ΑΝΤΩΝΙΟΣ ΜΙΧ ΜΑΡΑΓΚΟΥΔΑΚΗΣ
ΠΡΟΣΗΜΑΣΜΕΝΟΙ ΓΕΝΙΚΑ – ΜΕΙΟΝΕΚΤΗΜΑΤΑ – 8 ΜΠΙΤ

Τι σημαίνει "προσημασμένος αριθμός";

Ένας **προσημασμένος** αριθμός μπορεί να είναι **θετικός ή αρνητικός**.
Για να το δείξουμε αυτό, χρησιμοποιούμε το **πρώτο bit (το πιο αριστερό)** ως **bit πρόσημου (sign bit)**:

Bit πρόσημου	**Ερμηνεία**
0	Θετικός αριθμός
1	Αρνητικός αριθμός

Αναπαράσταση σε 8 bit

Με 8 bits έχουμε **1 bit για το πρόσημο** και **7 bits για την τιμή (magnitude)**.
Συνήθως χρησιμοποιούμε το **συμπλήρωμα ως προς 2 (Two's Complement)** για τα αρνητικά.

➤ **Εύρος τιμών:**

Είδος	**Δυαδικό**	**Δεκαδικό**
Ελάχιστο (αρνητικό)	10000000	−128
Μέγιστο (θετικό)	01111111	+127

Άρα το εύρος των 8-bit προσημασμένων είναι:

−128 ως +127

Θετικοί αριθμοί

Απλώς γράφονται κανονικά σε δυαδική μορφή.

Δεκαδικό	**Δυαδικό (8-bit)**
+5	00000101
+23	00010111
+127	01111111

Αρνητικοί αριθμοί – Συμπλήρωμα ως προς 2

Για να βρεις το δυαδικό ενός **αρνητικού** αριθμού:

Βήματα:

1. Γράψε τον θετικό αριθμό σε 8 bit
2. Αντιστροφή όλων των bits (0→1, 1→0)
3. Πρόσθεσε 1

Παράδειγμα: −5

1. +5 = 00000101
2. Αντιστροφή → 11111010
3. +1 → 11111011

✅ Άρα:
−5 = 11111011

Δεκαδικό	**8-bit Δυαδικό**
+5	00000101
−5	11111011
+1	00000001
−1	11111111
+127	01111111
−128	10000000

Πίνακας προσημασμένων δυαδικών (8-bit)

Δεκαδικός	**8-bit Δυαδικό**	**Επεξήγηση**
−8	11111000	συμπλήρωμα ως προς 2 του 00001000
−7	11111001	συμπλήρωμα ως προς 2 του 00000111
−6	11111010	συμπλήρωμα ως προς 2 του 00000110
−5	11111011	συμπλήρωμα ως προς 2 του 00000101
−4	11111100	συμπλήρωμα ως προς 2 του 00000100
−3	11111101	συμπλήρωμα ως προς 2 του 00000011
−2	11111110	συμπλήρωμα ως προς 2 του 00000010
−1	11111111	συμπλήρωμα ως προς 2 του 00000001
0	00000000	0
+1	00000001	θετικός αριθμός
+2	00000010	θετικός αριθμός

Δεκαδικός	8-bit Δυαδικό	Επεξήγηση
+3	00000011	θετικός αριθμός
+4	00000100	θετικός αριθμός
+5	00000101	θετικός αριθμός
+6	00000110	θετικός αριθμός
+7	00000111	θετικός αριθμός
+8	00001000	θετικός αριθμός

Ολόκληρο εύρος 8-bit

Για αναφορά:

Ελάχιστο	Μέγιστο	Εύρος
−128	+127	256 διαφορετικές τιμές

Ας δούμε **τα μειονεκτήματα της αναπαράστασης "προσημασμένου μεγέθους"** (ή αλλιώς *sign–magnitude representation*), δηλαδή της μεθόδου όπου το **πρώτο bit δείχνει το πρόσημο** (0 = +, 1 = −) και τα υπόλοιπα bits δείχνουν το μέγεθος του αριθμού.

◆ **Τι είναι η αναπαράσταση προσημασμένου μεγέθους**

Σε αυτή τη μορφή, για 8-bit αριθμό:

Πρόσημο	Μέγεθος	Παράδειγμα
0xxxxxxx	Θετικός αριθμός	+5 → 00000101
1xxxxxxx	Αρνητικός αριθμός	−5 → 10000101

Μειονεκτήματα

1 Δύο διαφορετικές αναπαραστάσεις του μηδενός

Υπάρχει:

- **+0 → 00000000**
- **−0 → 10000000**

➞ Αυτό προκαλεί **ασάφεια** και **περιττό πλεονασμό**, γιατί έχουμε δύο διαφορετικούς δυαδικούς για την ίδια αριθμητική τιμή (μηδέν).

Δυσκολία στις πράξεις (πρόσθεση, αφαίρεση)

Για πρόσθεση ή αφαίρεση πρέπει να ελέγχεται το **bit πρόσημου** ξεχωριστά και να εφαρμόζονται διαφορετικοί κανόνες αν οι αριθμοί έχουν ίδιο ή διαφορετικό πρόσημο.

→ Οι κυκλικές υλοποιήσεις (όπως οι ALU στους επεξεργαστές) γίνονται **πιο περίπλοκες**.

Δυσκολία στην αναπαράσταση και σύγκριση

Για να συγκρίνεις δύο προσημασμένους αριθμούς, πρέπει να ελέγξεις πρώτα το πρόσημο και μετά το μέγεθος.

→ Δεν υπάρχει **γραμμική σχέση** στη δυαδική ακολουθία (π.χ. το "μεγαλύτερο" bit δεν σημαίνει απαραίτητα μεγαλύτερη τιμή).

Περιορισμένη συμβατότητα με αριθμητική επεξεργασία

Επειδή δεν μπορείς να χρησιμοποιήσεις **τους ίδιους αλγορίθμους** για θετικούς και αρνητικούς αριθμούς (π.χ. πρόσθεση bit προς bit), οι υπολογισμοί γίνονται **λιγότερο αποδοτικοί**.

Δεν αξιοποιεί πλήρως το εύρος τιμών

Για 8-bit αριθμό:

- Το εύρος είναι από **−127 έως +127**
  (όχι −128 έως +127, όπως στο συμπλήρωμα ως προς 2)

→ Δηλαδή "χάνουμε" μία τιμή εξαιτίας του διπλού μηδενός.

Μειονέκτημα	**Περιγραφή**
Διπλό μηδέν	+0 και −0 ξεχωριστά
Πολύπλοκες πράξεις	Πρέπει να ελέγχεται το πρόσημο
Περιορισμένο εύρος	Από $-(2^{n-1}-1)$ έως $+(2^{n-1}-1)$
Μη γραμμική σειρά τιμών	Δυσκολία σε συγκρίσεις
Όχι κατάλληλο για hardware	Οι ALU προτιμούν συμπλήρωμα ως προς 2

Ας δούμε **ένα συγκεκριμένο παράδειγμα** που δείχνει καθαρά **γιατί η πρόσθεση δεν λειτουργεί σωστά** με την αναπαράσταση **προσημασμένου μεγέθους (sign–magnitude)**.

◆ **Υπενθύμιση:**

Στην αναπαράσταση **προσημασμένου μεγέθους**:

- Το **αριστερό bit** είναι το **πρόσημο**
  → 0 = θετικός, 1 = αρνητικός
- Τα υπόλοιπα bits δείχνουν το **μέγεθος** (όπως αν ήταν απόλυτη τιμή)

Παράδειγμα

Θέλουμε να υπολογίσουμε:

$(+5)+(-3)$

➤ Σε 8-bit προσημασμένο μέγεθος:

Δεκαδικό	Δυαδικό (sign–magnitude)
+5	00000101
−3	10000011

Αν τα προσθέσουμε δυαδικά κανονικά:

```
 00000101
+ 10000011
----------
 10001000
```

Ερμηνεία αποτελέσματος:

10001000 →
bit πρόσημου = 1 (άρα αρνητικός)
μέγεθος = 0001000 = 8

✓ Άρα το αποτέλεσμα φαίνεται να είναι **−8**.

Όμως το σωστό μαθηματικά είναι:

$(+5)+(-3)=+2$

και όχι −8!

Τι πήγε στραβά:

Η δυαδική πρόσθεση **δεν λαμβάνει υπόψη το πρόσημο χωριστά**.
Στην αναπαράσταση προσημασμένου μεγέθους πρέπει να:

- Ελέγξεις πρώτα τα πρόσημα
- Αν είναι διαφορετικά, να κάνεις **αφαίρεση των μεγεθών**
- Και μετά να βάλεις το πρόσημο του μεγαλύτερου αριθμού

➞ Δηλαδή η **απλή δυαδική πρόσθεση bit–προς–bit δεν ισχύει** όπως στο "συμπλήρωμα ως προς 2".

Σύγκριση με συμπλήρωμα ως προς 2

Αν κάναμε την ίδια πράξη στο **two's complement**:

Δεκαδικό	**Two's complement (8-bit)**
+5	00000101
−3	11111101

00000101

+ 11111101

00000010 ← (απορρίπτεται η μεταφορά)

Αποτέλεσμα = 00000010 = **+2**, που είναι **σωστό**.

Συμπέρασμα

Η αναπαράσταση **προσημασμένου μεγέθους**:

- Δεν υποστηρίζει σωστά **πράξεις με διαφορετικά πρόσημα**
- Δεν επιτρέπει απευθείας χρήση της **δυαδικής πρόσθεσης**
- Γι' αυτό οι υπολογιστές **δεν τη χρησιμοποιούν**, αλλά προτιμούν το **συμπλήρωμα ως προς 2 (two's complement)**