# 4

# Πράξεις με δεδομένα

Εισαγωγή στην Επιστήμη των Υπολογιστών ©
Εκδόσεις Κλειδάριθμος



## Στόχοι

**Μετά την ολοκλήρωση αυτού του κεφαλαίου, ο σπουδαστής θα είναι σε θέση:**

- ❑ Να προσδιορίζει τις τρεις κατηγορίες πράξεων που εκτελούνται σε δεδομένα.
- ❑ Να εκτελεί μονομελείς και διμελείς λογικές πράξεις σε σχήματα bit.
- ❑ Να ξεχωρίζει τις λογικές πράξεις μετατόπισης από τις αριθμητικές πράξεις μετατόπισης.
- ❑ Να εκτελεί πράξεις πρόσθεσης και αφαίρεσης σε ακεραίους που είναι αποθηκευμένοι σε μορφή συμπληρώματος ως προς δύο.
- ❑ Να εκτελεί πράξεις πρόσθεσης και αφαίρεσης σε ακεραίους που είναι αποθηκευμένοι σε μορφή προσήμου και μεγέθους.
- ❑ Να εκτελεί πράξεις πρόσθεσης και αφαίρεσης σε πραγματικούς αριθμούς που είναι αποθηκευμένοι σε μορφή κινητής υποδιαστολής.



## 4-1 ΛΟΓΙΚΕΣ ΠΡΑΞΕΙΣ

Στο Κεφάλαιο 3 περιγράψαμε τον τρόπο αποθήκευσης των δεδομένων στο εσωτερικό του υπολογιστή ως σχημάτων bit. Οι **λογικές πράξεις** είναι αυτές κατά τις οποίες εφαρμόζεται η ίδια βασική πράξη σε μεμονωμένα bit ενός σχήματος ή σε δύο αντίστοιχα bit δύο σχημάτων. Αυτό σημαίνει ότι μπορούμε να ορίζουμε λογικές πράξεις σε **επίπεδο bit** και σε **επίπεδο σχήματος**. Μία λογική πράξη σε επίπεδο σχήματος ισούται με *n* λογικές πράξεις του ίδιου τύπου σε επίπεδο bit, όπου *n* είναι το πλήθος των bit του σχήματος.



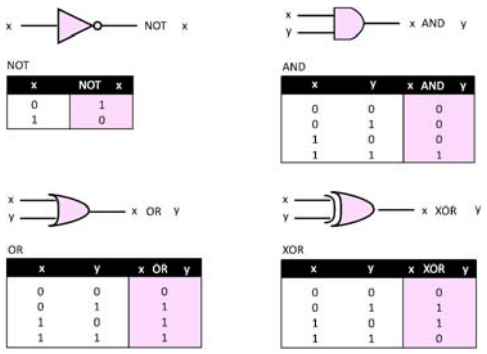
## Λογικές πράξεις σε επίπεδο bit

Ένα bit μπορεί να πάρει μία από δύο τιμές: 0 ή 1. Αν ερμηνεύσουμε το 0 ως την τιμή *ψευδής* (false) και το 1 ως την τιμή *αληθής* (true), τότε για τον χειρισμό των bit μπορούμε να εφαρμόζουμε τις πράξεις που ορίζονται στην άλγεβρα Μπουλ (Boolean algebra). Η **άλγεβρα Μπουλ**, που ονομάστηκε έτσι προς τιμή του George Boole, κατατάσσεται σε ένα ειδικό πεδίο των μαθηματικών που ονομάζεται *λογική*. Η άλγεβρα Μπουλ και ο τρόπος εφαρμογής της για τη δημιουργία λογικών κυκλωμάτων σε υπολογιστές περιγράφονται συνοπτικά στο Παράρτημα Ε. Σε αυτή την ενότητα παρουσιάζουμε συνοπτικά τέσσερις πράξεις σε επίπεδο bit οι οποίες χρησιμοποιούνται για τον χειρισμό bit: **NOT**, **AND**, **OR**, και **XOR**.

i

**Η άλγεβρα Μπουλ και τα λογικά κυκλώματα περιγράφονται στο Παράρτημα Ε.**






**Εικόνα 4.1** Λογικές πράξεις σε επίπεδο bit



### NOT

Ο τελεστής NOT είναι μονομελής: δηλαδή, δέχεται μόνο μία είσοδο. Το bit εξόδου είναι το συμπλήρωμα της εισόδου.

### AND

Ο τελεστής AND είναι διμελής, δηλαδή δέχεται δύο εισόδους. Το bit εξόδου είναι 1 όταν και οι δύο είσοδοι είναι 1, ενώ είναι 0 σε οποιαδήποτε από τις άλλες τρεις περιπτώσεις.

i

**Για $x = 0$ ή 1 $x$ AND 0 → 0 0 AND $x$ → 0**

## OR

Ο τελεστής OR είναι και αυτός διμελής, δηλαδή δέχεται δύο εισόδους. Το bit εξόδου είναι 0 όταν και οι δύο είσοδοι είναι 0, ενώ είναι 1 σε οποιαδήποτε από τις άλλες τρεις περιπτώσεις.

**Για $x = 0$ ή 1 $\quad x$ OR 1 → 1 $\quad$ 1 OR $x$ → 1**

## XOR

Ο τελεστής XOR (προφέρεται exclusive OR, αποκλειστικό OR) είναι επίσης διμελής όπως ο τελεστής OR, με μία διαφορά: η έξοδος είναι 0 όταν και οι δύο είσοδοι είναι 1.

**Για $x = 0$ ή 1**
**1 XOR $x$ → NOT $x$ $\quad$ $x$ XOR 1 → NOT $x$**

4.7

## Παράδειγμα 4.1

Στα Ελληνικά, ο σύνδεσμος "ή" (or) χρησιμοποιείται ορισμένες φορές ως συμπεριληπτικό OR, και άλλες φορές ως αποκλειστικό OR.

α. Για παράδειγμα, στην πρόταση "Θα ήθελα να έχω ένα αυτοκίνητο ή ένα σπίτι", το "ή" χρησιμοποιείται με τη συμπεριληπτική έννοια — θα ήθελα να έχω ένα αυτοκίνητο, ένα σπίτι, ή και τα δύο.

β. Στην πρόταση "Σήμερα είναι Δευτέρα ή Τρίτη", το "ή" χρησιμοποιείται με την αποκλειστική έννοια — δηλαδή, σήμερα είναι είτε Δευτέρα είτε Τρίτη, όχι όμως και τα δύο.

4.8

## Παράδειγμα 4.2

Ο τελεστής XOR στην πραγματικότητα δεν είναι καινούργιος. Μπορούμε σε κάθε περίπτωση να τον προσομοιώνουμε χρησιμοποιώντας τους άλλους τρεις τελεστές. Για παράδειγμα, οι ακόλουθες δύο παραστάσεις είναι ισοδύναμες

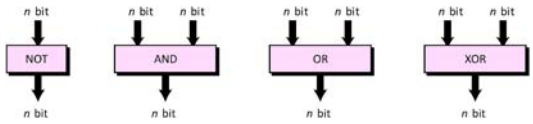
$$x \text{ XOR } y \leftrightarrow [x \text{ AND } (\text{NOT } y)] \text{ OR } [(\text{NOT } x) \text{ AND } y]$$

Η ισοδυναμία αποδεικνύεται αν δημιουργήσουμε τους πίνακες αληθείας και για τις δύο.

4.9

## Λογικές πράξεις σε επίπεδο σχήματος

Και οι τέσσερις τελεστές (NOT, AND, OR, και XOR) μπορούν να εφαρμοστούν σε ένα σχήμα με $n$ bit. Το αποτέλεσμα είναι το ίδιο όπως αν εφαρμόζαμε κάθε τελεστή σε κάθε μεμονωμένο bit για τον τελεστή NOT, και σε κάθε αντίστοιχο ζεύγος bit για τους άλλους τρεις τελεστές. Στην Εικόνα 4.2 παρουσιάζονται αυτοί οι τέσσερις τελεστές με σχήματα εισόδου και εξόδου.




**Εικόνα 4.2** Εφαρμογή λογικών τελεστών σε σχήματα bit

4.10

## Παράδειγμα 4.3

Χρησιμοποιήστε τον τελεστή NOT στο σχήμα bit 10011000.

**Λύση**

Η λύση φαίνεται παρακάτω. Παρατηρήστε ότι ο τελεστής NOT αλλάζει κάθε 0 σε 1 και κάθε 1 σε 0.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| NOT | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | Είσοδος |
| | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | Έξοδος |

4.11

## Παράδειγμα 4.4

Χρησιμοποιήστε τον τελεστή AND στα σχήματα bit 10011000 και 00101010.

**Λύση**

Η λύση φαίνεται παρακάτω. Παρατηρήστε ότι μόνο ένα bit της εξόδου είναι 1, ενώ και οι δύο αντίστοιχες είσοδοι είναι 1.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | Είσοδος 1 |
| AND | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | Είσοδος 2 |
| | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | Έξοδος |

4.12

**Παράδειγμα 4.5**

Χρησιμοποιήστε τον τελεστή OR στα σχήματα bit 10011001 και 00101110.

**Λύση**

Η λύση φαίνεται παρακάτω. Παρατηρήστε ότι μόνο ένα bit της εξόδου είναι 0, ενώ και οι δύο αντίστοιχες είσοδοι είναι 0.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| OR | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** |

Είσοδος 1
Είσοδος 2
Έξοδος



**Παράδειγμα 4.6**

Χρησιμοποιήστε τον τελεστή XOR στα σχήματα bit 10011001 και 00101110.

**Λύση**

Η λύση φαίνεται στη συνέχεια. Συγκρίνετε την έξοδο σε αυτό το παράδειγμα με αυτή στο Παράδειγμα 4.5. Η μοναδική διαφορά είναι ότι, όταν είναι και οι δύο είσοδοι 1, το αποτέλεσμα είναι 0 (λόγω του αποκλεισμού).

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| XOR | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| | **1** | **0** | **1** | **1** | **0** | **1** | **1** | **1** |

Είσοδος 1
Είσοδος 2
Έξοδος



## Εφαρμογές

Για την τροποποίηση ενός σχήματος bit μπορούν να εφαρμοστούν οι τέσσερις λογικές πράξεις.

- **Συμπλήρωμα (NOT)**
- **Απενεργοποίηση (AND)**
- **Ενεργοποίηση (OR)**
- **Αντιστροφή (XOR)**



**Παράδειγμα 4.7**

Χρησιμοποιήστε μια μάσκα για να απενεργοποιήσετε τα πέντε αριστερότερα bit ενός σχήματος. Ελέγξτε τη μάσκα με το σχήμα 10100110.

**Λύση**

Η μάσκα είναι 00000111. Το αποτέλεσμα της εφαρμογής της μάσκας είναι το εξής:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| AND | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| | **0** | **0** | **0** | **0** | **0** | **1** | **1** | **0** |

Είσοδος
Μάσκα
Έξοδος



**Παράδειγμα 4.8**

Χρησιμοποιήστε μια μάσκα για να ενεργοποιήσετε τα πέντε αριστερότερα bit ενός σχήματος. Ελέγξτε τη μάσκα με το σχήμα 10100110.

**Λύση**

Η μάσκα είναι 11111000. Το αποτέλεσμα της εφαρμογής της μάσκας είναι το εξής:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| OR | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** |

Είσοδος
Μάσκα
Έξοδος



**Παράδειγμα 4.9**

Χρησιμοποιήστε μια μάσκα για να αντιστρέψετε τα πέντε αριστερότερα bit ενός σχήματος. Ελέγξτε τη μάσκα με το σχήμα 10100110.

**Λύση**

Η μάσκα είναι 11111000. Το αποτέλεσμα της εφαρμογής της μάσκας είναι το εξής:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| XOR | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| | **0** | **1** | **0** | **1** | **1** | **1** | **1** | **0** |

Είσοδος 1
Μάσκα
Έξοδος

## 4-2 ΠΡΑΞΕΙΣ ΜΕΤΑΤΟΠΙΣΗΣ

Οι **πράξεις μετατόπισης** (ή ολίσθησης) μετακινούν τα bit σε ένα σχήμα, αλλάζοντας τις θέσεις τους. Τα bit μπορούν να μετακινηθούν προς τα αριστερά ή προς τα δεξιά. Οι πράξεις μετατόπισης χωρίζονται σε δύο κατηγορίες: στις **λογικές πράξεις μετατόπισης** και στις **αριθμητικές πράξεις μετατόπισης**.



## Λογικές πράξεις ολίσθησης

Λογικές πράξεις μετατόπισης εφαρμόζονται σε σχήματα τα οποία δεν αναπαριστούν προσημασμένους αριθμούς. Αυτό συμβαίνει επειδή αυτές οι πράξεις μετατόπισης ενδέχεται να αλλάξουν το πρόσημο ενός αριθμού το οποίο καθορίζεται από το αριστερότερο bit του σχήματος. Υπάρχουν δύο τύποι λογικών πράξεων μετατόπισης, οι οποίοι αναφέρονται στη συνέχεια:

- **Λογική μετατόπιση**
- **Κυκλική μετατόπιση**




**Εικόνα 4.3** Λογικές πράξεις μετατόπισης



**Παράδειγμα 4.10**

Χρησιμοποιήστε μια λογική πράξη αριστερής μετατόπισης στο σχήμα bit 10011000.

**Λύση**

Η λύση φαίνεται παρακάτω. Το αριστερότερο bit χάνεται και προστίθεται ένα 0 ως το δεξιότερο bit.





**Εικόνα 4.4** Πράξεις κυκλικής μετατόπισης

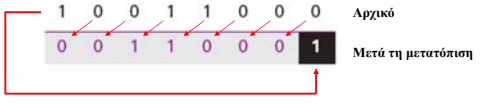


**Παράδειγμα 4.11**

Χρησιμοποιήστε μια πράξη αριστερής κυκλικής μετατόπισης στο σχήμα bit 10011000.

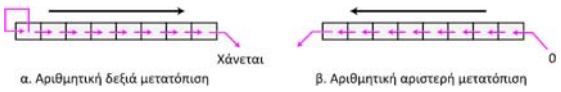
**Λύση**

Η λύση φαίνεται παρακάτω. Το αριστερότερο bit μετατοπίζεται κυκλικά και γίνεται το δεξιότερο bit.



## Αριθμητικές πράξεις μετατόπισης

Οι αριθμητικές πράξεις μετατόπισης προϋποθέτουν ότι το σχήμα bit είναι ένας προσημασμένος ακέραιος σε μορφή συμπληρώματος ως προς δύο. Η αριθμητική πράξη δεξιάς μετατόπισης χρησιμοποιείται για τη διαίρεση ενός ακεραίου με το δύο, ενώ η αριθμητική πράξη αριστερής μετατόπισης χρησιμοποιείται για τον πολλαπλασιαμό ενός ακεραίου με το δύο.




**Εικόνα 4.5** Αριθμητικές πράξεις μετατόπισης

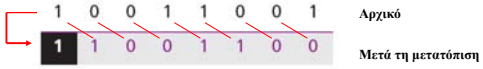


**Παράδειγμα 4.12**

Χρησιμοποιήστε μια αριθμητική πράξη δεξιάς μετατόπισης στο σχήμα bit 10011001. Το σχήμα είναι ένας ακέραιος σε μορφή συμπληρώματος ως προς δύο.

**Λύση**

Η λύση φαίνεται παρακάτω. Το αριστερότερο bit διατηρείται και αντιγράφεται επίσης στο δεξιό παρακείμενο bit.




Ο αρχικός αριθμός ήταν το −103 και ο νέος αριθμός είναι το −52, το οποίο είναι το αποτέλεσμα της διαίρεσης του −103 με το 2 με περικοπή στον μικρότερο ακέραιο.



**Παράδειγμα 4.13**

Χρησιμοποιήστε μια αριθμητική πράξη αριστερής μετατόπισης στο σχήμα bit 11011001. Το σχήμα είναι ένας ακέραιος σε μορφή συμπληρώματος ως προς δύο.

**Λύση**

Η λύση φαίνεται στη συνέχεια. Το αριστερότερο bit χάνεται και προστίθεται ένα 0 ως το δεξιότερο bit.


Ο αρχικός αριθμός ήταν το −39 και ο νέος αριθμός είναι το −78. Δηλαδή, ο αρχικός αριθμός πολλαπλασιάζεται με το δύο. Η πράξη είναι έγκυρη αφού δεν προέκυψε υποχείλιση.



**Παράδειγμα 4.14**

Χρησιμοποιήστε μια αριθμητική πράξη αριστερής μετατόπισης στο σχήμα bit 01111111. Το σχήμα είναι ένας ακέραιος σε μορφή συμπληρώματος ως προς δύο.

**Λύση**

Η λύση φαίνεται παρακάτω. Το αριστερότερο bit χάνεται και προστίθεται ένα 0 ως το δεξιότερο bit.


Ο αρχικός αριθμός ήταν το 127 και ο νέος αριθμός είναι το −2. Εδώ το αποτέλεσμα δεν είναι έγκυρο επειδή προέκυψε υπερχείλιση. Αυτό συνέβη επειδή το αναμενόμενο αποτέλεσμα 127 × 2 = 254 δεν μπορεί να αναπαρασταθεί από ένα σχήμα μήκους 8 bit.



**Παράδειγμα 4.15**

Ο συνδυασμός λογικών πράξεων και λογικών πράξεων μετατόπισης μας παρέχει ορισμένα εργαλεία για τον χειρισμό σχημάτων bit. Ας υποθέσουμε ότι έχουμε ένα σχήμα και πρέπει να χρησιμοποιήσουμε το τρίτο bit (από τα δεξιά) του σχήματος σε μια διαδικασία λήψης αποφάσεων. Επομένως, πρέπει να γνωρίζουμε αν το συγκεκριμένο bit είναι 0 ή 1. Εδώ βλέπετε πώς μπορούμε να το ελέγξουμε αυτό.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | h | g | f | e | d | **c** | b | a | Αρχικό |
| | 0 | h | g | f | e | d | **c** | b | Μία δεξιά μετατόπιση |
| | 0 | 0 | h | g | f | e | d | **c** | Δύο δεξιές μετατοπίσεις |
| AND | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Μάσκα |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **c** | Αποτέλεσμα |

Τώρα μπορούμε να ελέγξουμε το αποτέλεσμα: αν είναι ένας μη προσημασμένος ακέραιος 1, το bit που μας ενδιαφέρει ήταν 1, ενώ αν το αποτέλεσμα είναι ένας μη προσημασμένος ακέραιος 0, τότε το bit ήταν 0.



## 4-3 ΑΡΙΘΜΗΤΙΚΕΣ ΠΡΑΞΕΙΣ

Στις αριθμητικές πράξεις περιλαμβάνονται η πρόσθεση, η αφαίρεση, ο πολλαπλασιασμός, και η διαίρεση. Αυτές οι πράξεις μπορούν να εφαρμοστούν σε ακεραίους και σε αριθμούς κινητής υποδιαστολής.

## Αριθμητικές πράξεις σε ακεραίους

Στους ακέραιους αριθμούς μπορούν να εφαρμοστούν όλες οι αριθμητικές πράξεις, όπως η πρόσθεση, η αφαίρεση, ο πολλαπλασιασμός, και η διαίρεση. Παρόλο που ο πολλαπλασιασμός ή η διαίρεση ακεραίων μπορεί να υλοποιηθεί με επαναλαμβανόμενες προσθέσεις ή αφαιρέσεις, αντίστοιχα, η διαδικασία δεν είναι αποδοτική. Υπάρχουν πιο αποδοτικές διαδικασίες για τον πολλαπλασιασμό και τη διαίρεση, όπως ο **αλγόριθμος Booth**, όμως αυτές βρίσκονται πέρα από την εμβέλεια του συγκεκριμένου βιβλίου. Για τον λόγο αυτό, στην ενότητα αυτή θα περιγράψουμε μόνο την πρόσθεση και την αφαίρεση ακεραίων.



## Ακέραιοι συμπληρώματος ως προς δύο

Για να εκτελέσει την πράξη της αφαίρεσης, ο υπολογιστής απλώς τη μετατρέπει στην πράξη της πρόσθεσης, χρησιμοποιώντας όμως το συμπλήρωμα ως προς δύο του δεύτερου αριθμού. Με άλλα λόγια:

$$A - B \leftrightarrow A + (\overline{B} + 1)$$

όπου B είναι το συμπλήρωμα ως προς ένα του B και $(\overline{B} + 1)$ είναι το συμπλήρωμα ως προς δύο του B

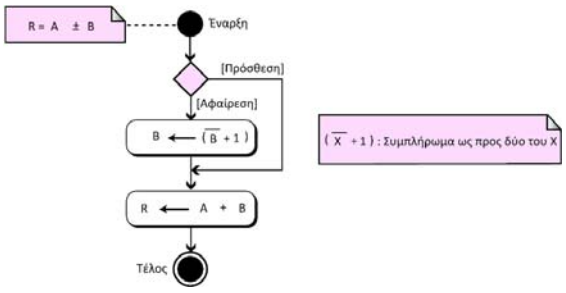


Θα πρέπει να θυμάστε ότι προσθέτουμε ακεραίους στήλη προς στήλη. Στον επόμενο πίνακα παρουσιάζονται τα αθροίσματα και τα κρατούμενα.

**Πίνακας 4.1** Το κρατούμενο και το άθροισμα που προκύπτουν από την πρόσθεση δύο bit

| *Στήλη* | *Κρατούμενο* | *Άθροισμα* | *Στήλη* | *Κρατούμενο* | *Άθροισμα* |
|---|---|---|---|---|---|
| Κανένα 1 | 0 | 0 | Δύο 1 | 1 | 0 |
| Ένα 1 | 0 | 1 | Τρία 1 | 1 | 1 |






**Εικόνα 4.6** Πρόσθεση και αφαίρεση ακεραίων σε μορφή συμπληρώματος ως προς δύο



### Παράδειγμα 4.16

Δύο ακέραιοι A και B είναι αποθηκευμένοι σε μορφή συμπληρώματος ως προς δύο. Δείξτε τον τρόπο πρόσθεσης του B στον A.

$$A = (00010001)_2 \quad B = (00010110)_2$$

**Λύση**

Η πράξη που πρέπει να εκτελεστεί είναι πρόσθεση. Ο A προστίθεται στον B και το αποτέλεσμα αποθηκεύεται στο R. (+17) + (+22) = (+39).

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 1 | | | | | | Κρατούμενο |
| | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | A |
| + | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | B |
| | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | R |



### Παράδειγμα 4.17

Δύο ακέραιοι A και B είναι αποθηκευμένοι σε μορφή συμπληρώματος ως προς δύο. Δείξτε τον τρόπο πρόσθεσης του B στον A.

$$A = (00011000)_2 \quad B = (11101111)_2$$

**Λύση**

Η πράξη που πρέπει να εκτελεστεί είναι πρόσθεση. Ο A προστίθεται στον B και το αποτέλεσμα αποθηκεύεται στο R. (+24) + (–17) = (+7).

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | | | | Κρατούμενο |
| | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | A |
| + | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | B |
| | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | R |

**Παράδειγμα 4.18**

Δύο ακέραιοι Α και Β είναι αποθηκευμένοι σε μορφή συμπληρώματος ως προς δύο. Δείξτε τον τρόπο αφαίρεσης του Β από τον Α.

$$\mathbf{A = (00011000)_2} \qquad \mathbf{B = (11101111)_2}$$

**Λύση**

Η πράξη που πρέπει να εκτελεστεί είναι αφαίρεση. Ο Α προστίθεται στο $(\overline{B} + 1)$ και το αποτέλεσμα αποθηκεύεται στο R. (+24) − (−17) = (+41).

| | | | | | | | | | Κρατούμενο |
|---|---|---|---|---|---|---|---|---|---|
| | | | | 1 | | | | | |
| | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | A |
| + | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | $(\overline{B}+1)$ |
| | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | R |

4.37

**Παράδειγμα 4.19**

Δύο ακέραιοι Α και Β είναι αποθηκευμένοι σε μορφή συμπληρώματος ως προς δύο. Δείξτε τον τρόπο αφαίρεσης του Β από τον Α.

$$\mathbf{A = (11011101)_2} \qquad \mathbf{B = (00010100)_2}$$

**Λύση**

Η πράξη που πρέπει να εκτελεστεί είναι αφαίρεση. Ο Α προστίθεται στο $(\overline{B} + 1)$ και το αποτέλεσμα αποθηκεύεται στο R. (−35) − (+20) = (−55).

| | | | | | | | | | Κρατούμενο |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 1 | 1 | 1 | 1 | | | |
| | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | A |
| + | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | $(\overline{B}+1)$ |
| | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | R |

4.38

**Παράδειγμα 4.20**

Δύο ακέραιοι Α και Β είναι αποθηκευμένοι σε μορφή συμπληρώματος ως προς δύο. Δείξτε τον τρόπο πρόσθεσης του Β στον Α.

$$\mathbf{A = (01111111)_2} \qquad \mathbf{B = (00000011)_2}$$

**Λύση**

Η πράξη που πρέπει να εκτελεστεί είναι πρόσθεση. Ο Α προστίθεται στον Β και το αποτέλεσμα αποθηκεύεται στο R.

| | | | | | | | | | Κρατούμενο |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | |
| | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | A |
| + | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | B |
| | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | R |

Παρόλο που θα περιμέναμε το αποτέλεσμα να είναι 127 + 3 = 130, η απάντηση είναι −126. Το σφάλμα αυτό οφείλεται σε υπερχείλιση, επειδή η αναμενόμενη απάντηση (+130) δεν βρίσκεται στο διάστημα τιμών −128 έως +127.

4.39

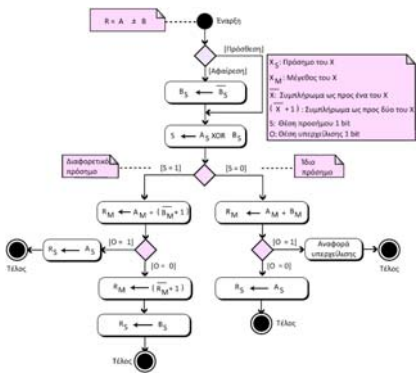
**Όταν εκτελούμε αριθμητικές πράξεις με αριθμούς σε έναν υπολογιστή, θα πρέπει να θυμόμαστε ότι κάθε αριθμός, αλλά και το αποτέλεσμα, πρέπει να ανήκουν στο διάστημα τιμών που ορίζονται από τη δέσμευση bit.**

4.40

## Ακέραιοι προσήμου και μεγέθους

Η πρόσθεση και η αφαίρεση ακεραίων στην αναπαράσταση προσήμου και μεγέθους φαίνονται ιδιαίτερα περίπλοκες. Υπάρχουν τέσσερις διαφορετικοί συνδυασμοί προσήμων (δύο πιθανά πρόσημα για κάθε μία από τις δύο τιμές) για την πρόσθεση, και τέσσερις διαφορετικοί συνδυασμοί για την αφαίρεση. Αυτό σημαίνει ότι πρέπει να λαμβάνουμε υπόψη μας οκτώ διαφορετικές καταστάσεις. Ωστόσο, αν πρώτα ελέγξουμε τα πρόσημα, τότε μπορούμε να μειώσουμε το πλήθος των περιπτώσεων, όπως βλέπετε στην Εικόνα 4.7.

4.41



**Εικόνα 4.7** Πρόσθεση και αφαίρεση ακεραίων σε μορφή προσήμου και μεγέθους

4.42

**Παράδειγμα 4.22**

Δύο ακέραιοι Α και Β είναι αποθηκευμένοι σε μορφή προσήμου και μεγέθους. Δείξτε τον τρόπο πρόσθεσης του Β στον Α.

**Λύση** $\mathbf{A = (0\ 0010001)_2}$ $\mathbf{B = (1\ 0010110)_2}$

Η πράξη που πρέπει να εκτελεστεί είναι πρόσθεση: το πρόσημο του Β δεν αλλάζει. Επειδή $S = A_S$ XOR $B_S = 1$, $R_M = A_M + (\overline{B}_M + 1)$. Επειδή δεν υπάρχει υπερχείλιση, πρέπει να πάρουμε το συμπλήρωμα ως προς δύο του RM. Το πρόσημο του R είναι το πρόσημο του Β. (+17) + ( −22) = (−5).

| | Χωρίς υπερχείλιση | | | | | | | | Κρατούμενο |
|---|---|---|---|---|---|---|---|---|---|
| $A_S$ | 0 | | 0 | 0 | 1 | 0 | 0 | 1 | $A_M$ |
| $B_S$ | 1 | + | 1 | 1 | 0 | 1 | 0 | 1 | 0 $(\overline{B}_M+1)$ |
| | | | 1 | 1 | 1 | 1 | 0 | 1 | 1 $R_M$ |
| $R_S$ | 1 | | 0 | 0 | 0 | 0 | 1 | 0 | 1 $R_M = (\overline{R}_M+1)$ |

4.43

**Παράδειγμα 4.23**

Δύο ακέραιοι Α και Β είναι αποθηκευμένοι σε μορφή προσήμου και μεγέθους. Δείξτε τον τρόπο αφαίρεσης του Β από τον Α.

**Λύση** $\mathbf{A = (1\ 1010001)_2}$ $\mathbf{B = (1\ 0010110)_2}$

Η πράξη που πρέπει να εκτελεστεί είναι αφαίρεση: $B_S = \overline{B}_S$. $S = A_S$ XOR $B_S = 1$, $R_M = A_M + (\overline{B}_M + 1)$. Επειδή υπάρχει υπερχείλιση, η τιμή του $R_M$ είναι η τελική. Το πρόσημο του R είναι το πρόσημο του Α. (−81) − (−22) = (−59).

| | Υπερχείλιση→ 1 | | | | | | | | Κρατούμενο |
|---|---|---|---|---|---|---|---|---|---|
| $A_S$ | 1 | | 1 | 0 | 1 | 0 | 0 | 1 | $A_M$ |
| $B_S$ | 1 | + | 1 | 1 | 0 | 1 | 0 | 1 | 0 $(\overline{B}_M+1)$ |
| $R_S$ | 1 | | 0 | 1 | 1 | 1 | 0 | 1 | 1 $R_M$ |

4.44

## Αριθμητικές πράξεις σε πραγματικούς αριθμούς

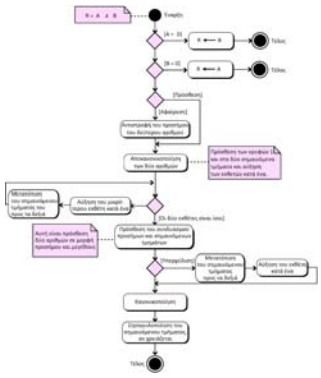
Στους πραγματικούς αριθμούς που είναι αποθηκευμένοι στη μορφή κινητής υποδιαστολής μπορούν να εφαρμοστούν όλες οι αριθμητικές πράξεις, όπως η πρόσθεση, η αφαίρεση, ο πολλαπλασιασμός, και η διαίρεση. Ο πολλαπλασιασμός δύο πραγματικών αριθμών απαιτεί τον πολλαπλασιασμό δύο ακεραίων στην αναπαράσταση προσήμου και μεγέθους. Παρόμοια, η διαίρεση δύο πραγματικών αριθμών απαιτεί τη διαίρεση δύο ακεραίων στην αναπαράσταση προσήμου και μεγέθους. Επειδή δεν περιγράψαμε τον πολλαπλασιασμό και τη διαίρεση ακεραίων στην αναπαράσταση προσήμου και μεγέθους, δεν θα περιγράψουμε ούτε τον πολλαπλασιασμό και τη διαίρεση πραγματικών αριθμών· θα δείξουμε μόνο τις πράξεις της πρόσθεσης και της αφαίρεσης για πραγματικούς αριθμούς.

4.45

## Πρόσθεση και αφαίρεση πραγματικών αριθμών

Η πρόσθεση και η αφαίρεση πραγματικών αριθμών αποθηκευμένων στη μορφή κινητής υποδιαστολής περιορίζεται στην πρόσθεση και την αφαίρεση δύο ακεραίων αποθηκευμένων στη μορφή προσήμου και μεγέθους (συνδυασμός προσήμου και σημαινόμενου τμήματος) μετά τη στοίχιση των υποδιαστολών. Στην Εικόνα 4.8 παρουσιάζεται με απλουστευμένο τρόπο η διαδικασία (υπάρχουν ορισμένες ειδικές περιπτώσεις που έχουμε παραλείψει).

4.46



**Εικόνα 4.8** **Πρόσθεση και αφαίρεση πραγματικών αριθμών σε μορφή κινητής υποδιαστολής**

4.47

**Παράδειγμα 4.24**

Δείξτε πώς ο υπολογιστής παράγει το αποτέλεσμα (5,75) + (+161,875) = (+167,625).

**Λύση**

Όπως είδαμε στο Κεφάλαιο 3, αυτοί οι δύο αριθμοί αποθηκεύονται σε μορφή κινητής υποδιαστολής, όπως βλέπετε στη συνέχεια, όμως πρέπει να θυμόμαστε ότι κάθε ένας από αυτούς έχει ένα κρυφό 1 (το οποίο δεν αποθηκεύεται αλλά εννοείται).

| | S | E | M |
|---|---|---|---|
| A | 0 | 10000001 | 01110000000000000000000 |
| B | 0 | 10000110 | 01000011110000000000000 |

4.48

**Παράδειγμα 4.24** (Συνέχεια)

Τα πρώτα βήματα του διαγράμματος UML (Εικόνα 4.8) δεν χρειάζονται. Έτσι προχωρούμε στην αποκανονικοποίηση των αριθμών προσθέτοντας το κρυφό 1 στο σημαινόμενο τμήμα και αυξάνοντας τον εκθέτη. Τώρα και τα δύο αποκανονικοποιημένα σημαινόμενα τμήματα είναι 24 bit και περιλαμβάνουν το κρυφό 1. Αυτό σημαίνει ότι θα πρέπει να αποθηκευτούν σε μια θέση που μπορεί να φιλοξενήσει και τα 24 bit. Κάθε εκθέτης αυξάνεται.

| | S | E | Αποκανονικοποιημένο M |
|---|---|---|---|
| A | 0 | 10000010 | 101110000000000000000000 |
| B | 0 | 10000111 | 101000011110000000000000 |



**Παράδειγμα 4.24** (Συνέχεια)

Τώρα εκτελούμε πρόσθεση στην αναπαράσταση προσήμου και μεγέθους, αφού μπορούμε να χειριστούμε το πρόσημο και το σημαινόμενο τμήμα κάθε αριθμού ως ακέραιο αποθηκευμένο στην αναπαράσταση προσήμου και μεγέθους.

| | S | E | Αποκανονικοποιημένο M |
|---|---|---|---|
| R | 0 | 10000111 | 10100111101000000000000 |

Επειδή δεν υπάρχει υπερχείλιση, προχωρούμε στην κανονικοποίηση του σημαινόμενου τμήματος.

| | S | E | M |
|---|---|---|---|
| R | 0 | 10000110 | 01001111010000000000000 |

Το σημαινόμενο τμήμα είναι μόνο 23 bit, και έτσι δεν χρειάζεται να στρογγυλοποιήσουμε. $E = (10000110)_2 = 134$ $M = 0100111101$. Με άλλα λόγια, το αποτέλεσμα είναι $(1{,}0100111101)_2 \times 2^{134-127} = (10100111{,}101)_2 = \mathbf{167{,}625}$.



**Παράδειγμα 4.25**

Δείξτε πώς ο υπολογιστής παράγει το αποτέλεσμα (+5,75) + (−7,0234375) = −1,2734375.

**Λύση**

Αυτοί οι δύο αριθμοί μπορούν να αποθηκευτούν σε μορφή κινητής υποδιαστολής, με τον τρόπο που παρουσιάζεται παρακάτω:

| | S | E | M |
|---|---|---|---|
| A | 0 | 10000001 | 01110000000000000000000 |
| B | 1 | 10000001 | 11000001100000000000000 |

Με την αποκανονικοποίηση παίρνουμε το εξής αποτέλεσμα:

| | S | E | Αποκανονικοποιημένο M |
|---|---|---|---|
| A | 0 | 10000010 | 101110000000000000000000 |
| B | 1 | 10000010 | 111000001100000000000000 |



**Παράδειγμα 4.25** (Συνέχεια)

Επειδή δεν απαιτείται ευθυγράμμιση (και οι δύο εκθέτες έχουν ίδια τιμή), εφαρμόζουμε την πράξη της πρόσθεσης στους συνδυασμούς προσήμου και σημαινόμενου τμήματος. Το πρόσημο του αποτελέσματος, που παρουσιάζεται παρακάτω, είναι αρνητικό:

| | S | E | Αποκανονικοποιημένο M |
|---|---|---|---|
| R | 1 | 10000010 | 00101000110000000000000 |

Τώρα πρέπει να εφαρμόσουμε κανονικοποίηση. Μειώνουμε τον εκθέτη τρεις φορές και μετατοπίζουμε το αποκανονικοποιημένο σημαινόμενο τμήμα τρεις θέσεις προς τα αριστερά:

| | S | E | M |
|---|---|---|---|
| R | 1 | 01111111 | 01000110000000000000000 |



**Παράδειγμα 4.25** (Συνέχεια)

Το σημαινόμενο τμήμα τώρα είναι 24 bit, γι' αυτό το στρογγυλοποιούμε στα 23 bit.

| | S | E | M |
|---|---|---|---|
| R | 1 | 01111111 | 01000110000000000000000 |

Το αποτέλεσμα είναι $R = -2^{127-127} \times 1{,}0100011 = \mathbf{-1{,}2734375}$, που είναι και το αναμενόμενο.