Φαρδιές Ποσότητες, Φαρδιές Μνήμες: Little-Endian versus Big-Endian, Ευθυγράμμιση γιά Ταχύτητα Πρόσβασης

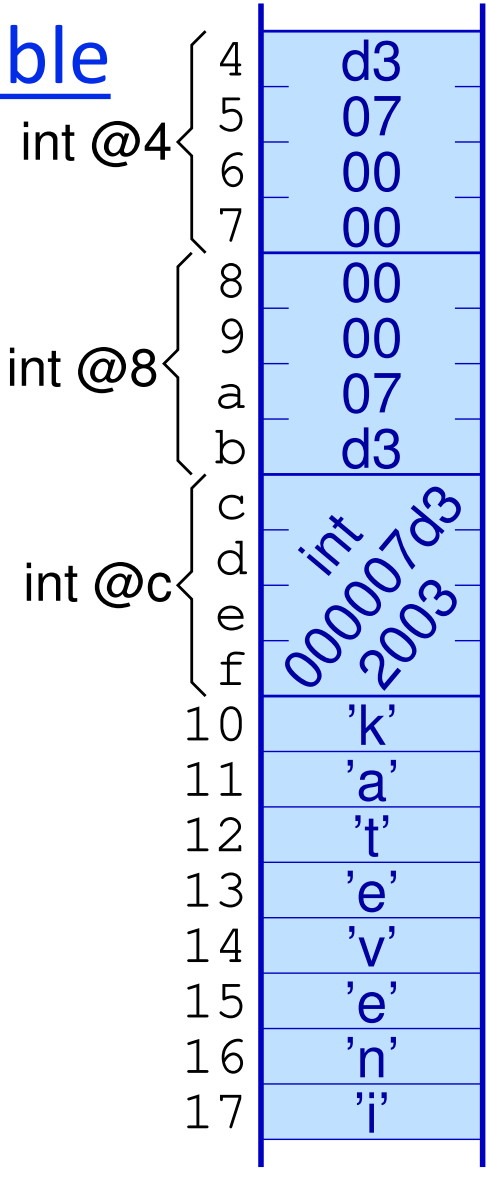
03b (§3.2-3.3) – 19-21/2/2025 – Μανόλης Κατεβαίνης

Ποσότ. >1 By σε Μν. Byte-Adddressable

- Μνήμη Byte-Addressable
- Πίνακες κατά *αύξουσες* & συνεχόμενες διευθύνσεις, *πάντα*
  - π.χ. array of char @ base addr. `10`

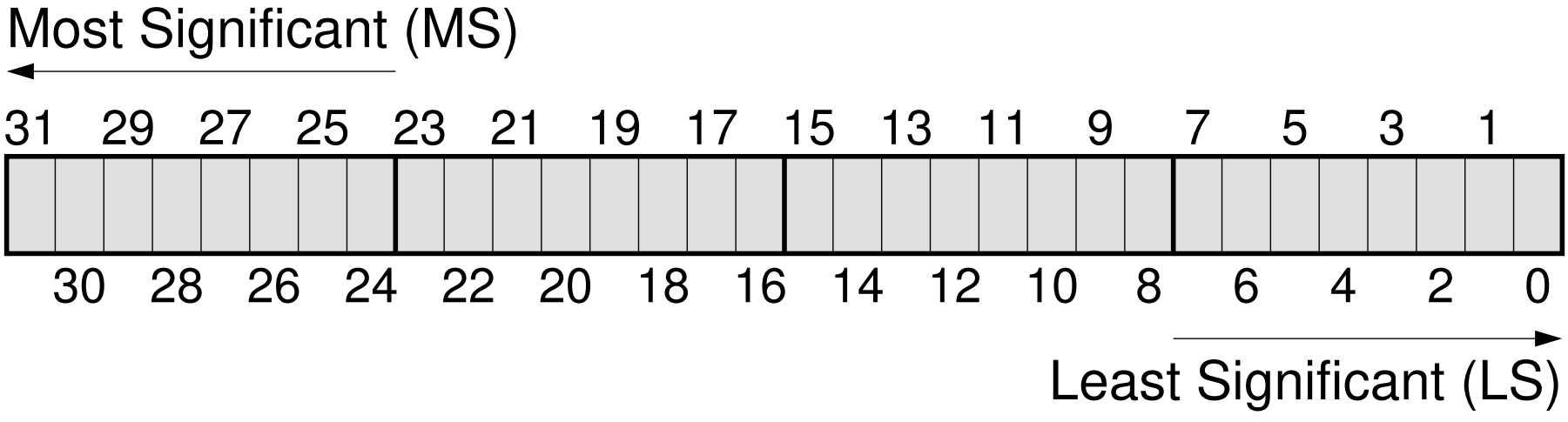
Ποσότητες μεγαλύτερες από 1 Byte:

- Διεύθυνση = η διεύθυνση εκείνου από τα Bytes τους που έχει τη *μικρότερη* διεύθυνση, *πάντα*
- Τα Bytes μέσα στην ποσότητα: *πού;;*
  - π.χ. ακέραιος 2003 (δεκαεξαδικό 7d3)
  - πού πάνε τα Bytes 00, 00, 07, d3 ?

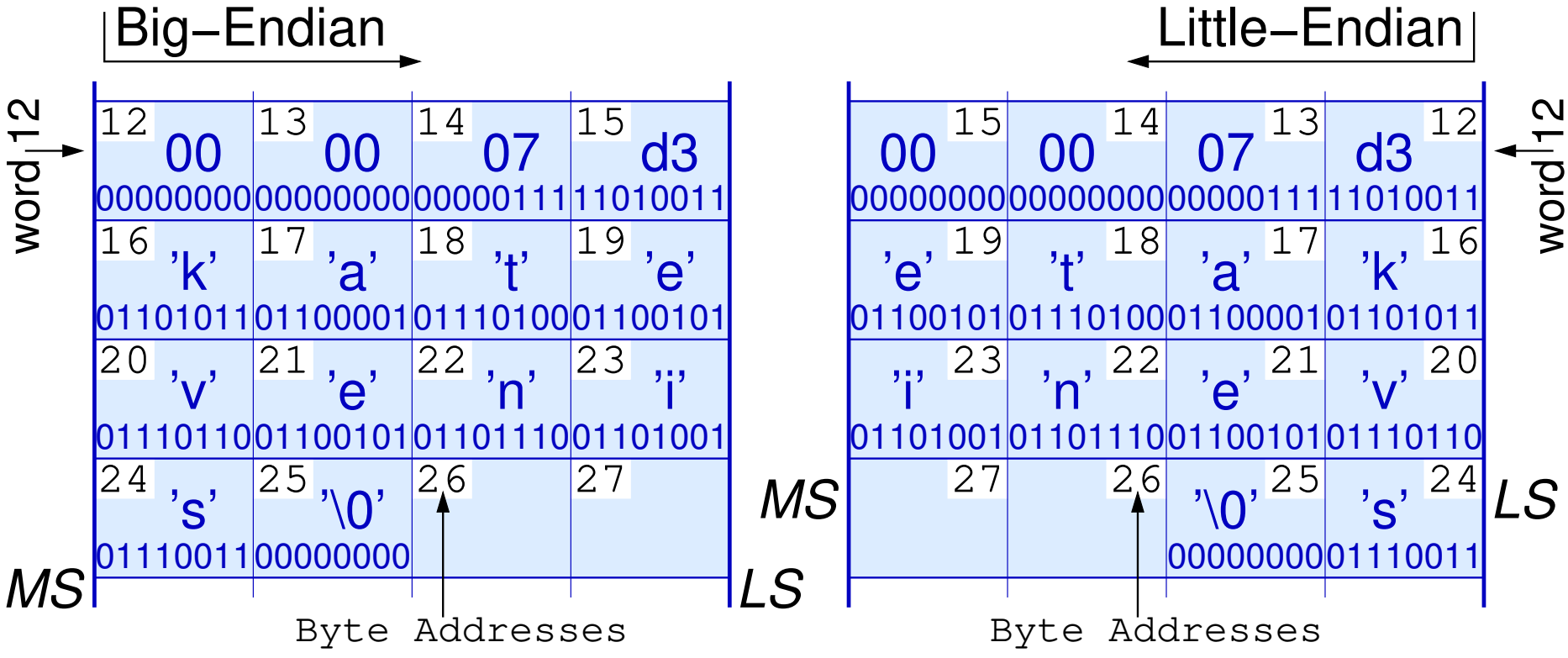


Αρίθμηση των bits μέσα σε μία Λέξη Ακεραίου

Ακέραιος = $b_{n-1}\times 2^{n-1} + b_{n-2}\times 2^{n-2} + \ldots + b_2\times 2^2 + b_1\times 2^1 + b_0\times 2^0$



Αρίθμηση των Bytes μέσα σε μία Λέξη Ακεραίου;



Δύο στρατόπεδα κατασκευαστών («Ιερός πόλεμος»)

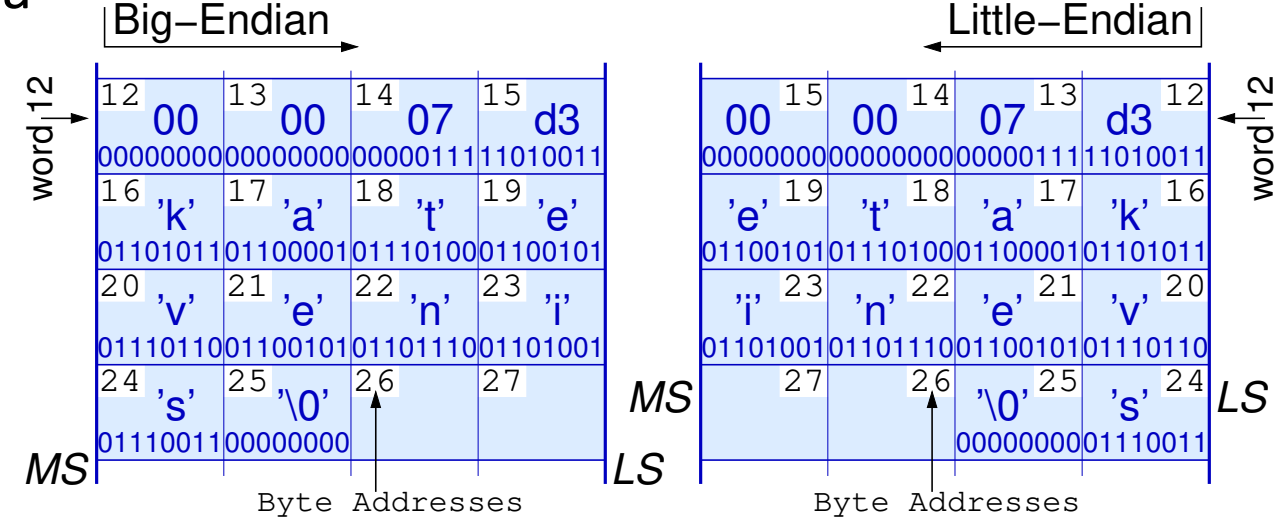
Big-Endians versus Little-Endians: ο ιερός πόλεμος

- Big-Endians: «τα strings να μπορούμε να τα διαβάζουμε»!
- Little-Endians: «όπως αριθμούμε τα bits, έτσι & τα Bytes»!
- rfc-editor.org/ien/ien137.txt "On Holy Wars and a Plea for Peace"
- *Big-Endians*: IBM S/360, Motorola 68000, early PowerPC
- *Bi-Endians* (configured as either): *ARM*, PowerPC, MIPS
- *Little-Endians*: Intel *x86*, AMD, *RISC-V*, early ARM

Τι με νοιάζει εμένα; – (1) κανονικά προγράμματα

- char buf[10] με base addr. 16
- γράφω buf[0] = ‘k’
- γράφω buf[1] = ‘a’
- διαβάζω buf[0] ⟶ ‘k’
- διαβάζω buf[1] ⟶ ‘a’
- ίδια, σωστή συμπεριφορά σε ό,τι μηχανή και να τρέξω το πρόγραμμά μου

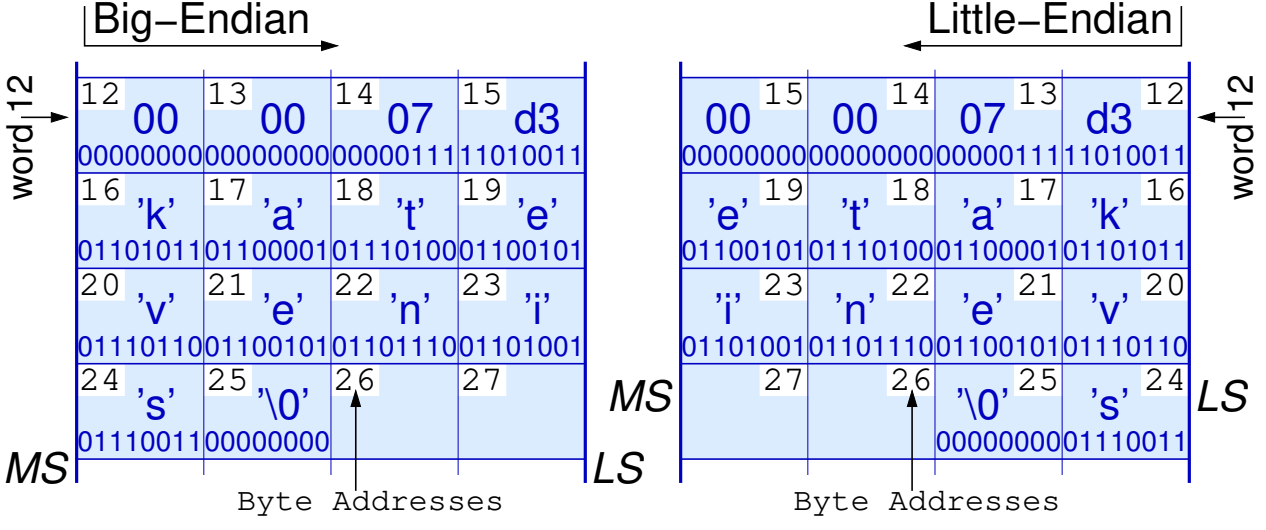
- int year=2003 στη διεύθυνση 12
- διαβάζω int από 12, βρίσκω 2003
- ίδια, σωστή συμπεριφορά σε ό,τι μηχανή και να τρέξω



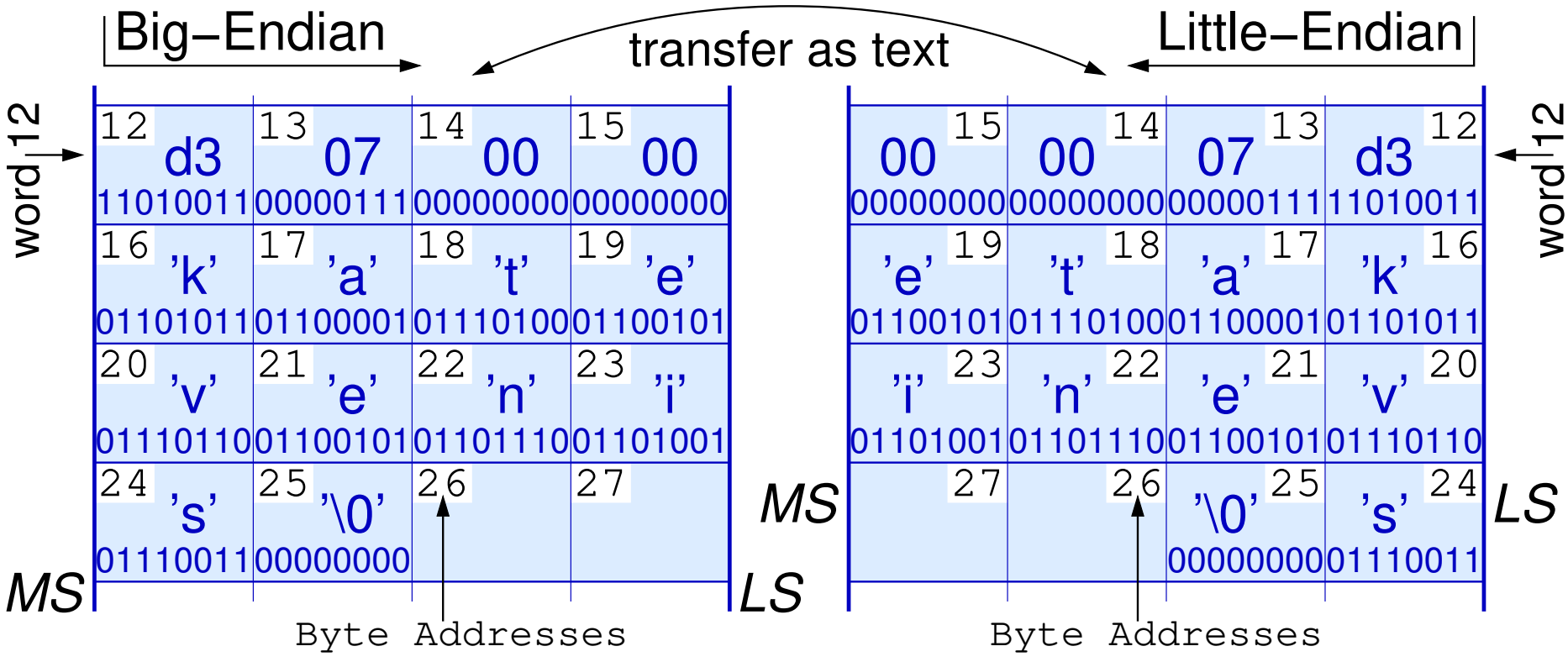
Τι με νοιάζει εμένα; – (2) «παράξενα» προγράμματα

- γράφω *string* “katevenis” ξεκινώντας από τη διευθ. 16
- διαβάζω *int* (!) από δ. 16:
- αλλού (big-e.) ⟶ 6b617465
- αλλού (little-e.) ⟶ 6574616b
- *non-portable* program !
- η συμπεριφορά αλλάζει από μηχανή σε μηχανή
- (και μάλλον *ανόητο* πρόγραμμα)

- γράφω *int* year=2003 στη δ. 12
- διαβάζω *char* (!) από διευθ. 12:
- αλλού (big-e.) βρίσκω: ‘\0’ (00)
- αλλού (little-e.) βρίσκω: ‘Σ’ (d3)

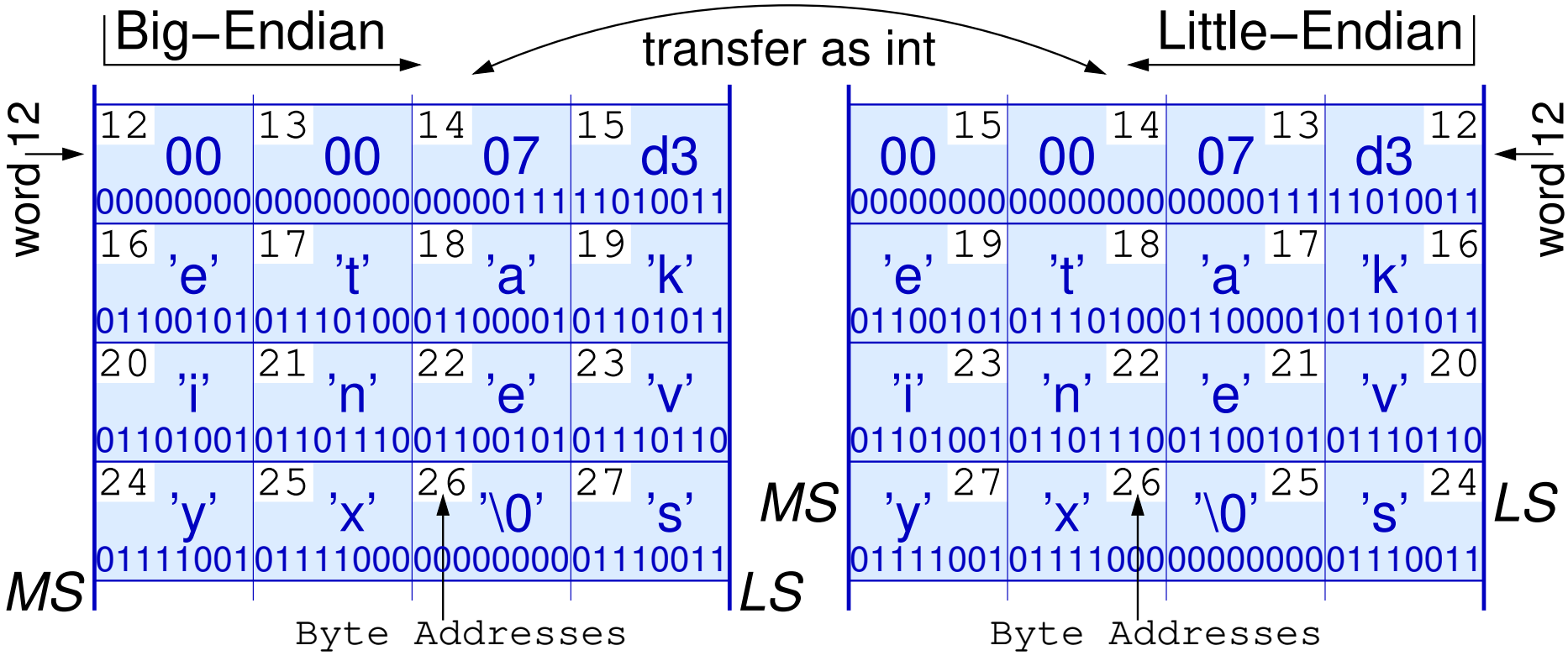


Τι με νοιάζει εμένα; – (3) μεταφορά αρχείου ως text



Ο ακέραιος 2003 γίνεται -754.515.968 στην άλλη μηχανή!

Τι με νοιάζει εμένα; – (4) μτφ. αρχείου ως binary (int)

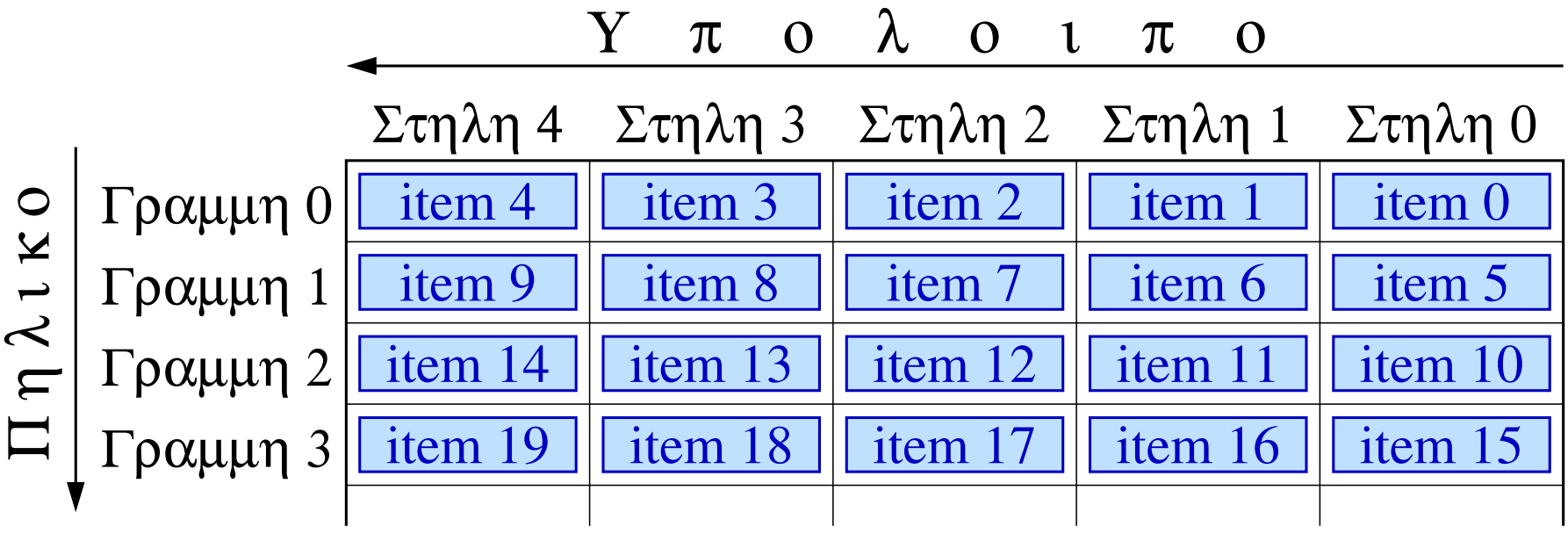


To string @16 γίνεται “etakinevyx” στην άλλη μηχανή!

Προσπελάσεις σε Φαρδιές Μνήμες: Ευθυγράμμιση

- Οι πραγματικές μνήμες συνήθως φαρδιές (32, 64,… bits)
  - γιά λόγους ταχύτητας: περισσότερα Bytes σε κάθε πρόσβαση
  - εάν ο επεξεργαστής χρειάζεται λιγότερα Bytes, η μνήμη δίνει όλα τα Bytes στην «γραμμή» εκείνη, και ο επεξεργαστής, εσωτερικά, επιλέγει και κρατά εκείνα που θέλει
  - χωριστό σήμα *writeEnable* γιά κάθε Byte position: γιά να γράψουμε λιγότερα Bytes από μία ολόκληρη «γραμμή», ανάβουμε μόνον τα *wrEnab*'s εκεί που θέλουμε να γράψουμε
- Η πραγματική μνήμη προσπελάζει πάντα *ευθυγραμμισμένες* «γραμμές», στην κάθε πρόσβαση
- *Πόσες προσπελάσεις* χρειάζονται γιά ποσότητα R Bytes?

Υπενθύμιση: Γραμμές, Στήλες ⇔ Πηλίκο, Υπόλοιπο



- Γραμμές μεγέθους Δ ⇒ Δ είναι ο διαιρέτης, εδώ Δ=5
- Το στοιχείο *M* είναι στην Ομάδα = Γραμμή = *Πηλίκο* (*M*/Δ),
- στη θέση = Στήλη = *Υπόλοιπο* (*M*/Δ)

HY-120 Εργ. 6a (slide 5)
www.csd.uoc.gr/~hy120/23f/06a_m_d_wrap.pdf

Πλάτος = 2^k ⇒ Γραμμή: n-k MS, Στήλη: k LS Addr. bits

000001	000000
000011	000010
000101	000100

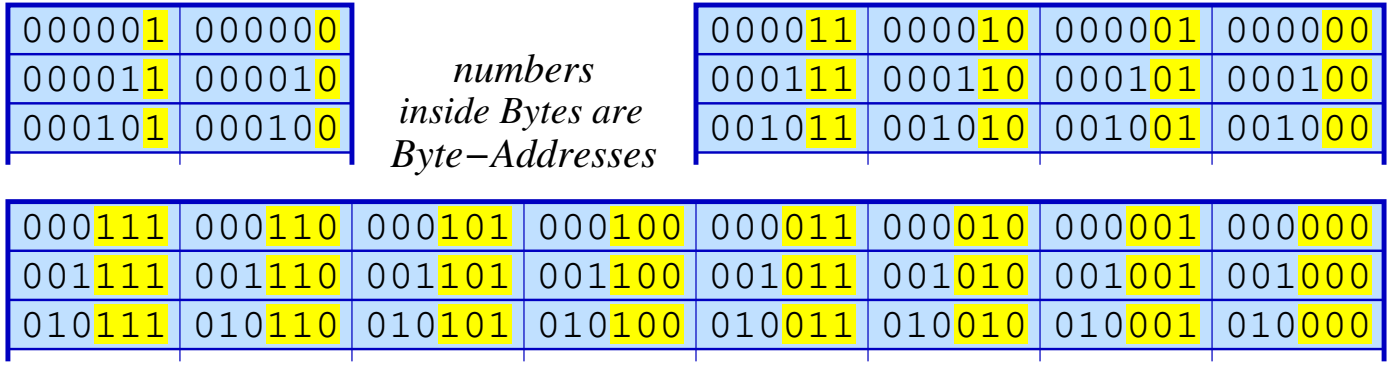
numbers inside Bytes are Byte–Addresses

000011	000010	000001	000000
000111	000110	000101	000100
001011	001010	001001	001000

000111	000110	000101	000100	000011	000010	000001	000000
001111	001110	001101	001100	001011	001010	001001	001000
010111	010110	010101	010100	010011	010010	010001	010000

- Όταν Πλάτος Μνήμης = Διαιρέτης $\Delta = 2^k$ Bytes:
  - Γραμμή = Πηλίκο (Byte-Address / 2^k) = n-k MS Addr. bits (γαλάζια)
  - Στήλη = Υπόλοιπο (Byte-Address / 2^k) = k LS Addr. bits (κίτρινα)

Διεύθ. («Ευθυγραμμισμένης») Ποσότητας μεγέθους Γραμμής



- Όταν Πλάτος Μνήμης = Διαιρέτης $\Delta = 2^k$ Bytes:
  - Γραμμή = Πηλίκο (Byte-Address / 2^k) = n-k MS Addr. bits (γαλάζια)
  - Στήλη = Υπόλοιπο (Byte-Address / 2^k) = k LS Addr. bits (κίτρινα)
- Διεύθυνση ποσότητας αποτελούμενης από *τα Bytes μιας Γραμμής* = διεύθυνση εκείνου από τα Bytes με την μικρότερη διεύθυνση = εκείνου με Υπόλοιπο 0, δηλαδή στη Στήλη 00…0 = εκείνου με Διεύθ. *ακέραιο πολλαπλάσιο* του μεγέθους της ποσότητας = μεγ. γραμμής

Πόσες προσπελάσεις για R Bytes σε Μν. πλάτους W?

- Για να διαβάσουμε/γράψουμε μια Ποσότητα μεγέθους R Bytes από/σε μια Μνήμη πλάτους W Bytes:
  - Minimum $\lceil R/W \rceil$ προσπελάσεις
- *Πώς* να εξασφαλίσουμε *πάντα* αυτό το minimum?
  - Όταν το *ίδιο* πρόγραμμα (binary executable) τρέχει σε διάφορα μοντέλα υπολογιστών με *διάφορα W*
  - Όταν $R=2^r$ και $W=2^w$ είναι δυνάμεις του 2

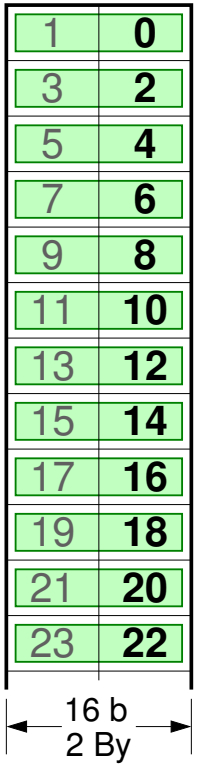
Πώς να εξασφαλίσουμε *πάντα* τις min. προσπελάσεις;

Minimum = $\lceil R/W \rceil$, με $R=2^r$ και $W=2^w$ δυνάμεις του 2:

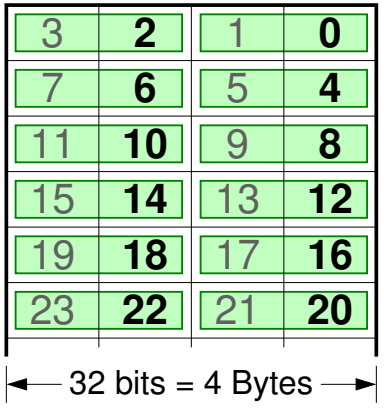
- Όταν $R=2^r > W=2^w$ τότε R/W είναι ακέραιος >1, και τότε:
  - Πρέπει κάθε κομμάτι W της Ποσότητας R να είναι πλήρης γραμμή της Μνήμης W, άρα να έχει διεύθυνση πολλαπλάσιο του $W=2^w$
  - Διεύθυνση Ποσότητας = διεύθ. «πρώτου» κομματιού = πολ. W
- Όταν $R=2^r < W=2^w$ τότε $\lceil R/W \rceil = 1$, και τότε:
  - Πρέπει η Ποσότ. R να μην διασταυρώνει γραμμή της Μνήμης W
  - Εάν έχει διεύθυνση πολλαπλάσιο του $R=2^r$ εξασφαλίζεται πάντα
- Όταν $R=2^r = W=2^w$ τότε $R/W = 1$, και τότε:
  - Πρέπει η Ποσότητα R να είναι γραμμή της Μνήμης W άρα να έχει διεύθυνση ακέραιο πολλαπλάσιο του $R=2^r = W=2^w$

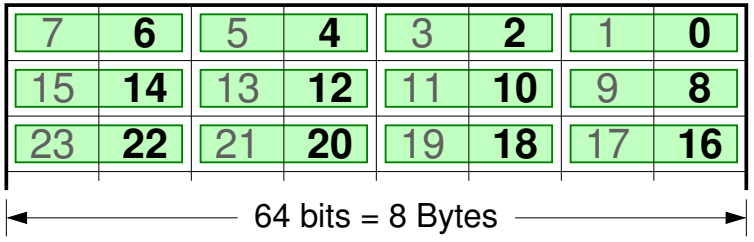
$\Rightarrow$ Συνολικά, για min. για κάθε W: **Διεύθ. πολλαπλ. του R**

2–Byte "Half Words" Aligned on Addresses that are integer multiples of 2



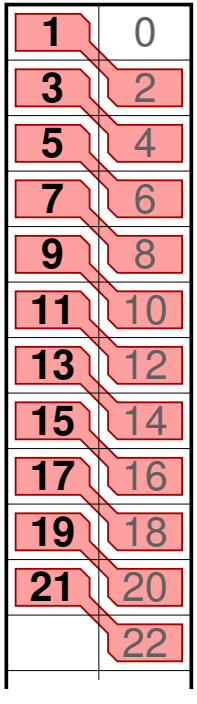
Addresses drawn assuming: *Little Endian layout*



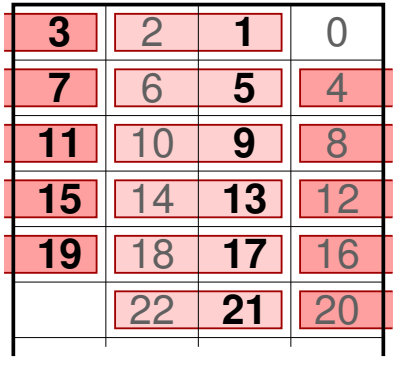


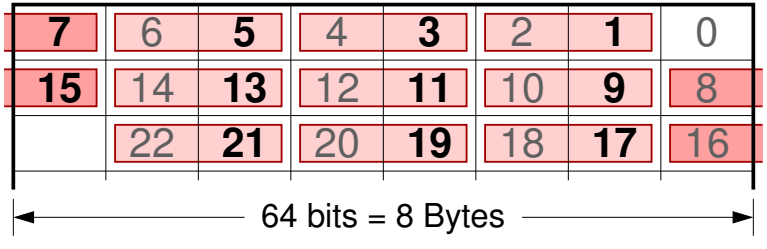
- Numbers inside boxes are Byte Addresses –NOT Contents
- The address of each 2–Byte "half word" is shown in Bold

(the address of a multi–Byte quantity is the address of that Byte inside it that has the smallest address among all the Bytes inside the quantity)



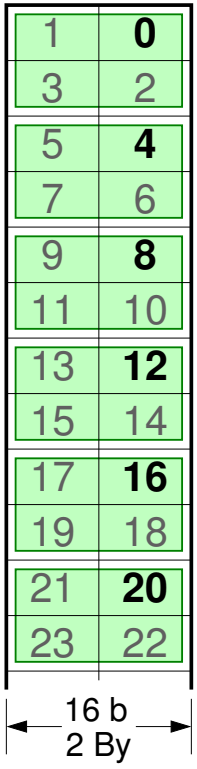
2–Byte "Half Words" at Addresses that are NOT integer multiples of 2



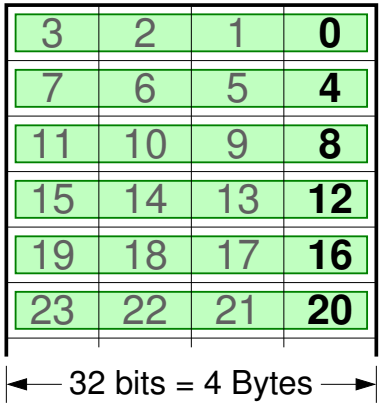


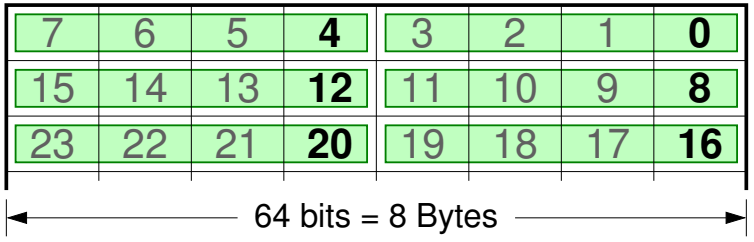
- Some (even if not all) of these 2–Byte half–w. incur a performance penalty when accessed

4–Byte "Words" Aligned on Addresses that are integer multiples of 4



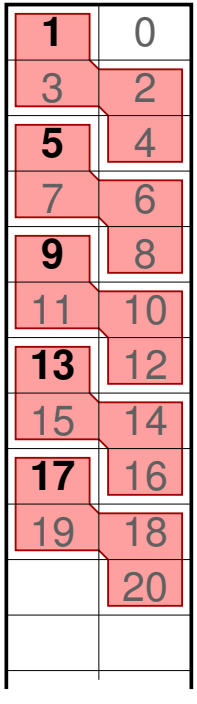
Addresses drawn assuming: *Little Endian layout*



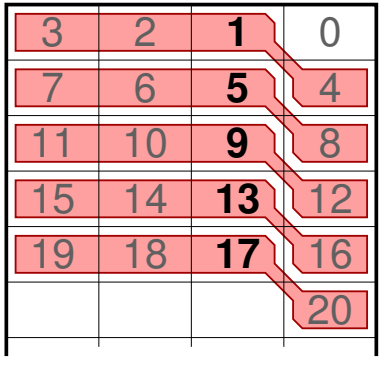


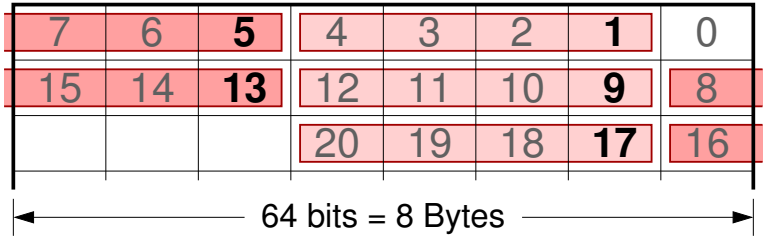
- Numbers inside boxes are Byte Addresses –NOT Contents
- The address of each 4–Byte "word" is shown in Bold

(the address of a multi–Byte quantity is the address of that Byte inside it that has the smallest address among all the Bytes inside the quantity)



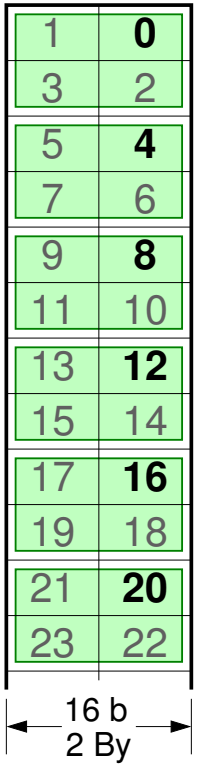
4–Byte "Words" at Addresses that are NOT multiples of 4, but are 1–off, i.e. Addr mod 4 == 1



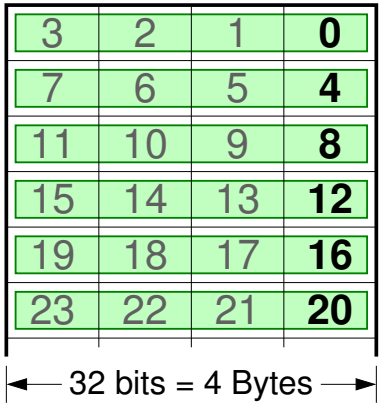


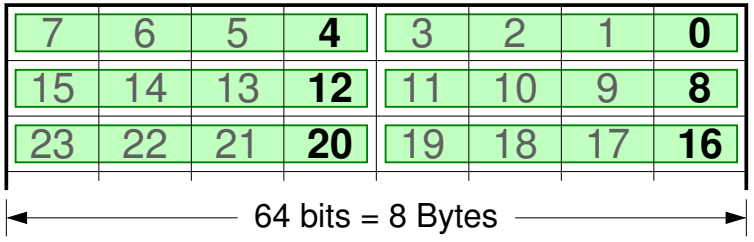
- Some (even if not all) of these 4–Byte words incur a performance penalty when accessed

4–Byte "Words" Aligned on Addresses that are integer multiples of 4



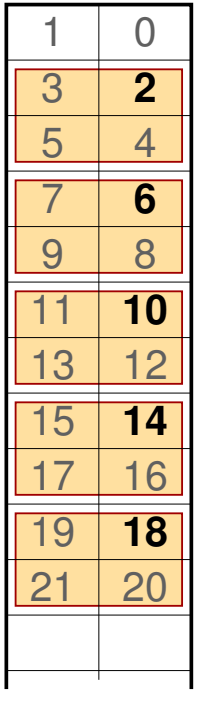
Addresses drawn assuming: *Little Endian layout*





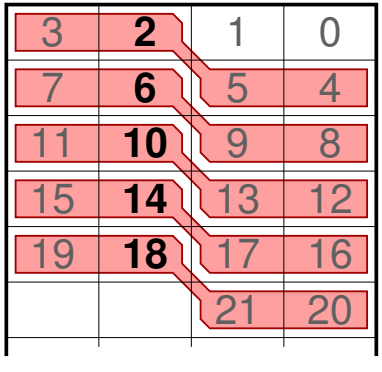
- Numbers inside boxes are Byte Addresses –NOT Contents
- The address of each 4–Byte "word" is shown in Bold

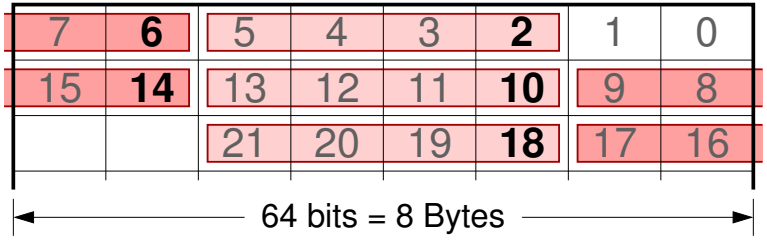
(the address of a multi–Byte quantity is the address of that Byte inside it that has the smallest address among all the Bytes inside the quantity)



4–Byte words at multiples of 2 but not of 4 are OK in 2–Byte wide memories, but NOT in wider!

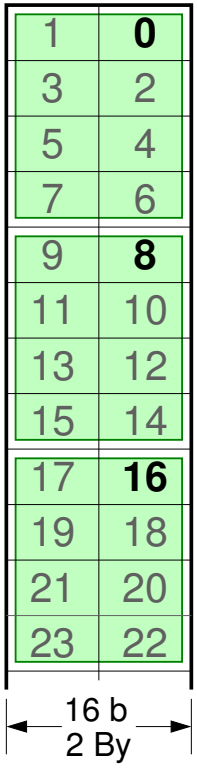
4–Byte "Words" at Addresses that are multiples of 2, but NOT multiples of 4 (i.e. Addr mod 4 == 2)



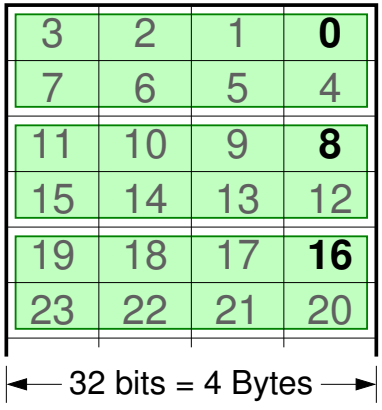


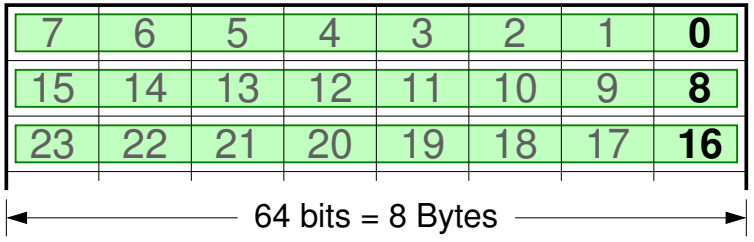
- Some (even if not all) of these 4–Byte words incur a performance penalty when accessed

8–Byte "Double Words" Aligned on Addresses that are integer multiples of 8



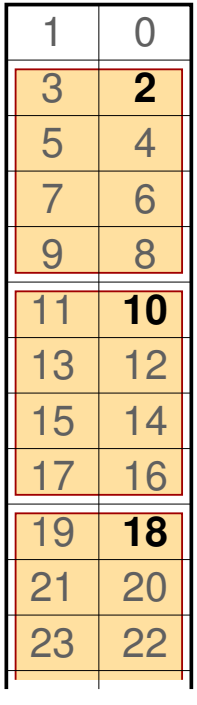
Addresses drawn assuming: *Little Endian layout*





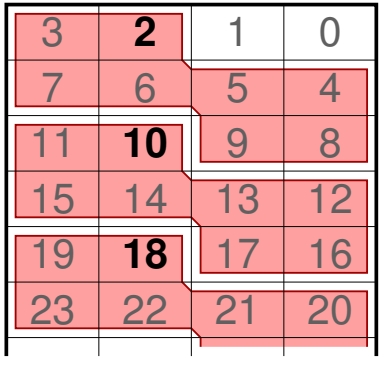
- Numbers inside boxes are Byte Addresses –NOT Contents
- The address of each 8–Byte "double word" is shown in Bold

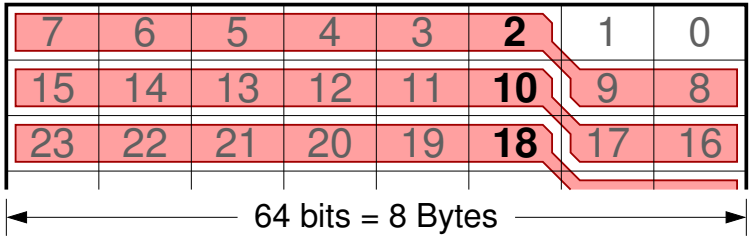
(the address of a multi–Byte quantity is the address of that Byte inside it that has the smallest address among all the Bytes inside the quantity)



8–Byte doubles at multiples of 2 but not of 4 or 8 are OK in 2–Byte wide memories, but NOT in wider!

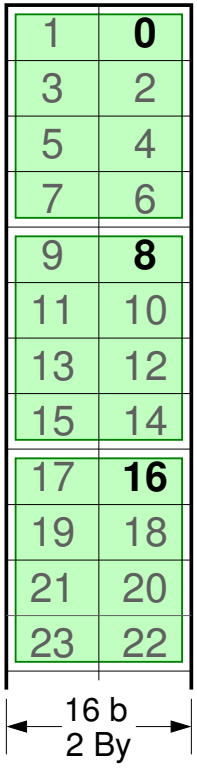
8–Byte "Doubles" at Addresses that are multiples of 2, but NOT multiples of 4 or 8 (i.e. Addr mod 8 == 2)



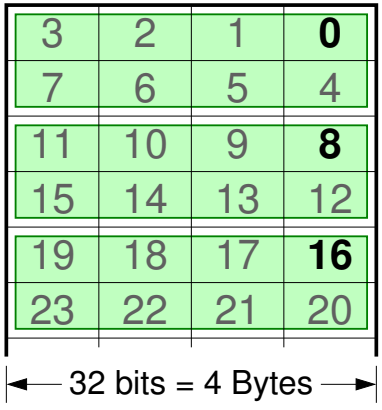


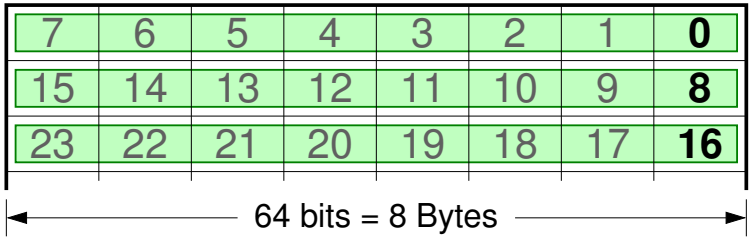
- In 4– and 8–wide memories, all of these doubles incur a performance penalty when accessed

8–Byte "Double Words" Aligned on Addresses that are integer multiples of 8



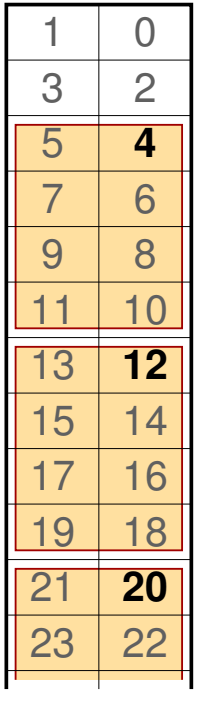
Addresses drawn assuming: *Little Endian layout*





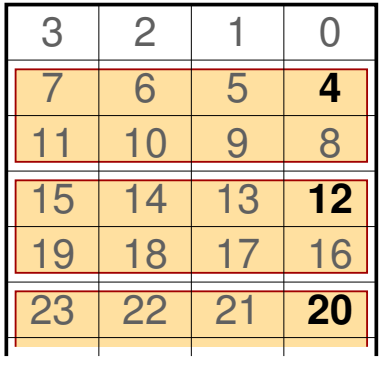
- Numbers inside boxes are Byte Addresses –NOT Contents
- The address of each 8–Byte "Double word" is shown in Bold

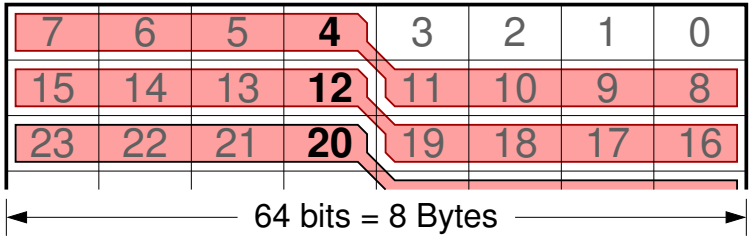
(the address of a multi–Byte quantity is the address of that Byte inside it that has the smallest address among all the Bytes inside the quantity)



8–Byte Doubles at multiples of 4 but not of 8 are OK in 2– & 4– wide memories, but NOT in wider!

8–Byte "Doubles" at Addresses that are multiples of 4, but NOT multiples of 8 (i.e. Addr mod 8 == 4)





- In 8– Byte wide memories, these Doubles incur a performance penalty when accessed

Γιά ταχύτητα: Ευθυγράμμιση στα «φυσικά όρια»

Ηθικόν Δίδαγμα:

- Γιά τον *ελάχιστο* δυνατό αριθμό προσπελάσεων μνήμης,
- σε μνήμη *οιουδήποτε* πλάτους (πάντα δύναμη του 2)
  - δηλαδή γιά προγράμματα με "*portable* efficiency":
- Half-words (*2 Bytes*) σε διευθ. ακέραια πολλαπλ. του **2**
- Words (*4 Bytes*) σε διευθύνσεις ακέραια πολλαπλ. του **4**
- Double-words (*8 Bytes*) σε διευθ. ακέρ. πολλαπλ. του **8**
- Γενικά: *Ευθυγράμμιση* της κάθε ποσότητας (2^r By) στα «*φυσικά*» της όρια (ακερ. πολ. του 2^r)

Οδηγίες στον Assembler γιά Ευθυγράμμιση

- Σε μερικούς επεξ. υποχρεωτική η ευθυγράμ. (π.χ. MIPS)
- Σε άλλους *προαιρετική*: **RISC-V** (κ.α.)
  - παντού συμφέρει σε ταχύτητα, έστω και με κάποια κενά στη χρησιμοπ. του χώρου μνήμης (λιγότερα εάν βελτιστοποιηθούν)
- Οδηγίες Assembler να ευθυγραμμίσει το επόμενο item:
  - `.align` **2** ⇒ σε ακέραιο πολλαπλ. του $2^{\mathbf{2}} = 4$
  - `.align` **3** ⇒ σε ακέραιο πολλαπλ. του $2^{\mathbf{3}} = 8$

    κάνε τη διεύθ. όπου θα αρχίσει το επόμενο πράγμα στο data segment ακερ. πολ. του 2^r (αυξάνοντας την λίγο αν δεν είναι ήδη)
  - `.space 32` ⇒ κράτησε εδώ χώρο 32 Bytes (άφησέ τον κενό)

Παράδειγμα: .align . space

```
mystr:  .asciz "katevenis"
        .align 3  # πήδα στο επόμενο
                    ακέρ. πολ. του 2^3=8
p_dbl:  .space 8  # ονόμασε τη θέση
                    "p_dbl" και κράτα
                    χώρο 8 Bytes εκεί
i_wrd:  .space 4  # επόμ. θέση ονόματι
                    "i_wrd" και κράτα
                    χώρο 4 Bytes εκεί
```