ΗΥ 134
Εισαγωγή στην Οργάνωση και στον Σχεδιασμό Υπολογιστών Ι

Διάλεξη 9

Αριθμητική Υπολογιστών
(Κεφάλαιο 3)

Νίκος Μπέλλας
Τμήμα Μηχανικών Η/Υ, Τηλεπικοινωνιών και Δικτύων

Αριθμητική για υπολογιστές

- Αναπαράσταση ακεραίων αριθμών
  - Απρόσημοι αριθμοί
  - Προσημασμένοι αριθμοί και μορφή συμπλήρωμα του 2 (2's complement)
- Πράξεις σε ακεραίους
  - Πρόσθεση και αφαίρεση
  - Πολλαπλασιασμός και διαίρεση
  - Ζήτημα υπερχείλισης
- Πραγματικοί αριθμοί κινητής υποδιαστολής
  - Αναπαράσταση και πράξεις

Απρόσημοι δυαδικοί ακέραιοι *(unsigned integers)*

- Ο απρόσημοςδυαδικός αριθμός:

$$b = b_{n-1}b_{n-2}\ldots b_1 b_0, b_i \in \{0,1\}$$

έχει τιμή: $b = b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \cdots + b_1 2^1 + b_0 2^0$

- Εύρος τιμών: 0 έως $+2^n - 1$
- Παράδειγμα
  - $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1011_2$
    $= 0 + \ldots + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
    $= 0 + \ldots + 8 + 0 + 2 + 1 = 11_{10}$
- Χρησιμοποιώντας 32 bits
  - 0 έως +4,294,967,295

Προσημασμένοι ακέραιοι στη μορφή Συμπλήρωμα του 2 (2s-Complement)

- Ένας προσημασμένος δυαδικός αριθμός:

$$b = b_{n-1}b_{n-2}\ldots b_1b_0, b_i \in \{0,1\}$$

έχει τιμή: $b = -b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \cdots + b_1 2^1 + b_0 2^0$

- Εύρος: -2^{n-1} έως $+2^{n-1} - 1$
- Παράδειγμα
  - $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1100_2$
    $= -1 \times 2^{31} + 1 \times 2^{30} + \ldots + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$
    $= -2{,}147{,}483{,}648 + 2{,}147{,}483{,}644 = -4_{10}$
- Χρησιμοποιώντας 32 bits
  - –2,147,483,648 to +2,147,483,647

Προσημασμένοι ακέραιοι στη μορφή Συμπλήρωμα του 2 (2s-Complement)

- Το πιό σημαντικό bit (most significant bit) είναι το πρόσημο ενός αριθμού:
  - 1 για αρνητικούς αριθμούς
  - 0 για μη αρνητικούς αριθμούς
- Ερώτηση:
  - Ποια η δεκαδική τιμή του 1011_2?
- Απάντηση:
  - Χρειαζόμαστε και επιπλέον πληροφορία σχετικά με το εάν ο αριθμός είναι προσημασμένος
  - 11_{10} (απρόσημος)
  - -5_{10} (προσημασμένος)

Προσημασμένοι αριθμοί των 4-bits

Προσημασ. Δεκαδικοί	Δυαδικοί
-8	1000
-7	1001
-6	…
-5	1011
…	…
-1	1111
0	0000
1	0001
…	…
5	0101
6	0110
7	0111

Υπολογισμός αντιθέτου

- Αντιστρέψτε και προσθέστε 1
  - Αντιστροφή σημαίνει 1 → 0, 0 → 1

$$x + \bar{x} = 1111...111_2 = -1$$

$$\bar{x} + 1 = -x$$

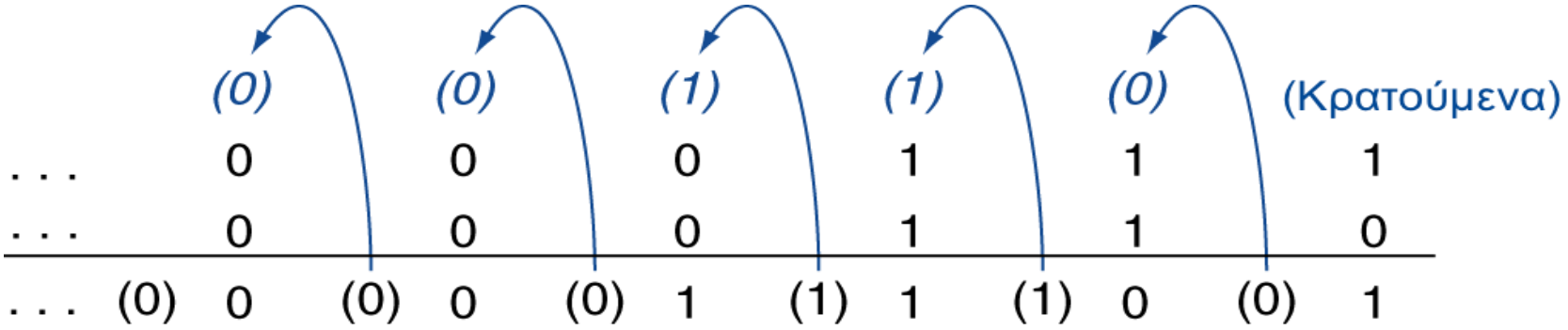
- Παράδειγμα: το αντίθετο του +2
  - +2 = 0000 0000 … 0010_2
  - –2 = 1111 1111 … 1101_2 + 1
    = 1111 1111 … 1110_2

Επέκταση προσήμου

- Αναπαράσταση αριθμού με περισσότερα bits
  - Διατηρώντας την αριθμητική τιμή του
- Αντιγράψτε το πρόσημο αριστερά
  - π.χ. απρόσημες τιμές: επεκτείνονται με μηδενικά
- Παραδείγματα: 8-bit σε 16-bit
  - +2: 0000 0010 => 0000 0000 0000 0010
  - –2: 1111 1110 => 1111 1111 1111 1110

Πρόσθεση ακεραίων

- Παράδειγμα: 7 + 6



- Παράδειγμα με πολλαπλούς ακεραίους: 10 + 15 + 23 = 48

+	0	0	1	0	1	0
	0	0	1	1	1	1
	0	1	0	1	1	1
	1	1	0	0	0	0

Αφαίρεση ακεραίων

- Προσθέστε το αντίθετο του δεύτερου τελεστέου
- Παράδειγμα: 7 – 6 = 7 + (–6)

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

- Τόσο η πρόσθεση όσο και η αφαίρεση μπορούν να οδηγήσουν σε υπερχείλιση αποτελέσματος (overflow)

Υπερχείλιση (overflow)

- Υπερχείλιση: κατάσταση μη έγκυρου αποτελέσματος (το αποτέλεσμα μιας πράξης χρειάζεται περισσότερα από τα διαθέσιμα ψηφία για να αναπαρασταθεί)
- Παράδειγμα μη προσημασμένης πρόσθεσης
  - Έστω ότι θέλουμε να προσθέσουμε δύο **απρόσημους** αριθμούς , και να τοποθετήσουμε το αποτέλεσμα σε ένα καταχωρητή των 4-bits.

  +14: 1110
  + 3: 0011
  +17: **10001**

  - Το πιο σημαντικό bit δεν χωράει στον καταχωρητή → overflow
  - Άρα, όταν το κρατούμενο είναι $c_n = 1$, τότε έχουμε υπερχείλιση στην πρόσθεση

Υπερχείλιση (overflow)

- Παράδειγμα προσημασμένης πρόσθεσης
  - Έστω ότι θέλουμε να προσθέσουμε δύο **προσημασμένους**, θετικούς αριθμούς , και να τοποθετήσουμε το αποτέλεσμα σε ένα καταχωρητή των 5-bits.

+12: 01100
+ 8: 01000
+20: **010100**

  - Ο αριθμός είναι αρνητικός → overflow
  - Άρα, όταν το κρατούμενο είναι $c_n <> c_{n-1}$, τότε έχουμε υπερχείλιση στην πρόσθεση προσημασμένων αριθμών. Εδώ $c_n = 0$, $c_{n-1} = 1$

Υπερχείλιση (overflow)

- Παράδειγμα προσημασμένης πρόσθεσης
  - Έστω ότι θέλουμε να προσθέσουμε δύο **προσημασμένους**, αρνητικούς αριθμούς , και να τοποθετήσουμε το αποτέλεσμα σε ένα καταχωρητή των 5-bits.

  -12: 10100
  - 8: 11000
  -20: **101100**

  - Άρα, όταν το κρατούμενο είναι $c_n <> c_{n-1}$, τότε έχουμε υπερχείλιση στην πρόσθεση προσημασμένων αριθμών. Εδώ $c_n = 1$, $c_{n-1} = 0$
  - Δεν μπορούμε ποτέ να έχουμε υπερχείλιση όταν προσθέτουμε έναν θετικό και έναν αρνητικό αριθμό
  - Δεν μπορούμε ποτέ να έχουμε υπερχείλιση όταν αφαιρούμε έναν θετικό από έναν θετικό αριθμό ή το αντίθετο

Υπερχείλιση (overflow)

- Υπερχείλιση πρόσθεσης A+B μη προσημασμένων αριθμών, n bit:
  - $A+B \geq 2^n \Leftrightarrow c_n = 1$ (c_n : κρατούμενο εξόδου από το MSB)
- Η αφαίρεση A-B ή B-A μη προσημασμένων αριθμών δεν παρουσιάζει υπερχείλιση
- Υπερχείλιση πρόσθεσης A+B προσημασμένων αριθμών:
  - εάν $A \geq 0, B \geq 0$: $A+B \geq 2^{n-1}$ εάν $c_n = 0$: $c_{n-1} = 1$

    εάν $A < 0, B < 0$: $A+B < -2^{n-1}$ εάν $c_n = 1$: $c_{n-1} = 0$

    εάν $A \cdot B < 0$ (περίπτωση χωρίς υπερχείλιση) τότε πάντοτε είναι $c_n = c_{n-1} = 0$ ή $c_n = c_{n-1} = 1$ (c_{n-1} : κρατούμενο εισόδου προς το MSB)
  - Συνολική συνθήκη υπερχείλισης: MSB(X) = MSB(Y) ≠ MSB(X+Y)
  - Ισοδύναμη συνθήκη υπερχείλισης: (c_n xor c_{n-1}) = 1

Αριθμητικές και συγκριτικές πράξεις στον MIPS

- Αριθμητικές πράξεις προσημασμένων αριθμών:

  add, sub, addi

- Συγκριτικές πράξεις προσημασμένων αριθμών:

  slt, slti

- Η σταθερά 16-bit στις εντολές addi και slti - καθώς και στις εντολές διακλαδώσεων και μεταφοράς δεδομένων όπου αποτελεί σχετική μετατόπιση - είναι *προσημασμένη* (δηλ. είναι είτε θετική είτε αρνητική) και λαμβάνει επέκταση προσήμου μέχρι τα 32-bits
- Τυχόν υπερχείλιση στις εντολές add, addi, sub προκαλεί *εξαίρεση* την οποία διαχειρίζεται η αρμόδια ρουτίνα του λειτουργικού συστήματος

Αριθμητικές και συγκριτικές πράξεις στον MIPS

- Αριθμητικές πράξεις μη προσημασμένων αριθμών:

  addu $s0,$s1,$s2 ⇔ $s0 = $s1 + $s2 ; no_overflow

  subu $s0,$s1,$s2 ⇔ $s0 = $s1 - $s2 ; no_overflow

  addiu $s0,$s1,100 ⇔ $s0 = $s1 + 100 ; no_overflow
- Οι addu, subu, addiu παράγουν το ίδιο αποτέλεσμα με τις add, sub, addi αλλά δεν προκαλούν εξαίρεση σε πιθανή υπερχείλιση

  => χρησιμοποιούνται - εκτός από το χειρισμό μη προσημασμένων δεδομένων, π.χ. *διευθύνσεων* - όταν δεν θέλουμε να διακοπεί η ροή του προγράμματος από το λειτουργικό σύστημα σε περίπτωση υπερχείλισης (τότε η ύπαρξη υπερχείλισης θα πρέπει να ελέγχεται μέσα από το ίδιο το πρόγραμμα)
- Συγκριτικές πράξεις μη προσημασμένων αριθμών:

  sltu $s0,$s1,$s2 ⇔ $s0 = ($s1 < $s2)$_u$

  sltiu $s0,$s1,100 ⇔ $s0 = ($s1 < 100)$_u$
- Προσοχή. Στις εντολές addiu και sltiu η 16-bit σταθερά είναι *προσημασμένη* και λαμβάνει επέκταση προσήμου μέχρι τα 32-bits, και μόνο η τελική πράξη της πρόσθεσης ή της σύγκρισης αφορά μη προσημασμένους αριθμούς

Άλλες εντολές πράξεων στον MIPS

- Εντολές ολίσθησης (εκτός της sll):

  srl $s0,$s1,10 ⇔ $s0 = $s1 >> 10 ; zero_extend

  sra $s0,$s1,10 ⇔ $s0 = $s1 >> 10 ; sign_extend

  sllv $s0,$s1,$s2 ⇔ $s0 = $s1 << $s2

  srlv $s0,$s1,$s2 ⇔ $s0 = $s1 >> $s2 ; zero_extend

  srav $s0,$s1,$s2 ⇔ $s0 = $s1 >> $s2 ; sign_extend

- Η *αριθμητική* δεξιά ολίσθηση κατά n-bits ισοδυναμεί με διαίρεση δια 2^n για όσους προσημασμένους - θετικούς ή αρνητικούς - αριθμούς διαιρούνται *ακριβώς* με το 2^n (δηλ. για εκείνους που έχουν τουλάχιστον n ψηφία 0 στις λιγότερο σημαντικές θέσεις τους)

  Προσοχή. Εάν ένας αριθμός δεν διαιρείται ακριβώς με το 2^n τότε η αριθμητική δεξιά ολίσθηση δίνει το αποτέλεσμα της διαίρεσής του με το 2^n στρογγυλοποιημένο προς το *μικρότερο* ακέραιο

  => για θετικούς αριθμούς δίνει ακριβώς το πηλίκο της διαίρεσης

  => για αρνητικούς αριθμούς δίνει το πηλίκο της διαίρεσης μείον 1