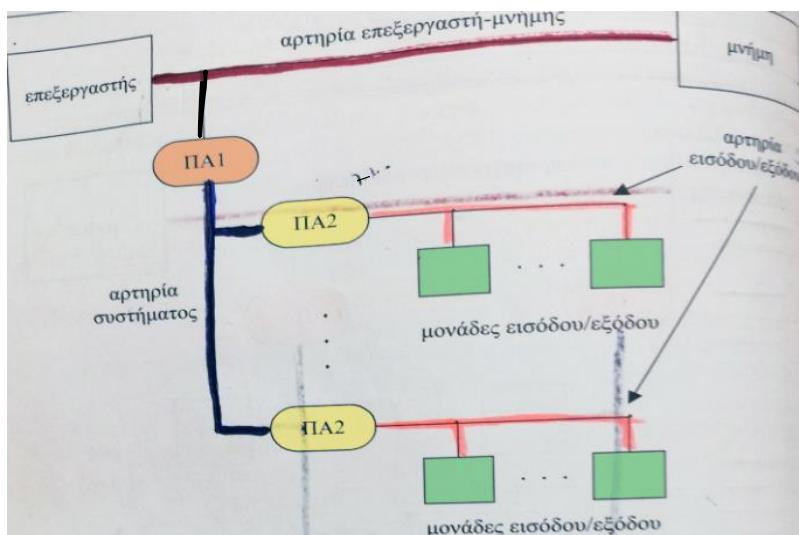


## Κεφάλαιο 6 – Αρτηρίες

### 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 σύνδεση 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).

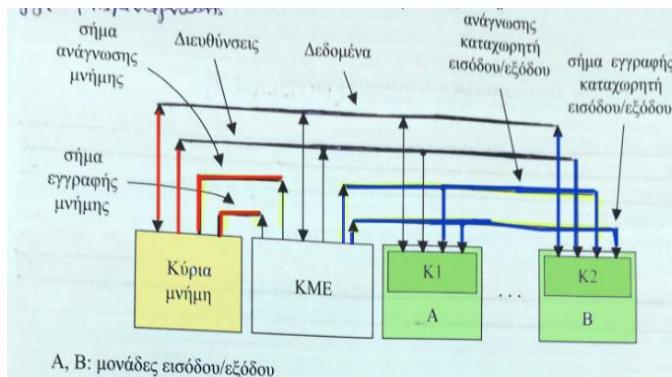


Μια κλασσική αρτηρία συστήματος είναι η αρτηρία PCI. Αρτηρίες εισόδου – εξόδου είναι οι ISA, EISA, SCSI. Σε μια **αρτηρία σύγχρονη** ένα από τα σήματα που μεταδίδονται είναι ένα σήμα χρονισμού (ρολόι) και όλες οι λειτουργίες γίνονται βάσει αυτού. Είναι γρήγορες, αλλά το πλήθος των μονάδων που μπορούν 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

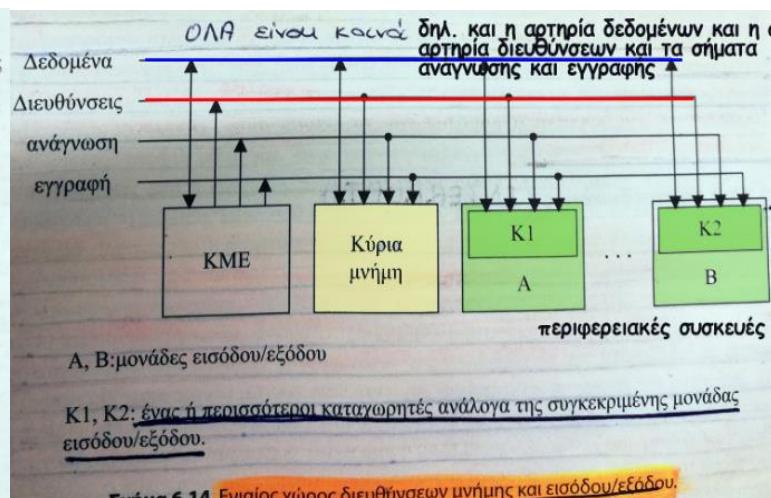
ικανοποιείται, εφόσον υπάρχουν απαιτήσεις από μονάδες υψηλότερης προτεραιότητας, άσχετα με το χρόνο αναμονής. Τα κριτήρια επιλογής του κατάλληλου κάθε φορά σχήματος διαιτησίας είναι ο αριθμός των μονάδων που συνδέονται σε αυτή, το μήκος της αρτηρία, ο απαιτούμενος βαθμός δικαιοσύνης κ.λ.π.

**Η απόδοση μιας αρτηρίας εξαρτάται** από την καθυστέρηση (latency) που παρουσιάζει και το ρυθμό μεταφοράς της. Ο τελευταίος, προσδιορίζεται από i) την ύπαρξη διακριτών γραμμών διευθύνσεων και δεδομένων, ii) από τη δυνατότητα ομαδικής μεταφοράς δεδομένων και iii) από το εύρος της αρτηρίας δεδομένων.

Αναφορικά με τους **τρόπους διακίνησης πληροφορίας ελέγχου μεταξύ KME και μονάδων E/E (Εισόδου/Εξόδου)**, είναι η ύπαρξη **διακριτών** χώρων διευθύνσεων μνήμης και E/E και η ύπαρξη **ενιαίου** χώρου διευθύνσεων μνήμης και E/E.



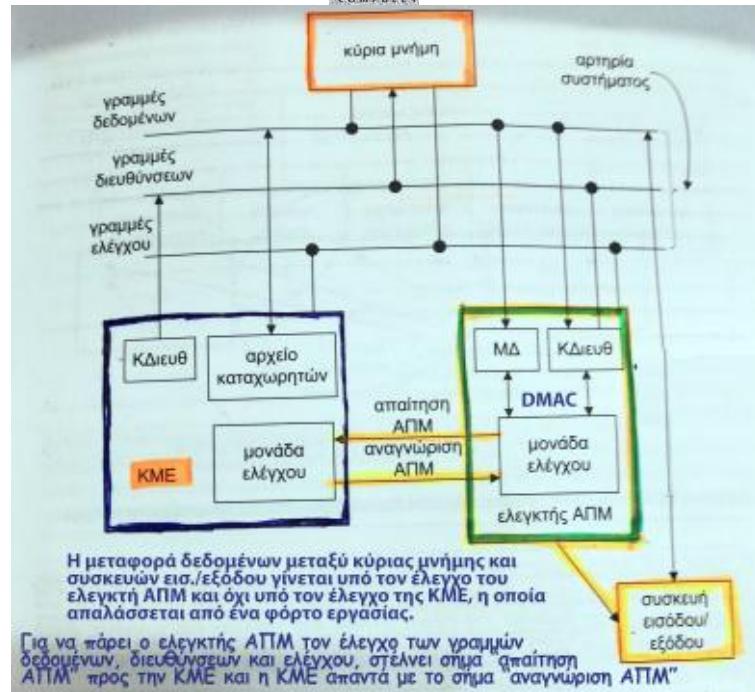
**Σχήμα 6.13** Διάκριτοι χώροι διευθύνσεων μνήμης και εισόδου/εξόδου.



**Σχήμα 6.14** Ενιαίος χώρος διευθύνσεων μνήμης και εισόδου/εξόδου.

Αναφορικά με τους **τρόπους διακίνησης πληροφορίας από κάποια συσκευή E/E προς την KME**, υπάρχουν δύο προσεγγίσεις: **ο χρονοπρογραμματισμένος έλεγχος (polling)** και **η χρήση σημάτων διακοπής (interrupts)**. Αναφορικά με την πρώτη προσέγγιση, η KME διαβάζει περιοδικά (σε τακτά χρονικά διαστήματα) τον καταχωρητή κατάστασης του ελεγκτή μιας μονάδας E/E για να διαπιστώσει την κατάστασή της. Αυτό συμβαίνει στην περίπτωση που η μονάδα E/E είναι το πληκτρολόγιο. Το polling έχει απλή υλοποίηση, αλλά μεγάλη καθυστέρηση, ιδιαίτερα όταν το πλήθος των μονάδων E/E 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 KME για να διακόψει τη λειτουργία της. Τότε η KME 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 KME στη διαδικασία εισόδου – εξόδου**, υπάρχουν τρεις περιπτώσεις: a) Η προγραμματισμένη E/E (η απόδοση του συστήματος υποβαθμίζεται σημαντικά), b) η προγραμματισμένη E/E με χρήση σημάτων διακοπής (σε αυτήν την περίπτωση αποδεσμεύεται η KME από το συνεχή έλεγχο της κατάστασης της συσκευής E/E), c) με χρήση άμεσης προσπέλασης μνήμης (APM), όπου ο ελεγκτής της συσκευής E/E καλείται σε αυτήν την περίπτωση ελεγκτής άμεσης προσπέλασης μνήμης (DMAC – DMA Controller). Ο ελεγκτής APM ζητά στην περίπτωση αυτή από την KME 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 KME, η οποία μπορεί να ασχοληθεί με δικές της διεργασίες. Όταν ολοκληρωθούν οι μεταφορές, ο έλεγχος των διαύλων επιστρέφει πίσω στην KME.



### 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 **διατησία βάζει καθυστέρηση 5 nsec**. Υπολογίστε το **μέσο ρυθμό μεταφοράς πληροφορίας σε ψηφιολέξεις ανά δευτερόλεπτο** για κάθε μια από τις ακόλουθες περιπτώσεις.

a. Η αρτηρία είναι **σύγχρονη** με περίοδο του σήματος χρονισμού (ρολογιού) 5 nsec

- χωρίς δυνατότητα εισαγωγής κύκλων ρολογιού καθυστέρησης (wait states)
- με δυνατότητα εισαγωγής κύκλων ρολογιού καθυστέρησης

b. Η αρτηρία είναι ασύγχρονη και απαιτεί 10 nsec επιπλέον για κάθε μεταφορά λόγω του πρωτοκόλλου χειραψίας.

### Λύση

$$\text{Ο μέσος ρυθμός μεταφοράς πληροφορίας (εντολών/δεδομένων)} = \frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}}$$

a. i) Όταν η αρτηρία είναι **σύγχρονη** και δεν χρησιμοποιούνται κύκλοι ρολογιού καθυστέρησης (wait states ή wait cycles)<sup>1</sup>, 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)** =  $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{4 \text{ bytes}}{65 \text{ nsec}} = \frac{4 \times 10^9 \text{ bytes}}{65 \text{ sec}} = 0.0615 \times 10^9 \text{ bytes/sec} = 0.0615 \times 10^3$

$$\times 10^6 \text{ bytes/sec} = 61.5 \text{ MB/sec} \text{ ή σε KB/sec θα έχουμε: } \frac{4 \text{ bytes}}{65 \text{ nsec}} = \frac{4 \times 10^9 \text{ bytes}}{65 \text{ sec}} = 0.0615 \times 10^6 \times 10^3 \text{ bytes/sec} = 61500 \text{ KB/sec.}$$

**Παρατηρηση:** Όταν αναφερόμαστε σε **ρυθμό μεταφοράς πληροφορίας** χρησιμοποιούμε δυνάμεις του «10» και όχι δυνάμεις του «2» ως συνήθως.

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 που προστίθεται σε κάθε περίπτωση, δηλ. χρόνος μεταφοράς =  $0.20 \times 15 \text{ nsec} + 0.30 \times 25 \text{ nsec} + 0.40 \times 35 \text{ nsec} + 0.10 \times 65 \text{ nsec} = 31 \text{ nsec}$ . Επομένως ο **μέσος ρυθμός μεταφοράς πληροφορίας (εντολών/δεδομένων)** =  $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{4 \text{ bytes}}{31 \text{ nsec}} = \frac{4 \text{ bytes}}{31 \times 10^{-9} \text{ sec}} = \frac{4 \text{ bytes} \times 10^9}{31 \text{ sec}} = 0.129 \times 10^9 \text{ bytes/sec} = 0.129 \times 10^3 \times 10^6 = 129$

MB/sec. Παρατηρούμε ότι μέσος ρυθμός μεταφοράς πληροφορίας υπερδιπλασιάζεται σε σχέση με πριν.

<sup>1</sup> Τα wait states χρησιμοποιούνται όταν μια γρήγορη KME επικοινωνεί με αργές περιφερειακές συσκευές προκειμένου να επιβραδυθεί η λειτουργία της

**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 διθέντες 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 KME χρησιμοποιείται πρωτόκολλο χειραψίας με χρήση σημάτων χειραψίας (handshaking) που είναι τα σήματα της απαίτησης (request) και της αναγνώρισης (acknowledge). Στην περίπτωση αυτή θα πρέπει, **εκτός από την καθυστέρηση των συσκευών και της διαιτησίας, να λάβουμε υπόψη και την καθυστέρηση λόγω του χρησιμοποιούμενου πρωτοκόλλου χειραψίας**. Ο νέος χρόνος μεταφοράς =  $0.20 \times (10 + 10 + 5)\text{nsec} + 0.30 \times (20 + 10 + 5)\text{nsec} + 0.40 \times (30 + 10 + 5) \text{nsec} + 0.10 \times (60 + 10 + 5) \text{nsec} = 41 \text{nsec}$ . Επομένως ο **μέσος ρυθμός μεταφοράς πληροφορίας** (εντολών/δεδομένων) =  $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{4 \text{ bytes}}{41 \text{ nsec}} = 0.09756 \times 10^9 \text{ bytes/sec} = 97.56 \text{ MB/sec}$ .

## Ασκηση 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 σε ψηφιολέξεις ανά δευτερόλεπτο για κάθε μια από τις ακόλουθες περιπτώσεις.

α. Η αρτηρία είναι **σύγχρονη** με περίοδο του σήματος χρονισμού (ρολογιού) **5 nsec**

- χωρίς δυνατότητα εισαγωγής κύκλων ρολογιού καθυστέρησης (wait states)
- 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 διαιτησίας**, δηλ. χρόνος μεταφοράς =  $65 \text{nsec} + 5 \text{nsec} = 70 \text{nsec}$ . **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) =  $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{8 \text{ bytes}}{70 \text{ nsec}} = \frac{8 \text{ bytes}}{70 \times 10^{-9} \text{ sec}} = \frac{8 \text{ bytes} \times 10^9}{70 \text{ sec}} = 0.114 \times 10^9 \text{ bytes/sec}$ .

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 που προστίθεται σε κάθε περίπτωση**, δηλ. χρόνος μεταφοράς =  $0.20 \times 20 \text{nsec} + 0.30 \times 25 \text{nsec} + 0.40 \times 35 \text{nsec} + 0.10 \times 70 \text{nsec} = 32.5 \text{nsec}$ . 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  $12 \text{nsec} \rightarrow 15 \text{nsec}$ , τα  $19 \text{nsec} \rightarrow 20 \text{nsec}$  κ.ο.κ. Ο **μέσος ρυθμός μεταφοράς πληροφορίας** (εντολών/δεδομένων) =  $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{8 \text{ bytes}}{32.5 \text{ nsec}} = \frac{8 \text{ bytes}}{32.5 \times 10^{-9} \text{ sec}} = \frac{8 \text{ bytes} \times 10^9}{32.5 \text{ sec}} = 0.246 \times 10^9 \text{ bytes/sec}$ . 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 διθέντες 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 Ε/Ε με την ΚΜΕ χρησιμοποιείται το **πρωτόκολλο χειραψίας**, με τα σήματα χειραψίας, που είναι τα σήματα της απαίτησης (request) και της αναγνώρισης (acknowledge). Στην περίπτωση αυτή 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 =  $0.20 \times (12 + 10 + 5) \text{ nsec} + 0.30 \times (19 + 10 + 5) \text{ nsec} + 0.40 \times (28 + 10 + 5) \text{ nsec} + 0.10 \times (62 + 10 + 5) \text{ nsec} = 40.5 \text{ nsec}$ . Επομένως ο **μέσος ρυθμός μεταφοράς πληροφορίας** (εντολών/δεδομένων) =  $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{8 \text{ bytes}}{40.5 \text{ nsec}} = \frac{8 \text{ bytes} \times 10^9}{40.5 \text{ sec}} = 0.1975 \times 10^9 \text{ bytes/sec}$ .

## 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, 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 =  $0.08 \times 45 \text{ nsec} + 0.30 \times 30 \text{ nsec} + 0.17 \times 35 \text{ nsec} + 0.44 \times 60 \text{ nsec} = 45 \text{ nsec}$ . 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** =  $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{6 \text{ bytes}}{45 \text{ nsec}} = 0.133 \times 10^9 \text{ bytes/sec} = 0.133 \text{ bytes} \times 10^3 \times 10^6 \text{ bytes/sec} = 133.33 \times 10^6 \text{ bytes/sec} = 133.33 \text{ MB/sec} \rightarrow 133 \text{ MB/sec}$ . Ο ρυθμός μεταφοράς πληροφορίας χρησιμοποιεί δυνάμεις του «10» και όχι του «2» ως συνήθως, γιατό και το  $1 \text{ MB} = 10^6 \text{ bytes}$  και όχι  $2^{20}$ .

## 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 οπότε για τον υπολογισμό του **χρόνου μεταφοράς λαμβάνουμε υπόψη την πιο αργή συσκευή και την καθυστέρηση λόγω διαιτησίας**, δηλ. χρόνος μεταφοράς = 55

nsec + 5 nsec = 60 nsec. Επομένως ο μέσος ρυθμός μεταφοράς πληροφορίας (εντολών/δεδομένων) = ποσότητα μεταφερόμενης πληροφορίας =  $\frac{6 \text{ bytes}}{60 \text{ nsec}} = 0.1 \times 10^9 \text{ bytes/sec} = 0.1 \times 10^3 \times 10^6 \text{ bytes/sec} = 100 \text{ MB/sec}$ .

## 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 A, B, Γ και Δ συνδεδεμένες στην αρτηρία. Εάν στην αρτηρία ήταν συνδεδεμένη μόνο μια μονάδα εισόδου, η μονάδα A ή B 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 A, στο 36% των περιπτώσεων από την μονάδα B, στο 27% των περιπτώσεων από την μονάδα Γ και στο 17% των περιπτώσεων από την μονάδα Δ. **Θεωρήστε επίσης ότι η διατησία 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 10 nsec. 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) =  $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{7 \text{ bytes}}{50 \text{ nsec}} = 0.14 \times 10^9 \text{ bytes/sec} = 0.14 \times 10^3 \times 10^6 \text{ bytes/sec} = 140 \text{ MB/sec}$

## Άσκηση 6.3 βιβλίου με polling/interrupts

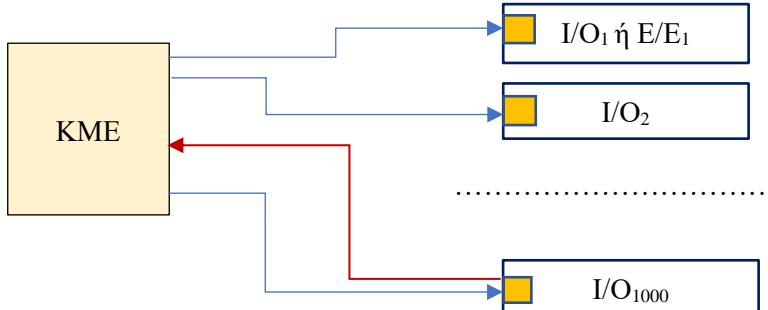
Θεωρήστε ένα υπολογιστή με επεξεργαστή συχνότητας λειτουργίας 1 GHz που, εκτός των άλλων, χρησιμοποιείται για να ελέγχει τη λειτουργία 1.000 μονάδων ενός εργοστασίου. Κατά μέσο όρο οι μονάδες εισ/εξ απαιτούν συνολικά εξυπηρέτηση 500 φορές το λεπτό. Να υπολογίσετε το ποσοστό του χρόνου του υπολογιστή 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 KME τη δυνατότητα εξυπηρέτησης σημάτων διακοπής. Το κόστος διακοπής της λειτουργίας της KME 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 =  $\frac{1}{1 \text{ GHz}} = \frac{1}{10^9 \text{ Hz}} = 10^{-9} \text{ sec} = 1 \text{ nsec}$ .

α. Στην περίπτωση **της χρονοπρογραμματισμένης διαδικασίας (polling)**, θα πρέπει να λάβουμε υπόψη δύο χρόνους: έναν που απαιτείται για τον έλεγχο κάθε συσκευής (μονάδας) E/E και έναν που απαιτείται για την **εξυπηρέτηση** μόνο εκείνης της μονάδας που χρειάζεται εξυπηρέτηση. Στην Εικόνα, η KME ελέγχει περιοδικά τον καταχωρητή κατάστασης κάθε μονάδας E/E, 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 E/E, υπολογίζεται από τον τύπο:  $10 \text{ φορές} \times 500 \text{ κ.ρ.} \times 1 \text{ nsec} \times 1000 = 10 \times 500 \text{ κ.ρ.} \times 10^{-9} \text{ sec} \times 10^3 = 10 \times 5 \times 10^2 \times 10^{-9} \text{ sec} \times 10^3 = 5 \times 10^{-3} \text{ sec} = 5 \text{ msec}$ .

Ο **δεύτερος χρόνος** που απαιτείται για την **εξυπηρέτηση μόνο εκείνης** της μονάδας E/E που χρειάζεται εξυπηρέτηση, υπολογίζεται από τον τύπο:  $500 \frac{\text{φορές}}{\text{min}} \times 500 \text{ κ.ρ.} \times 1 \text{ nsec} \times 1 = \frac{250000}{60 \text{ sec}} \times 10^{-9} \text{ sec} = \frac{25}{6} \times 10^{-6} \text{ sec} = 4.166 \text{ msec}$ . Επομένως, ο **συνολικός χρόνος που απαιτείται για το polling είναι το άθροισμα των δύο προηγούμενων χρόνων**, δηλ.  $5 \text{ msec} + 4.166 \text{ msec} = 5 \times 10^{-3} \text{ sec} + 4.166 \times 10^{-6} \text{ sec} = 0.005 \text{ sec} + 0.000004166 \text{ sec} = 0.005004166 \text{ sec} \rightarrow$  μετατροπή σε ποστό **0.5004166%**. Επομένως, η KME δαπανά το **0.5%** του χρόνου της για τον έλεγχο και την εξυπηρέτηση μονάδων E/E.

β. Στην περίπτωση χρήσης σημάτων διακοπής (Interrupts) έχουμε **μόνο χρόνο εξυπηρέτησης**  $500 \frac{\text{φορές}}{\text{min}} \times 1300 \text{ κ.ρ.} \times 1 \text{ nsec} \times 1 = \frac{500}{60 \text{ sec}} \times 1300 \text{ κ.ρ.} \times 10^{-9} \text{ sec} = \frac{65}{6} \times 10^{-6} \text{ sec} = 0.000018083 \text{ sec} \rightarrow$  μετατροπή σε ποστό **0.0018083%**. Επομένως σε αυτή την περίπτωση η KME δαπανά το **0.001%** του χρόνου της για την εξυπηρέτηση των μονάδων E/E.

γ. Παρατηρούμε ότι στη δεύτερη περίπτωση το ποσοστό απασχόλησης της KME είναι μικρότερο, γεγονός αναμενόμενο αφού στην περίπτωση επικοινωνίας της KME με συσκευές E/E το σημαντικό είναι ότι δεν χρειάζεται έλεγχος εκ' μέρους της KME για το ποια συσκευή θέλει εξυπηρέτηση.

### 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 KME τη δυνατότητα εξυπηρέτησης σημάτων διακοπής. Το κόστος διακοπής της λειτουργίας της KME 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  $1 \times 10^{-9} \text{ sec} = 1 \text{ nsec}$ .

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 KME κάθε δευτερόλεπτο απασχολείται με τον έλεγχο των μονάδων εισ/εξ για χρόνο ίσο με:  $10 \times 500 \text{ κ.ρ.} \times 10^{-9} \text{ sec} \times 2.000 = 10^{-2} \text{ sec}$ .

Οι μονάδες εισ/εξ απαιτούν συνολικά εξυπηρέτηση 1000 φορές το λεπτό, δηλαδή  $1000/60 = 1000/60 \text{ φορές το δευτερόλεπτο}$ . Επομένως οι μονάδες εισ/εξ απαιτούν για την **εξυπηρέτηση** τους επιπλέον  $\frac{1000}{60} \times 600 \text{ κ.ρ.} \times 10^{-9} \text{ sec} \times 1 = 10^{-5} \text{ sec}$ . Επομένως, η KME κάθε δευτερόλεπτο απασχολείται με τον έλεγχο και την εξυπηρέτηση των μονάδων εισ/εξ:  $10^{-2} \text{ sec} + 10^{-5} \text{ sec} = 0.01001$

→ μετατροπή σε ποσοστό: **1.001%**. Σε αυτό το ποσοστό απασχολείται η KME για τον έλεγχο και την εξυπηρέτηση των μονάδων εισ/εξ., όταν χρησιμοποιείται η χρονοπρογραμματισμένη διαδικασία (polling).

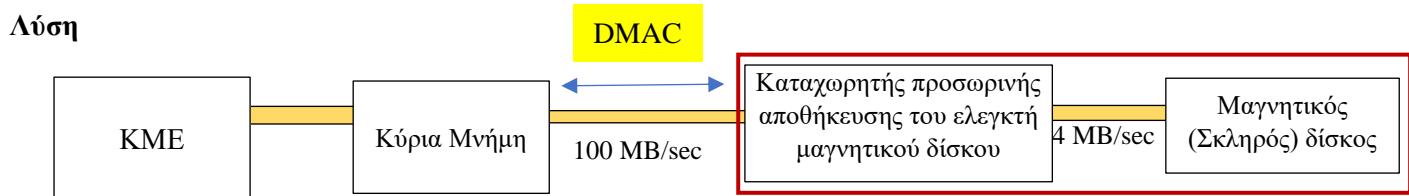
β. Αφού η εξυπηρέτηση κάθε μονάδας γίνεται 1.000 φορές το λεπτό και κάθε σήμα διακοπής απαιτεί 1.200 κύκλους ρολογιού, η KME αφιερώνει για την **εξυπηρέτηση** των μονάδων εισ/εξ:  $\frac{1000}{60} \times 1200 \text{ κ.ρ.} \times 10^{-9} \text{ sec} \times 1 = 0.002 \times 10^{-2} \text{ sec}$ .

Επομένως σε αυτή την περίπτωση η KME δαπανά το **0.002%** του χρόνου της για την εξυπηρέτηση των μονάδων εισ/εξ.

#### 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 4 Μψηφιολέξεις/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 100 Μψηφιολέξεις/δευτερόλεπτο και η συχνότητα λειτουργίας της KME είναι 1 GHz. Θεωρήστε ότι το πρόγραμμα εκτελείται μόνο του στον εν λόγω H/Y 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.



Από τη συχνότητα λειτουργίας του επεξεργαστή θα υπολογίσουμε τη διάρκεια του κύκλου ρολογιού  $t_{cycle} = \frac{1}{\text{συχνότητα λειτουργίας}} = \frac{1}{1 \text{ GHz}} = 1 \text{ nsec}$ . 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  $\times \frac{6 \text{ κ.ρ.}}{\text{εντολή}} \times 1 \frac{\text{nsec.}}{\text{κ.ρ.}} = 30 \times 10^6 \times 10^{-9} \text{ sec} = 30 \times 10^{-3} \text{ sec} = 30 \text{ msec}$ . 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  $t_1 = 2000 \text{ κ.ρ.} \times 1 \frac{\text{nsec.}}{\text{κ.ρ.}} = 2 \times 10^{-6} \text{ sec} = 2 \text{ μsec}$ . 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:  $\text{χρόνος μεταφοράς} = \frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{ρυθμός μεταφοράς δεδομένων}} = \frac{4 \text{ KB}}{4 \text{ MB/sec}}$

$$\text{χρόνος μεταφοράς} = \frac{2^{10} \text{ bytes}}{10^6 \text{ bytes/sec}} = \frac{2^{10} \text{ sec}}{10^6} = 1024 \times 10^{-6} \text{ sec} = 1024 \text{ μsec}$$

Αναφορικά με τον υπολογισμό του χρόνου  $t_3$  έχουμε ότι: **χρόνος μεταφοράς μεταξύ καταχωρητή προσωρινής αποθήκευσης και κύριας μνήμης** =  $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{ρυθμός μεταφοράς δεδομένων}} =$

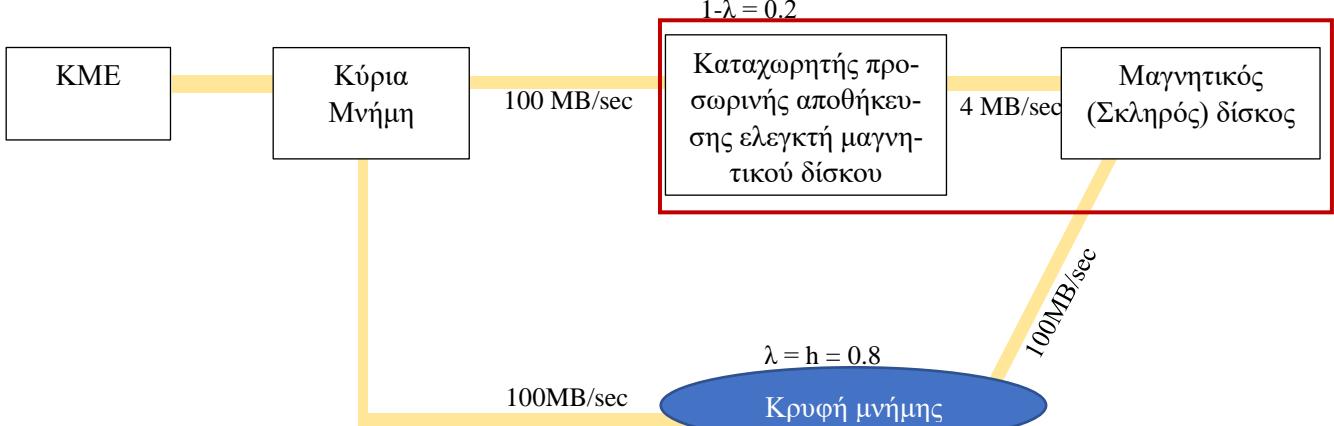
$\frac{4 \text{ KB}}{100 \text{ MB/sec}} = \frac{4 \times 2^{10} \text{ bytes}}{100 \times 10^6 \text{ bytes/sec}} = \frac{4 \times 2^{10} \text{ sec}}{10^8} = 0.00004096 \text{ sec} = 40.96 \text{ usec}$ . Ο χρόνος για τις **100 εντολές εισόδου/εξόδου** είναι:  $(2 \text{ usec} + 1024 \text{ usec} + 40.96 \text{ usec}) \times 100 = 1066.96 \times 10^{-6} \times 10^2 \text{ sec} = 1066.96 \times 10^{-4} \text{ sec} = 106.696 \text{ msec}$ . 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.

#### Άνση



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 =  $\frac{1}{\text{συχνότητα λειτουργίας}} = \frac{1}{1 \text{ GHz}} = 1 \text{ nsec}$ . Επίσης ισχύει ότι για τις 5.000.000 εντολές που δεν είναι εισόδου/εξόδου δεν αλλάζει ο χρόνος, οπότε έχουμε:  $5000000 \times \frac{6 \text{ κ.ρ.}}{\text{εντολή}} \times 1 \frac{\text{nsec.}}{\text{κ.ρ.}} = 30 \times 10^6 \times 10^{-9} \text{ sec} = 30 \times 10^{-3} \text{ sec} = 30 \text{ msec}$ . Για τις υπόλοιπες **100 εντολές που είναι εισόδου/εξόδου**, θα χρησιμοποιηθεί η τεχνική ΑΠΜ και ο χρόνος που απαιτείται για την εκτέλεσή τους είναι:  $T = 100 * [0.8 * \frac{4 \text{ KB}}{100 \text{ MB/sec}} + 0.2 * (\frac{4 \text{ KB}}{4 \text{ MB/sec}} + \frac{4 \text{ KB}}{100 \text{ MB/sec}})] = 100 * [0.8 * \frac{4 \times 2^{10} \text{ bytes}}{100 \times 10^6 \text{ bytes/sec}} + 0.2 * (\frac{4 \times 2^{10} \text{ bytes}}{4 \times 10^6 \text{ bytes/sec}} + \frac{4 \times 2^{10} \text{ bytes}}{100 \times 10^6 \text{ bytes/sec}})] = 100 * [0.8 * 0.00004096 \text{ sec} + 0.2 * (0.001024 \text{ sec} + 0.00004096 \text{ sec})] = 100 * [0.000032768 + 0.00021299] \text{ sec} = 0.024576 \text{ sec} = 24.576 \text{ msec}$ . 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».

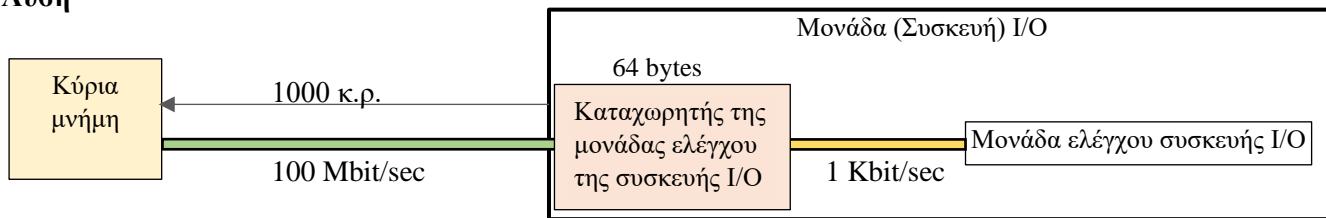
### Άσκηση με προγραμματισμένη είσοδο – έξοδο με χρήση διακοπών

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



Ο ρυθμός μεταφοράς δεδομένων =  $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}}$  ⇒ 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 E/E και του αντίστοιχου καταχωρητή της και την άλλη φορά μεταξύ της μονάδας ελέγχου συσκευής E/E και κύριας μνήμης.

i) **Χρόνος μεταφοράς μεταξύ μονάδας ελέγχου συσκευής E/E και του αντίστοιχου καταχωρητή της** =  $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{ρυθμός μεταφοράς δεδομένων}} = \frac{64 \text{ bytes}}{1 \text{ Kbit/sec}} = \frac{64 \times 8 \text{ bits}}{10^3 \text{ bits/sec}} = \frac{512 \text{ sec}}{1000} = 0.512 \text{ sec}$ . 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 E/E και κύριας μνήμης** =  $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{ρυθμός μεταφοράς δεδομένων}} = \frac{64 \text{ bytes}}{100 \text{ Mbit/sec}} = \frac{64 \times 8 \text{ bits}}{100 \times 10^6 \text{ bits/sec}} = \frac{512 \text{ sec}}{10^8} = 5.12 \times 10^{-6} \text{ sec} = 5.12 \text{ μsec}$ . 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  $1000 \text{ κ.ρ.} \times 10 \text{ nsec/κ.ρ.} = 10 \times 10^{-6} \text{ sec} = 10 \text{ μsec}$ . Άρα συνολικά έχουμε  $5.12 \times 10^{-6} \text{ sec} + 10 \times 10^{-6} \text{ sec} = 15.12 \times 10^{-6} \text{ sec}$ .

Για να υπολογίσουμε το **ποσοστό απασχόλησης της KME** (επεξεργαστή) για τη μεταφορά πληροφορίας από τον ελεγκτή της συσκευής E/E στην κύρια μνήμη, θα χρησιμοποιήσουμε μια απλή μέθοδο των τριών, ως εξής:

Κάθε  $0.512 \text{ sec}$  που γεμίζει ο καταχωρητής  $\text{έχουμε απασχόληση της KME κατά } 15.12 \times 10^{-6} \text{ sec}$   
100 x;

$x = 15.12 \times 10^{-6} \text{ sec} \times 100 / 0.512 = 2.953 \times 10^{-3} = 0.002953\%$ . Αυτό το μέγεθος είναι ήδη εκφρασμένο σε ποσοστό.

### Άσκηση με διαιτησία αρτηρίας

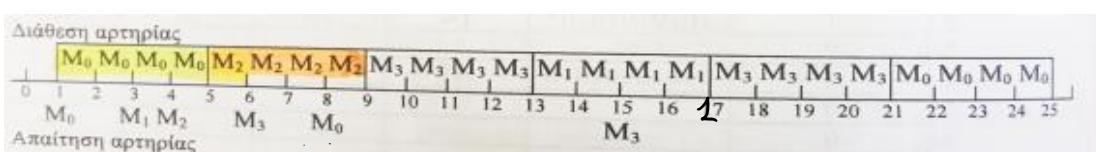
Θεωρείστε ότι σε σύγχρονη αρτηρία με συγχρότητα 1 GHz (περίοδο ρολογιού 1ns) είναι συνδεδεμένες 4 κύριες μονάδες (masters)  $M_0, M_1, M_2$  και  $M_3$ . Θεωρήστε επίσης ότι χρησιμοποιείται η διαιτησία με χρήση αλυσίδας προτεραιότητας και ότι η μονάδα με μεγαλύτερο δείκτη έχει μεγαλύτερη προτεραιότητα 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 αιτιολογήσετε 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 κυριοτητά 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 A, B, Γ και Δ. Η μεταφορά πληροφορίας στον κύριο από τη μονάδα εισόδου A, B, 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 B 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 χειραψίας.

### Λύση

Το χαρακτηριστικό σε αυτήν την άσκηση είναι ότι λαμβάνουμε υπόψη **μόνο τη μονάδα B**, γιατί αυτό αναφέρεται σχετικά στην εκφρώνηση.

α) i) Ρυθμός μεταφοράς πληροφορίας =  $8 \text{ bytes}/(60 + 5) \text{ nsec} = 8 \text{ bytes}/65 \text{ nsec} = 8 \text{ bytes}/65 \times 10^{-9} \text{ sec}$ , λαμβάνουμε υπόψη μόνο την πιο αργή συσκευή.

ii) Ρυθμός μεταφοράς πληροφορίας =  $8 \text{ bytes}/(20 + 5) \text{ nsec} = 8 \text{ bytes}/25 \text{ nsec} = 0.32 \times 10^9 \text{ bytes/sec}$ , λαμβάνουμε υπόψη μόνο τη B.

β) Ρυθμός μεταφοράς πληροφορίας =  $8 \text{ bytes}/(18 + 8 + 5) \text{ nsec} = 8 \text{ bytes}/31 \text{ nsec} = 0.258 \times 10^9 \text{ bytes/sec}$ , λαμβάνουμε

### Θέμα με σύγχρονη και ασύγχρονη αρτηρία

Υπολογίστε τον ρυθμό μεταφοράς δεδομένων μεταξύ κεντρικής μονάδας επεξεργασίας (KME) 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.

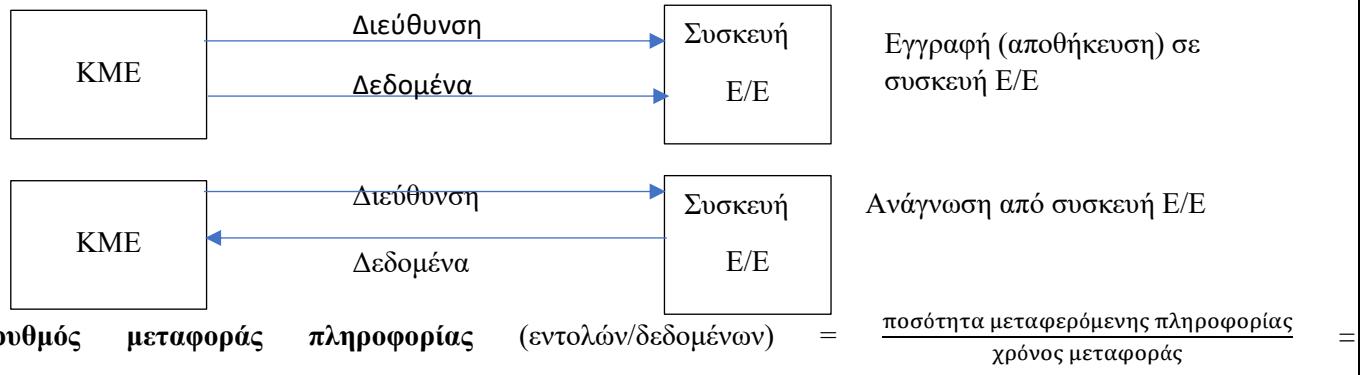
α. Η αρτηρία διαθέτει διαφορετικές γραμμές δεδομένων και διεύθυνσεων.

β. Η αρτηρία διαθέτει πολυπλεγμένες γραμμές δεδομένων και διεύθυνσεων.

Θεωρήστε ότι το 50% των μεταφορών αφορούν ανάγνωση από τη μνήμη και το 50% εγγραφή σ' αυτή και 1 λέξη = 4 ψηφιολέξεις (bytes).

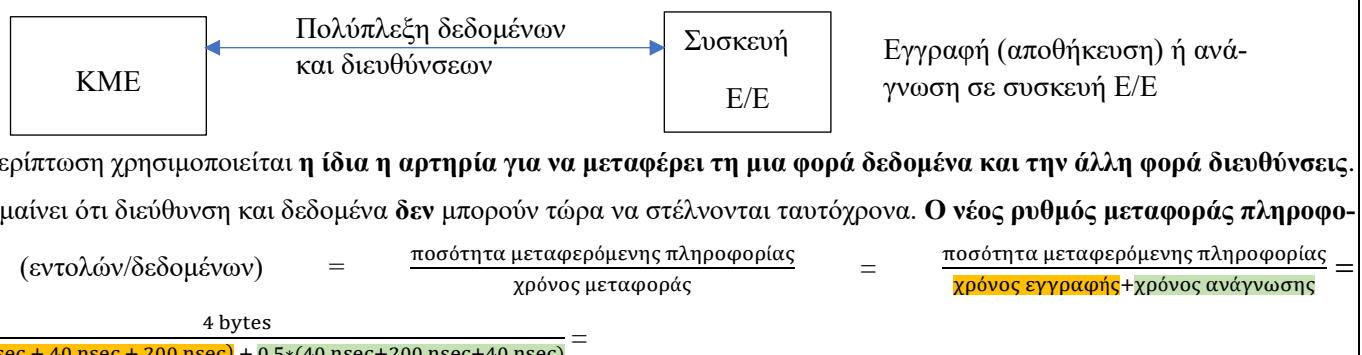
### Λύση

#### 1α. Αρτηρία σύγχρονη με ξεχωριστές γραμμές διεύθυνσεων και δεδομένων



Αναφορικά με το χρόνο εγγραφής, μεταφέρονται στον ίδιο χρόνο (40 nsec) τόσο η διεύθυνση της συσκευής E/E 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 E/E 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 KME (40 nsec).

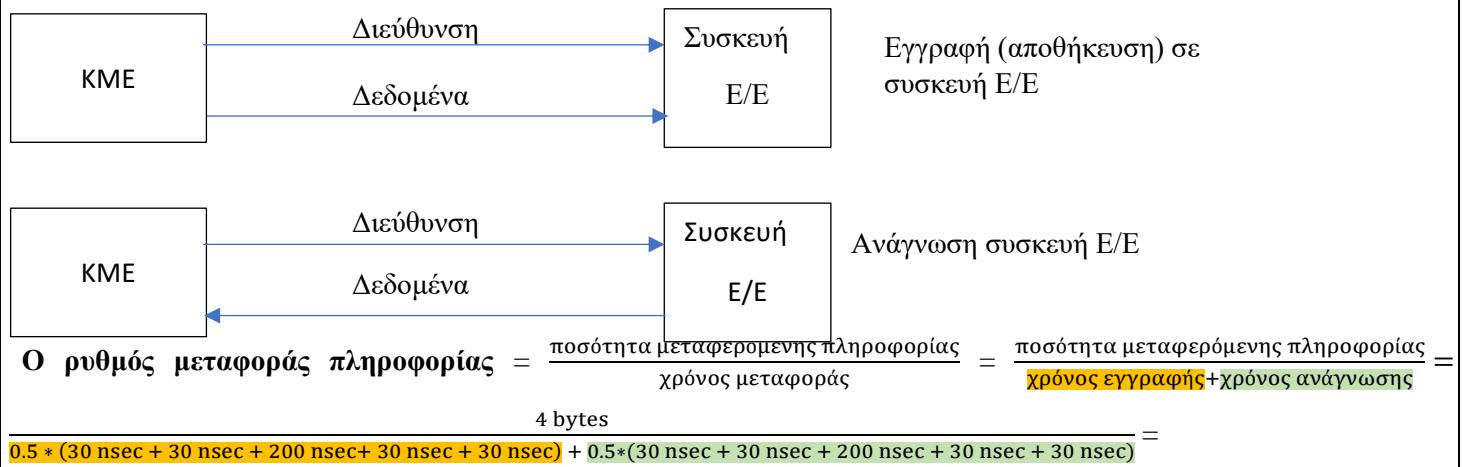
#### 1β. Πολυπλεγμένες γραμμές διεύθυνσεων και δεδομένων



Αναφορικά με το χρόνο εγγραφής, απαιτείται ένας χρόνος της τάξης των 40 nsec για να σταλθεί διεύθυνση της συσκευής E/E 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 άλλαζει κάτι σε σχέση με πριν:** δηλ. χρειάζονται τρεις χρόνοι: ένας για την αποστολή της διεύθυνσης της συσκευής E/E 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 KME (40 nsec).

#### 2α. Αρτηρία ασύγχρονη με ξεχωριστές γραμμές διεύθυνσεων 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.



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.

## 2a. 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** =  $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος εγγραφής} + \text{χρόνος ανάγνωσης}}$  =  
 $4 \text{ bytes}$   
 $0.5 * (30 \text{ nsec} + 30 \text{ nsec} + 30 \text{ nsec} + 30 \text{ nsec} + 200 \text{ nsec} + 30 \text{ nsec} + 30 \text{ nsec}) + 0.5 * (30 \text{ nsec} + 200 \text{ nsec} + 30 \text{ nsec} + 30 \text{ nsec} + 30 \text{ nsec}) =$

Οι πρώτες τέσσερις τιμές των 30 nsec αφορούν ενεργοποίηση και απενεργοποίηση σημάτων χειραψίας για αποστολή της διεύθυνσης της συσκευής E/E 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/εξόδου (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.

A. 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

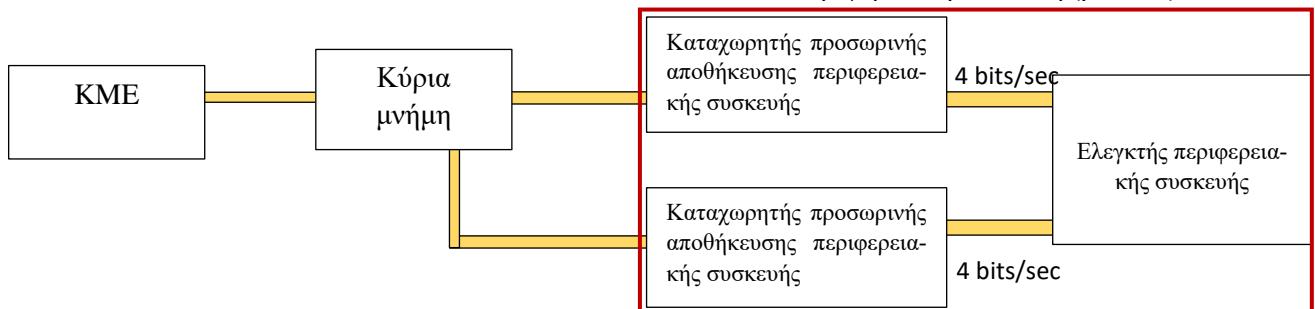
ακέραια ψηφία, υποδιαστολή και δύο ψηφία δεξιά της υποδιαστολής, π.χ. 04.30. Για να περιορίσετε σε δύο το πλήθος των ψηφίων δεξιά της υποδιαστολής να εφαρμόσετε την τεχνική της περικοπής.

B. 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.

### Αύση

Περιφερειακή συσκευή (μονάδα)



α) **Στο χρονοπρογραμματισμένο έλεγχο υπολογίζονται δύο χρόνοι: ένας χρόνος ελέγχου και ένας χρόνος εξυπηρέτησης.** Κάθε έλεγχος της κατάστασης του ελεγκτή της περιφερειακής συσκευής από την κεντρική μονάδα επεξεργασίας (KME) 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, άρα αθροιστικά έχουμε  $0.005 \text{ sec} + 0.00009 \text{ sec} = 0.00509 \text{ sec}$ . Στο 30% των ελέγχων απαιτείται εξυπηρέτηση (μεταφορά δεδομένων) και κάθε ελεγκτής ελέγχεται 5 φορές ανά sec και σε μια μεταφορά έχουμε  $0.006 \text{ sec} + 0.00509 \text{ sec}$ . Σε 1 sec για 7 περιφερειακές μονάδες (E/E) θα έχουμε 7 συσκευές\* 5 έλεγχοι/sec \*  $(0.006 + 0.3 * 0.00509 \text{ sec}) = 0.263445 \rightarrow 26.34\%$

β) Επειδή ο ελεγκτής της περιφερειακής συσκευής περιέχει 2 καταχωρητές, η μεταφορά δεδομένων στους καταχωρητές του ελεγκτή είναι  $\frac{4 \text{ bits}}{0.5 \text{ sec}} = \frac{x \text{ bits}}{1 \text{ sec}} \Rightarrow 4 = 0.5 * x \Rightarrow x = 8$

γ)

Μεταφράσεις  
Γιατί η παραγωγή είναι μεγαλύτερη από την παραγωγή της σε δεδομένου ότι το παραγόμενο δεδομένο είναι μεγαλύτερο από την παραγωγή της.

$$\frac{1 \text{ bit}}{0,125 \text{ sec}} = \frac{x \text{ bit}}{1 \text{ sec}}$$

Αποτέλεσμα 8bit , 0,006sec έτηγκος, 0,002sec αριθμ.  
0,00006sec εγγραφή!

A. 20% → Μεταφράσεις  
9 φορές σε 1sec έτεγχοι

$$\Sigma 1 \text{ Μεταφράσεις} = 0,006 \text{ sec} + 0,002 \text{ sec} + 0,00006 \text{ sec} = \\ \text{έτηγκος} \quad \text{αριθμ.} \quad \text{εγγραφή}$$

- (A) • 0,006 sec έτηγκος
- 0,002 sec + 0,00006 sec = 0,00206 sec αριθμ. εγγραφή!
- 20% ήταν ποσό δεδ. 9 έτεγχοι σε 1 sec
- 6 λογάριθμοι σε εξέλιξη

$$\Sigma 1 \text{ sec} \rightarrow 6 * 9 * (0,006 + 0,00206) = 0,3462 \text{ sec}$$

Άρα 34,62%

(B) Νέα σειρά ρυθμών θεσαφοράς:  $\frac{1 \text{ bit}}{0,125 \text{ sec}}$

$$\frac{1 \text{ bit}}{0,125 \text{ sec}} = \frac{x \text{ bit}}{1 \text{ sec}} \Rightarrow \boxed{x = 8}$$

### 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 KME τη δυνατότητα εξυπηρέτησης σημάτων διακοπής. Όλες οι μονάδες εισ/εξ μιας ομάδας στέλνουν σήμα διακοπής στην KME μέσω μιας κοινής γραμμής. Κάθε ομάδα χρησιμοποιεί διαφορετική γραμμή. Η KME 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 áθροισμα των ν πρώτων όρων μιας αριθμητικής προόδου, δίνεται από τη σχέση  $\Sigma_v = [(\alpha_1 + \alpha_v)v]/2$ , όπου  $\alpha_1$  είναι ο πρώτος όρος και  $\alpha_v$  ο τελευταίος.

A. 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. .

B. Θεωρήστε ότι όταν ζήτησε εξυπηρέτηση μια μονάδα εισ/εξ δεν υπήρχε άλλη εξυπηρέτηση σε εξέλιξη.

B.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.

B.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

A)  $19 \text{ ομάδες} \times 7 \text{ nsec} + 3160 \text{ κ.ρ.} \times 7 \frac{\text{nsec}}{\text{κ.ρ.}} = 133 + 22120 = 22253 \text{ nsec}$ . Αυτός είναι ο συνολικός χρόνος για την εξυπηρέτηση μιας μονάδας.

B)  $19 \text{ ομάδες} \times 7 \text{ nsec} + 69 \times 3160 \text{ κ.ρ.} \times 7 \frac{\text{nsec}}{\text{κ.ρ.}} = 1526413$

Γ)  $\Sigma_v = [(\alpha_1 + \alpha_v)v]/2 = [(1 + 19)19]/2 = 190 \rightarrow 190 \times 69 \times 7 = 91770 \text{ nsec}$

Γ)  $\Sigma_v = [(\alpha_1 + \alpha_v)v]/2 = [(1 + 69)69]/2 = 2415 \rightarrow 2415 \times 19 \times 7 = 321195 \text{ nsec}$

Γ)  $\Sigma_v = [(\alpha_1 + \alpha_v)v]/2 = [(1 + 1311)1311]/2 = 1720032 \rightarrow 1720032 \times 7 = 6020112 \text{ nsec}$

## 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 KME τη δυνατότητα εξυπηρέτησης σημάτων διακοπής. Όλες οι μονάδες εισ/εξ μιας ομάδας στέλνουν σήμα διακοπής στην KME μέσω μιας κοινής γραμμής. Κάθε ομάδα χρησιμοποιεί διαφορετική γραμμή. Η KME 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 áθροισμα των ν πρώτων όρων μιας αριθμητικής προόδου, δίνεται από τη σχέση  $\Sigma_v = [(\alpha_1 + \alpha_v)v]/2$ , όπου  $\alpha_1$  είναι ο πρώτος όρος και  $\alpha_v$  ο τελευταίος.

A. 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. .

B. Θεωρήστε ότι όταν ζήτησε εξυπηρέτηση μια μονάδα εισ/εξ δεν υπήρχε άλλη εξυπηρέτηση σε εξέλιξη.

B.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.

B.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

A. 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

$$(612+1836)*5 \text{ nsec}$$

B1. 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

$$(31*612)*5\text{nsec} + (612+1836)*5\text{nsec}$$

Εναλλακτικά θα μπορούσαμε να το γράψουμε ως εξής:  $32 * 612 \text{ k.p.} * 5 \text{ nsec} + 1836 \text{ k.p.} * 5 \text{ nsec}$

B2. 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

$$\text{B2) O Xρονος ειναι } \Sigma v = [(t1(A \text{ ερωτημα})+t2(B \text{ ερωτημα}))*32]/2$$

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:

$$t1 = (612\text{k}\mu\text{s} + 1836\text{k}\mu\text{s}) * 5\text{nsec} = 12.240\text{nsec}$$

B) Ο μεγαλύτερος χρόνος που είναι δυνατόν να απαιτηθεί για την εξυπηρέτηση μονάδας, είναι όταν εξυπηρετηθούν όλες οι μονάδες κατά σειρά. Οπότε είναι:

$$t2 = (612\text{k}\mu\text{s} + 1836\text{k}\mu\text{s}) * 5\text{nsec} * 864 \text{ μονάδες} = 10.575.360 \text{ nsec}$$

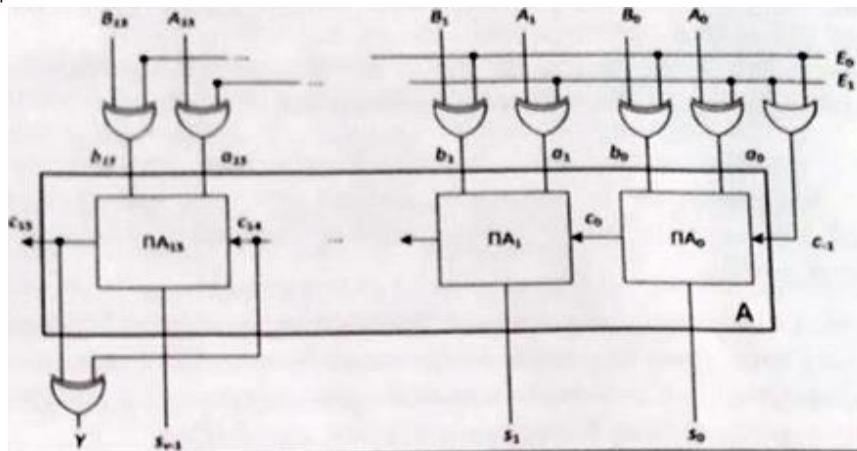
$$\text{B2) O Xρονος ειναι } \Sigma v = [(t1(A \text{ ερωτημα})+t2(B \text{ ερωτημα}))*32]/2$$

Και μετα  $\Sigma v/32$  για να βρουμε τον μεσο χρονο

## Θέματα Ιανουαρίου 2022

### Θέμα 1

Η κεντρική μονάδα επεξεργασίας επεξεργαστή διαθέτει τον αθροιστή/αφαιρέτη του διπλανού σχήματος και καταχωρητές των 16 δυαδικών ψηφίων.



- Για την πράξη  $B-A$  εάν οι αριθμοί είναι **σε αναπαράσταση συμπληρώματος ως προς 2** και  $Y=0$  τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
- Για την πράξη  $B-A$  εάν οι αριθμοί είναι **σε αναπαράσταση συμπληρώματος ως προς 2** και  $Y=1$  τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
- Για την πράξη  $B+A$  εάν οι αριθμοί είναι **χωρίς πρόσημο (θετικοί)** και  $Y=1$  τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
- Για την πράξη  $B-A$  εάν οι αριθμοί είναι **σε αναπαράσταση συμπληρώματος ως προς 2** και  $C_{15}=1$  τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.



5. Για την πράξη B+A εάν οι αριθμοί είναι **χωρίς πρόσημο (θετικοί)** και **Y=0** τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
6. Για την πράξη A-B εάν οι αριθμοί είναι **σε αναπαράσταση συμπληρώματος ως προς 2** και **Y=1** τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
7. Για την πράξη A-B εάν οι αριθμοί είναι σε **αναπαράσταση συμπληρώματος ως προς 2** και **Y=0** τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
8. Για την πράξη B+A εάν οι αριθμοί είναι σε **αναπαράσταση συμπληρώματος ως προς 2** και **C<sub>15</sub>=0** τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
9. Για την πράξη B+A εάν οι αριθμοί είναι σε **αναπαράσταση συμπληρώματος ως προς 2** και **Y=1** τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
10. Για την πράξη B+A εάν οι αριθμοί είναι **χωρίς πρόσημο (θετικοί)** και **C<sub>15</sub>=0** τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
11. Για την πράξη B+A εάν οι αριθμοί είναι **χωρίς πρόσημο (θετικοί)** και **C<sub>15</sub>=1** τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
12. Για την πράξη B+A εάν οι αριθμοί είναι **σε αναπαράσταση συμπληρώματος ως προς 2** και **C<sub>15</sub>=1** τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
13. Για την πράξη B+A εάν οι αριθμοί είναι σε **αναπαράσταση συμπληρώματος ως προς 2** και **Y=0** τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
14. Για την πράξη B-A εάν οι αριθμοί είναι **σε αναπαράσταση συμπληρώματος ως προς 2** και **C<sub>15</sub>=0** τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
15. Για την πράξη B-A εάν οι αριθμοί είναι **χωρίς πρόσημο (θετικοί)** και **Y=1** τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
16. Για την πράξη B-A εάν οι αριθμοί είναι **χωρίς πρόσημο (θετικοί)** και **C<sub>15</sub>=0** τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
17. Για την πράξη B-A εάν οι αριθμοί είναι **χωρίς πρόσημο (θετικοί)** και **C<sub>15</sub>=1** τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
18. Για την πράξη B-A εάν οι αριθμοί είναι **χωρίς πρόσημο (θετικοί)** και **Y=0** τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.

### Λύση

1. Λ, 2. Σ, 3. Λ, 4. Λ, 5. Λ, 6. Σ, 7. Λ, 8. Λ, 9. Σ, 10. Λ, 11. Σ, 12. Λ, 13. Λ, 14. Λ, 15. Λ, 16. Λ, 17. Σ, 18. Λ

### Θέμα 2

Θεωρήστε 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 A, B, Γ και Δ. Η μεταφορά πληροφορίας στον κύριο από τη μονάδα εισόδου A, B, Γ και Δ θα απαιτούσε 14, 18, 31 και 56 nsec αντίστοιχα εάν ήταν η μόνη μονάδα που συνδέονταν μέσω της αρτηρίας με τον κύριο. Θεωρήστε επίσης ότι η διαιτησία βάζει καθυστέρηση 5 nsec. Να υπολογίστε το **ρυθμό μεταφοράς πληροφορίας σε ψηφιολέξεις ανά δευτερόλεπτο** από τη **μονάδα Γ στον κύριο** για κάθε μια από τις ακόλουθες περιπτώσεις.

- a. Η αρτηρία είναι **σύγχρονη** με περίοδο του σήματος χρονισμού (ρολογιού) **5 nsec**.
  - i. χωρίς δυνατότητα εισαγωγής κύκλων ρολογιού καθυστέρησης
  - ii. με δυνατότητα εισαγωγής κύκλων ρολογιού καθυστέρησης
- β. Η αρτηρία είναι **ασύγχρονη** απαιτεί και 8 nsec επιπλέον για κάθε μεταφορά λόγω του πρωτοκόλλου χειραψίας.

### Λύση

Το χαρακτηριστικό σε αυτήν την άσκηση είναι ότι λαμβάνουμε υπόψη **μόνο τη μονάδα Γ**, όπως αναφέρεται σχετικά στην εκφώνηση.

- α) i) Ρυθμός μεταφοράς πληροφορίας = 8 bytes/(60 + 5) nsec = 8 bytes/65 nsec = 8 bytes/65 x 10<sup>-9</sup> sec = 0.123 x 10<sup>9</sup> bytes/sec

- ii) Ρυθμός μεταφοράς πληροφορίας =  $8 \text{ bytes}/(35 + 5) \text{ nsec} = 8 \text{ bytes}/40 \text{ nsec} = 0.2 \times 10^9 \text{ bytes/sec}$ , λαμβάνουμε υπόψη μόνο τη Γ.  
 β) Ρυθμός μεταφοράς πληροφορίας =  $8 \text{ bytes}/(31 + 8 + 5) \text{ nsec} = 8 \text{ bytes}/44 \text{ nsec} = 0.18 \times 10^9 \text{ bytes/sec}$ , λαμβάνουμε υπόψη μόνο τη Γ.

### Θέμα 3

Θεωρήστε επεξεργαστή με 16 καταχωρητές των 32 δυαδικών ψηφίων ο καθένας που στο σύνολο των εντολών του περιλαμβάνονται οι εντολές του Πίνακα 1.

**Πίνακας 1**

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

**Προσοχή:** σε όλες τις ανωτέρω εντολές χρησιμοποιείται ο κατ' ευθείαν τρόπος διευθυνσιοδότησης (direct addressing mode)

Στον Πίνακα 2 δίνονται τα περιεχόμενα τμήματος της κύριας μνήμης και στον Πίνακα 3 τα περιεχόμενα των καταχωρητών

**Πίνακας 2**

Διεύθυνση θέσης μνήμης στο δεκαεξαδικό	περιεχόμενο θέσης μνήμης στο δεκαεξαδικό πριν την εκτέλεση των εντολών	περιεχόμενο θέσης μνήμης στο δεκαεξαδικό μετά την εκτέλεση των εντολών
110	87	
111	00	
112	11	
113	E3	
114	E4	
115	00	
116	AB	
117	F0	
118	01	
119	B6	
11A	05	
11B	32	
11C	AE	
11D	BC	
11E	10	
11F	C4	
120	89	
121	1A	
122	A0	

πριν την εκτέλεση των εντολών του Πίνακα 4.

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. Να συμπληρώσετε στον Πίνακα 2 το περιεχόμενο των θέσεων μνήμης και στον Πίνακα 3 το περιεχόμενο των καταχωρητών μετά την εκτέλεση των εντολών του Πίνακα 4.

**Πίνακας 3**

Καταχωρητής	Περιεχόμενο του καταχωρητή στο δεκαεξαδικό πριν την εκτέλεση της εντολής	Περιεχόμενο του καταχωρητή στο δεκαεξαδικό μετά την εκτέλεση των εντολών
R1=	0F 0F 0F 0F	
R2=	0F 0F 0F 0F	

R3=	0F 0F 0F 0F	
R4=	0F 0F 0F 0F	
R5=	0F 0F 0F 0F	
R6=	00 FF FF 18	
R7=	FF 09 24 56	
R8=	6A 23 AE 00	
R9=	0F 0F 0F 0F	

Πίνακας 4
εντολή
LOADSB R1, 110 <sub>(16)</sub>
LOADSHW R2, 116 <sub>(16)</sub>
LOADW R3, 111 <sub>(16)</sub>
LOADUB R4, 112 <sub>(16)</sub>
LOADUHW R5, 119 <sub>(16)</sub>
STOREB 121 <sub>(16)</sub> , R6
STOREHW 11B <sub>(16)</sub> , R7
STOREW 11D <sub>(16)</sub> , R8
LOADSB R9, 122 <sub>(16)</sub>

### Λύση

Οι απαντήσεις συνοψίζονται στους δύο επόμενους πίνακες. Σε κάθε ένα από αυτούς, φαίνονται με επισήμανση τα περιεχόμενα των θέσεων μνήμης/καταχωρητών που έχουν **αλλάξει** μετά την εκτέλεση των εντολών.

**Πίνακας 2**

Διεύθυνση θέσης μνήμης στο δεκαεξαδικό	Περιεχόμενο θέσης μνήμης στο δεκαεξαδικό πριν την εκτέλεση των εντολών	Περιεχόμενο θέσης μνήμης στο δεκαεξαδικό μετά την εκτέλεση των εντολών
110	87	87
111	00	00
112	11	11
113	E3	E3
114	E4	E4
115	00	00
116	AB	AB
117	F0	F0
118	01	01
119	B6	B6
11A	05	05
11B	32	24
11C	AE	56
11D	BC	6A
11E	10	23
11F	C4	AE
120	89	00
121	1A	18
122	A0	A0

**Πίνακας 3**

Καταχωρητής	Περιεχόμενο του καταχωρητή στο δεκαεξαδικό πριν την εκτέλεση της εντολής	Περιεχόμενο του καταχωρητή στο δεκαεξαδικό μετά την εκτέλεση των εντολών
R1=	0F 0F 0F 0F	FF FF FF 87
R2=	0F 0F 0F 0F	FF FF AB F0
R3=	0F 0F 0F 0F	00 11 E3 E4
R4=	0F 0F 0F 0F	00 00 00 11
R5=	0F 0F 0F 0F	00 00 B6 05
R6=	00 FF FF 18	00 FF FF 18
R7=	FF 09 24 56	FF 09 24 56
R8=	6A 23 AE 00	6A 23 AE 00
R9=	0F 0F 0F 0F	FF FF FF A0

#### Θέμα 4

Να σχεδιάσετε μονάδα μνήμης με χωρητικότητα 144 Mbits και 9 δυαδικά ψηφία ανά θέση, η οποία να διαθέτει είσοδο επιλογής μονάδας και σε κάθε προσπέλαση θα προσπελαύνεται μία θέση (9 δυαδικά ψηφία). Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης:

OK0 με χωρητικότητα 32 Mbits και 8 δυαδικά ψηφία ανά θέση,

OK1 με χωρητικότητα 4 Mbits και 1 δυαδικό ψηφίο ανά θέση,

Καθώς και οποιονδήποτε ή οποιουσδήποτε αποκωδικοποιητές και πύλες χρειάζεστε.

A. Πόσα ολοκληρωμένα κυκλώματα του τύπου OK0 και πόσα ολοκληρωμένα κυκλώματα του τύπου OK1 απαιτούνται για την υλοποίηση της ζητούμενης μονάδας μνήμης ώστε το **πλήθος των ολοκληρωμένων που θα χρησιμοποιηθούν να είναι το ελάχιστο**:

B. Να δώσετε ένα σχήμα στο οποίο να φαίνεται όλη η απαιτούμενη πληροφορία.

**Λύση**

**OK0**

1 θ.μ. 8 bits

x; 32 Mbits

$$x = 2^5 \times 2^{20}/2^3 = 2^{22} \text{ θ.μ.}$$

**OK1**

1 θ.μ. 1 bits

x; 4 Mbits

$$x = 2^{20} \times 2^2/1 = 2^{22} \text{ θ.μ.}$$

**Κύρια μνήμη**

1 θ.μ. 9 bits

x; 144 Mbits

$$x = 144 \times 2^{20}/9 = 16 \times 2^{20} \text{ θ.μ.} = 2^{24} \text{ θ.μ.}$$

Οριζόντια επανάληψη: 2 (8 bits + 1 bits), πάντα **δημιουργεί το επιθυμητό εύρος μνήμης**

Κατακόρυφη επανάληψη:  $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{2^{24} \text{ θ.μ.}}{2^{22} \text{ θ.μ.}} = 4$ . **Επιστρέφεται ακέραιο πηλίκο**

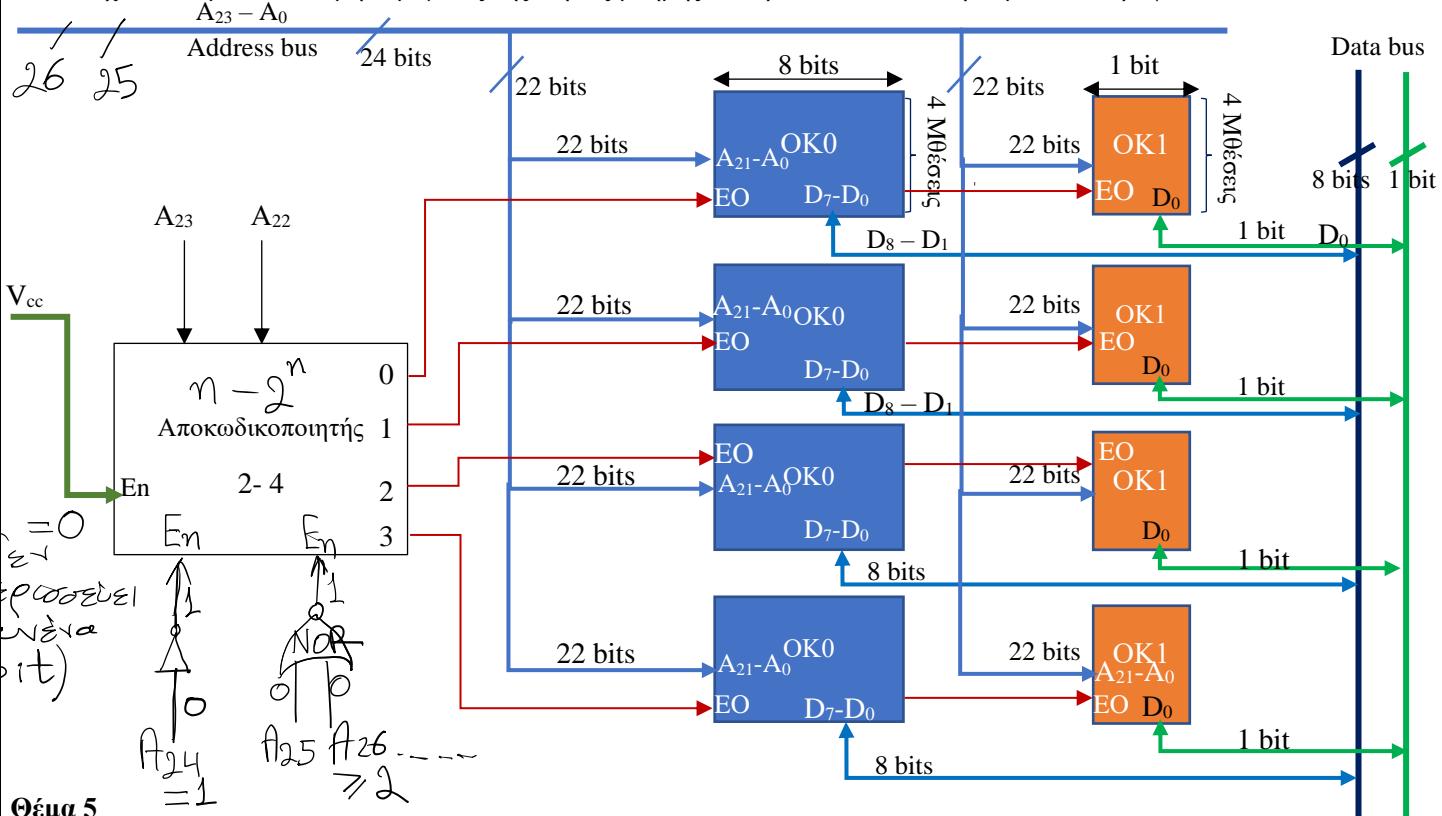
Κατακόρυφη επανάληψη:  $\frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{2^{24} \text{ θ.μ.}}{2^{22} \text{ θ.μ.}} = 4$ . **Επιστρέφεται ακέραιο πηλίκο**

- Κάθε OK0 αφού περιέχει  $2^{22}$  θ.μ., έχει  $2^{22} = 2^2 \cdot 2^{20} = 4$  Μθέσεις του ενός byte η καθεμία.

- Κάθε OK1 αφού περιέχει  $2^{22}$  θ.μ., έχει  $2^{22} = 2^2 \cdot 2^{20} = 4$  Μθέσεις του ενός bit η καθεμία.

- Η **κύρια μνήμη** που θα σχεδιάσουμε έχει  $16 \times 2^{20}$  θ.μ. = 16 Μθέσεις των 9 bits η καθεμία =  $2^{24}$  θ.μ.

Για να σχεδιάσουμε το επιθυμητό μέγεθος της κύριας μνήμης, θα πρέπει να επαναλάβουμε κατακόρυφα 4 OK0 και 4 OK1



#### Θέμα 5

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:





16. 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. μη προσημασμένα δεδομένα μεγέθους μιας ψηφιολέξης, 2. προσημασμένα δεδομένα μεγέθους μιας ψηφιολέξης.
17. 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 λέξης, 2. προσημασμένα δεδομένα μεγέθους μιας λέξης.
18. 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.
19. 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. μη προσημασμένα δεδομένα μεγέθους μισής λέξης, 3. μη προσημασμένα δεδομένα μεγέθους μιας λέξης.

### Λύση

1. Σ, 2. Λ, 3. Σ, 4. Σ, 5. Λ, 6. Λ, 7. Σ, 8. Λ, 9. Λ, 10. Λ, 11. Σ, 12. Λ, 13. Σ, 14. Λ, 15. Σ, 16. Λ, 17. Σ, 18. Σ, 19. Λ

## Θέματα Σεπτεμβρίου 2022

### Θέμα 1

Θεωρήστε επεξεργαστή με καταχωρητές γενικού σκοπού των 32 δυαδικών ψηφίων και καταχωρητής κινητής υποδιαστολής των 32 δυαδικών ψηφίων. Ο καταχωρητής γενικού σκοπού R1 περιέχει τον αριθμό **-8** σε αναπαράσταση συμπληρώματος ως προς 2, ενώ ο καταχωρητής κινητής υποδιαστολής FR1 περιέχει τον αριθμό **-8** σε μορφή κινητής υποδιαστολής, σύμφωνα με το πρότυπο IEEE 754 απλής ακρίβειας. Να δώσετε το περιεχόμενο των καταχωρητών R1 και FR1 στο δυαδικό.

**Υπενθύμιση.** Το πρότυπο απλής ακρίβειας της IEEE:

$N = NaN$  εάν  $E = 255$  και  $\Sigma_K \neq 0$ ,

$N = (-1)^{\Pi} \infty$  εάν  $E = 255$  και  $\Sigma_K = 0$ ,

$N = (-1)^{\Pi} \times 2^{E-127} \times (1.\Sigma_K)$  εάν  $0 < E < 255$  (κανονικοποιημένοι αριθμοί),

$N = (-1)^{\Pi} \times 2^{126} \times (0.\Sigma_K)$  εάν  $E = 0$  και  $\Sigma_K \neq 0$  (μη κανονικοποιημένοι αριθμοί),

$N = (-1)^{\Pi} 0$  εάν  $E = 0$  και  $\Sigma_K = 0$  (διπλή αναπαράσταση του μηδενός).

R1	
FR1	

### Λύση

**R1:**  $-8 \rightarrow 8 \rightarrow 01000 \rightarrow 10111 + 1 = 11000 \rightarrow$  κάνουμε **επέκταση προσήμου** με το MSB για να **αποκτήσει μέγεθος 32 bits**. Και έτσι να γεμίσει πλήρως ο καταχωρητής R1 που περιέχει το συγκεκριμένο αριθμό.

**FR1:** Ξεκινάμε με το θετικό  $8 \rightarrow 1000 = 1000 \times 2^0 = 1000.0 \times 2^0 = 1.0000 \times 2^3$  (κάνουμε κανονικοποίηση 1.XXXX...X). Στη συνέχεια, προσθέτουμε στον εκθέτη την πόλωση (127 για απλή ακρίβεια) και αυτό γίνεται  $3 + 127 = 130 = 10000010$

R1	FFFFFFFFFF8 = 1111 1111 1111 1111 1111 1111 1000
----	--

	$\pi$	E	$\Sigma_K$
FR1	1	10000010	0000...000000
	/ — 1 — /	/ — 8 — /	/ — 23 — /

Το ακριβές περιεχόμενο του καταχωρητή FR είναι: **11000001000000....0000**

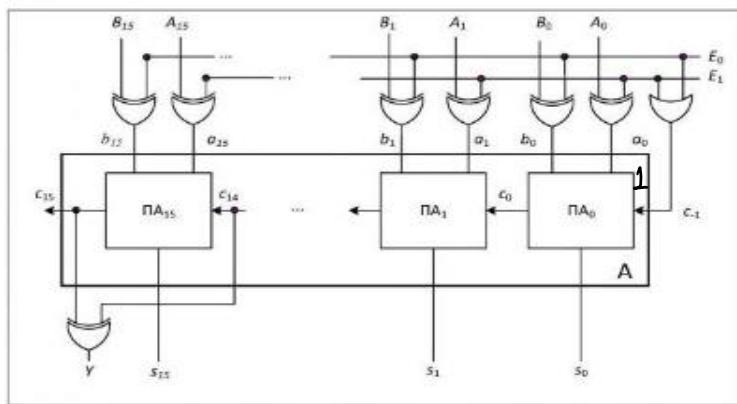
**Σημείωση 1:** για τον αριθμό -8.5 θα μετατρέπαμε χωριστά στο δυαδικό σύστημα το ακέραιο και το κλασματικό μέρος του αριθμού, οπότε θα είχαμε:  $8 = 1000$  και το  $0.5 = 0.1$ , οπότε:  $8.5 = 1000.1 \times 2^0 = 1000.1 \times 2^0 = 1.0001 \times 2^3$  (κάνουμε κανονικοποίηση 1.XXXX...X). Στη συνέχεια, προσθέτουμε στον εκθέτη την πόλωση (127 για απλή ακρίβεια) και αυτό γίνεται  $3 + 127 = 130 = 10000010$

### Σημείωση 2:

για τον αριθμό -22.125 θα μετατρέπαμε χωριστά στο δυαδικό σύστημα το ακέραιο και το κλασματικό μέρος του αριθμού, οπότε θα είχαμε:  $22 = 10110$  και το  $0.125 = 0.001$ , οπότε:  $22.125 = 10110.001 \times 2^0 = 1.0110001 \times 2^4$  (κάνουμε κανονικοποίηση 1.XXXX...X). Στη συνέχεια, προσθέτουμε στον εκθέτη την πόλωση (127 για απλή ακρίβεια) και αυτό γίνεται  $4 + 127 = 131 = 10000011$  και το ακριβές περιεχόμενο του FR1 είναι: **[1 10000011 011000100....0]**. Όσον αφορά το περιεχόμενο του καταχωρητή R1 έχουμε το εξής:  $22.125 = 010110.001 \rightarrow 101001.110 + 1 = 101001.111 \rightarrow 1....11 101001.111$

## Θέμα 2

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  $S_{15}, S_{14}, \dots, S_0$  του αθροιστή. Λ



Χρησιμοποιήστε τους συμβολισμούς  $Y'$  για να δηλώσετε το συμπλήρωμα ως προς 1 του  $Y$  και  $Y''$  για να δηλώσετε το συμπλήρωμα ως προς 2 του  $Y$

### Λύση

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:

$$E_0 = 0 \text{ και } E_1 = 0 \rightarrow A + B \text{ (ή } B + A\text{).}$$

$$E_0 = 0 \text{ και } E_1 = 1 \rightarrow B - A = B + A' + 1$$

$$E_0 = 1 \text{ και } E_1 = 0 \rightarrow A - B = A + B' + 1$$

$$E_0 = 1 \text{ και } E_1 = 1 \rightarrow -A - B \text{ ή } -B - A$$

## Θέμα 3

Να σημειώσετε ποιες από τις ακόλουθες προτάσεις ισχύουν (χωρίς αιτιολόγηση).

1. Για την πράξη  $B+A$  εάν οι αριθμοί είναι χωρίς πρόσημο (θετικοί) και  $C_{15}=0$  τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
2. Για την πράξη  $B-A$  εάν οι αριθμοί είναι σε αναπαράσταση συμπληρώματος ως προς 2 και  $Y=1$  τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
3. Για την πράξη  $B-A$  εάν οι αριθμοί είναι σε αναπαράσταση συμπληρώματος ως προς 2 και  $Y=0$  τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
4. Για την πράξη  $B-A$  εάν οι αριθμοί είναι χωρίς πρόσημο (θετικοί) και  $Y=1$  τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
5. Για την πράξη  $B+A$  εάν οι αριθμοί είναι σε αναπαράσταση συμπληρώματος ως προς 2 και  $Y=1$  τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
6. Για την πράξη  $B-A$  εάν οι αριθμοί είναι χωρίς πρόσημο (θετικοί) και  $Y=0$  τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
7. Για την πράξη  $B-A$  εάν οι αριθμοί είναι σε αναπαράσταση συμπληρώματος ως προς 2 και  $C_{15}=1$  τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
8. Για την πράξη  $B+A$  εάν οι αριθμοί είναι χωρίς πρόσημο (θετικοί) και  $Y=0$  τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
9. Για την πράξη  $B-A$  εάν οι αριθμοί είναι χωρίς πρόσημο (θετικοί) και  $C_{15}=0$  τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
10. Για την πράξη  $B-A$  εάν οι αριθμοί είναι χωρίς πρόσημο (θετικοί) και  $C_{15}=1$  τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
11. Για την πράξη  $B+A$  εάν οι αριθμοί είναι χωρίς πρόσημο (θετικοί) και  $Y=1$  τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
12. Για την πράξη  $B+A$  εάν οι αριθμοί είναι χωρίς πρόσημο (θετικοί) και  $C_{15}=0$  τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.

13. Για την πράξη  $B+A$  εάν οι αριθμοί είναι χωρίς πρόσημο (θετικοί) και  $C_{15}=1$  τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
14. Για την πράξη  $A-B$  εάν οι αριθμοί είναι σε αναπαράσταση συμπληρώματος ως προς 2 και  $Y=0$  τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
15. Για την πράξη  $B+A$  εάν οι αριθμοί είναι σε αναπαράσταση συμπληρώματος ως προς 2 και  $C_{15}=1$  τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.
16. Για την πράξη  $B-A$  εάν οι αριθμοί είναι σε αναπαράσταση συμπληρώματος ως προς 2 και  $C_{15}=0$  τότε το αποτέλεσμα της πράξης δεν χωράει σε ένα καταχωρητή.

### Λύση

1. Α
2. Σ
3. Α
4. Λ
5. Σ
6. Α
7. Λ
8. Λ
9. Λ
10. Σ
11. Λ
12. Λ
13. Σ
14. Λ
15. Λ
16. Λ

### 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.** Θεωρήστε ότι σε όλες τις εντολές στις οποίες αναφερόμαστε στη συνέχεια, 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.

1. 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: α. μη προσημασμένα δεδομένα μεγέθους μισής λέξης, β. προσημασμένα δεδομένα μεγέθους μισής λέξης.
2. 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: α. μη προσημασμένα δεδομένα μεγέθους μιας λέξης, β. προσημασμένα δεδομένα μεγέθους μιας λέξης.
3. 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: α. μη προσημασμένα δεδομένα μεγέθους μιας ψηφιολέξης, β. μη προσημασμένα δεδομένα μεγέθους μισής λέξης, γ. μη προσημασμένα δεδομένα μεγέθους μιας λέξης.
4. 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: α. μη προσημασμένα δεδομένα μεγέθους μιας ψηφιολέξης, β. μη προσημασμένα δεδομένα μεγέθους μισής λέξης, γ. μη προσημασμένα δεδομένα μεγέθους μιας λέξης.
5. 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: α. προσημασμένα δεδομένα μεγέθους μιας ψηφιολέξης, β. προσημασμένα δεδομένα μεγέθους μισής λέξης, γ. προσημασμένα δεδομένα μεγέθους μιας λέξης.
6. 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: α. μη προσημασμένα δεδομένα μεγέθους μιας λέξης, β. προσημασμένα δεδομένα μεγέθους μισής λέξης.
7. 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: α. μη προσημασμένα δεδομένα μεγέθους μιας λέξης, β. προσημασμένα δεδομένα μεγέθους μισής λέξης.

## Λύση

1. Σ
2. Λ
3. Σ
4. Λ
5. Λ
6. Λ
7. Σ

## Θέμα 5

Θεωρήστε επεξεργαστή με καταχωρητές γενικού σκοπού των 32 δυαδικών ψηφίων και καταχωρητές κινητής υποδιαστολής των 32 δυαδικών ψηφίων. Θεωρήστε ότι ο επεξεργαστής μεταξύ των άλλων εντολών υποστηρίζει και τις εντολές του πίνακα που ακολουθεί.

STORE FR1	\$ XXXX	κατευθείαν τρόπος διευθυνσιοδότησης (direct addressing mode), X: δεκαεξαδικό ψηφίο
STOREB R1	\$ XXXX	μεταφορά byte, κατευθείαν τρόπος διευθυνσιοδότησης, X: δεκαεξαδικό ψηφίο
LOADSB R2	\$ XXXX	μεταφορά byte με αναπαράσταση συμπληρώματος ως προς 2, κατευθείαν τρόπος διευθυνσιοδότησης, X: δεκαεξαδικό ψηφίο
LOADW R3	\$ XXXX	μεταφορά λέξης (4 bytes), κατευθείαν τρόπος διευθυνσιοδότησης, X: δεκαεξαδικό ψηφίο

Θεωρήστε τα περιεχόμενα των καταχωρητών και της κύριας μνήμης που δίνονται στο ακόλουθο σχήμα.

FR1:	5467DAC0	διεύθυνση μνήμη	περιεχόμενο
R1:	FFFFFFFA4	0100	23
R2:	FFFFFFFFF	0101	A3
R3:	FFFFFFFFF	0102	56
R4:	FFFFFFFFF	0103	90
		0104	33
		0105	FE
		0106	45
		0107	ED
		0108	25
		0109	A4
		010A	17
		010B	34
		010C	3D
		010D	23
		010E	99
		010F	AD
		0110	9D

Για καθεμία των κάτωθι περιπτώσεων να δώσετε στο δεκαεξαδικό τα περιεχόμενα των καταχωρητών και της μνήμης μετά την εκτέλεση των εντολών του πίνακα που ακολουθεί.

- Απεικόνιση ανάλογη του μεγέθους (little endian)
- Απεικόνιση αντιστρόφως ανάλογη του μεγέθους big endian

STORE FR1	\$ 0104
STOREB R1	\$ 0109
LOADSB R2	\$ 010A
LOADW R3	\$ 010C
LOADSB R4	\$ 0110

- a. Απεικόνιση ανάλογη του μεγέθους (little endian)

FR1:	
R1:	
R2:	
R3:	
R4:	

- β. Απεικόνιση αντιστρόφως ανάλογη του μεγέθους (big endian)

FR1:	
R1:	
R2:	
R3:	
R4:	

## Λύση

- a. Απεικόνιση ανάλογη του μεγέθους (little endian)

FR1:	5467DAC0
R1:	FFFFFFFA4

διεύθυνση μνήμη	περιεχόμενο
0100	
0101	
0102	
0103	
0104	
0105	
0106	
0107	
0108	
0109	
010A	
010B	
010C	
010D	
010E	
010F	
0110	

διεύθυνση μνήμη	περιεχόμενο
0100	
0101	
0102	
0103	
0104	
0105	
0106	
0107	
0108	
0109	
010A	
010B	
010C	
010D	
010E	
010F	
0110	



R2:	<b>00000017</b>
R3:	<b>AD99233D</b>
R4:	<b>FFFFFFFFFF9D</b>

Διεύθυνση μνήμη	Περιεχόμενο (little endian)	Περιεχόμενο (big endian)
0100	23	23
0101	A3	A3
0102	56	56
0103	90	90
0104	<b>C0</b>	<b>54</b>
0105	<b>DA</b>	<b>67</b>
0106	<b>67</b>	<b>DA</b>
0107	<b>54</b>	<b>C0</b>
0108	25	25
0109	<b>A4</b>	<b>A4</b>
010A	17	17
010B	34	34
010C	3D	3D
010D	23	23
010E	99	99
010F	AD	AD
0110	9D	9D

### β. Απεικόνιση αντιστρόφως ανάλογη του μεγέθους (big endian)

FR1:	5467DAC0
R1:	FFFFFFFA4
R2:	<b>00000017</b>
R3:	<b>3D2399AD</b>
R4:	<b>FFFFFFFFFF9D</b>

Διεύθυνση μνήμη	Περιεχόμενο
0100	23
0101	A3
0102	56
0103	90
0104	<b>54</b>
0105	<b>67</b>
0106	<b>DA</b>
0107	<b>C0</b>
0108	25
0109	<b>A4</b>
010A	17
010B	34
010C	3D
010D	23
010E	99
010F	AD
0110	9D

## Θέμα 6

Οι απαντήσεις για την άσκηση αυτή να μπουν στον πίνακα που ακολουθεί. Τα R1, R2, R3 κ.λ.π. δηλώνουν καταχωρητές, ενώ τα A, B, C, D και Y δηλώνουν διευθύνσεις θέσεων μνήμης.

Α. Στην αριστερή στήλη του κάτωθι πίνακα υπάρχει ένα πρόγραμμα.

Α.1. Να δώσετε την αριθμητική έκφραση που υπολογίζει αυτό το πρόγραμμα.

Α.2. Λαμβάνοντας υπόψη τις εντολές του προγράμματος, να αναφέρετε την αρχιτεκτονική του επεξεργαστή.

Β. Θεωρήστε ότι το σύνολο των εντολών του επεξεργαστή σε επίπεδο γλώσσας μηχανής **αποτελείται από 235 εντολές**, ο επεξεργαστής **διαθέτει 16 καταχωρητές** γενικού σκοπού των 64 δυαδικών ψηφίων, **το μήκος κάθε εντολής σε δυαδικά ψηφία είναι ακέραιο πολλαπλάσιο του 8** και το **εύρος της αρτηρίας διευθύνσεων είναι 64 δυαδικά ψηφία**. Για κάθε διαφορετική εντολή της συμβολικής γλώσσας, να σχεδιάσετε τη μορφή της εντολής σε επίπεδο γλώσσας μηχανής (δηλαδή τα πεδία από τα οποία αποτελείται η εντολή σε επίπεδο γλώσσας μηχανής και το εύρος κάθε πεδίου).

Γ. Να δώσετε το πλήθος των ψηφιολέξεων που καταλαμβάνει το πρόγραμμα στην κύρια μνήμη.

Δ. θεωρήστε ότι τα **δεδομένα είναι μίας λέξης** (64 δυαδικά ψηφία), η αρτηρία δεδομένων μεταξύ της Κεντρικής Μονάδας Επεξεργασίας (KME) και του συστήματος μνήμης **έχει εύρος 2 ψηφιολέξεις (byte)** και **σε κάθε προσπέλαση του συστήματος μνήμης προσπελαύνονται δύο ψηφιολέξεις**. Πόσες **προσπελάσεις** με μνήμη απαιτεί η εκτέλεση του προγράμματος;

	Μορφή εντολής σε επίπεδο γλώσσας μηχανής	Πλήθος ψηφιολέξεων που καταλαμβάνει η εντολή στη μνήμη	Πλήθος προσπελάσεων στη μνήμη που απαιτεί η προσπέλαση και εκτέλεση της εντολής
LOAD R0, B			
LOAD R1, C			
MUL R0, R1			
LOAD R1, D			
LOAD R2, E			
MUL R1, R2			
ADD R0, R1			
LOAD R1, A			
ADD R0, R1			
STORE Y, R0			
A.1. Αριθμητική έκφραση που υπολογίζει το ανωτέρω πρόγραμμα:	A.2. Αρχιτεκτονική του επεξεργαστή:	Πλήθος ψηφιολέξεων που καταλαμβάνει το πρόγραμμα:	Πλήθος προσπελάσεων για την εκτέλεση του προγράμματος:

### Άλση

Από τη διαίρεση  $\frac{\Delta \text{δεδομένα}}{\text{Εύρος αρτηρίας δεδομένων}} = \frac{64 \text{ bits}}{16 \text{ bits}} = 4$  έχουμε **4 επιπλέον προσπελάσεις** σε εντολές που έχουν έντελο μνήμη

Από αριθμό εντολών υπολογίζουμε το μήκος του opcode, και συγκεκριμένα  $2^x \geq 235 \Rightarrow x = 8 \text{ bits}$  και από αριθμό καταχωρητών υπολογίζουμε το μήκος του κάθε καταχωρητή ως έντελο, και συγκεκριμένα  $2^x \geq 16 \Rightarrow x = 4 \text{ bits}$ .

	Μορφή εντολής σε επίπεδο γλώσσας μηχανής	Πλήθος ψηφιολέξεων που καταλαμβάνει η εντολή στη μνήμη	Πλήθος προσπελάσεων στη μνήμη που απαιτεί η προσπέλαση και εκτέλεση της εντολής								
LOAD R0, B	<table border="1"> <tr> <td>8</td><td>4</td><td>64</td><td>4</td></tr> <tr> <td>opcode</td><td>R0</td><td>B</td><td>X</td></tr> </table>	8	4	64	4	opcode	R0	B	X	10 bytes	Επειδή σε κάθε προσπέλαση του συστήματος μνήμης προσπελαύνονται <b>δύο ψηφιολέξεις</b> , αυτό σημαίνει ότι για την προσπέλαση των 10 ψηφιολέξεων απαιτούνται 5 προσπελάσεις και επιπλέον, άλλες 4 επειδή προσπελαύνεται μνήμη (από τη διαίρεση $\frac{\Delta \text{δεδομένα}}{\text{αρτηρία δεδομένων}}$ ). Άρα συνολικά απαιτούνται <b>9 προσπελάσεις</b>
8	4	64	4								
opcode	R0	B	X								
LOAD R1, C	<table border="1"> <tr> <td>8</td><td>4</td><td>64</td><td>4</td></tr> <tr> <td>opcode</td><td>R1</td><td>C</td><td>X</td></tr> </table>	8	4	64	4	opcode	R1	C	X	10 bytes	9 προσπελάσεις
8	4	64	4								
opcode	R1	C	X								
MUL R0, R1	<table border="1"> <tr> <td>8</td><td>4</td><td>4</td><td></td></tr> <tr> <td>opcode</td><td>R0</td><td>R1</td><td></td></tr> </table>	8	4	4		opcode	R0	R1		2 bytes	Επειδή σε κάθε προσπέλαση του συστήματος μνήμης προσπελαύνονται δύο ψηφιολέξεις, αυτό σημαίνει ότι για την προσπέλαση των 2 ψηφιολέξεων απαιτείται <b>1 προσπέλαση</b>
8	4	4									
opcode	R0	R1									
LOAD R1, D	<table border="1"> <tr> <td>8</td><td>4</td><td>64</td><td>4</td></tr> <tr> <td>opcode</td><td>R1</td><td>D</td><td>X</td></tr> </table>	8	4	64	4	opcode	R1	D	X	10 bytes	9 προσπελάσεις
8	4	64	4								
opcode	R1	D	X								
LOAD R2, E	<table border="1"> <tr> <td>8</td><td>4</td><td>64</td><td>4</td></tr> <tr> <td>opcode</td><td>R2</td><td>E</td><td>X</td></tr> </table>	8	4	64	4	opcode	R2	E	X	10 bytes	9 προσπελάσεις
8	4	64	4								
opcode	R2	E	X								
MUL R1, R2	<table border="1"> <tr> <td>8</td><td>4</td><td>4</td><td></td></tr> <tr> <td>opcode</td><td>R1</td><td>R2</td><td></td></tr> </table>	8	4	4		opcode	R1	R2		2 bytes	1 προσπελάσεις
8	4	4									
opcode	R1	R2									
ADD R0, R1	<table border="1"> <tr> <td>8</td><td>4</td><td>4</td><td></td></tr> <tr> <td>opcode</td><td>R1</td><td>R2</td><td></td></tr> </table>	8	4	4		opcode	R1	R2		2 bytes	1 προσπελάσεις
8	4	4									
opcode	R1	R2									
LOAD R1, A	<table border="1"> <tr> <td>8</td><td>4</td><td>64</td><td>4</td></tr> <tr> <td>opcode</td><td>R1</td><td>A</td><td>X</td></tr> </table>	8	4	64	4	opcode	R1	A	X	10 bytes	9 προσπελάσεις
8	4	64	4								
opcode	R1	A	X								
ADD R0, R1	<table border="1"> <tr> <td>8</td><td>4</td><td>4</td><td></td></tr> <tr> <td>opcode</td><td>R0</td><td>R1</td><td></td></tr> </table>	8	4	4		opcode	R0	R1		2 bytes	1 προσπελάσεις
8	4	4									
opcode	R0	R1									
STORE Y, R0	<table border="1"> <tr> <td>8</td><td>64</td><td>4</td><td>4</td></tr> <tr> <td>opcode</td><td>Y</td><td>R0</td><td>X</td></tr> </table>	8	64	4	4	opcode	Y	R0	X	10 bytes	9 προσπελάσεις
8	64	4	4								
opcode	Y	R0	X								
A.1. Αριθμητική έκφραση που υπολογίζει το ανωτέρω πρόγραμμα:	A.2. Αρχιτεκτονική του επεξεργαστή:	Πλήθος ψηφιολέξεων που καταλαμβάνει το πρόγραμμα: <b>68 bytes</b>	Πλήθος προσπελάσεων για την εκτέλεση του προγράμματος: <b>58 προσπελάσεις</b>								

### A.1.

Αριθμητική έκφραση που υπολογίζει το πρόγραμμα:  $\mathbf{Y} = \mathbf{B}^* \mathbf{C} + \mathbf{D}^* \mathbf{E} + \mathbf{A}$ , διότι:

$R0 \leftarrow B$

$R1 \leftarrow C$

$R0 \leftarrow B * C$

$R1 \leftarrow D$

$R2 \leftarrow E$

$R1 \leftarrow D * E$

$R0 = B * C + D * E$

$R1 \leftarrow A$

$R0 \leftarrow B * C + D * E + A$

$\mathbf{Y} \leftarrow \mathbf{B}^* \mathbf{C} + \mathbf{D}^* \mathbf{E} + \mathbf{A}$

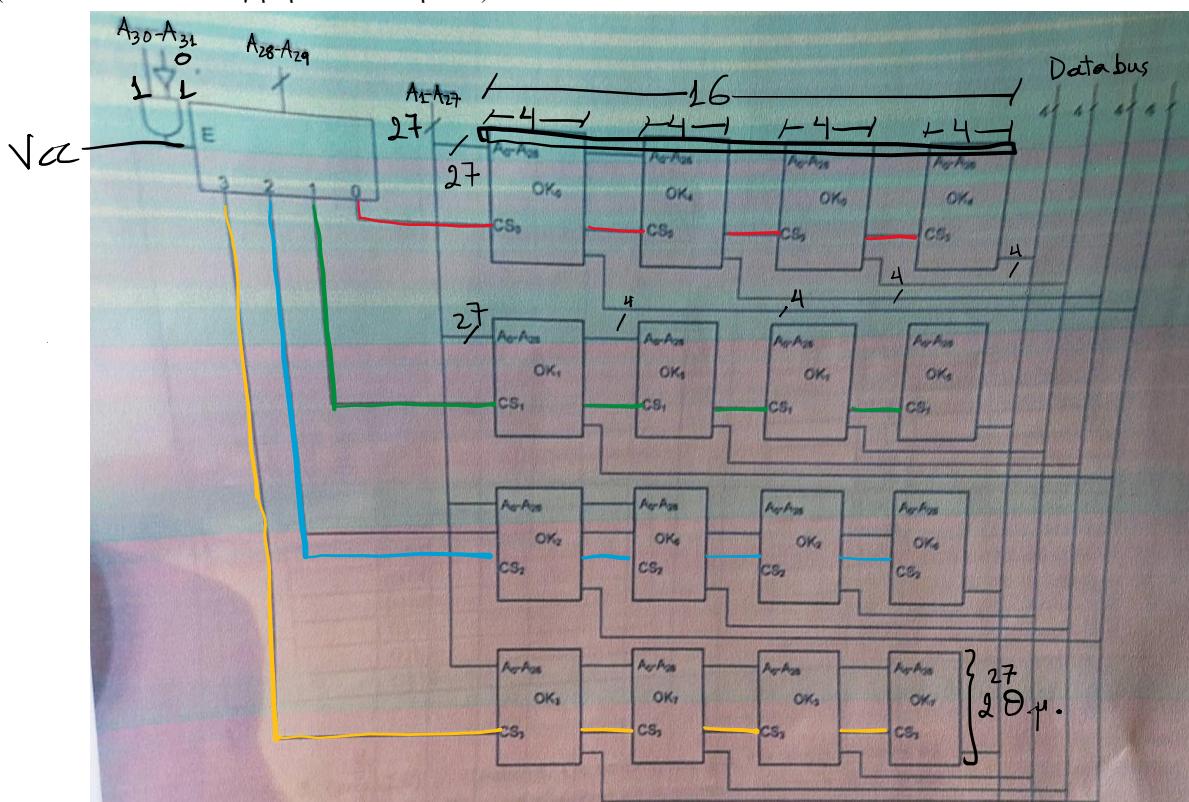
### A.2.

#### Αρχιτεκτονική καταχωρητή – καταχωρητή με 2 τελούμενα

B. Από το πλήθος των εντολών, υπολογίζουμε το μήκος του opcode, όταν αυτό δεν δίνεται από την εκφώνηση.  $2^x \geq 235 \Rightarrow x = 8 \text{ bits}$  για opcode και από το πλήθος των καταχωρητών (και όχι το μέγεθός τους) υπολογίζουμε το μήκος του κάθε καταχωρητή που εμφανίζεται ως έντελο:  $2^x \geq 16 \Rightarrow x = 4 \text{ bits}$  για καταχωρητή.

### Θέμα 7

Σ' ένα υπολογιστή ο επεξεργαστής έχει αρτηρία διευθύνσεων των 32 δυαδικών ψηφίων και αρτηρία δεδομένων των 16 δυαδικών ψηφίων. Δίνεται το ακόλουθο σχήμα μνήμης. Κάθε θέση του συστήματος μνήμης είναι των 8 δυαδικών ψηφίων. Να συμπληρώσετε τον ακόλουθο πίνακα (δεν απαιτείται αιτιολόγηση των απαντήσεων).



Πλήθος θέσεων μνήμης κάθε ολοκληρωμένου κυκλώματος $OK_i$	
Χωρητικότητα κάθε ολοκληρωμένου κυκλώματος σε ψηφιολέξεις	
πλήθος δυαδικών ψηφίων ανά θέση μνήμης κάθε ολοκληρωμένου κυκλώματος $OK_i$	
πλήθος δυαδικών ψηφίων που αντιστοιχεί σε κάθε διεύθυνση του συστήματος μνήμης	
πλήθος ψηφιολέξεων (bytes) που προσπελαύνεται ανά προσπέλαση του συστήματος μνήμης	
Οργάνωση του συστήματος μνήμης (διαφύλλωση)	
Χωρητικότητα του συστήματος μνήμης	
Η περιοχή διευθύνσεων που καλύπτεται από το σύστημα μνήμης (πρώτη και τελευταία διεύθυνση στο δεκαεξαδικό)	

### Λύση

Πλήθος θέσεων μνήμης κάθε ολοκληρωμένου κυκλώματος $OK_i$	$2^{27} \text{ θ.μ.} = 2^{7}2^{20} = 128 \text{ Μθέσεις}$
Χωρητικότητα κάθε ολοκληρωμένου κυκλώματος σε ψηφιολέξεις	$2^{26} \text{ ψηφιολέξεις}$
Πλήθος δυαδικών ψηφίων ανά θέση μνήμης κάθε ολοκληρωμένου κυκλώματος $OK_i$	4 bits



Πλήθος δυαδικών ψηφίων που αντιστοιχεί σε κάθε διεύθυνση του συστήματος μνήμης	16 bits
Πλήθος ψηφιολέξεων (bytes) που προσπελαύνεται ανά προσπέλαση του συστήματος μνήμης	2 ψηφιολέξεις
Οργάνωση του συστήματος μνήμης (διαφύλλωση)	Χαμηλής τάξης διαφύλλωση
Χωρητικότητα του συστήματος μνήμης	$2^{30}$ bytes = 1 GByte
Η περιοχή διευθύνσεων που καλύπτεται από το σύστημα μνήμης (πρώτη και τελευταία διεύθυνση στο δεκαεξαδικό)	40000000 έως 7FFFFFFF

1 θ.μ. 4 bits

2<sup>27</sup> θ.μ. x;

Έχουμε  $x = 4 \text{ bits} * 2^{27} = 2^2 * 2^7 * 2^{20} \text{ bits} = 2^{29} \text{ bits} = 2^{26} \text{ bytes} = 2^6 2^{20} \text{ bytes} = 64 \text{ MB}$

Για τη χωρητικότητα του συστήματος μνήμης έχουμε συνολικά 16 OKM \*  $\frac{2^{26} \text{ ψηφιολέξεις}}{\text{OKM}} = 2^4 * 2^{26} \text{ bytes} = 2^{30} \text{ bytes} = 1 \text{ GB.}$

Για περιοχή διευθύνσεων που καλύπτεται από το σύστημα μνήμης, πρέπει  $A_{31}A_{30} = 01$  για να είναι επιλεγμένος ο decoder, οπότε:

$A_3A_2A_1A_0$	$A_3A_2A_1A_0$
0100	0000 0000 0000 0000 0000 0000 0000 → 40000000
0111	1111 1111 1111 1111 1111 1111 1111 → 7FFFFFFF

## Θέμα 8

Θεωρήστε μια αρτηρία που χρησιμοποιείται για τη μεταφορά πληροφορίας 32 ψηφιολέξεων κάθε φορά από μια μονάδα εισόδου (υπηρέτης-slave) στον κύριο (master). Θεωρήστε ότι έχουμε πέντε διαφορετικές μονάδες εισόδου A, B, Γ, Δ και E. Η μεταφορά πληροφορίας στον κύριο από τη μονάδα εισόδου A, B, Γ, Δ και E απαιτεί 23, 12, 31, 38 και 34 nsec αντίστοιχα. Θεωρήστε επίσης ότι η διαιτησία βάζει καθυστέρηση 5 nsec. **Να υπολογίσετε το ρυθμό μεταφοράς πληροφορίας σε ψηφιολέξεις ανά δευτερόλεπτο από τη μονάδα E στον κύριο** για κάθε μια από τις ακόλουθες περιπτώσεις:

Η αρτηρία είναι **σύγχρονη** με περίοδο του σήματος χρονισμού (ρολογιού) **5 nsec**.

α) i) **χωρίς** δυνατότητα εισαγωγής κύκλων ρολογιού καθυστέρησης (wait states)

ii) με δυνατότητα εισαγωγής κύκλων ρολογιού καθυστέρησης

β) Η αρτηρία είναι ασύγχρονη απαιτεί και 11 nsec επιπλέον για κάθε μεταφορά λόγω του πρωτοκόλλου χειραγίας.

## Λύση

α) i) Σε αυτή την περίπτωση λαμβάνουμε υπόψη **την πιο αργή μονάδα** (όποια είναι αυτή) που στην προκειμένη περίπτωση είναι η Δ με τα 38 nsec που γίνεται 40 nsec (πολλαπλάσιο της διάρκειας του κύκλου ρολογιού) και στη συνέχεια προσθέτουμε σε αυτό και τα 5 nsec που είναι η καθυστέρηση της διαιτησίας, οπότε ο ρυθμός μεταφοράς πληροφορίας =  $\frac{\text{ποσότητα μεταφερόμενης πληροφορίας}}{\text{χρόνος μεταφοράς}} = \frac{32 \text{ bytes}}{45 \text{ nsec}} = \frac{32 \text{ bytes}}{45 \times 10^{-9} \text{ sec}} = \frac{32}{45} \times 10^9 \text{ bytes/sec.}$

ii) Σε αυτή την περίπτωση λαμβάνουμε υπόψη τη μονάδα E με τα 34 nsec που γίνεται 35 nsec (πολλαπλάσιο της διάρκειας του κύκλου ρολογιού) και στη συνέχεια προσθέτουμε σε αυτό και τα 5 nsec που είναι η καθυστέρηση της διαιτησίας, οπότε ο ρυθμός μεταφοράς πληροφορίας =  $\frac{32 \text{ bytes}}{40 \text{ nsec}} = \frac{32}{40} \times 10^9 \text{ bytes/sec.}$

**Σημείωση:** Το 34 έχει γίνει 35nsec + 5nsec (διαιτησία) = 40 nsec.

β) Σε αυτή την περίπτωση λαμβάνουμε υπόψη τη μονάδα E με τα 34 nsec που διατηρούνται και στη συνέχεια προσθέτουμε σε αυτό και τα 5 nsec που είναι η καθυστέρηση της διαιτησίας, οπότε ο ρυθμός μεταφοράς πληροφορίας =  $\frac{32 \text{ bytes}}{(34 + 11 + 5) \text{nsec}} = \frac{32 \text{ bytes}}{50 \text{nsec}} = \frac{32}{50} \times 10^9 \text{ bytes/sec.}$

## Θέμα 9

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης OK0 με χωρητικότητα 160 Mbits και 15 δυαδικά ψηφία ανά θέση μνήμης, καθώς και ολοκληρωμένα κυκλώματα μνήμης OK1 με χωρητικότητα 1.25 Mbytes και 5 δυαδικά ψηφία ανά θέση μνήμης.

Πόσα ολοκληρωμένα κυκλώματα τύπου OK0 και πόσα τύπου OK1 χρειάζεστε για να σχεδιάσετε ένα σύστημα μνήμης με χωρητικότητα 160 Mbits και 15 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 OK0 και 12 τύπου OK1. Προσοχή μη βάλετε κενά μεταξύ των συμβόλων.

## Λύση

## OK0

$$1 \text{ θ.μ.} \quad 10 \text{ bits}$$

$$x; \quad 80 \text{ Mbits}$$

$$x = 80 \times 2^{20} / 10 = 2^{23} \text{ θ.μ.} =$$

## OK1

$$1 \text{ θ.μ.} \quad 5 \text{ bits}$$

$$x \quad 1.25 \text{ Mbytes}$$

$$x = 1.25 \times 2^{20} \times 2^3 / 5 = 0.25 \times 2^{20} \times 2^3$$

## Κύρια μνήμη

$$1 \text{ θ.μ.} \quad 15 \text{ bits}$$

$$x \quad 160 \text{ Mbits}$$

$$x = 160 \times 2^{20} / 15 = 10.66 \times 2^{20}$$

Οριζόντια επανάληψη: 2 (8 bits + 8 bits)

$$\text{Κατακόρυφη επανάληψη για OK0: } \frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{160 / 15 \times 2^{20}}{2^{23} \text{ θ.μ.}} = 160 / 15 \times 2^{-3} = \frac{160}{15 \times 8} = \frac{160}{120} = \frac{4}{3} = 1.33$$

$$\text{Κατακόρυφη επανάληψη για OK1: } \frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{160 / 15 \times 2^{20}}{2^{21} \text{ θ.μ.}} = \frac{160}{15 \times 2} = 5.33$$

Κάθε **OK0** αφού περιέχει  $2^{23}$  θ.μ., έχει  $2^3 \times 2^{20} = 8$  Μθέσεις των 10 bits η καθεμία.

Κάθε **OK1** αφού περιέχει  $2^{21}$  θ.μ., έχει  $2^1 \times 2^{20} = 2$  Μθέσεις των 5 bits η καθεμία.

Η **κύρια μνήμη** που θα σχεδιάσουμε έχει  $160 / 15 \times 2^{20} = 10.66$  Μθέσεις των 15 bits η καθεμία

Επειδή η K.M. δεν έχει ακέραιο αριθμό θ.μ. θα χρησιμοποιήσουμε τις χωρητικότητες των OKM για να υπολογίσουμε ποσα θα χρειαστούμε. Το OK0 περιέχει 80 Mbits και το OK1 περιέχει 1.25 MB =  $1.25 \times 8$  Mbits = 10 Mbits. Θα χρησιμοποιήσουμε 1 OK0 + 8 OK1 =  $1 \times 80$  Mbits +  $8 \times 80$  Mbits = 160 Mbits.

## Θέμα 10

Δίνονται οι αριθμοί  $A = 01000010010000000000000000000000$  και  $B = 11000011000100000000000000000000$  σε αναπαράσταση κινητής υποδιαστολής σύμφωνα με το πρότυπο IEEE 754.

Να υπολογίσετε και να δώσετε το άθροισμά τους σε αναπαράσταση κινητής υποδιαστολής απλής ακρίβειας σύμφωνα με το ίδιο πρότυπο.

### Αύση

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 = 132 και για τον αριθμό B, ο E = 134. Επειδή και για τους δύο αριθμούς ο  $0 < E < 255$ , ισχύει:  $N = (-1)^{\pi} * 2^{E-\text{πόλωση}} * 1.\Sigma_{\kappa}$  οπότε για τον αριθμό A έχουμε ότι είναι:  $N = (-1)^0 * 2^{132-127} * 1.5 = 2^5 * 1.5 = 48$ , όπου  $\Sigma_{\kappa} = 2^{-1} = 0.5$  ενώ για τον αριθμό B έχουμε ότι είναι:  $N = (-1)^0 * 2^{134-127} * 1.125 = 2^7 * 1.125 = 144$ , όπου  $\Sigma_{\kappa} = 1/8 = 0.125$ . Το άθροισμά τους σε παράσταση σταθερής υποδιαστολής είναι  $48 + (-144) = -96$ . Στη συνέχεια, πρέπει να μετατρέψουμε το άθροισμα αυτό πίσω σε αναπαράσταση κινητής υποδιαστολής, σύμφωνα με το πρότυπο IEEE 754. Θα πάρουμε το θετικό  $96 = 0110\ 0000.0 \times 2^0 = 1.100000 \times 2^6$ . Τώρα η πόλωση (127) πρέπει να **προστεθεί** στον εκθέτη, ο οποίος θα γίνει  $6 + 127 = 133 = 1000\ 0101$ . Ο συντελεστής Σκ είναι το κλασματικό μέρος του αριθμού που συμπληρώνεται στο τέλος με «0» για να φτάσει τα 23 bits. Το πρόσημο είναι «1», διότι ο αριθμός είναι αρνητικός.

**1 1000 0101 10000000000000000000000000000000**

**Σημείωση:** όταν ένας αριθμός μετατρέπεται από σταθερή σε κινητή υποδιαστολή, πρέπει πρώτα να **κανονικοποιηθεί**, δηλαδή να μετατραπεί στη μορφή **1.Σκπόλωση**. Όσες θέσεις μετακινείται προς τα **αριστερά/δεξιά** η υποδιαστολή, τόσες τιμές αυξάνεται/μειώνεται αντίστοιχα ο εκθέτης.

### Θέμα 11

Έχετε στη διάθεση σας ολοκληρωμένα κυκλώματα μνήμης OK0 με χωρητικότητα 4 MBytes και 8 δυαδικά ψηφία ανά θέση μνήμης, καθώς και ολοκληρωμένα κυκλώματα μνήμης OK1 με χωρητικότητα 8 Mbits και 8 δυαδικά ψηφία ανά θέση μνήμης.

Πόσα ολοκληρωμένα κυκλώματα τύπου OK0 και πόσα τύπου OK1 χρειάζεστε για να σχεδιάσετε ένα σύστημα μνήμης με χωρητικότητα 22 Mbytes 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 OK0 και 12 τύπου OK1. Προσοχή μη βάλετε κενά μεταξύ των συμβόλων.

### Λύση

#### OK0

1 θ.μ. 8 bits

x; 4 Mbytes

$$x = 2^2 \times 2^{20} \times 2^3 / 2^3 = 2^{22} \text{ θ.μ.}$$

#### OK2

1 θ.μ. 8 bits

x 8 Mbits

$$x = 2^3 \times 2^{20} / 2^3 = 2^{20} \text{ θ.μ.}$$

#### Κύρια μνήμη

1 θ.μ. 16 bits

x 22 MB

$$x = 22 \times 2^{20} \times 2^3 / 2^4 = 11 \times 2^{20} \text{ θ.μ.}$$

Οριζόντια επανάληψη: 2 (8 bits + 8 bits)

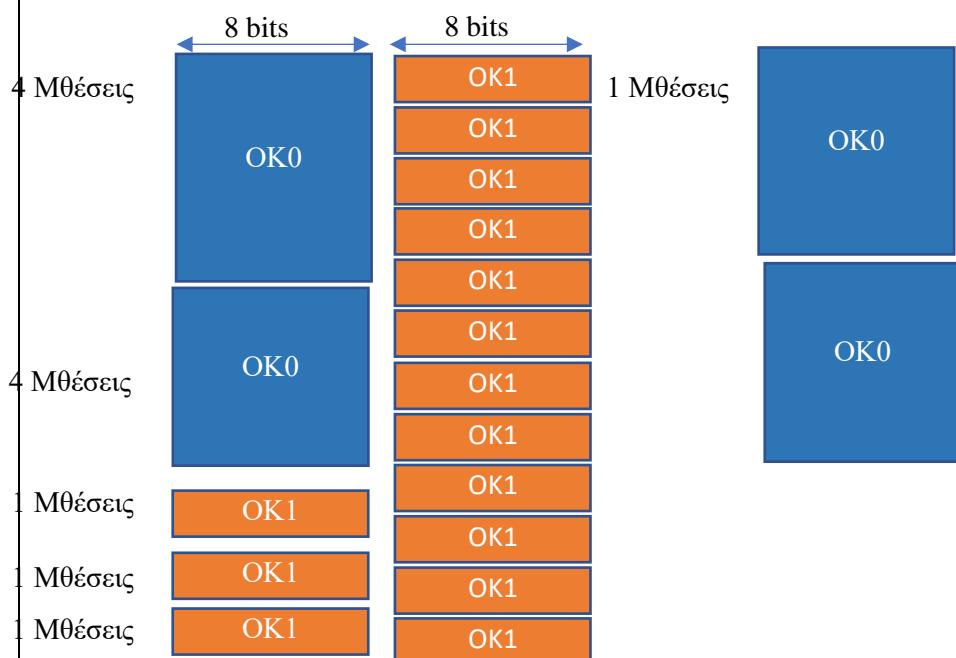
$$\text{Κατακόρυφη επανάληψη για OK0: } \frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{11 \times 2^{20}}{2^{22} \text{ θ.μ.}} = 11 \times 2^{-2} = 2.75 \text{ επαναλήψεις}$$

$$\text{Κατακόρυφη επανάληψη για OK1: } \frac{\text{θ.μ. που θέλουμε να έχουμε}}{\text{θ.μ. που διαθέτουμε από τα OKM}} = \frac{11 \times 2^{20}}{2^{20} \text{ θ.μ.}} = 11 \text{ επαναλήψεις}$$

Κάθε OK0 αφού περιέχει  $2^{22}$  θ.μ., έχει  $2^2 \times 2^{20} = 4$  Μθέσεις των 8 bits η καθεμία.

Κάθε OK1 αφού περιέχει  $2^{20}$  θ.μ., έχει 1 Μθέσεις των 8 bits η καθεμία.

Η κύρια μνήμη που θα σχεδιάσουμε έχει  $11 \times 2^{20} = 11$  Μθέσεις των 16 bits η καθεμία



Επομένως χρειαζόμαστε 4 OK0 + 6 OK1 → 04+06.

### Θέματα Ιανουαρίου 2022

#### Θέμα 1

Θεωρήστε μια αρτηρία που χρησιμοποιείται για τη μεταφορά πληροφορίας 16 ψηφιολέξεων κάθε φορά από μια μονάδα εισόδου (υπηρέτης – slave) στον κύριο (master). Θεωρήστε ότι έχουμε τέσσερις διαφορετικές μονάδες εισόδου A, B, Γ, και Δ αντίστοιχα.

Η μεταφορά πληροφορίας στον κύριο από τη μονάδα εισόδου Α, Β, Γ και Δ απαιτεί 13, 25, 31 και 45 nsec αντίστοιχα. Θεωρήστε επίσης ότι η διαιτησία βάζει καθυστέρηση 4 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) 4 nsec.

i. χωρίς δυνατότητα εισαγωγής κύκλων ρολογιού καθυστέρησης (wait states)

ii. με δυνατότητα εισαγωγής κύκλων ρολογιού καθυστέρησης

β. Η αρτηρία είναι ασύγχρονη απαιτεί και 10 nsec επιπλέον για κάθε μεταφορά λόγω του πρωτοκόλλου χειραψίας.

## Θέμα 2

Πίνακας 1	
εντολές	σχόλια
LOADSB R1, A	Φόρτωσε προσημασμένη ψηφιολέξη
LOADSHW R1, A	Φόρτωσε προσημασμένη μισή λέξη
LOADW R1, A	Φόρτωσε λέξη
LOADUB R1, A	Φόρτωσε απρόσημη ψηφιολέξη
LOADUHW R1, A	Φόρτωσε απρόσημη μισή λέξη
STOREB A, R1	αποθήκευσε ψηφιολέξη
STOREHW A, R1	αποθήκευση μισή λέξη
STOREW A, R1	αποθήκευσε λέξη

**Προσοχή:** σε όλες τις ανωτέρω εντολές χρησιμοποιείται ο κατ' ευθείαν τρόπος διεύθυνσιοδότησης (direct addressing mode)

Στον Πίνακα 2 δίνονται τα περιεχόμενα τμήματος της κύριας μνήμης και στον Πίνακα 3 τα περιεχόμενα των καταχωρητών πριν την εκτέλεση των εντολών του Πίνακα 4.

Θεωρήστε ότι χρησιμοποιείται απεικόνιση **ανάλογη του μεγέθους (little 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

Πίνακας 2		
διεύθυνση θέσης μνήμης στο δεκαεξαδικό πριν την εκτέλεση των εντολών	περιεχόμενο θέσης μνήμης στο δεκαεξαδικό μετά την εκτέλεση των εντολών	περιεχόμενο θέσης μνήμης στο δεκαεξαδικό μετά την εκτέλεση των εντολών
110	11	11
111	00	00
112	F1	F1
113	E3	E3
114	E4	E4
115	00	00
116	AB	AB
117	F0	F0
118	01	01
119	B6	B6
11A	05	05
11B	32	56
11C	AE	24
11D	BC	00
11E	10	AE
11F	C4	23
120	89	6A
121	1A	18
122	A0	A0

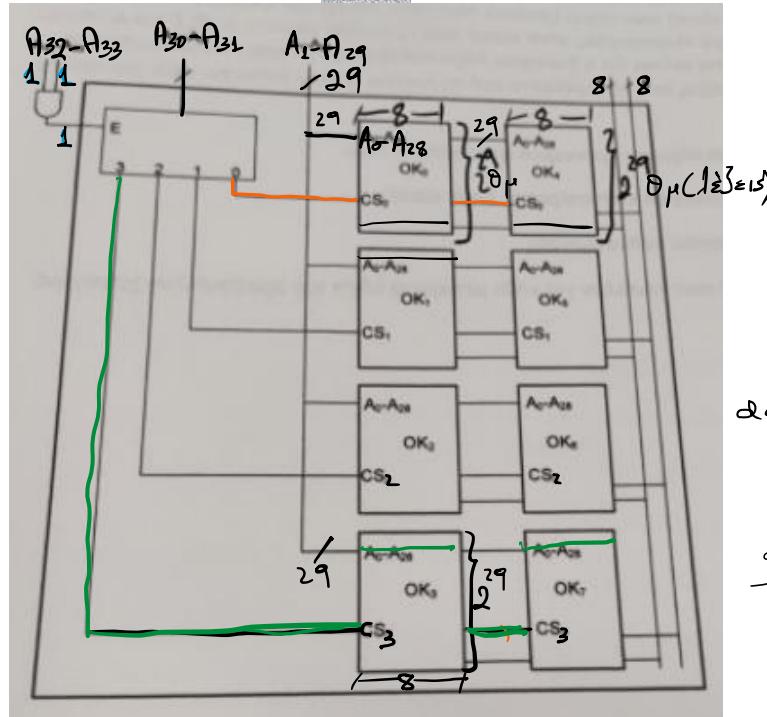
Να συμπληρώσετε στον Πίνακα 2 το περιεχόμενο των θέσεων μνήμης και στον Πίνακα 3 το περιεχόμενο των καταχωρητών μετά την εκτέλεση των εντολών του Πίνακα 4.

Πίνακας 3		
Καταχωρητής	Περιεχόμενο του καταχωρητή στο δεκαεξαδικό πριν την εκτέλεση της εντολής	Περιεχόμενο του καταχωρητή στο δεκαεξαδικό μετά την εκτέλεση της εντολής
R1=	0F 0F 0F 0F	00000011
R2=	0F 0F 0F 0F	FFFFF0AB
R3=	0F 0F 0F 0F	E4E3F100
R4=	0F 0F 0F 0F	000000A0
R5=	0F 0F 0F 0F	000005B6
R6=	00 FF FF 18	00 FF FF 18
R7=	FF 09 24 56	FF 09 24 56
R8=	6A 23 AE 00	6A 23 AE 00
R9=	0F 0F 0F 0F	FFFFFFFFFF

Πίνακας 4
εντολή
LOADSB R1, 110 <sub>(16)</sub>
LOADSHW R2, 116 <sub>(16)</sub>
LOADW R3, 111 <sub>(16)</sub>
LOADUB R4, 122 <sub>(16)</sub>
LOADUHW R5, 119 <sub>(16)</sub>
STOREB 121 <sub>(16)</sub> , R6
STOREHW 11B <sub>(16)</sub> , R7
STOREW 11D <sub>(16)</sub> , R8
LOADSB R9, 112 <sub>(16)</sub>

## Θέμα 3

Σ' ένα υπολογιστή ο επεξεργαστής έχει αρτηρία διεύθυνσεων των 34 δυαδικών ψηφίων και αρτηρία δεδομένων των 16 δυαδικών ψηφίων. Δίνεται το ακόλουθο σύστημα μνήμης. Κάθε θέση του συστήματος μνήμης έχεις εύρος μιας ψηφιολέξης. Να συμπληρώσετε τον ακόλουθο πίνακα (δεν απαιτείται αιτιολόγηση των απαντήσεων).



### Λύση

πλήθος θέσεων μνήμης κάθε ολοκληρωμένου κυκλώματος $OK_i$	$2^{29} \Theta\mu$
πλήθος δυαδικών ψηφίων ανά θέση μνήμης κάθε ολοκληρωμένου κυκλώματος $OK_i$	8
πλήθος δυαδικών ψηφίων που αντιστοιχεί σε κάθε διεύθυνση του συστήματος μνήμης	16
πλήθος ψηφιολέξεων που προσπελαύνεται ανά προσπέλαση του συστήματος μνήμης	2
Οργάνωση του συστήματος μνήμης (διαφύλλωση)	Χαμηλής τάξης διαφύλλωση
Χωρητικότητα του συστήματος μνήμης	4 GByte
Η περιοχή διευθύνσεων που καλύπτεται από το σύστημα μνήμης (πρώτη και τελευταία διεύθυνση στο δεκαεξαδικό)	300000000 έως 3FFFFFFF

Κάθε  $OK$  έχει  $2^{29} \Theta\mu$ .  $x \frac{8 \text{ bits}}{\Theta\mu} = 2^{29} \text{ bytes} = 2^9 2^{20} \text{ bytes} = 512 \text{ MB}$  και συνολικά υπάρχουν 8  $OK$ , οπότε η χωρητικότητα του συστήματος μνήμης είναι  $8 \times 512 \text{ MB} = 4096 \text{ MB} = 2^{12} \text{ MB} = 2^{12} 2^{20} \text{ bytes} = 2^{32} \text{ bytes} = 2^2 2^{30} \text{ bytes} = 4 \text{ GB}$ .

Για περιοχή διευθύνσεων που καλύπτεται από το σύστημα μνήμης, πρέπει  $A_{33}A_{32} = 11$  για να είναι επιλεγμένος ο decoder, οπότε:

$A_3A_2A_1A_0$	$A_3A_2A_1A_0$
11 0000   0000   0000   0000   0000   0000   000X → 300000000	
11 1111   1111   1111   1111   1111   1111   111X → 3FFFFFFF	

**Σημείωση 1:** η οργάνωση χαμηλής τάξης διαφύλλωση μνήμης διακρίνεται στη διαφορετικότητα της αρίθμησης των bits του address bus που εισέρχονται σε κάθε  $OK$  ( $A_1 - A_{29}$ ) και σε αυτά που περιέχονται σε κάθε ολοκληρωμένο ( $A_0 - A_{28}$ )

**Σημείωση 2:** αν η εκφώνηση ζητούσε όλο το χάρτη διευθύνσεων του δοθέντος συστήματος μνήμης, αυτός θα ήταν ο εξής:

OK	Πλήθος bits διεύθυνσης	Αρτηρία διεύθυνσεων
$OK_0-OK_4$	29	$A_3A_2A_1A_0$
$OK_0-OK_4$	29	από 11 00 000000000000..... 000X
$OK_0-OK_4$	29	έως 11 00 111111111111..... 111X
$OK_1-OK_5$	29	από 11 01 000000000000..... .000X
$OK_1-OK_5$	29	έως 11 01 111111111111..... 111X
$OK_2-OK_6$	29	από 11 10 000000000000..... ....000X
$OK_2-OK_6$	29	έως 11 10 111111111111..... 111X
$OK_3-OK_7$	29	από 11 11 000000000000..... .000X
$OK_3-OK_7$	29	έως 11 11 111111111111..... 111X

### Θέμα 4

A. Για τον υπολογισμό της έκφρασης " $Y = Ax(B+C)+DxE$ " να γραφεί πρόγραμμα για υπολογιστή που βασίζεται στη χρήση 8 καταχωριτών γενικού σκοπού των 32 δυαδικών ψηφίων ο κάθε ένας με εντολές μόνο καταχωρητή-καταχωρητή (ή φόρτωσης – αποθήκευσης) με δύο τελούμενα.

Β. Το σύνολο των εντολών σε επίπεδο γλώσσας μηχανής αποτελείται από 600 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 32 δυαδικά ψηφία. Για κάθε διαφορετική εντολή της συμβολικής γλώσσας που θα χρησιμοποιήσετε, να σχεδιάσετε τη μορφή της εντολής σε επίπεδο γλώσσας μηχανής (δηλαδή τα πεδία από τα οποία αποτελείται η εντολή σε επίπεδο γλώσσας μηχανής και το εύρος κάθε πεδίου).

Γ. Να δώσετε το πλήθος των ψηφιολέξεων που καταλαμβάνει το πρόγραμμα στην κύρια μνήμη.

Δ. Θεωρήστε ότι τα δεδομένα είναι μίας λέξης (32 δυαδικά ψηφία), η αρτηρία δεδομένων μεταξύ της Κεντρικής Μονάδας Επεξεργασίας (KME) και του συστήματος μνήμης έχει εύρος 1 ψηφιολέξης (byte) και σε κάθε προσπέλαση του συστήματος μνήμης προσπελάζεται μία ψηφιολέξη. Πόσες προσπελάσεις στη μνήμη απαιτεί η εκτέλεση του προγράμματος;

## Θέματα Σεπτεμβρίου 2023

### Θέμα 1

Θεωρήστε μια αρτηρία που χρησιμοποιείται για τη μεταφορά πληροφορίας 16 ψηφιολέξεων κάθε φορά από μια μονάδα εισόδου

<p><b>B</b></p> <p>Θέμα 1. (μον. 2.4) Θεωρήστε τα ακόλουθα περιεχόμενα των καταχωρητών:</p> <p>r1: 00010101, r2: 10011010, r3: 11010101, r4: 11011010, r5: 10000000, r6: 10000000,  r7: 01010101, r8: 01110101, r9: 00010101, r10: 10011010, r11: 10000000, r12: 10000000.</p> <p>Να εκτελέσετε κάθε μια αριθμό των σημαίων καθώς και να αιτιολογήσετε στον πίνακα το αποτέλεσμα, τις τιμές των σημαίων για κάθε μια τιμή των δύο περιπτώσεων: α. τα περιεχόμενα των βασιζόμενοι στις τιμές των σημαίων για κάθε μια τιμή της οποίας βιασίζετε το συμπλέρωμά σας, καταχωρητών είναι χωρίς 2. Η αιτιολογία να έχει την μορφή <math>X=0 \rightarrow</math> σωστό ή λάθος αποτέλεσμα ή <math>X=1 \rightarrow</math> συμπληρώματος ως προς 2. Η αιτιολογία να έχει την μορφή <math>X=0 \rightarrow</math> σωστό ή λάθος αποτέλεσμα ή <math>X=1 \rightarrow</math> σωστό ή λάθος αποτέλεσμα, όπου X είναι η σημαία στην τιμή της οποίας βιασίζετε το συμπλέρωμά σας. Τόσο για την πρόσθεση όσο και την αφαίρεση μεταξύ αριθμών χωρίς πρόσημο ή μεταξύ αριθμών σε αναπαράσταση συμπληρώματος ως προς 2 χρησιμοποιείται η ίδια μονάδα αθροιστή των 8 δυαδικών ψηφίων.</p> <p>ADD r1, r2 σημαίνει <math>r1 \leftarrow r1+r2</math> και SUB r1, r2 σημαίνει <math>r1 \leftarrow r1-r2</math></p> <p><b>Απάντηση</b></p>			
εντολή	αποτέλεσμα	C V Z	αιτιολογία σωστού ή λάθους αποτελέσματος
ADD r1, r2			αριθμοί χωρίς πρόσημο
ADD r3, r4			Αριθμοί σε αναπαράσταση συμπληρώματος ως προς 2
ADD r5, r6			
ADD r7, r8			
SUB r9, r10			
SUB r11, r12			

### Θέμα 2

**Θέμα 2. (μον. 2)** Θεωρήστε επεξεργαστή με καταχωρητές γενικού σκοπού των 32 διαδικών ψηφίων. Θεωρήστε, επίσης, ότι ο επεξεργαστής μεταξύ των άλλων εντολών υποστηρίζει και τις εντολές του πίνακα που ακολουθεί.

LOADW R1, \$XXXX	\$ XXXX	μεταφορά λέξης (32 bits), κατευθείαν τρόπος διεύθυνσης διεύθυνσης, X: δεκαεξαδικό ψηφίο
LOADSB R2, \$XXXX	\$ XXXX	μεταφορά λέξης (32 bits), με αναπαρόσταση συμπλ. πρόματος ως προς 2, κατευθείαν τρόπος διεύθυνσης διεύθυνσης, X: δεκαεξαδικό ψηφίο
LOADUB R3, \$XXXX	\$ XXXX	μεταφορά λέξης χωρίς πρόσημο, κατευθείαν τρόπος διεύθυνσης διεύθυνσης, X: δεκαεξαδικό ψηφίο
LOADSHW R4, \$XXXX	\$ XXXX	μεταφορά μισής λέξης με αναπαρόσταση συμπλ. πρόματος ως προς 2, κατευθείαν τρόπος διεύθυνσης διεύθυνσης, X: δεκαεξαδικό ψηφίο
STOREHW R5, (R6)	(R6)	μεταφορά μισής λέξης, έμμεσος τρόπος διεύθυνσης (indirect addressing mode) με χρήση καταχωρητή.

Θεωρήστε τα περιεχόμενα των καταχωρητών και της κύριας μνήμης που δίνονται στο ακόλουθο σχήμα.

R1:	00FEDCBA
R2:	0F0F0FOF
R3:	F0F0F0FO
R4:	01020304
R5:	00000303
R6:	0F0F0FOF

διεύθυνση μνήμη	περιεχόμενο
00000300	56
00000301	A3
00000302	56
00000303	90
00000304	33
00000305	FE
00000306	45
00000307	ED
00000308	15
00000309	F4
0000030A	9B
0000030B	84
0000030C	FD
0000030D	23
0000030E	99
0000030F	3D
00000310	1D

Για καθεμία των κάτωθι περιπτώσεων να δώσετε στο δεκαεξαδικό τα περιεχόμενα των καταχωρητών και της μνήμης μετά την εκτέλεση των εντολών του πίνακα που ακολουθεί.

- Απεικόνιση ανάλογη του μεγέθους (little endian)
- Απεικόνιση αντιστρόφως ανάλογη του μεγέθους big endian

R1:		R1:	
R2:		R2:	
R3:		R3:	
R4:		R4:	
R5:		R5:	
R6:		R6:	

διεύθυνση μνήμη	περιεχόμενο	διεύθυνση μνήμη	περιεχόμενο
00000300		00000300	
00000301		00000301	
00000302		00000302	
00000303		00000303	
00000304		00000304	
00000305		00000305	
00000306		00000306	
00000307		00000307	
00000308		00000308	
00000309		00000309	
0000030A		0000030A	
0000030B		0000030B	
0000030C		0000030C	
0000030D		0000030D	
0000030E		0000030E	
0000030F		0000030F	
00000310		00000310	

STOREHW R4, (R5) // (R5) ← R4 αποθήκευση του περιεχομένου του καταχωρητή R4 στη θ.μ. με διεύθυνση αυτή που δείχνει ο R5

LOADSHW R1, \$00000308

LOADSB R2, \$0000030A

LOADUB R3, \$0000030C

LOADSB R6, \$00000310

### Λύση

Αρχικό περιεχόμενο καταχωρητών	
R1	00FEDCBA
R2	0F0F0FOF
R3	F0F0F0FO
R4	01020304
R5	00000303
R6	0F0F0FOF

**Εντολές**  
 STOREHW R4, (R5) // (R5)  
 LOADSHW R1, \$00000308  
 LOADSB R2, \$0000030A  
 LOADUB R3, \$0000030C  
 LOADSB R6, \$00000310

BigEndian (αντιστρόφως ανάλογη του μεγέθους)	
R1	000015F4
R2	FFFFFFFFFF9B
R3	000000FD
R4	01020304

LittleEndian (ανάλογη του μεγέθους)	
R1	FFFFF415
R2	FFFFFFFFFF9B
R3	000000FD
R4	01020304
R5	00000303
R6	0000001D

R5	00000303	
R6	0000001D	

**BigEndian MNHMH**

Διεύθυνση μνήμης	Αρχικό Περιεχόμενο	Νέο Περιεχόμενο
00000300	56	56
00000301	A3	A3
00000302	56	56
00000303	90	03
00000304	33	04
00000305	FE	FE
00000306	45	45
00000307	ED	ED
00000308	15	15
00000309	F4	F4
0000030A	9B	9B
0000030B	84	84
0000030C	FD	FD
0000030D	23	23
0000030E	99	99
0000030F	3D	3D
00000310	1D	1D

**LittleEndian MNHMH**

Διεύθυνση μνήμης	Αρχικό Περιεχόμενο	Νέο Περιεχόμενο
00000300	56	56
00000301	A3	A3
00000302	56	56
00000303	90	04
00000304	33	03
00000305	FE	FE
00000306	45	45
00000307	ED	ED
00000308	15	15
00000309	F4	F4
0000030A	9B	9B
0000030B	84	84
0000030C	FD	FD
0000030D	23	23
0000030E	99	99
0000030F	3D	3D
00000310	1D	1D

## Ογκοδίαι Β

Σεμα 5 (μον. 2)

Έχετε στη διάθεσή σας ολοκληρωμένα κυκλώματα μνήμης με χωρητικότητα 2 MBytes το καθένα με 4 διαδικα ψηφία ανά θέση μνήμης και επιπλέον ένα οποιοδήποτε κύκλωμα και δύο οποιεσδήποτε πόλες.

a. Να σχεδιάσετε μονάδα μνήμης με χωρητικότητα 16 Mbytes με οργάνωση ενός byte ανά θέση μνήμης και είσοδο ενεργοποίησης.

a.1. Να δώσετε το χάρτη διευθύνσεων της μονάδας μνήμης.

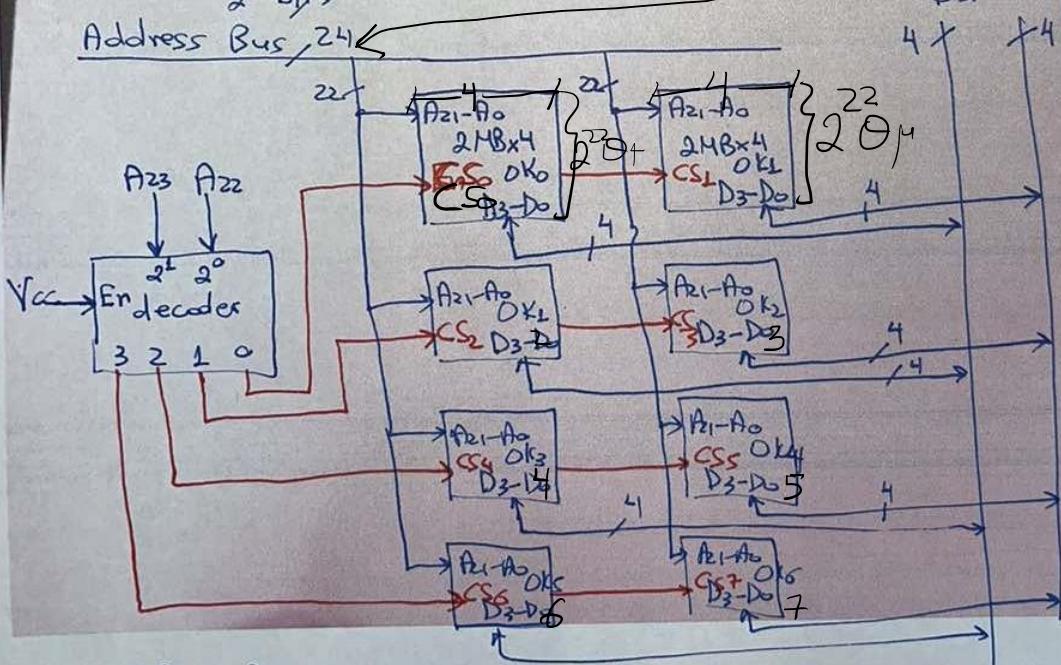
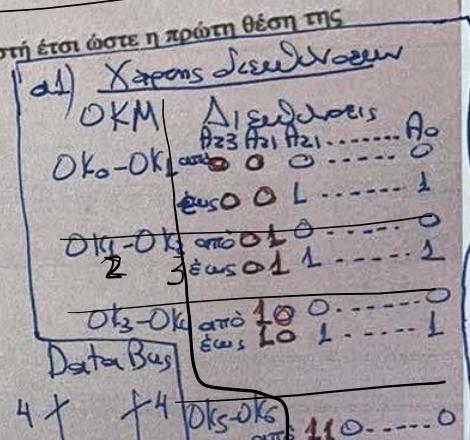
a.2. Να δώσετε ένα σχήμα στο οποίο να φαίνεται όλη η πληροφορία.

β. Σας δίνεται επεξεργαστής με 26 γραμμές διευθύνσεων.

β.1. Πόσες μονάδες μνήμης, από αυτή που σχεδιάσατε, χρειάζονται για να καλύψετε όλο το χώρο φυσικών διευθύνσεων του επεξεργαστή;

β.2 Να συνδέσετε τη μονάδα μνήμης, την οποία σχεδιάσατε, στον επεξεργαστή έτσι ώστε η πρώτη θέση της μονάδας μνήμης να έχει τη δεκαεξαδική διεύθυνση 2000000.

$$\begin{aligned}
 & \text{Απάντηση} \\
 & \text{Εγκάριση OKM } 2 \text{MB} \times 4 \rightarrow \text{Συστοιχία KM } 16 \text{MB} \times 8 \\
 & 1 \text{ Θέση } 4 \text{ bits} \\
 & X; 2 \text{MB} \\
 & X = \frac{2^4 \cdot 2^{20}}{2^2 \text{ bits}} = 2^{22} \text{ Θέση} \\
 & X = \frac{2^4 \cdot 2^{20}}{2} = 2^{24} \text{ Θέση}
 \end{aligned}$$



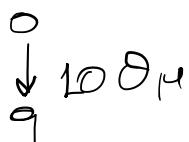
$$\beta) \frac{2^{26}}{2^{21}} = 2^2 = 4 \text{ μονάδες μνήμης θα χρειασούν για τα καθύσταρα σύλλογα}$$

β2) Αρχική διεύθυνση: 2000000H

Η K.M. του α' ερωτήματος καταλαμβάνει  $2^{24}$  Θ.μ. = 1 0000 0000 0000 0000 0000 0000 = 1000000H

Τελική διεύθυνση: 2000000H + 1000000H = 3000000H

Στη συνέχεια γράφουμε την αρχική και την τελική διεύθυνση από την οποία **αφαιρείται** μια θέση μνήμης, διότι στη συνολική μέτρηση λαμβάνεται υπόψη και η **αρχική** διεύθυνση μνήμης (2000000H) και γράφοντας ως τελική διεύθυνση την 2FFFFFH, βρίσκουμε από **αριστερά** τα **σταθερά bits σε όλο το εύρος των διευθύνσεων**. Είναι αυτά που θα χρησιμοποιηθούν για την επιλογή της K.M. Επειδή έχουμε 26 γραμμές διευθύνσεων στον επεξεργαστή, και η διεύθυνση που δίνεται (2000000H) είναι των 28 bits, θα διαγράψουμε τα 2 αριστερότερα bits της διεύθυνσης και θα χρησιμοποιήσουμε για την επιλογή της K.M. τα bits A<sub>25</sub>A<sub>24</sub>, όπως φαίνεται στη συνέχεια. Το En του decoder θεωρείται θετικής λογικής.



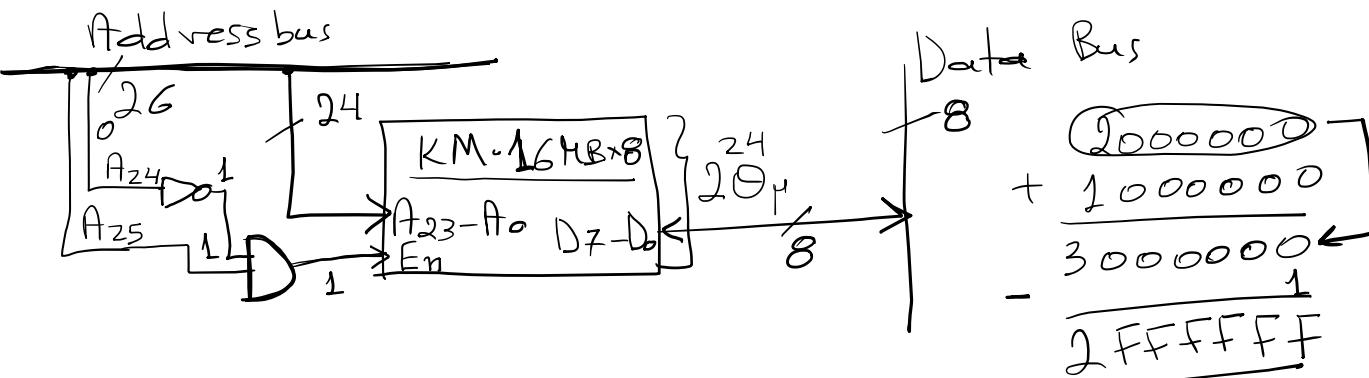
A<sub>27</sub>A<sub>26</sub>A<sub>25</sub>A<sub>24</sub>A<sub>23</sub>

A<sub>1</sub>A<sub>0</sub>

Αρχική διεύθυνση: 2000000H = 0010 0000 0000 0000 0000 0000 0000

Τελική διεύθυνση: 2FFFFFH = 0010 1111 1111 1111 1111 1111 1111

$$16MB = 2^4 \cdot 2^{20} \text{ bytes} = 2^{24} \Theta_{\mu} = 1000000$$



**Θέμα 4.** (μον. 1,6) Θεωρήστε μια αρτηρία που χρησιμοποιείται για τη μεταφορά πλ. προφορίας 6 νηφιολέξεων κάθε φορά από μια μονάδα εισόδου (υπηρέτης-slave) στον κύριο (master). Θεωρήστε ότι έχουμε πέντε διαφορετικές μονάδες εισόδου A, B, Γ, Δ και E. Η μεταφορά πλ. προφορίας στον κύριο από τη μονάδα εισόδου A, B, Γ, Δ και E απαιτεί 13, 52, 41, 35 και 29 nsec αντίστοιχα. Θεωρήστε επίσης ότι η διατησία βάζει τη μονάδα E στον κύριο για κάθε μια από τις ακόλουθες περιπτώσεις.

- a. Η αρτηρία είναι σύγχρονη με περίοδο του σήματος χρονισμού (ρολογιού) 3 nsec
- i. χωρίς δυνατότητα εισαγωγής κύκλων ρολογιού καθυστέρησης (wait states)
  - ii. με δυνατότητα εισαγωγής κύκλων ρολογιού καθυστέρησης

- β. Η αρτηρία είναι ασύγχρονη και απαιτεί 12 nsec επιπλέον για κάθε μεταφορά λόγω του πρωτοκόλλου χειραγίας.

Απάντηση

a) i) Ρυθμός μεταφορών =  $\frac{6 \text{ bytes}}{(54+3) \text{ nsec}} = \frac{6 \text{ bytes}}{57 \text{ nsec}}$

Η πιο αργή μονάδα είναι 52 nsec γιατί είναι 54 nsec (ανάφεση πολλαπλού του cycle)

ii) Ρυθμός μεταφορών =  $\frac{6 \text{ bytes}}{(30+3) \text{ nsec}} = \frac{6 \text{ bytes}}{33 \text{ nsec}}$

Παρθενότητα μέχρι την E

b) Ρυθμός μεταφορών =  $\frac{6 \text{ bytes}}{(29+12+3) \text{ nsec}} = \frac{6 \text{ bytes}}{44 \text{ nsec}}$

Παρθενότητα μέχρι την E και κατάλληλη

### Θέμα 1

r1: 01000101   r5: 00010111   r9: 10000000  
r2: 01111101   r6: 10011010   r10: 10000000  
r3: 11010000   r7: 00010111   r11: 10000001  
r4: 11011010   r8: 10011011   r12: 11000000

Στη συνέχεια ακολουθούν αναλυτικά οι πράξεις, οι τιμές των σημαιών και ο έλεγχος της ορθότητας του αποτελέσματος σε κάθε τερίτωση.

Εντολή	Αποτέλεσμα	C, V, Z	Ατιολογία σωστού ή λάθος αποτελέσματος	
			Αριθμοί χωρίς πρόσημο	Αριθμοί σε συμπλήρωμα ως προς 2
ADD r1, r2 // r1 $\leftarrow$ r1 + r2	<u>01000101+</u> <u>01111101</u> <b>11000010</b>	0, 1, 0	Σωστό, διότι C = 0	Λάθος, διότι V = 1
ADD r3, r4 // r3 $\leftarrow$ r3 + r4	<u>11010000 +</u> <u>11011010</u> 1)10101010	1, 0, 0	Λάθος, διότι C = 1	Σωστό, διότι V = 0
SUB r5, r6 // r5 $\leftarrow$ r5 - r6 μέσω πρόσθεσης με το συπλήρωμα ως προς 2	<u>00010111 +</u> <u>01100110</u> 01111101	0, 0, 0	Σωστό, διότι C = 0	Σωστό, διότι V = 0
ADD r7, r8 // r7 $\leftarrow$ r7 + r8	<u>00010111 +</u> <u>10011011</u> 10110010	0, 0, 0	Σωστό, διότι C = 0	Σωστό, διότι V = 0
ADD r9, r10 // r9 $\leftarrow$ r9 + r10	<u>10000000 +</u> <u>10000000</u> 1)00000000	1, 1, 1	Λάθος, διότι C = 1	Λάθος, διότι V = 1
SUB r11, r12 // r11 $\leftarrow$ r11 - r12 μέσω πρόσθεσης με το συπλήρωμα ως προς 2	<u>10000001 +</u> <u>01000000</u> 11000001	0, 0, 0	Σωστό, διότι C = 0	Σωστό, διότι V = 0

Άσκη 1. (μον. 2.4) Θεωρήστε τα ακόλουθα περιεχόμενα των καταχωρητών:  
 r1: 01000101, r2: 01111011, r3: 11010000, r4: 11011010, r5: 00010111, r6: 10011010,  
 r7: 00010111, r8: 10011011, r9: 10000000, r10: 10000000, r11: 10000001, r12: 10000001.

Να εκτελέσετε κάθε μια από τις εντολές του ακόλουθου πίνακα και να συμπληρώσετε στον πίνακα αποτέλεσμα, τις τιμές των σημαιών καθώς και να αιτιολογήσετε εάν το αποτέλεσμα είναι σωστό ή βασιζόμενοι στις τιμές των σημαιών για κάθε μία των δύο περιπτώσεων: α. τα περιεχόμενα των καταχωρητών είναι χωρίς πρόσημο και β. τα περιεχόμενα των καταχωρητών είναι σε αναπαράσταση συμπληρώματος ως προς 2. Η αιτιολογία να έχει την μορφή  $X=0 \rightarrow$  σωστό ή λάθος αποτέλεσμα. Όσο για την πρόσθεση όσο και την αφαίρεση μεταξύ αριθμών χωρίς πρόσημο ή μεταξύ αριθμών χωρίς πρόσημο ή μεταξύ αριθμών σε αναπαράσταση συμπληρώματος ως προς 2 χρησιμοποιείται η ίδια μονάδα αθροιστή των 8 bits.

ADD r1, r2 σημαίνει  $r1 \leftarrow r1 + r2$  και SUB r1, r2 σημαίνει  $r1 \leftarrow r1 - r2$

Απαντήση	εντολή	αποτέλεσμα	C V Z	αιτιολογία σωστού ή λάθους αποτέλεσματος
				αριθμοί χωρίς πρόσημο
ADD r1, r2		11000101 11000010 01111011 10011010	C=0, V=1, Z=0	C=0 → σωστό
ADD r3, r4		11010101010	C=1, V=0, Z=0	C=1 → λάθος
SUB r5, r6		110000000000 000101111010	C=0, V=0, Z=0	V=0 → σωστό
ADD r7, r8		10110000	C=0, V=0, Z=0	V=0 → σωστό
ADD r9, r10		100000000000 011110111010	C=1, V=1, Z=1	C=1 → λάθος
SUB r11, r12		00000000 100110111010	C=0, V=1, Z=1	C=0 → λάθος

$$\begin{array}{l}
 r1: 01000101 + 11010000 \\
 r2: 01111011 + 11011010 \\
 \hline
 r3: 06010111 \\
 r4: 12011011
 \end{array}$$

$$\begin{array}{l}
 r5: 00010111 - 10011010 \\
 r6: 10011010 - 11011010 \\
 \hline
 r7: 00010111 \\
 r8: 10011010
 \end{array}$$

### Άσκηση από άλλη ομάδα

Θεωρήστε τα ακόλουθα περιεχόμενα των καταχωρητών.

r1: 00010101      r4: 11011010      r7: 01010101      r10: 10011010  
 r2: 10011010      r5: 10000000      r8: 01110101      r11: 10000000  
 r3: 11010101      r6: 10000000      r9: 00010101

r12: 10000000 → r12' = 01111111 (συμπλήρωμα ως προς 1) → r12' + 1 = 10000000 (συμπλήρωμα ως προς 2)

Να εκτελέσετε καθεμία από τις εντολές του ακόλουθου πίνακα και να συμπληρώσετε στον πίνακα το αποτέλεσμα, τις τιμές των σημαιών καθώς και να αιτιολογήσετε αν το αποτέλεσμα είναι σωστό βασιζόμενοι στις τιμές των σημαιών για κάθε μία των δύο περιπτώσεων: α. τα περιεχόμενα των καταχωρητών είναι χωρίς πρόσημο και β. τα περιεχόμενα των καταχωρητών είναι σε αναπαράσταση συμπληρώματος ως προς 2. Η αιτιολογία να έχει τη μορφή  $X=0 \rightarrow$  σωστό ή λάθος αποτέλεσμα, όπου X είναι η τιμή της σημαίας στην τιμή της οποίας βασίζετε το συμπέρασμα. Όσο για την πρόσθεση όσο και για την αφαίρεση μεταξύ αριθμών χωρίς πρόσημο ή μεταξύ αριθμών σε αναπαράσταση συμπληρώματος ως προς 2 χρησιμοποιείται η ίδια μονάδα αθροιστή των 8 bits.

ADD r1, r2 σημαίνει  $r1 \leftarrow r1 + r2$  και SUB r1, r2 σημαίνει  $r1 \leftarrow r1 - r2$

## Λύση

Στη συνέχεια ακολουθούν αναλυτικά οι πράξεις, οι τιμές των σημαιών και ο έλεγχος της ορθότητας του αποτελέσματος

Εντολή	Αποτέλεσμα	C, V, Z	Ατιολογία σωστού ή λάθος αποτελέσματος	
			Αριθμοί χωρίς πρόσημο	Αριθμοί σε συμπλήρωμα ως προς 2
ADD r1, r2 //r1 ← r1 + r2	00010101+ <u>10011010</u> 10101111	0, 0, 0	Σωστό, διότι C = 0	Σωστό, διότι V = 0
ADD r3, r4 // r3 ← r3 + r4	11010101+ <u>11011010</u> 1)10101111	1, 0, 0	Λάθος, διότι C = 1	Σωστό, διότι V = 0
ADD r5, r6 // r5 ← r5 + r6	10000000+ <u>10000000</u> 1)00000000	1, 1, 1	Λάθος, διότι C = 1	Λάθος, διότι V = 1
ADD r7, r8 // r7 ← r7 + r8	01010101 <u>01110101</u> 10001010	0, 1, 0	Σωστό, διότι C = 0	Λάθος, διότι V = 1
SUB r9, r10 // r9 ← r9 - r10 μέσω πρόσθεσης με το συπλήρωμα ως προς 2	00010101 + <u>01100110</u> 01111011			
SUB r11, r12 // r11 ← r11 - r12 μέσω πρόσθεσης με το συπλήρωμα ως προς 2	10000000 + <u>10000000</u> 1)00000000	0, 0, 1	Σωστό, διότι C = 0	Σωστό, διότι V = 0
		1, 1, 1	Λάθος, διότι C = 1	Λάθος, διότι V = 1

## Θέματα Ιανουρίου 2023

### Θέμα 2

Α. Για τον υπολογισμό της έκφρασης "Y = A×(B+C)+D×E" να γραφεί πρόγραμμα για υπολογιστή που βασίζεται στη χρήση 8 καταχωρητών γενικού σκοπού των 32 δυαδικών ψηφίων ο κάθε ένας με εντολές μόνο καταχωρητή-καταχωρητή (ή φόρτωσης-αποθήκευσης) με δύο τελούμενα.

Β. Το σύνολο των εντολών σε επίπεδο γλώσσας μηχανής αποτελείται από 600 εντολές. Το μήκος κάθε εντολής σε δυαδικά ψηφία είναι ακέραιο πολλαλάσιο του 8. Όπου απαιτείται διευθυνσιοδότηση της μνήμης να χρησιμοποιήσετε τον κατευθείαν τρόπο διευθυνσιοδότησης (direct addressing mode). Το εύρος της αρτηρίας διευθύνσεων είναι 32 δυαδικά ψηφία. Για κάθε διαφορετική εντολή της συμβολικής γλώσσας που θα χρησιμοποιήσετε, να σχεδιάσετε τη μορφή της εντολής σε επίπεδο γλώσσας μηχανής (δηλαδή τα πεδία από τα οποία αποτελείται η εντολή σε επίπεδο γλώσσας μηχανής και το εύρος κάθε πεδίου).

Γ. Να δώσετε το πλήθος των ψηφιολέξεων που καταλαμβάνει το πρόγραμμα στην κύρια μνήμη.

Δ. Θεωρήστε ότι τα δεδομένα είναι μίας λέξης (32 δυαδικά ψηφία), η αρτηρία δεδομένων μεταξύ της Κεντρικής Μονάδας Επεξεργασίας (KME) και του συστήματος μνήμης έχει εύρος 1 ψηφιολέξης (byte) και σε κάθε προσπέλαση του συστήματος μνήμης προσπελάζεται μία ψηφιολέξη. Πόσες προσπελάσεις στη μνήμη απαιτεί η εκτέλεση του προγράμματος;

### Λύση

Για την υλοποίηση της έκφρασης  $Y=A \times (B + C) + D \times E$  χρησιμοποιείται **μόνο** η αρχιτεκτονική καταχωρητή – καταχωρητή με δύο τελούμενα:

a)

LOAD R1, B //R1 ← B

LOAD R2, C //R2 ← C

ADD R1, R2 //R1 ← R1 + R2 = B + C

LOAD R3, A //R3 ← A

MUL R1, R3 //R1 ← R1 x R3 = (B + C) x A

LOAD R2, D //R2 ← D

LOAD R3, E //R3 ← E

MUL R2, R3 //R2 ← R2 x R3 = D x E

ADD R1, R2 // E1 ← R1 + R2 = (B + C) x A + D x E

STORE Y, R1 //Y ← R1 = (B + C) x A + D x E

b) Από τη στιγμή που δεν δίνεται άμεσα το μέγεθος του κωδικού λειτουργία (opcode) των εντολών, **θα υπολογιστεί από το πλήθος των εντολών**, το οποίο είναι 600 εντολές, με βάση τον τύπο  $2^x \geq 600 \Rightarrow x = 10$  bits. Επίσης, από το πλήθος και όχι το μέγεθος των



καταχωρητών, θα υπολογιαστεί η χωρητικότητα των καταχωρητών, με βάση τον τύπο:  $2^x \geq 8 \Rightarrow x = 3 \text{ bits}$ . Στη συνέχεια σχεδιαζεται για κάθε εντολή που χρησιμοποιήθηκε στο πρόγραμμα του πρώτου ερωτήματος, η μορφή της σε γλώσσα μηχανής. Από το εύρος της **αρτηρίας διευθύνσεων** υπολογίζουμε το μήκος κάθε διένυθμησης που εμφανίζεται στην εντολή και το οποίο είναι ίσο με 32 bits. Για υπολογισμό προσπελάσεων μνήμης ισχύει ο κανόνας που αναφέρει **ότι οι εντολές που έχουν έντελο μνήμη** έχουν τόσες περισσότερες προσπελάσεις από bytes όσο είναι το πηλίκο  $\frac{\text{δεδομένα}}{\text{αρτηρία δεδομένων}} - \frac{32 \text{ bits}}{8 \text{ bits}} = 4$  (επιπλέον προσπελάσεις). Οι εντολές που δεν έχουν έντελο μνήμη έχουν τόσες προσπελάσεις όσα είναι και τα bytes.

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

Μεταβλητό μήκος εντολής = 45 bits  $\rightarrow$  48 bits

opcode LOAD	R1	B	X
10 bits	3 bits	32 bits	3 bits αρχησιμοποίητο μέρος

Χωρητικότητα  
48 bits = 6 bytes

Προσπελάσεις μνήμης  
10 προσπελάσεις

Μεταβλητό μήκος εντολής = 45 bits  $\rightarrow$  48 bits  
ποίηση μέρους

opcode LOAD	R2	C	X
10 bits	3 bits	32 bits	3 bits αρχησιμοποίητο μέρος

48 bits = 6 bytes

10 προσπελάσεις

Μεταβλητό μήκος εντολής = 45 bits  $\rightarrow$  48 bits

opcode LOAD	R3	A	X
10 bits	3 bits	32 bits	3 bits

16 bits = 2 bytes

2 προσπελάσεις

Μεταβλητό μήκος εντολής = 16 bits

opcode MUL	R1	R3
10 bits	3 bits	3 bits

16 bits = 2 bytes

2 προσπελάσεις

Μεταβλητό μήκος εντολής = 45 bits  $\rightarrow$  48 bits

opcode LOAD	R2	D	X
10 bits	3 bits	32 bits	3 bits

48 bits = 6 bytes

10 προσπελάσεις

Μεταβλητό μήκος εντολής = 45 bits  $\rightarrow$  48 bits

opcode LOAD	R3	E	X
10 bits	3 bits	32 bits	3 bits

48 bits = 6 bytes

10 προσπελάσεις

Μεταβλητό μήκος εντολής = 16 bits

opcode MUL	R2	R3
10 bits	3 bits	3 bits

16 bits = 2 bytes

2 προσπελάσεις

Μεταβλητό μήκος εντολής = 16 bits

opcode ADD	R1	R2
10 bits	3 bits	3 bits

16 bits = 2 bytes

2 προσπελάσεις

Μεταβλητό μήκος εντολής = 45 bits  $\rightarrow$  48 bits

opcode STORE	Y	R1	X
10 bits	32 bits	3 bits	3 bits

48 bits = 6 bytes

10 προσπελάσεις

Συνολική χωρητικότητα 44 bytes και συνολικές προσπελάσεις 68.