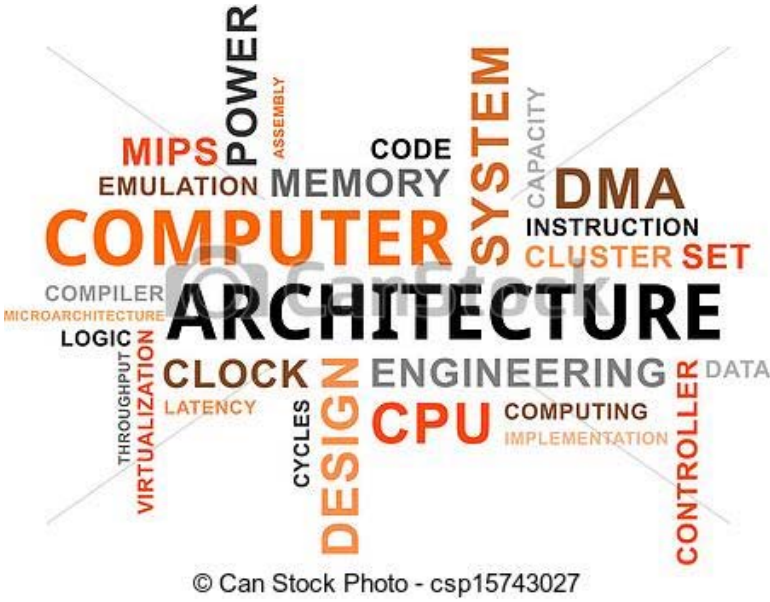
SCHOOL OF INFORMATICS



Αρχιτεκτονική Υπολογιστών

Διάλεξη 5 – Αριθμητική Υπολογιστών (Computer Arithmetics)

Γεώργιος Κεραμίδας, Επίκουρος Καθηγητής

3ο Εξάμηνο, Τμήμα Πληροφορικής

ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

Αντιστοίχιση με ύλη Βιβλίου

- **Το συγκεκριμένο σετ διαφανειών καλύπτει τα εξής κεφάλαια/ενότητες:**
  - **Κεφάλαιο 3: Όλες οι ενότητες εκτός της ενότητας 3.7**

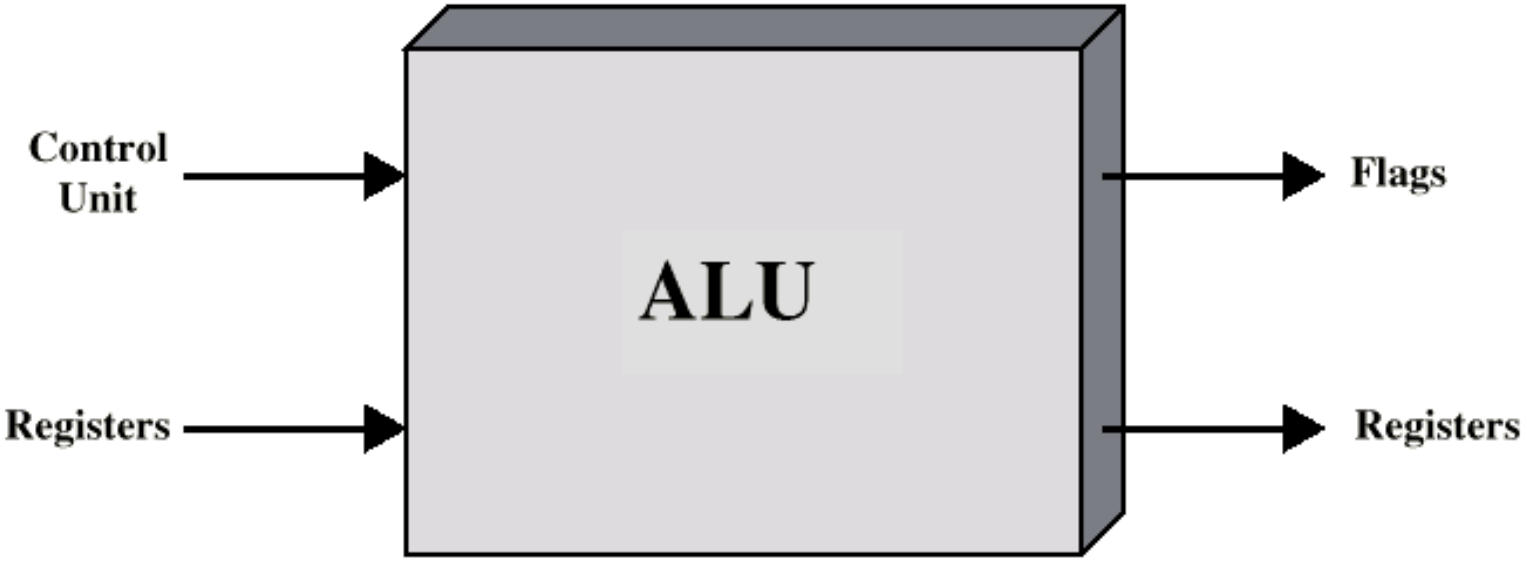
Three programmers walk into a bar...

Αριθμητική για υπολογιστές

- **Λειτουργίες (πράξεις) σε ακεραίους**
  - **Πρόσθεση και αφαίρεση**
  - **Πολλαπλασιασμός και διαίρεση**
  - **Χειρισμός της υπερχείλισης**
- **Πραγματικοί αριθμοί κινητής υποδιαστολής (floating-point)**
  - **Αναπαράσταση και λειτουργίες (πράξεις)**

```
a = 5 ;
b = 0 ;
c = a / b ;
```

Αριθμητική και Λογική Μονάδα



Προηγούμενη Γνώση

- **Μετατροπή Δεκαδικών ←→ Δυαδικών αριθμών**
- **Υπολογισμός Συμπληρώματος ως προς 1 και 2**
- **Γιατί όμως χρειαζόμαστε τα συμπληρώματα ως προς 1 και 2 ?**
- **Γιατί δεν μας κάνει το συμπλήρωμα ως προς 1 και χρειαζόμαστε το συμπλήρωμα ως προς 2?**

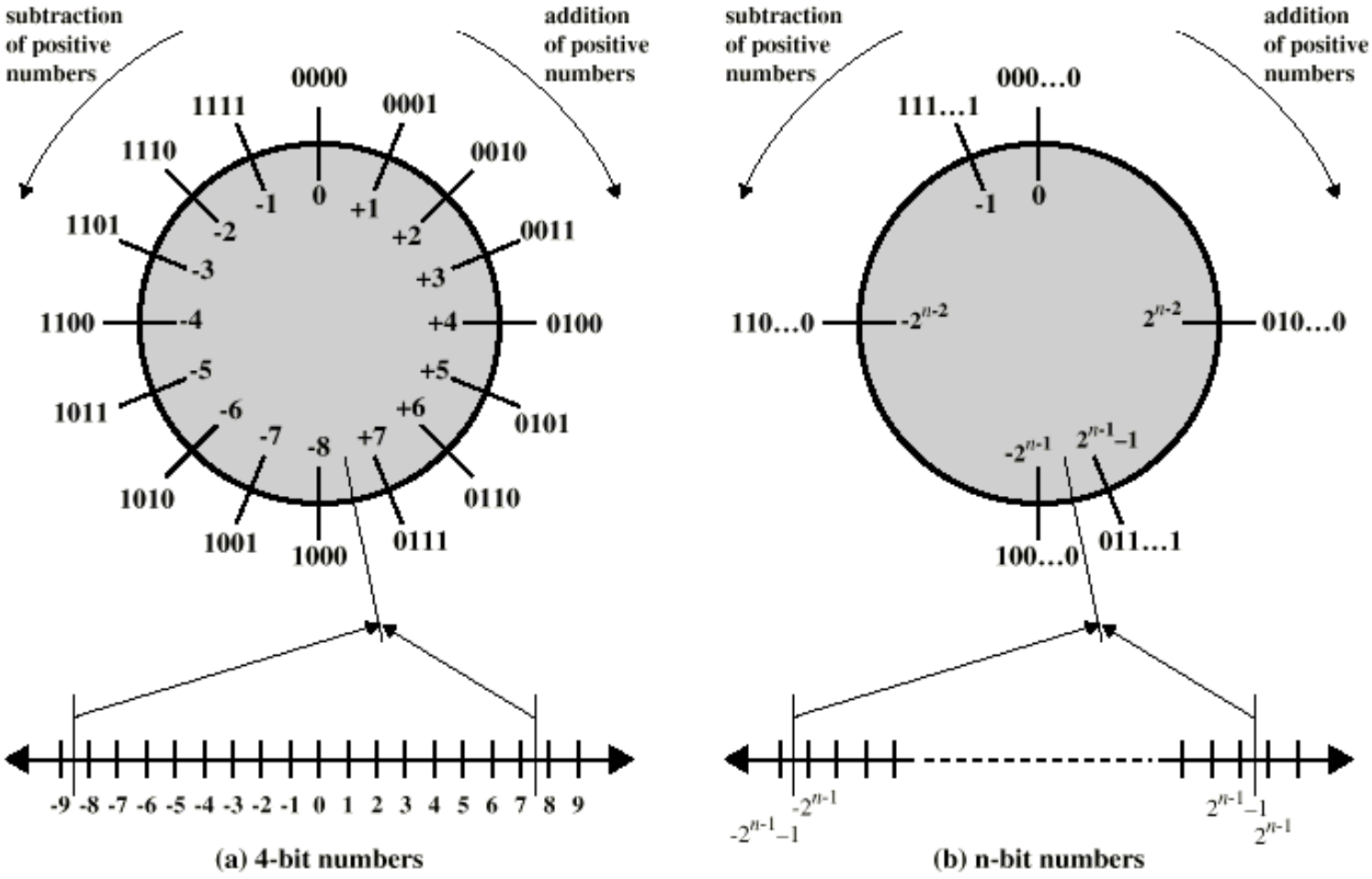
+7	0111	-1	1001
+6	0110	-2	1010
+5	0101	-3	1011
+4	0100	-4	1100
+3	0011	-5	1101
+2	0010	-6	1110
+1	0001	-7	1111

Αναπαράσταση με προσημασμένο μέτρο

Πίνακας

	Signed Magnitude	1's Complement	2's Complement
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
0	0000 1000	0000 1111	0000
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001
-8	-	-	1000

Συμπλήρωμα ως προς 2



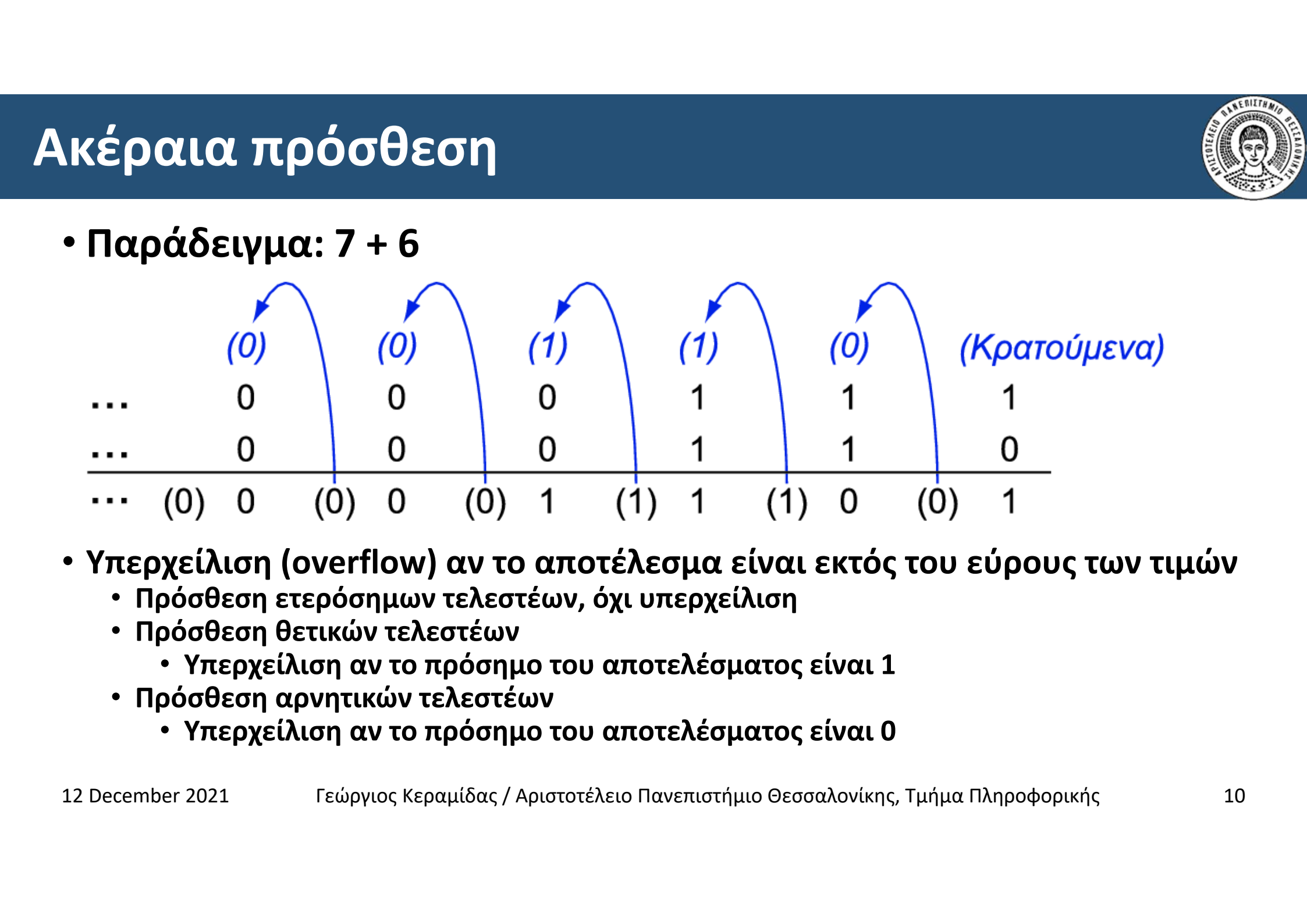
Εύρος

- **8 bit 2s compliment**
  - **+127 = 01111111 = 2^7 -1**
  - **-128 = 10000000 = -2^7**
- **16 bit 2s compliment**
  - **+32767 = 011111111 11111111 = 2^{15} - 1**
  - **-32768 = 100000000 00000000 = -2^{15}**

Αλλάζοντας το εύρος

- **Μη προσημασμένοι → zero extension**
  - **18 = 00010010**
  - **18 = 00000000 00010010**
- **Προσημασμένοι αριθμοί → sign extension**
  - **-18 = 10010010**
  - **-18 = 11111111 10010010**

Ακέραια πρόσθεση



- **Παράδειγμα: 7 + 6**

- **Υπερχείλιση (overflow) αν το αποτέλεσμα είναι εκτός του εύρους των τιμών**
  - **Πρόσθεση ετερόσημων τελεστέων, όχι υπερχείλιση**
  - **Πρόσθεση θετικών τελεστέων**
    - **Υπερχείλιση αν το πρόσημο του αποτελέσματος είναι 1**
  - **Πρόσθεση αρνητικών τελεστέων**
    - **Υπερχείλιση αν το πρόσημο του αποτελέσματος είναι 0**

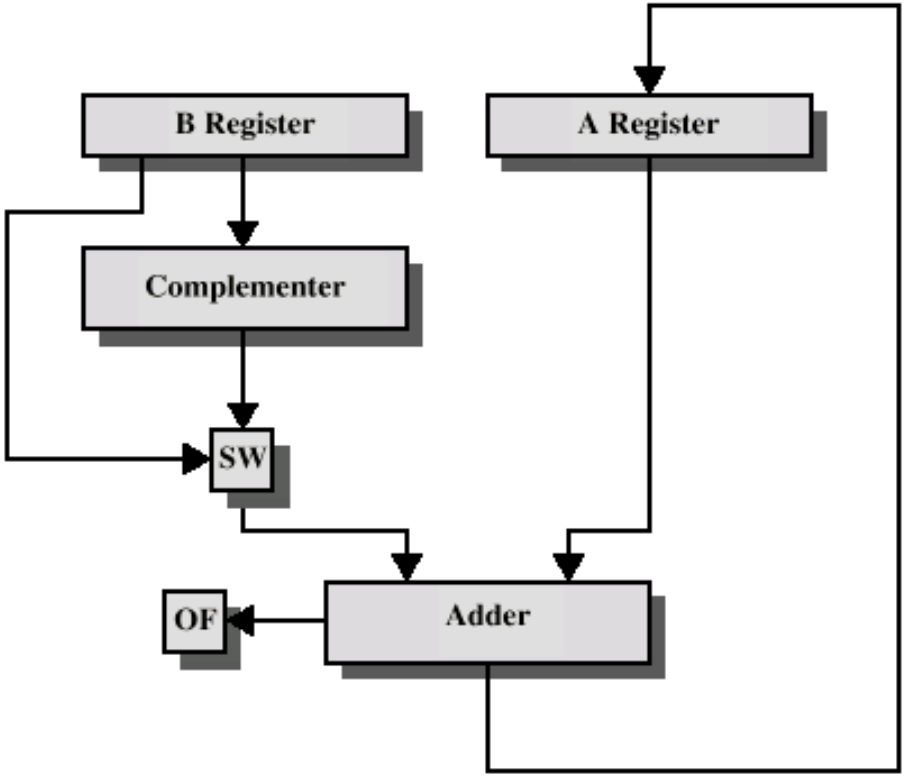
Ακέραια αφαίρεση

- **Πρόσθεση του αντιθέτου του δεύτερου τελεστέου**
- **Παράδειγμα: 7 – 6 = 7 + (–6)**

+7: 0000 0000 … 0000 0111
–6: 1111 1111 … 1111 1010
+1: 0000 0000 … 0000 0001

- **Υπερχείλιση αν το αποτέλεσμα είναι εκτός του εύρους των τιμών**
  - **Αφαίρεση δύο θετικών ή δύο αρνητικών, όχι υπερχείλιση**
  - **Αφαίρεση θετικού από αρνητικό τελεστέο**
    - **Υπερχείλιση αν το πρόσημο του αποτελέσματος είναι 0**
  - **Αφαίρεση αρνητικού από θετικό τελεστέο**
    - **Υπερχείλιση αν το πρόσημο του αποτελέσματος είναι 1**

Υλικό Προσθετή – Αφαιρέτη

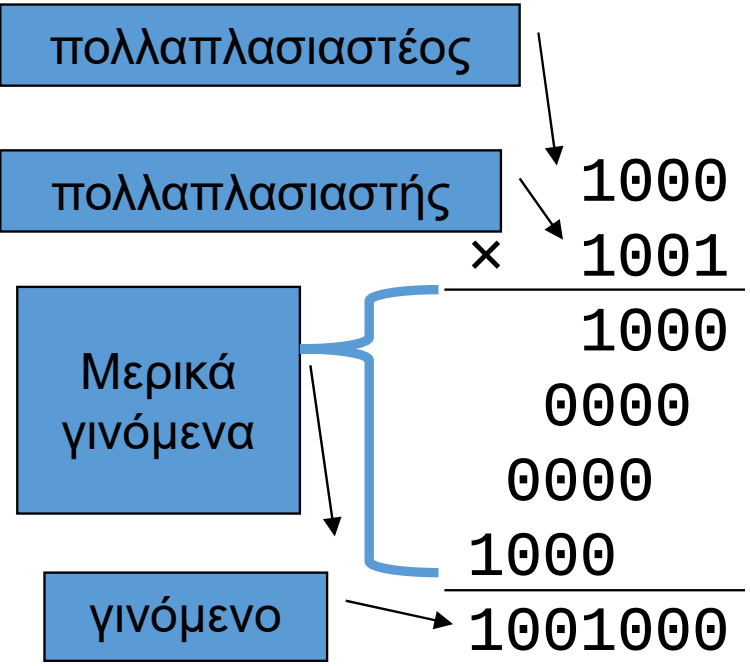


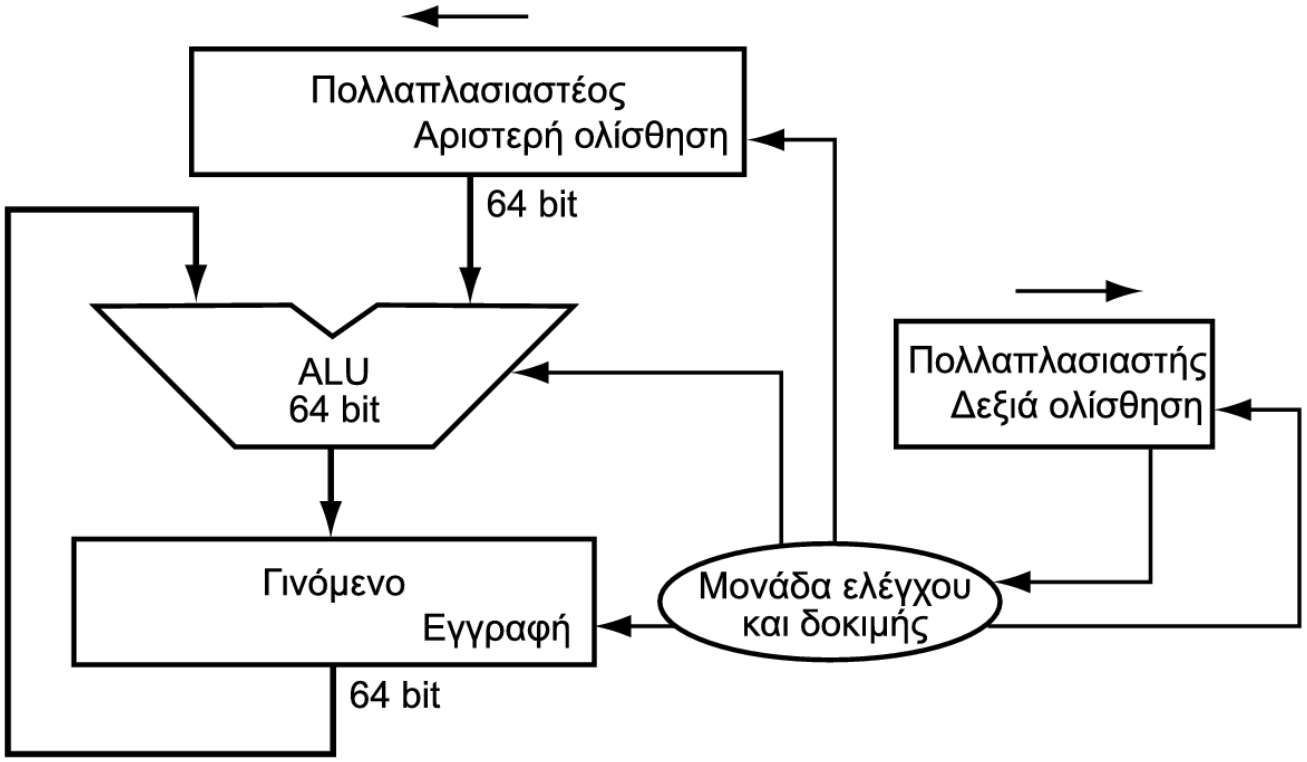
OF = overflow bit
SW = Switch (select addition or subtraction)

Το δεύτερο σημαντικό πλεονέκτημα του συμπληρώματος ως προς 2 είναι ότι μπορεί να χρησιμοποιηθεί ο ίδιος προσθετής για πράξεις προσημασμένων και μη-προσημασμένων αριθμών

Πολλαπλασιασμός – Διάγραμμα Ροής

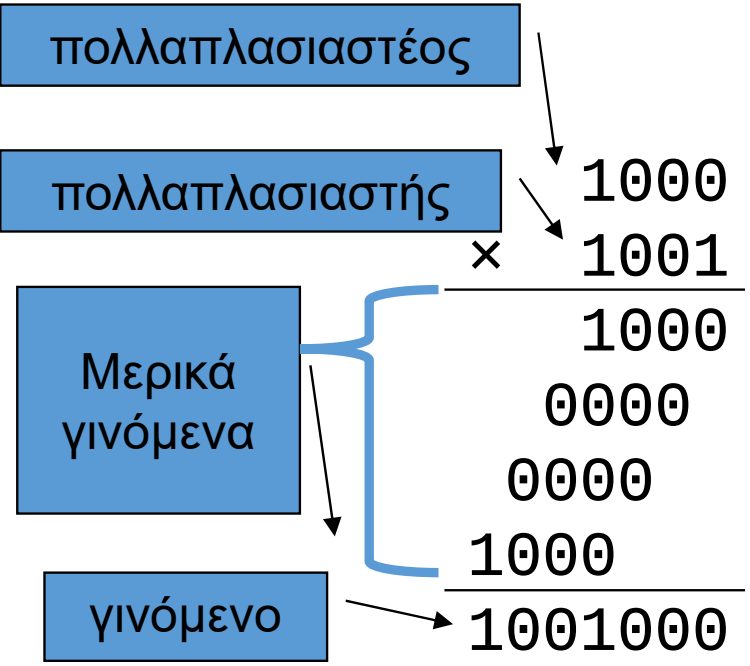
- Ξεκινάμε με τον πολ/σμό μεγάλου μήκους



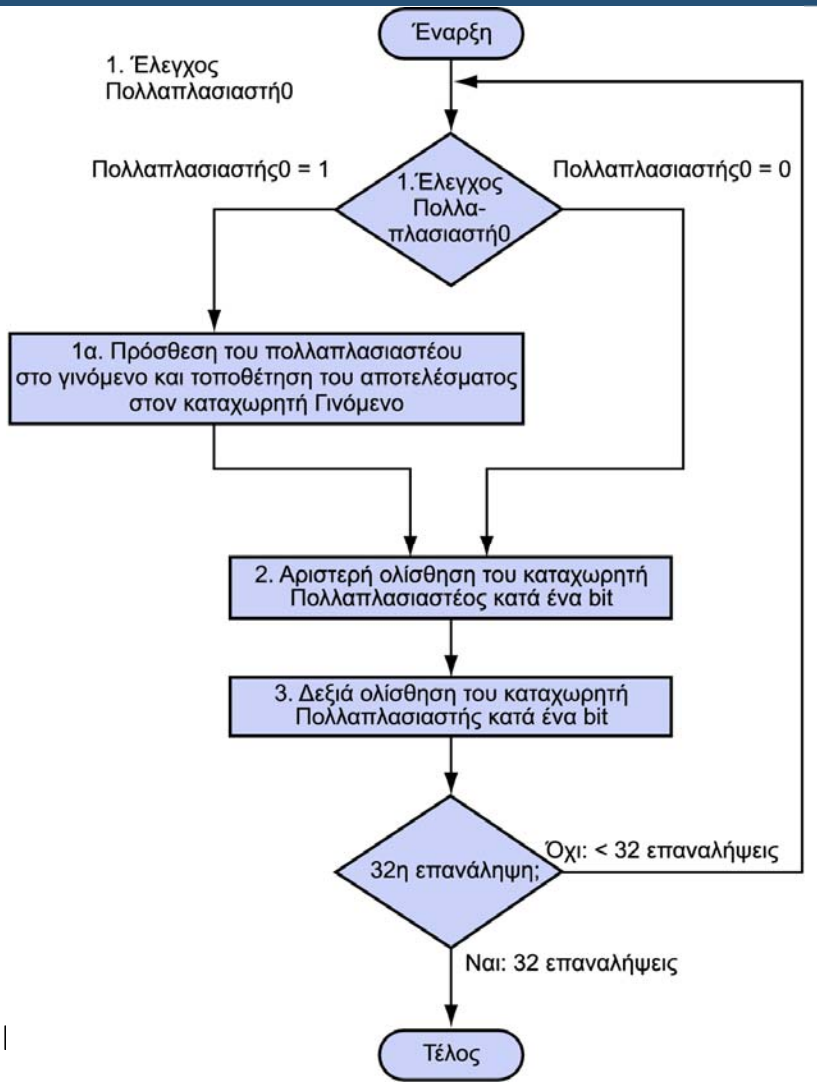


Το μήκος του γινομένου είναι το άθροισμα των μηκών των τελεστέων

Πολλαπλασιασμός – Διάγραμμα Ροής

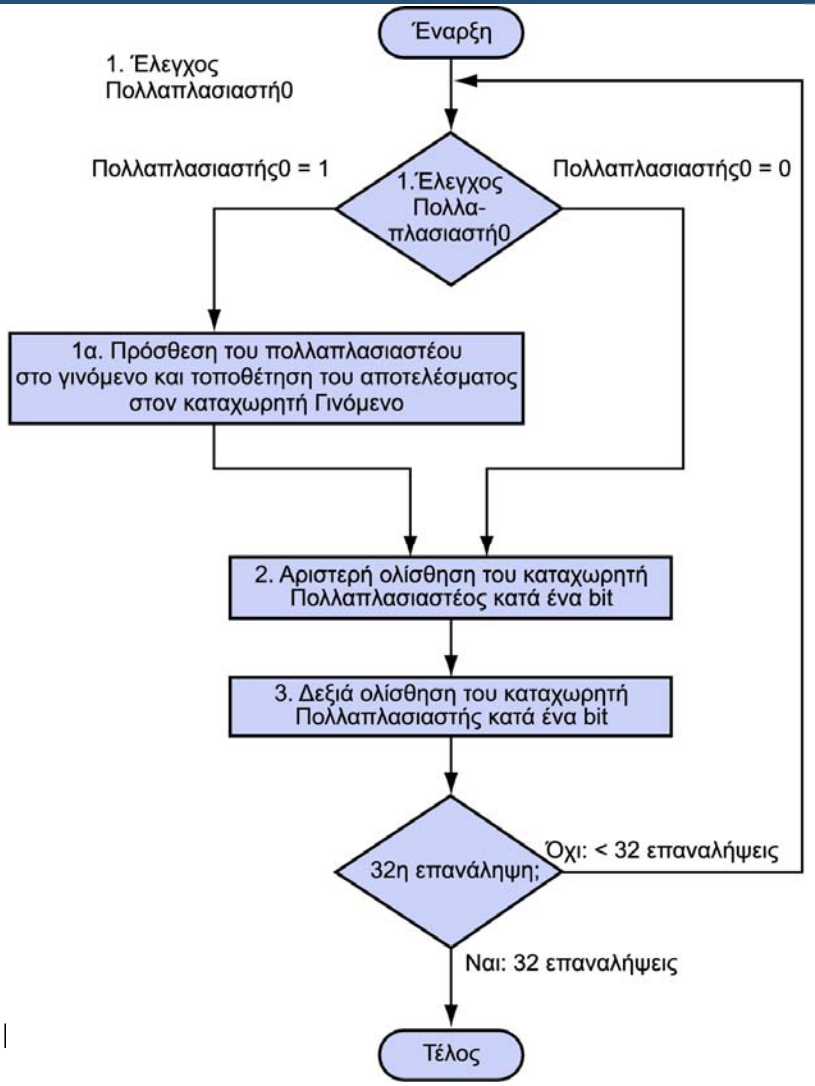


Το μήκος του γινομένου είναι το άθροισμα των μηκών των τελεστέων

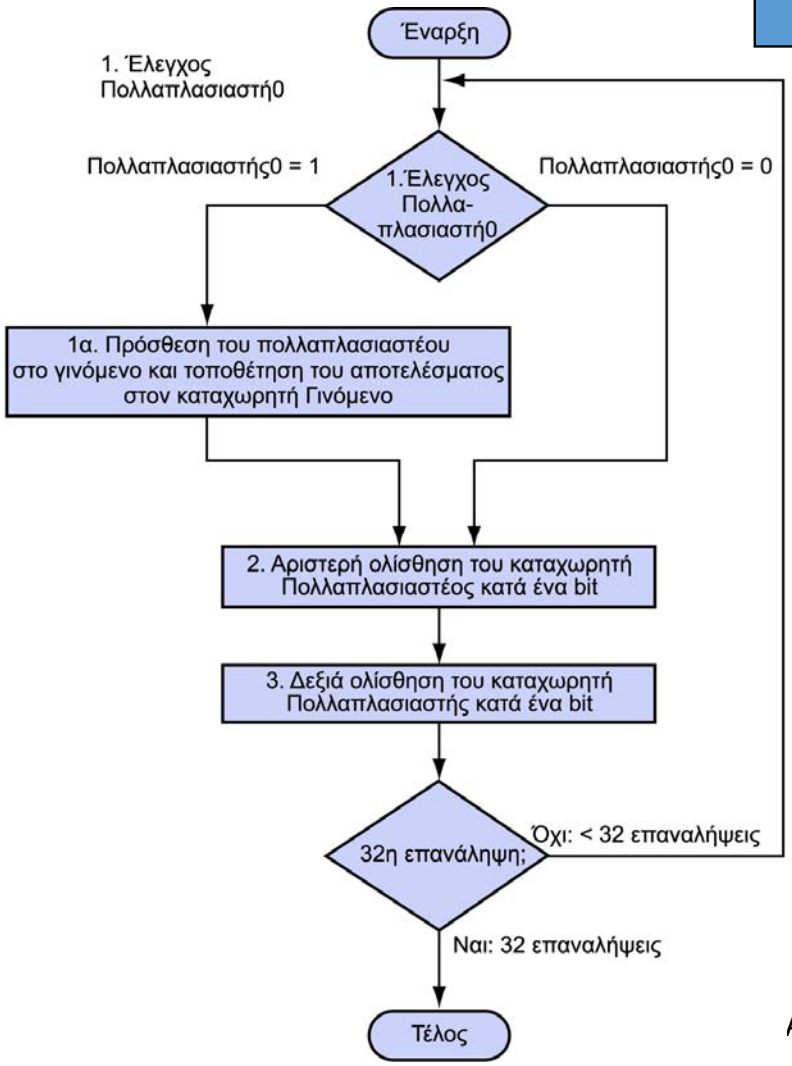


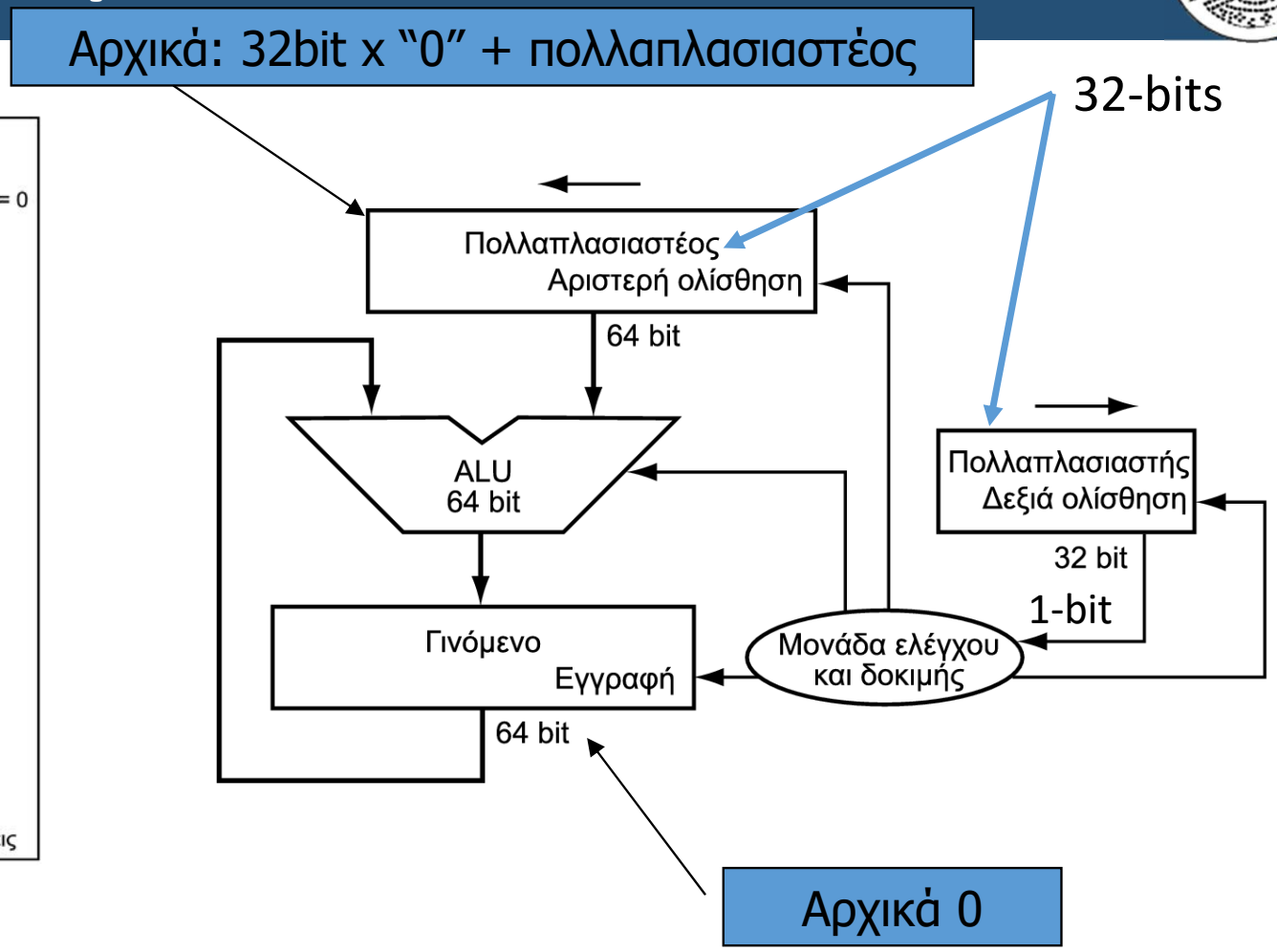
Πολλαπλασιασμός – Διάγραμμα Ροής

- **Τι υλικό (registers, adders, shifters) χρειάζομαι για να το υλοποιήσω?**
- **Πόσους registers και τι μεγέθους?**
- **Πόσους κύκλους θα χρειαστεί μέχρι το αποτέλεσμα να είναι έτοιμο?**



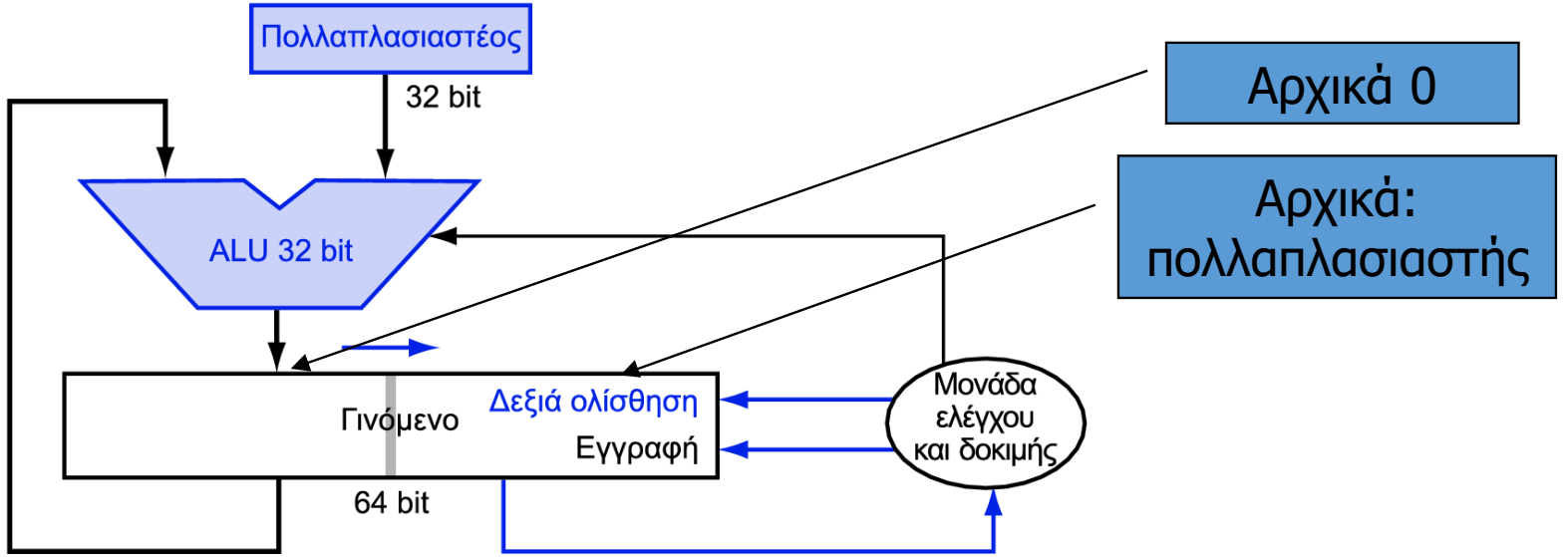
Υλικό πολλαπλασιασμού





Βελτιστοποιημένος πολλαπλασιαστής

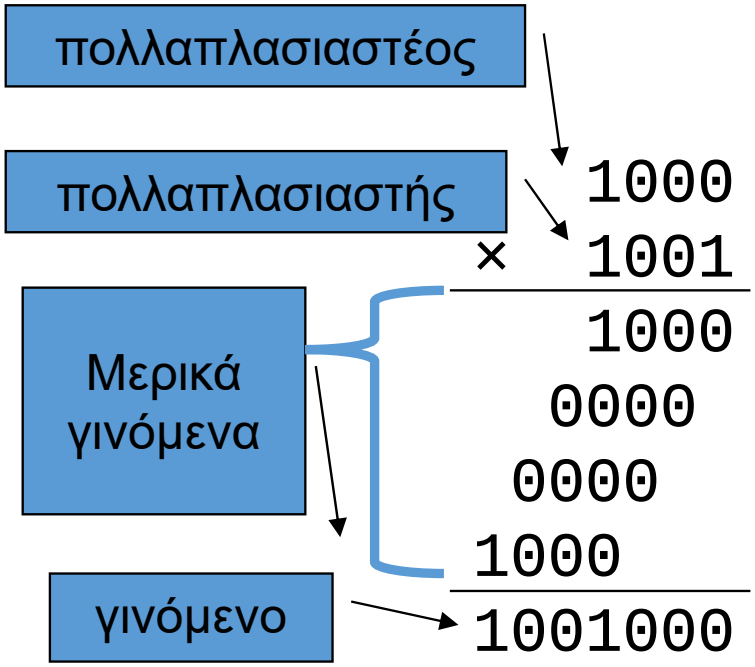
- **Εκτέλεση βημάτων παράλληλα: πρόσθεση/ολίσθηση**



- **Ένας κύκλος ανά πρόσθεση μερικού γινομένου**
  - **Είναι εντάξει, αν η συχνότητα εμφάνισης του πολλαπλασιασμού είναι χαμηλή**

Ταχύτερος πολλαπλασιαστής

- **Χρησιμοποιεί πολλούς αθροιστές**
  - **Συμβιβασμός κόστους/απόδοσης**



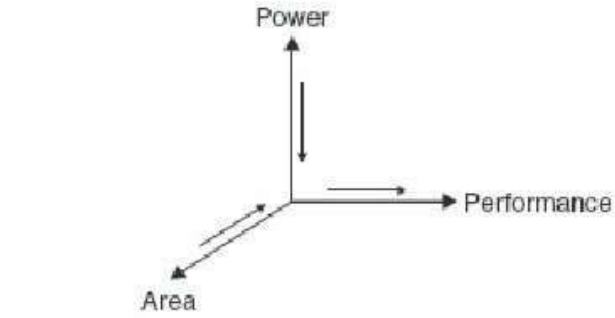
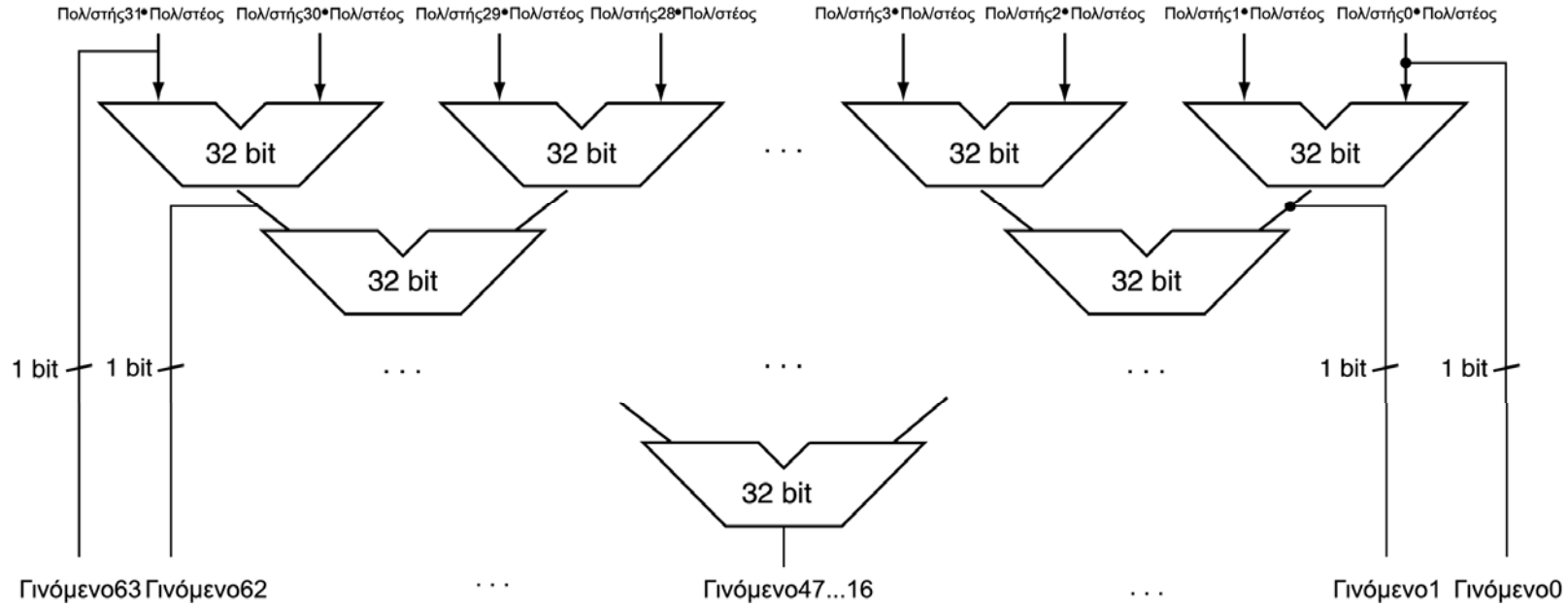


Figure 3.16: Performance, Area, and Power Trade-Off

Ταχύτερος πολλαπλασιαστής

- Χρησιμοποιεί πολλούς αθροιστές
  - Συμβιβασμός κόστους/απόδοσης



- Μπορεί να υλοποιηθεί με διοχέτευση (pipeline)
  - Πολλοί πολλαπλασιασμοί εκτελούνται παράλληλα

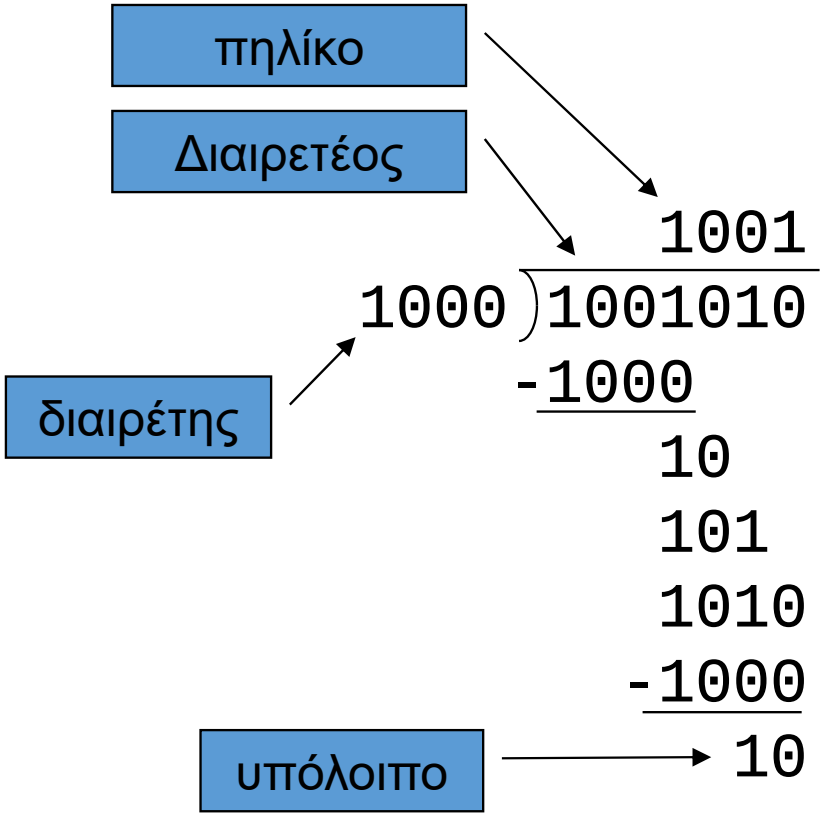
Πολλαπλασιασμός με Αρνητικούς Αριθμούς

- Ο προηγούμενος τρόπος δεν λειτουργεί
- Λύση 1
  - Μετατροπή σε θετικό (αν χρειάζεται)
  - Πολλαπλασιασμός (όπως εξηγήσαμε)
  - Ετερόσημοι → αρνητικό αποτέλεσμα
- Άλλες λύσεις
  - Booth’s algorithm, ….,

Πολλαπλασιασμός στον MIPS

- Δύο καταχωρητές των 32 bit για το γινόμενο
  - HI: τα περισσότερο σημαντικά 32 bit
  - LO: τα λιγότερο σημαντικά 32 bit
- Εντολές
  - mult rs, rt / multu rs, rt
    - γινόμενο των 64 bit στους HI/LO
  - mfhi rd / mflo rd
    - Μεταφορά από (move from) του HI/LO στον rd
    - Μπορούμε να ελέγξουμε τη τιμή του HI για να δούμε αν το γινόμενο ξεπερνά τα 32 bit
  - mul rd, rs, rt
    - Τα λιγότερο σημαντικά 32 bit του γινομένου –> rd

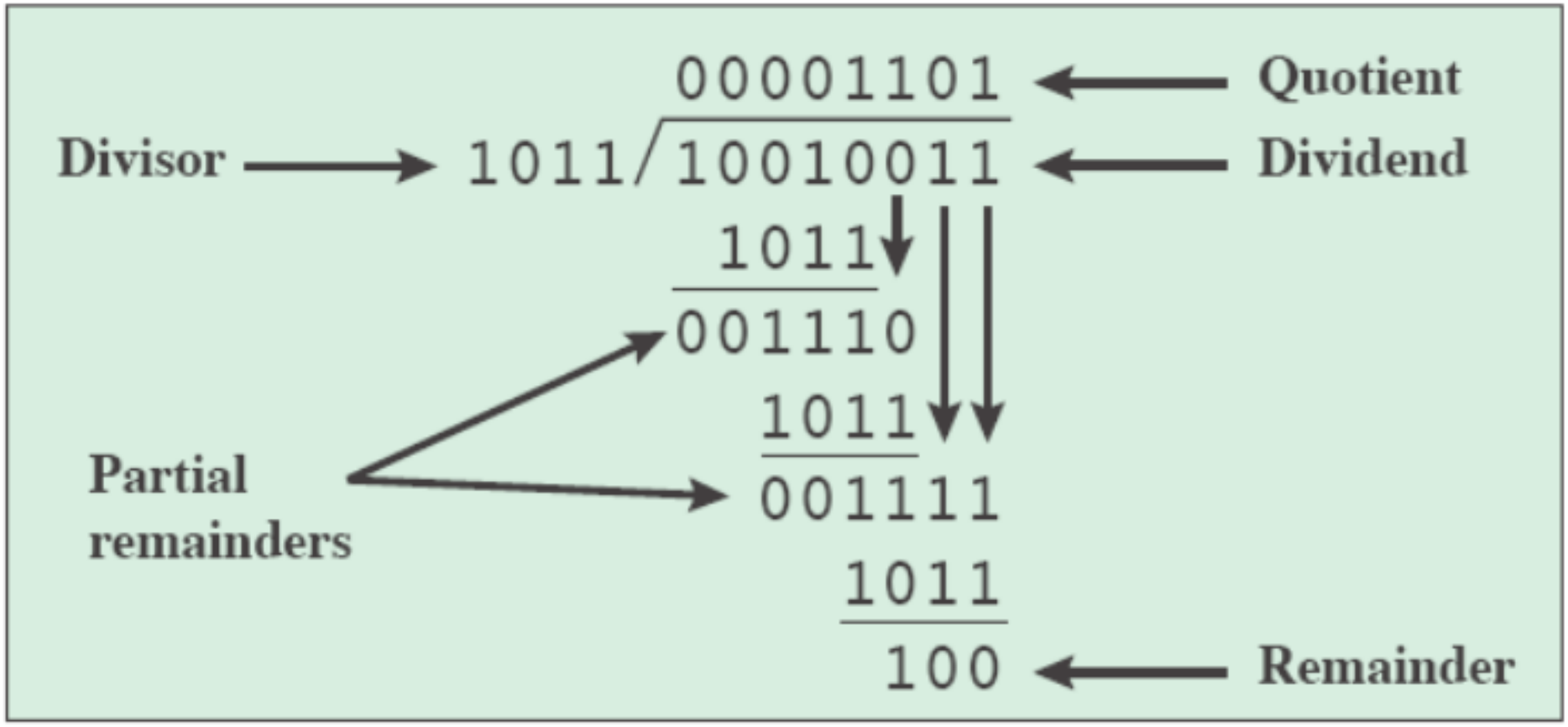
Διαίρεση



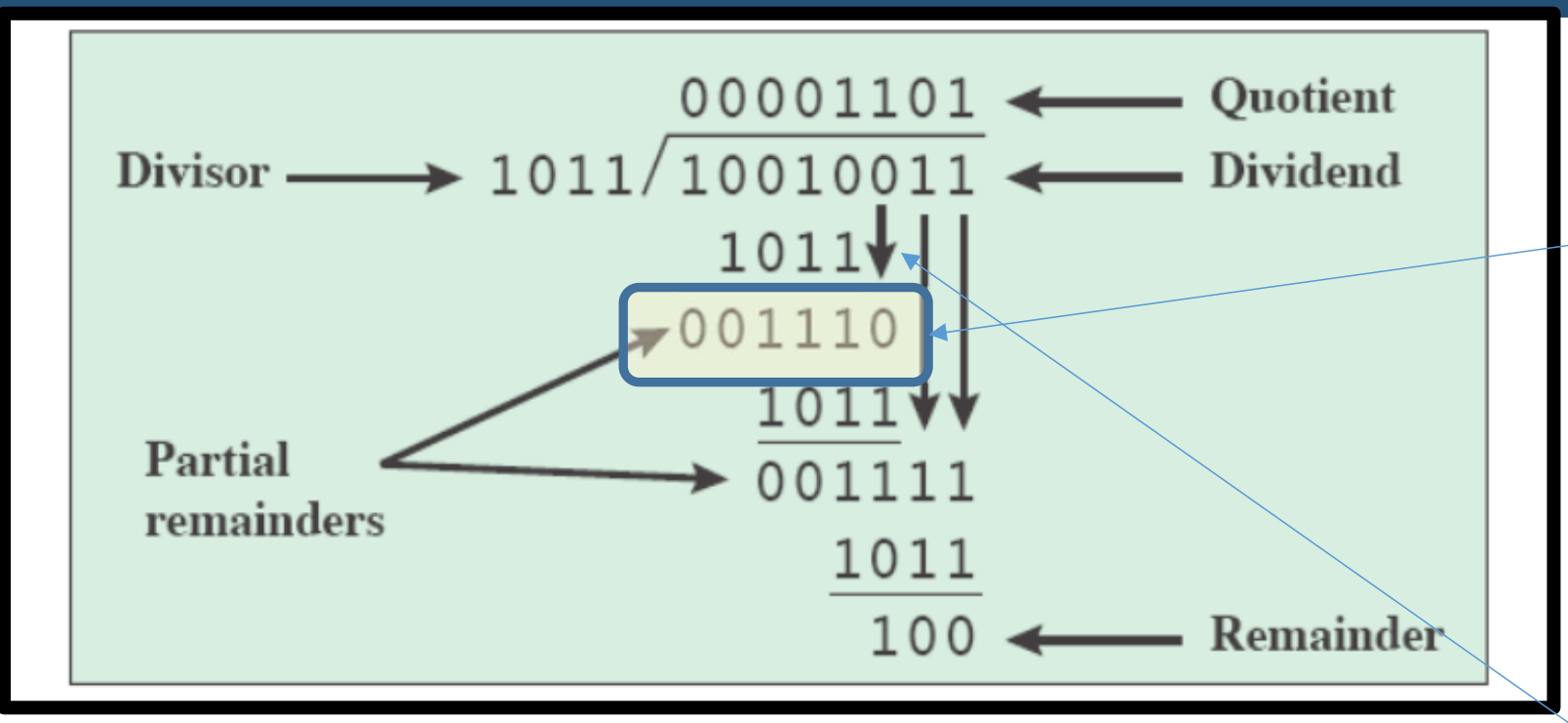
τελεστέοι των *n* bit δίνουν πηλίκο και υπόλοιπο των *n* bit

- Έλεγχος για μηδενικό διαιρέτη
- Διαίρεση μεγάλου μήκους
  - Αν διαιρέτης ≤ από τον διαιρετέο
    - 1 bit στο πηλίκο, αφαίρεση
  - Αλλιώς (ή διαιρέτης ≤ από τα bits του διαιρετέου)
    - 0 bit στο πηλίκο, κατέβασμα του επόμενου bit του διαιρετέου
- Προσημασμένη διαίρεση
  - Κάνε τη διαίρεση με τις απόλυτες τιμές
  - Ρύθμισε το πρόσημο του πηλίκου και του υπολοίπου όπως απαιτείται

Διαίρεση (Παράδειγμα)



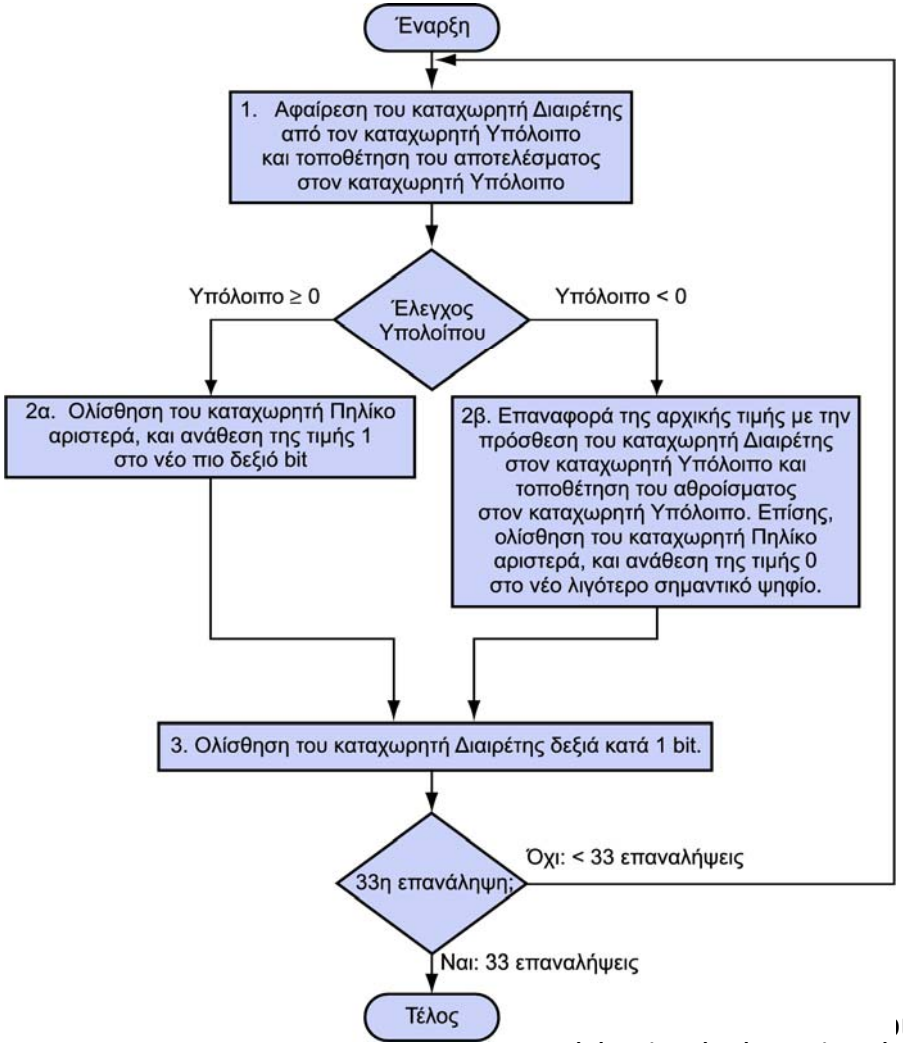
Διαίρεση (Παράδειγμα)

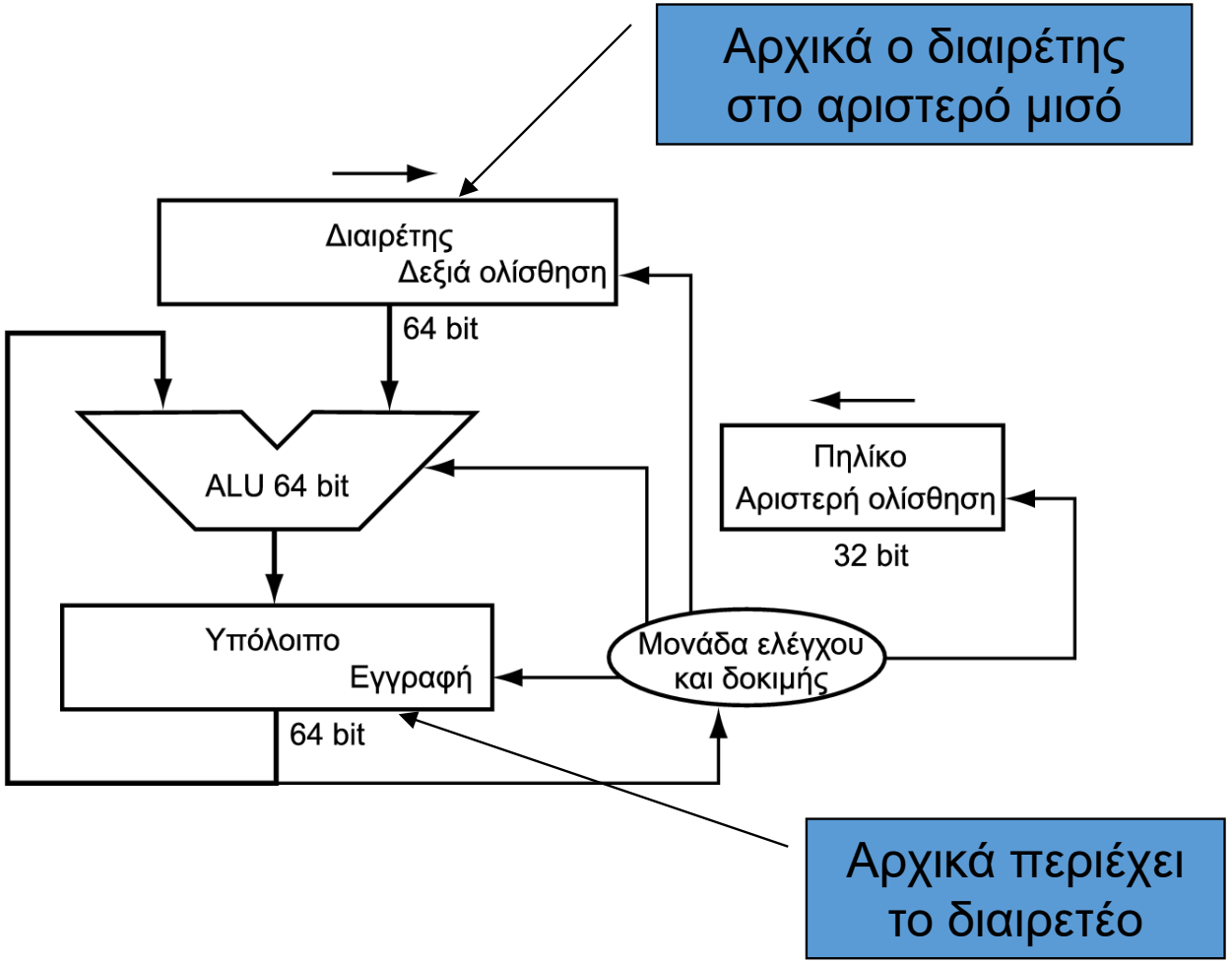


Ας το ονομάσουμε αυτό υπόλοιπο

Στην ουσία συνεχίζουμε την ολίσθηση του υπολοίπου μέχρι το αποτέλεσμα είναι μεγαλύτερο του 0

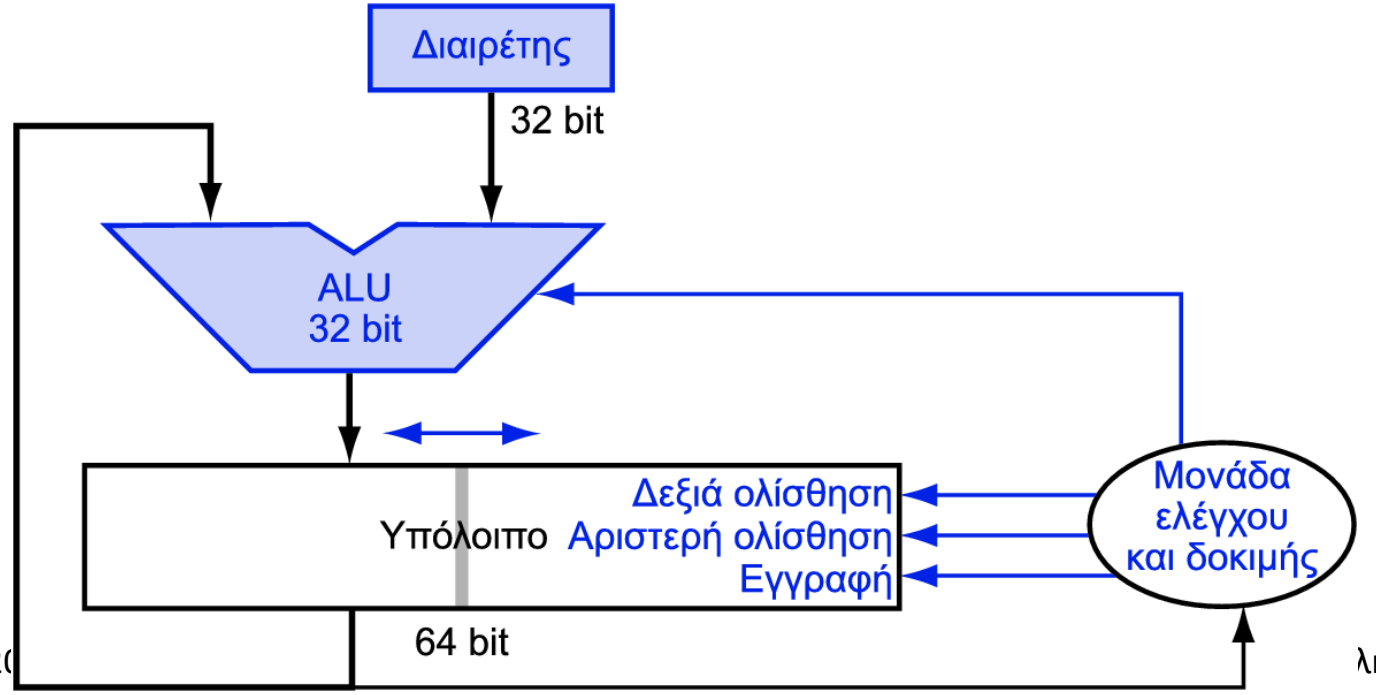
Υλικό διαίρεσης





Βελτιστοποιημένος διαιρέτης

- Ένας κύκλος για κάθε αφαίρεση μερικού υπολοίπου
- Μοιάζει πολύ με πολλαπλασιαστή!
  - Το ίδιο υλικό μπορεί να χρησιμοποιηθεί και για τις δύο πράξεις



Ταχύτερη διαίρεση

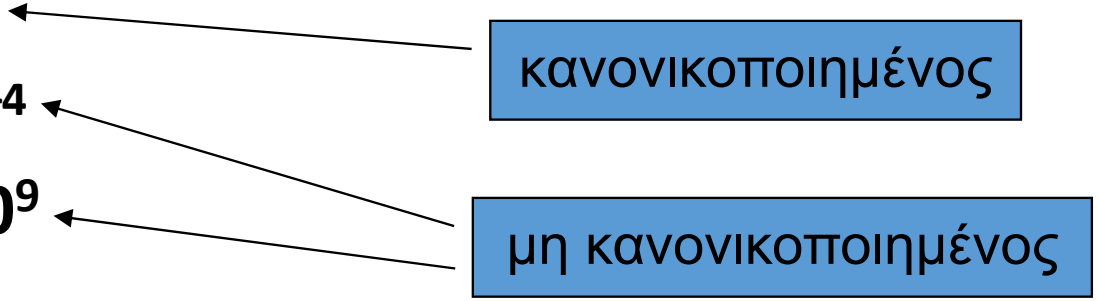
- **Δεν μπορεί να χρησιμοποιηθεί παράλληλο υλικό όπως στον πολλαπλασιαστή**
  - **Η αφαίρεση εκτελείται υπό συνθήκη, ανάλογα με το πρόσημο του υπολοίπου**
- **Ταχύτεροι διαιρέτες (π.χ. διαίρεση SRT) δημιουργούν πολλά bit του πηλίκου σε κάθε βήμα**
  - **Και πάλι απαιτούνται πολλά βήματα**

Διαίρεση στο MIPS

- Χρήση των καταχωρητών HI/LO για το αποτέλεσμα
  - HI: υπόλοιπο 32 bit
  - LO: πηλίκο 32 bit
- Εντολές
  - div rs, rt / divu rs, rt
  - Όχι έλεγχος για υπερχείλιση ή διαίρεση με το 0
    - Το λογισμικό πρέπει να εκτελεί τους ελέγχους αν αυτό απαιτείται
  - Χρήση των mfhi, mflo για προσπέλαση του αποτελέσματος

Κινητή υποδιαστολή

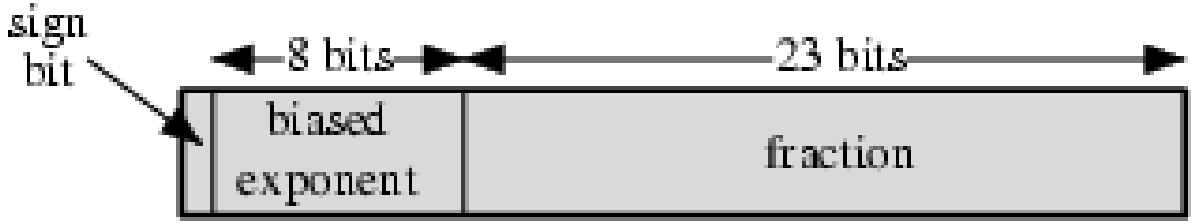
- **Αναπαράσταση για μη ακεραίους αριθμούς**
  - **Περιλαμβάνει και πολύ μικρούς και πολύ μεγάλους αριθμούς**
- **Όπως η επιστημονική σημειογραφία (scientific notation)**
- **-2.34×10^{56}**
- **$+0.002 \times 10^{-4}$**
- **$+987.02 \times 10^{9}$**
- **Σε δυαδικό**
  - **$\pm 1.xxxxxxx_2 \times 2^{yyyy}$**
- **Οι τύποι float και double της C**



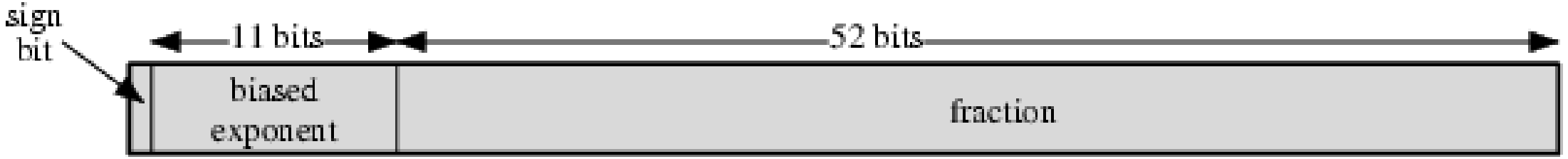
Πρότυπο κινητής υποδιαστολής

- Ορίζεται από το IEEE Std 754-1985
- Αναπτύχθηκε ως λύση στην απόκλιση των αναπαραστάσεων
  - Ζητήματα φορητότητας (portability) για τον κώδικα επιστημονικών εφαρμογών
- Πλέον είναι σχεδόν οικουμενικά αποδεκτό
- Δύο αναπαραστάσεις κινητής υποδιαστολής (floating point)
  - Απλή ακρίβεια – single precision (32 bit) – το float στην C
  - Διπλή ακρίβεια – double precision (64 bit) – το double στην C

IEEE 754 Formats



(a) Single format



(b) Double format

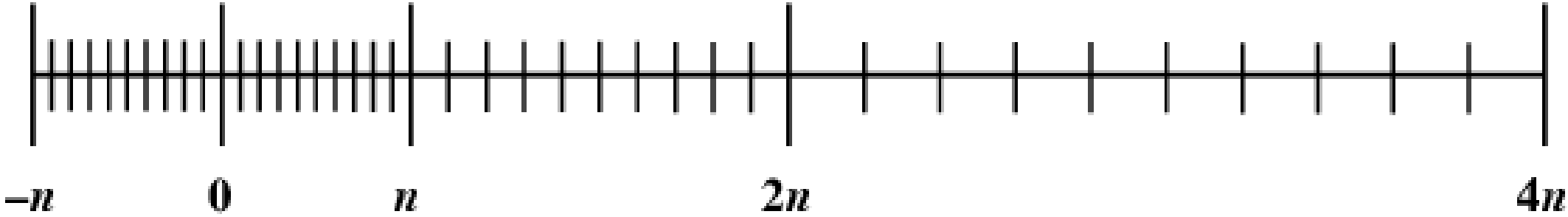
Μορφή κινητής υποδιαστολής IEEE

	single: 8 bit double: 11 bit	single: 23 bit double: 52 bit
S	Εκθέτης	Κλάσμα

$$x = (-1)^S \times (1 + \text{Κλάσμα}) \times 2^{(\text{Εκθέτης} - \text{Πόλωση})}$$

- **Εκθέτης (exponent) – Κλάσμα (fraction)**
- **S: bit προσήμου (0 → θετικός, 1 → αρνητικός)**
- **Κανονικοποίηση του σημαντικού (significand): 1.0 ≤ |significand| < 2.0**
  - **Έχει πάντα ένα αρχικό bit 1 πριν την υποδιαστολή, και συνεπώς δε χρειάζεται ρητή αναπαράστασή του («κρυμμένο» bit)**
  - **Το σημαντικό (significand) είναι το κλάσμα (fraction) μαζί με το κρυμμένο "1"**
- **Εκθέτης: αναπαράσταση «με υπέρβαση» (excess): πραγματικός εκθέτης + πόλωση (bias)**
  - **Εγγυάται ότι ο εκθέτης είναι απρόσημος**
  - **Απλή ακρίβεια: Πόλωση = 127 – Διπλή ακρίβεια: Πόλωση = 1023**

Παραδείγματα

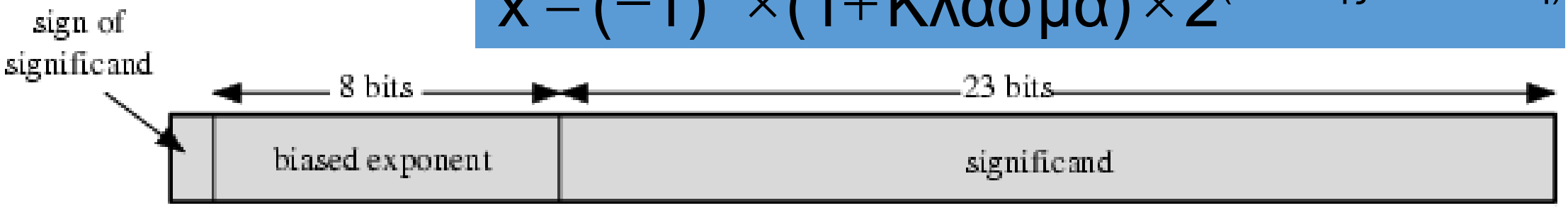


Προσοχή στην ακρίβεια σε σχέση με έναν int.

Έχουμε μόνο 2^{32} αριθμούς

Αναπαράσταση Αριθμών

$$x = (-1)^S \times (1 + \text{Κλάσμα}) \times 2^{(\text{Εκθέτης} - \text{Πόλωση})}$$



(a) Format

Πολωση: 127

147

```
 1.1010001 × 2^10100  = 0 10010011 10100010000000000000000 =  1.638125 × 2^20
-1.1010001 × 2^10100  = 1 10010011 10100010000000000000000 = -1.638125 × 2^20
 1.1010001 × 2^-10100 = 0 01101011 10100010000000000000000 =  1.638125 × 2^-20
-1.1010001 × 2^-10100 = 1 01101011 10100010000000000000000 = -1.638125 × 2^-20
```

107

(b) Examples

Εύρος απλής ακρίβειας

- Οι εκθέτες 00000000 και 11111111 δεσμεύονται
- Μικρότερη τιμή
  - Εκθέτης: 00000001 → πραγματικός εκθέτης = 1 – 127 = –126
  - Κλάσμα: 000…00 → σημαντικό = 1.0
  - $\pm 1.0 \times 2^{-126} \approx \pm 1.2 \times 10^{-38}$
- Μεγαλύτερη τιμή
  - Εκθέτης: 11111110 → πραγματικός εκθέτης = 254 – 127 = +127
  - Κλάσμα: 111…11 → σημαντικό ≈ 2.0
  - $\pm 2.0 \times 2^{+127} \approx \pm 3.4 \times 10^{+38}$

Εύρος διπλής ακρίβειας

Πόλωση

- Οι εκθέτες 0000…00 και 1111…11 δεσμεύονται
- Μικρότερη τιμή
  - Εκθέτης: 00000000001 → πραγματικός = 1 – 1023 = –1022
  - Κλάσμα: 000…00 → σημαντικό = 1.0
  - $\pm 1.0 \times 2^{-1022} \approx \pm 2.2 \times 10^{-308}$
- Μεγαλύτερη τιμή
  - Εκθέτης: 11111111110 → πραγματικός εκθέτης = 2046 – 1023 = +1023
  - Κλάσμα: 111…11 → σημαντικό ≈ 2.0
  - $\pm 2.0 \times 2^{+1023} \approx \pm 1.8 \times 10^{+308}$

Ακρίβεια κινητής υποδιαστολής

- **Σχετική ακρίβεια**
  - **Όλα τα bit του κλάσματος είναι σημαντικά**
  - **Απλή: περίπου 2^{-23}**
    - **Ισοδύναμο με $23 \times \log_{10}2 \approx 23 \times 0.3 \approx 7$ δεκαδικά ψηφία ακρίβειας**
  - **Διπλή: περίπου 2^{-52}**
    - **Ισοδύναμο με $52 \times \log_{10}2 \approx 52 \times 0.3 \approx 16$ δεκαδικά ψηφία ακρίβειας**

Παράδειγμα κινητής υποδιαστολής

- **Αναπαράσταση του –0.75**
  - **$-0.75 = (-1)^1 \times 0.11_2 \times 2^0 = (-1)^1 \times 1.1_2 \times 2^{-1}$**
  - **S = 1**
  - **Κλάσμα = $1000\ldots00_2$**
  - **Εκθέτης = –1 + Πόλωση**
    - **Απλή: –1 + 127 = 126 = 01111110_2**
    - **Διπλή: –1 + 1023 = 1022 = 01111111110_2**
  - **Απλή: 101111110100…00**
  - **Διπλή: 1011111111101000…00**

Παράδειγμα κινητής υποδιαστολής

- **Ποιος αριθμός αναπαρίσταται από τον απλής ακρίβειας κινητής υποδιαστολής αριθμό;**

  1100000001010000…00

  - **S = 1**
  - **Κλάσμα = $01000\ldots00_2$**
  - **Εκθέτης = $10000001_2 = 129$**

- **$x = (-1)^1 \times (1 + .01_2) \times 2^{(129 - 127)}$**

  $= (-1) \times 1.25 \times 2^2$

  $= -5.0$

Μη κανονικοποιημένοι (denormals)

- **Εκθέτης = 000…0 → το «κρυμμένο» bit είναι 0**

$$x = (-1)^S \times (0 + \text{Κλάσμα}) \times 2^{-\text{Πόλωση}}$$

- **Μικρότεροι από τους κανονικοποιημένους**

- **Denormal με κλάσμα = 000…0**

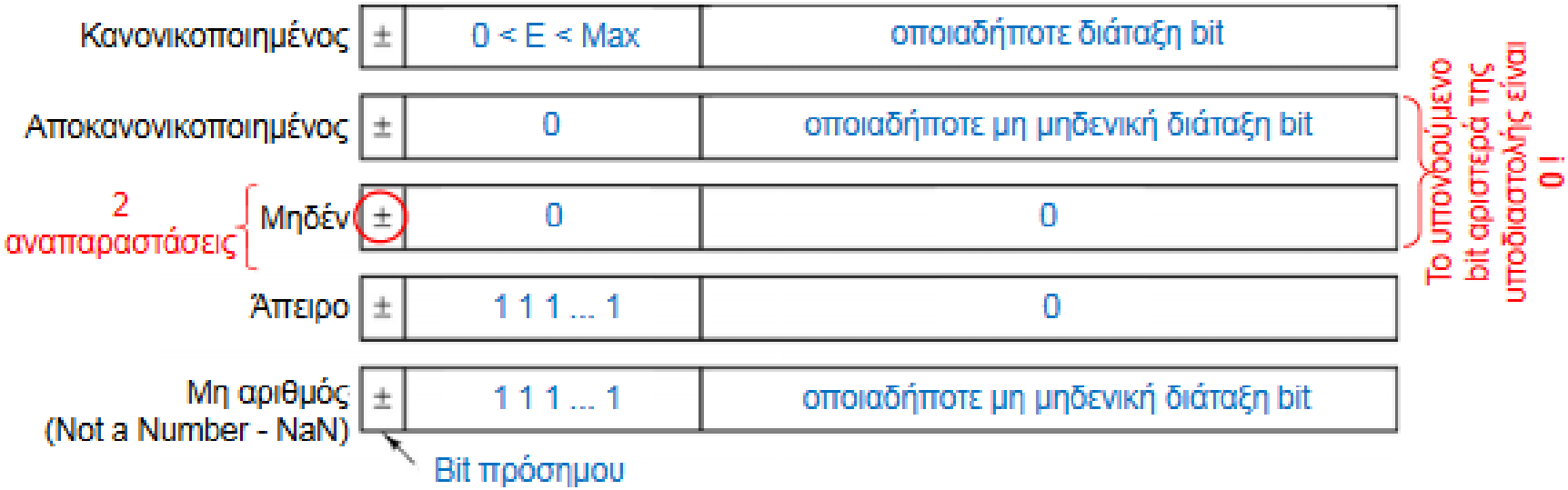
$$x = (-1)^S \times (0 + 0) \times 2^{-\text{Πόλωση}} = \pm\ 0.0$$

Δύο αναπαραστάσεις του 0.0!

Άπειρα και όχι αριθμοί (NaN)

- Εκθέτης = 111...1, Κλάσμα = 000...0
  - ±Άπειρο
  - Μπορεί να χρησιμοποιηθεί σε επόμενους υπολογισμούς
- Εκθέτης = 111...1, Κλάσμα ≠ 000...0
  - Όχι αριθμός (Not-a-Number – NaN)
  - Δείχνει ένα άκυρο ή απροσδιόριστο αποτέλεσμα
    - π.χ., 0.0 / 0.0
  - Μπορεί να χρησιμοποιηθεί σε επόμενους υπολογισμούς

Συνοπτικά



Με στόχο τη μείωση του προβλήματος ανεπάρκειας (underflow) η IEEE εισήγαγε τους αποκανονικοποιημένους (denormalized) αριθμούς. Σε αυτή την περίπτωση το υπονοούμενο bit αριστερά της υποδιαστολής από 1 γίνεται 0.

- Ο μικρότερος θετικός κανονικοποιημένος αριθμός απλής ακρίβειας έχει E=1 και F=0 και είναι ο $1,0 \times 2^{-126}$.
- Ο μεγαλύτερος θετικός αποκανονικοποιημένος αριθμός απλής ακρίβειας έχει E=0 και F=111...1 και είναι ο $0,9999999 \times 2^{-127}$.
- Ο μικρότερος θετικός αποκανονικοποιημένος αριθμός απλής ακρίβειας έχει E=0 και F=00...01 και είναι ο $2^{-23} \times 2^{-127} = 2^{-150}$.

Πρόσθεση κινητής υποδιαστολής

- Ένα δεκαδικό παράδειγμα με 4 ψηφία
  - $9.999 \times 10^{1} + 1.610 \times 10^{-1}$
- 1. Ευθυγράμμιση υποδιαστολών
  - Ολίσθηση αριθμού με το μικρότερο εκθέτη
  - $9.999 \times 10^{1} + 0.016 \times 10^{1}$
- 2. Πρόσθεση σημαντικών
  - $9.999 \times 10^{1} + 0.016 \times 10^{1} = 10.015 \times 10^{1}$
- 3. Κανονικοποίηση αποτελέσματος & έλεγχος υπερχείλισης/ανεπάρκειας
  - 1.0015×10^{2}
- 4. Στρογγυλοποίηση και επανακανονικοποιήση αν είναι απαραίτητο
  - 1.002×10^{2}

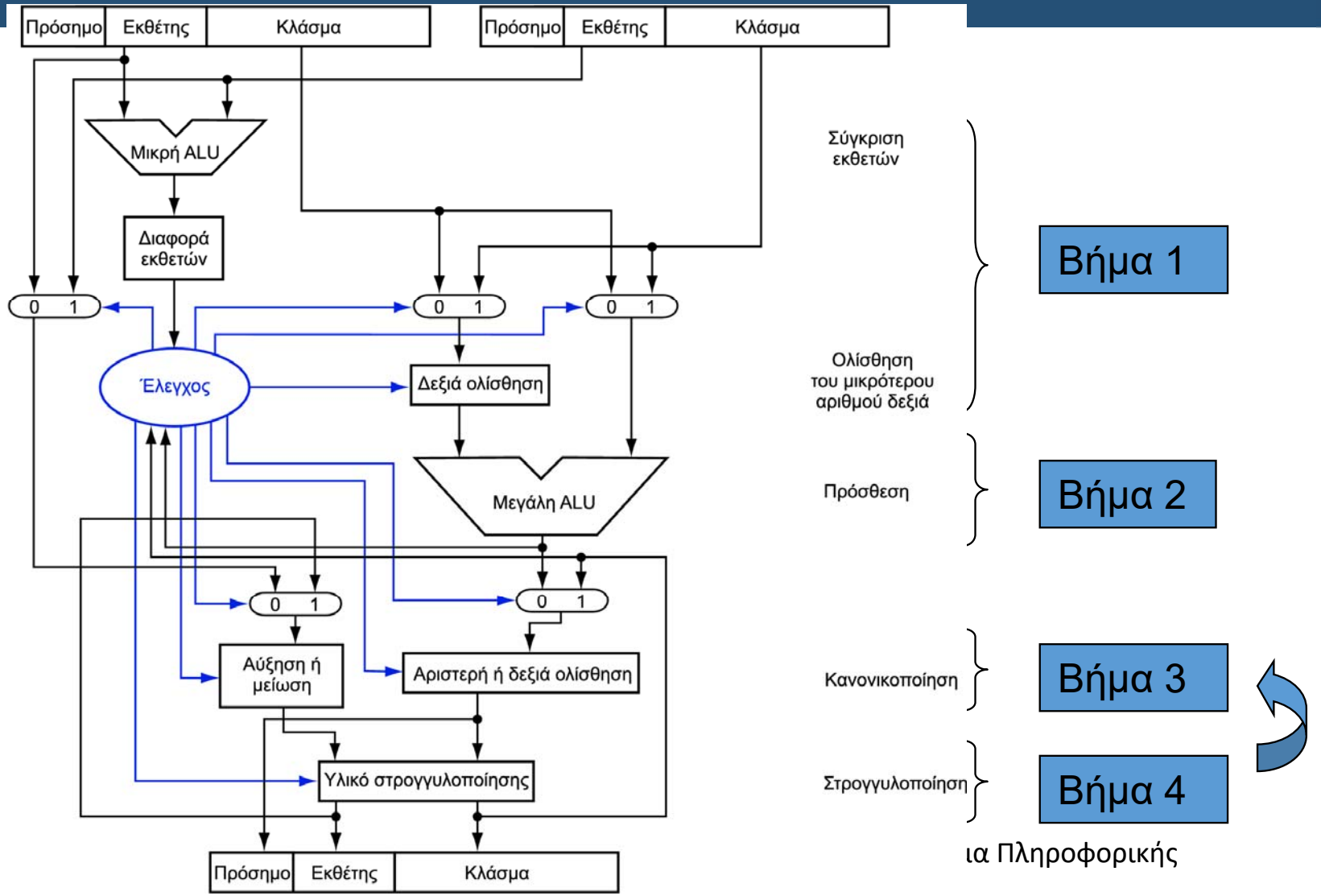
Πρόσθεση κινητής υποδιαστολής

- Τώρα ένα δυαδικό παράδειγμα με 4 ψηφία
  - $1.000_2 \times 2^{-1} + -1.110_2 \times 2^{-2}$ (0.5 + –0.4375)
- 1. Ευθυγράμμιση υποδιαστολών
  - Ολίσθηση αριθμού με το μικρότερο εκθέτη
  - $1.000_2 \times 2^{-1} + -0.111_2 \times 2^{-1}$
- 2. Πρόσθεση σημαντικών
  - $1.000_2 \times 2^{-1} + -0.111_2 \times 2^{-1} = 0.001_2 \times 2^{-1}$
- 3. Κανονικοποίηση αποτελέσματος και έλεγχος υπερχείλισης/ανεπάρκειας
  - $1.000_2 \times 2^{-4}$, χωρίς υπερχείλιση/ανεπάρκεια
- 4. Στρογγυλοποίηση και επανακανονικοποιήση αν είναι απαραίτητο
  - $1.000_2 \times 2^{-4}$ (καμία αλλαγή) = 0.0625

Υλικό αθροιστή κιν. υποδ.

- Πολύ πιο πολύπλοκο από του ακέραιου αθροιστή
- Για να γίνει σε έναν κύκλο πρέπει να έχει πολύ μεγάλη διάρκεια
  - Πολύ μεγαλύτερη από τις ακέραιες λειτουργίες
  - Το πιο αργό ρολόι θα επιβάρυνε όλες τις εντολές
- Ο αθροιστής κινητής υποδιαστολής συνήθως παίρνει πολλούς κύκλους
  - Μπορεί να υπολοποιηθεί με διοχέτευση

Υλικό αθροιστή κιν. υποδ.



Πολλαπλασιασμός κιν. υποδ.

- Ένα δεκαδικό παράδειγμα με 4 ψηφία
  - $1.110 \times 10^{10} \times 9.200 \times 10^{-5}$
- 1. Πρόσθεση εκθετών
  - Για πολωμένους εκθέτες, αφαίρεση της πόλωσης από το άθροισμα
  - Νέος εκθέτης = 10 + –5 = 5
- 2. Πολλαπλασιασμός σημαντικών
  - $1.110 \times 9.200 = 10.212 \rightarrow 10.212 \times 10^{5}$
- 3. Κανονικοποίηση αποτελέσματος & έλεγχος υπερχείλισης/ανεπάρκειας
  - 1.0212×10^{6}
- 4. Στρογγυλοποίηση και επανακανονικοποίηση αν είναι απαραίτητο
  - 1.021×10^{6}
- 5. Καθορισμός του προσήμου του αποτελέσματος από τα πρόσημα των τελεστέων
  - $+1.021 \times 10^{6}$

Πολλαπλασιασμός κιν. υποδ.

- Τώρα ένα δυαδικό παράδειγμα με 4 ψηφία
  - $1.000_2 \times 2^{-1} \times -1.110_2 \times 2^{-2}$ (0.5 × –0.4375)
- 1. Πρόσθεση εκθετών
  - Χωρίς πόλωση: –1 + –2 = –3
  - Με πόλωση: (–1 + 127) + (–2 + 127) → –3 + 254 – 127 = –3 + 127
- 2. Πολλαπλασιασμός σημαντικών
  - $1.000_2 \times 1.110_2 = 1.110_2$ → $1.110_2 \times 2^{-3}$
- 3. Κανονικοποίηση αποτελέσματος και έλεγχος υπερχείλισης/ανεπάρκειας
  - $1.110_2 \times 2^{-3}$ (καμία αλλαγή) χωρίς υπερχείλιση/ανεπάρκεια
- 4. Στρογγυλοποίηση και επανακανονικοποίηση αν είναι απαραίτητο
  - $1.110_2 \times 2^{-3}$ (καμία αλλαγή)
- 5. Καθορισμός προσήμου: +ve × –ve → –ve
  - $-1.110_2 \times 2^{-3} = -0.21875$

Υλικό αριθμητικής κιν. υποδ.

- Ο πολλαπλασιαστής ΚΥ έχει παρόμοια πολυπλοκότητα με τον αθροιστή ΚΥ
  - Αλλά χρησιμοποιεί πολλαπλασιαστή για τα σημαντικά αντί για αθροιστή
- Το υλικό αριθμητικής κιν. υποδ. συνήθως εκτελεί
  - Πρόσθεση, αφαίρεση, πολλαπλασιασμό, διαίρεση, αντίστροφο, τετραγωνική ρίζα
  - Μετατροπή ΚΥ → ακέραιο
- Οι λειτουργίες συνήθως διαρκούν πολλούς κύκλους
  - Μπορούν να υπολοποιηθούν με διοχέτευση

Εντολές ΚΥ στο MIPS

- Το υλικό ΚΥ είναι ο συνεπεξεργαστής (coprocessor)
  - Επιπρόσθετος επεξεργαστής που επεκτείνει την αρχιτεκτονική συνόλου εντολών
- Ξεχωριστοί καταχωρητές ΚΥ
  - 32 απλής ακρίβειας: $f0, $f1, … $f31
  - Ζευγάρια για διπλή ακρίβεια: $f0/$f1, $f2/$f3, …
    - Η έκδοση 2 του συνόλου εντολών MIPS υποστηρίζει 32 × 64 bit καταχωρητές ΚΥ
- Εντολές ΚΥ επενεργούν μόνο σε καταχωρητές ΚΥ
  - Γενικά τα προγράμματα δεν εκτελούν ακέραιες πράξεις σε δεδομένα ΚΥ, ή αντίστροφα
  - Περισσότεροι καταχωρητές με ελάχιστη επίδραση στο μέγεθος του κώδικα
- Εντολές φόρτωσης και αποθήκευσης ΚΥ
  - lwc1, ldc1, swc1, sdc1
    - π.χ., ldc1 $f8, 32($sp)

Εντολές ΚΥ στον MIPS

- **Αριθμητική απλής ακρίβειας**
  - **add.s, sub.s, mul.s, div.s**
    - **π.χ., add.s $f0, $f1, $f6**
- **Αριθμητική διπλής ακρίβειας**
  - **add.d, sub.d, mul.d, div.d**
    - **π.χ., mul.d $f4, $f4, $f6**
- **Σύγκριση απλής και διπλής ακρίβειας**
  - **c.xx.s, c.xx.d (xx είναι eq, lt, le, …)**
  - **Δίνει τη τιμή 1 ή 0 σε bit κωδικών συνθήκης ΚΥ (FP condition-code bit)**
    - **π.χ. c.lt.s $f3, $f4**
- **Διακλάδωση σε αληθή ή ψευδή κωδικό συνθήκης ΚΥ**
  - **bc1t, bc1f**
    - **π.χ., bc1t TargetLabel**

Παραδειγμα ΚΥ: βαθμοί °F σε °C

- **Κώδικας C:**

```
float f2c (float fahr) {
  return ((5.0/9.0)*(fahr - 32.0));
}
```

  - **fahr στον $f12, αποτέλεσμα στον $f0, οι σταθερές στο χώρο της καθολικής μνήμης**

- **Μεταγλωττισμένος κώδικας MIPS:**

```
f2c:    lwc1  $f16, const5($gp)
        lwc2  $f18, const9($gp)
        div.s $f16, $f16, $f18
        lwc1  $f18, const32($gp)
        sub.s $f18, $f12, $f18
        mul.s $f0,  $f16, $f18
        jr    $ra
```

Παράδειγμα ΚΥ: Πολλαπλασιασμός πινάκων

X = X + Y × Z

- **Όλοι πίνακες 32 × 32, με στοιχεία 64 bit διπλής ακρίβειας**

Κώδικας C:

```
void mm (double x[][],double y[][],double z[][]){
  int i, j, k;
  for (i = 0; i! = 32; i = i + 1)
    for (j = 0; j! = 32; j = j + 1)
      for (k = 0; k! = 32; k = k + 1)
        x[i][j] = x[i][j] + y[i][k] * z[k][j];
}
```

- **Διευθύνσεις των x, y, z στους $a0, $a1, $a2, και των i, j, k στους $s0, $s1, $s2**

Παράδειγμα ΚΥ: Πολλαπλασιασμός πινάκων

- **Κώδικας MIPS:**

```
      li   $t1, 32          # $t1 = 32 (row size/loop end)
      li   $s0, 0           # i = 0; initialize 1st for loop
L1:   li   $s1, 0           # j = 0; restart 2nd for loop
L2:   li   $s2, 0           # k = 0; restart 3rd for loop
      sll  $t2, $s0, 5      # $t2 = i * 32 (size of row of x)
      addu $t2, $t2, $s1    # $t2 = i * size(row) + j
      sll  $t2, $t2, 3      # $t2 = byte offset of [i][j]
      addu $t2, $a0, $t2    # $t2 = byte address of x[i][j]
      l.d  $f4, 0($t2)      # $f4 = 8 bytes of x[i][j]
L3:   sll  $t0, $s2, 5      # $t0 = k * 32 (size of row of z)
      addu $t0, $t0, $s1    # $t0 = k * size(row) + j
      sll  $t0, $t0, 3      # $t0 = byte offset of [k][j]
      addu $t0, $a2, $t0    # $t0 = byte address of z[k][j]
      l.d  $f16, 0($t0)     # $f16 = 8 bytes of z[k][j]
    …
```

Παράδειγμα ΚΥ: Πολλαπλασιασμός πινάκων

....

```
sll   $t0, $s0, 5       # $t0 = i*32 (size of row of y)
addu  $t0, $t0, $s2     # $t0 = i*size(row) + k
sll   $t0, $t0, 3       # $t0 = byte offset of [i][k]
addu  $t0, $a1, $t0     # $t0 = byte address of y[i][k]
l.d   $f18, 0($t0)      # $f18 = 8 bytes of y[i][k]
mul.d $f16, $f18, $f16  # $f16 = y[i][k] * z[k][j]
add.d $f4, $f4, $f16    # f4=x[i][j] + y[i][k]*z[k][j]
addiu $s2, $s2, 1       # $k k + 1
bne   $s2, $t1, L3      # if (k != 32) go to L3
s.d   $f4, 0($t2)       # x[i][j] = $f4
addiu $s1, $s1, 1       # $j = j + 1
bne   $s1, $t1, L2      # if (j != 32) go to L2
addiu $s0, $s0, 1       # $i = i + 1
bne   $s0, $t1, L1      # if (i != 32) go to L1
```

Διερμηνεία των δεδομένων

- Τα bit δεν έχουν έμφυτη σημασία
  - Η διερμηνεία εξαρτάται από τις εντολές που εφαρμόζονται
- Αναπαράσταση των αριθμών στους υπολογιστές
  - Πεπερασμένο εύρος και ακρίβεια
  - Πρέπει να λαμβάνονται υπόψη στα προγράμματα

Προσεταιριστικότητα

- **Δεν ισχύει πάντα στις πράξεις κινητής υποδιαστολής**

Άσκηση (1)

- **Δίνονται οι αριθμοί κινητής υποδιαστολής**

```
Y = 0 0111 1111 1100 0000 … …
X = 0 1001 1011 0000 0100 … …
Z = 1 1001 1011 0000 0100 … …
```

- **Να υπολογίσετε την τιμή των εκφράσεων Y + (X + Z) και (Y + X) + Z) και να συγκρίνετε τα αποτέλεσμα και να τα δικαιολογήσετε**

Άσκηση (2)

```
Y = 0 0111 1111 1100 0000 … …
X = 0 1001 1011 0000 0100 … …
Z = 1 1001 1011 0000 0100 … …
```

```
Πρόσημο Y = Θετικός
Εκθέτης Y = 127 (δεκαδικό)
Κλάσμα  Y = 1,1100 0000 … …
```

Άσκηση (3)

```
Y = 0 0111 1111 1100 0000 … …
X = 0 1001 1011 0000 0100 … …
Z = 1 1001 1011 0000 0100 … …

Πρόσημο X = θετικός
Εκθέτης X = 155 (δεκαδικός)
Κλάσμα  X = 1,000001 … …
```

Άσκηση (4)

```
Y = 0 0111 1111 1100 0000 … …
X = 0 1001 1011 0000 0100 … …
Z = 1 1001 1011 0000 0100 … …

Πρόσημο Z = αρνητικός
Εκθέτης Z = 155 (δεκαδικός)
Κλάσμα  Z = 1,000001
```

Y + (X + Z)

$$X + Z = 2^{155} \times 1{,}000001 - 2^{155} \times 1{,}000001 = 0$$

$$Y + (X + Z) = Y$$

(Y + X) + Z

$$Y + X = 2^{127} \times 1{,}11 + 2^{155} \times 1{,}000001$$

Ολισθαίνουμε το Y κατά (155-127=) 28 θέσεις προς τα δεξιά:

$$Y = 2^{127} \times 1{,}11 = 2^{155} \times 0{,}00000$$

underflow

Οπότε:

$$Y + X = 2^{155} \times 0{,}000000 + 2^{155} \times 1{,}000001 = X$$

$$(Y + X) + Z = 2^{155} \times 1{,}000001 - 2^{155} \times 1{,}000001$$
$$= X + Z = 0$$

Άσκηση – Συμπεράσματα

```
X = 0 0111 1111 1100 0000 → μικρός (σχετικά)
Y = 0 1001 1011 0000 0100 → μεγαλύτερος
Z = 1 1001 1011 0000 0100 → μεγαλύτερος

Y + (X + Z) = Y
(Y + X) + Z = 0
```

- **Η προσεταιριστική ιδιότητα δεν ισχύει πάντα. Αυτό συμβαίνει όταν προσθέτουμε ένα μεγάλο αριθμό με ένα μικρό αριθμό αντίθετου προσήμου**

Προσεταιριστικότητα

- **Δεν ισχύει πάντα στις πράξεις κινητής υποδιαστολής**

- **Τα παράλληλα προγράμματα μπορεί να «πλέκουν» τις λειτουργίες με μη αναμενόμενη σειρά**
  - **υποθέσεις προσεταιριστικότητας μπορεί να αποτύχουν**
- **Πρέπει να επιβεβαιώνεται η λειτουργία των παράλληλων προγραμμάτων σε διαφορετικούς βαθμούς παραλληλίας**

Δεξιά ολίσθηση και διαίρεση

- Η αριστερή ολίσθηση κατά i θέσεις πολλαπλασιάζει έναν ακέραιο με 2^i
- Η δεξιά ολίσθηση διαιρεί με το 2^i ;
  - Μόνο σε απρόσημους ακεραίους
- Για προσημασμένους ακεραίους
  - Αριθμητική δεξιά ολίσθηση: επανάληψη του προσήμου
  - π.χ., –5 / 4
    - $11111011_2 >> 2 = 11111110_2 = -2$
    - Στρογγυλοποιεί προς το –∞
  - σύγκριση $11111011_2 >> 2 = 00111110_2 = +62$

Ποιος νοιάζεται για την ακρίβεια ΚΥ;

- Σημαντική για επιστημονικό κώδικα
  - Αλλά για καθημερινή χρήση;
    - "Το υπόλοιπό μου στη τράπεζα διαφέρει κατά 0.0002 σεντ!"
- Το σφάλμα της διαίρεσης ΚΥ του Intel Pentium (FDIV bug)
  - Η αγορά αναμένει ακρίβεια
  - Δείτε Colwell, The Pentium Chronicles

Συμπερασματικές παρατηρήσεις

- **Οι αρχιτεκτονικές συνόλου εντολών υποστηρίζουν αριθμητική**
  - **Προσημασμένων και απρόσημων ακεραίων**
  - **Προσεγγίσεων κινητής υποδιαστολής για τους πραγματικούς**

- **Πεπερασμένο εύρος και ακρίβεια**
  - **Οι λειτουργίες μπορεί να οδηγήσουν σε υπερχείλιση (overflow) και ανεπάρκεια (underflow)**