Αρχιτεκτονική Υπολογιστών

Κεφάλαιο 3

Αριθμητική για υπολογιστές

[Έχει χρησιμοποιηθεί υλικό από τις διαφάνειες
Computer Organization and Design, 4th Edition,
Patterson & Hennessy, © 2008, MK]

Εισαγωγή

- Ακέραιοι αριθμοί
  - Αναπαράσταση
  - Πρόσθεση και αφαίρεση
  - Πολλαπλασιασμός και διαίρεση
  - Χειρισμός της υπερχείλισης
- Πραγματικοί αριθμοί κινητής υποδιαστολής
  - Αναπαράσταση και πράξεις

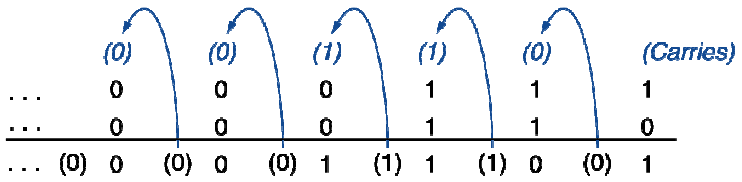
Αναπαράσταση προσημασμένων

- **Αναπαράσταση σε συμπλήρωμα ως προς 2**
  - Το πρώτο bit δείχνει το πρόσημο
    - 0 θετικός, 1 αρνητικός

- $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2 = 0_{10}$
- $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_2 = 1_{10}$
- $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_2 = 2_{10}$
- .....
- $0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_2 = 2.147.483.645_{10}$
- $0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_2 = 2.147.483.646_{10}$
- **$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_2 = 2.147.483.647_{10}$**
- **$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2 = -2.147.483.648_{10}$**
- $1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_2 = -2.147.483.647_{10}$
- $1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_2 = -2.147.483.646_{10}$
- .....
- $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_2 = -3_{10}$
- $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_2 = -2t_{10}$
- $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_2 = -1_{10}$

Πρόσθεση ακεραίων

- Παράδειγμα: 7 + 6



Αφαίρεση ακεραίων

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

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

Υπερχείλιση

- Το πρόβλημα της υπερχείλισης (overflow)
  - Δεν χωράει το αποτέλεσμα στα 32 bit

Πράξη	Τελεστέος A	Τελεστέος B	Αποτέλεσμα που δείχνει υπερχείλιση
A + B	≥ 0	≥ 0	< 0
A + B	< 0	< 0	≥ 0
A – B	≥ 0	< 0	< 0
A – B	< 0	≥ 0	≥ 0

Χειρισμός υπερχείλισης

- Τι γίνεται με τους απρόσημους ακεραίους;
  - Χρησιμοποιούνται συνήθως για διευθύνσεις μνήμης (ή χαρακτήρες) όπου οι υπερχειλίσεις αγνοούνται
- Ο MIPS παρέχει ένα τρόπο για την αγνόηση της υπερχείλισης σε μερικές περιπτώσεις και την αναγνώρισή της σε άλλες
- Ύπαρξη δύο ειδών αριθμητικών εντολών:
  - **add (`add`), add immediate (`addi`), subtract (`sub`)**
    - προκαλούν εξαιρέσεις (exceptions) κατά την υπερχείλιση
  - **add unsigned (`addu`), add immediate unsigned (`addiu`), subtract unsigned (`subu`)**
    - **δεν** προκαλούν εξαιρέσεις κατά την υπερχείλιση

Χειρισμός υπερχείλισης

- Μερικές γλώσσες (π.χ., C) αγνοούν την υπερχείλιση
  - Χρήση των εντολών `addu`, `addui`, `subu`
- Άλλες εντολές (π.χ., Ada, Fortran) απαιτούν να προκληθεί εξαίρεση
  - Χρήση των εντολών `add`, `addi`, `sub`

Εξαίρεση

- Ο MIPS εντοπίζει τις υπερχειλίσεις με μια **εξαίρεση (exception)**
  - Σε πολλούς υπολογιστές ονομάζεται και διακοπή (interrupt)
- Εξαίρεση : Μη προγραμματισμένο συμβάν που διακόπτει την εκτέλεση του προγράμματος
- Όταν προκύψει εξαίρεση από υπερχείλιση εκτελούνται κάποιες ενέργειες:
  - Αποθηκεύεται η διεύθυνση της εντολής που υπερχείλισε
  - Ο υπολογιστής πραγματοποιεί άλμα σε μια προκαθορισμένη διεύθυνση ώστε να καλέσει την κατάλληλη ρουτίνα εξαίρεσης (exception routine)
  - Το πρόγραμμα μπορεί να συνεχίσει μετά την εκτέλεση της ρουτίνας εξαίρεσης από εκεί που είχε σταματήσει

Αριθμητική για πολυμέσα

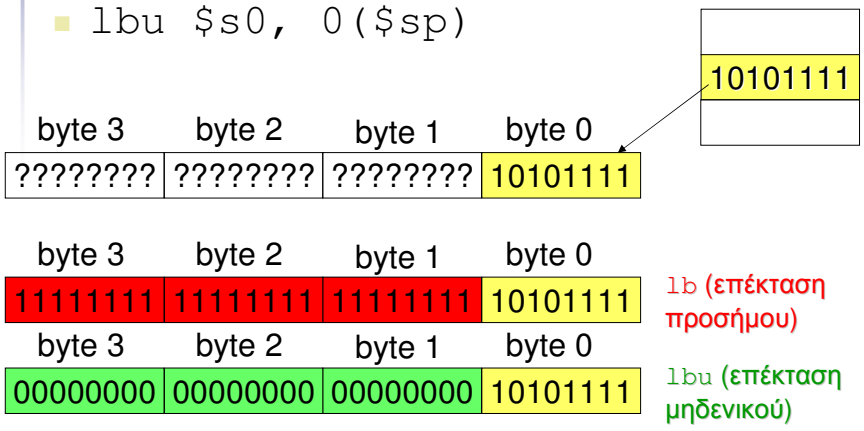
- Η επεξεργασία γραφικών λειτουργεί σε διανύσματα των 8-bit και 16-bit
  - Χρήση αθροιστή των 64-bit adder, με διαμέριση στις αλυσίδες κρατουμένων
    - Πράξεις σε διανύσματα των 8×8-bit, 4×16-bit ή 2×32-bit
  - SIMD (single-instruction, multiple-data)
- Λειτουργίες κορεσμού (saturation)
  - Στην υπερχείλιση, το αποτέλεσμα παίρνει τη μεγαλύτερη τιμή που μπορεί να αναπαρασταθεί
  - Δεν χρησιμοποιείται σε επεξεργαστές γενικού σκοπού
    - Σε επεξεργαστές ψηφιακής επεξεργασίας σήματος (digital signal processors)

Φόρτωση προσημασμένων

- Πρέπει να διατηρηθεί το πρόσημο του αριθμού
  - επέκταση προσήμου (sign extension)
- Στις απρόσημες φορτώσεις byte και half word
  - επέκταση μηδενικού (zero extension)
- Δύο παραλλαγές φόρτωσης byte:
  - **load byte** `(lb)`: για προσημασμένους αριθμούς
    - Επέκταση προσήμου
  - **load byte unsigned** `(lbu)`: για απρόσημους αριθμούς
    - Επέκταση μηδενικού
- Παρόμοια, για φορτώσεις half word:
  - **load half** `(lh)`
  - **load half-word unsigned** `(lhu)`

Επέκταση προσήμου

- `lb $s0, 0($sp)`
- `lbu $s0, 0($sp)`



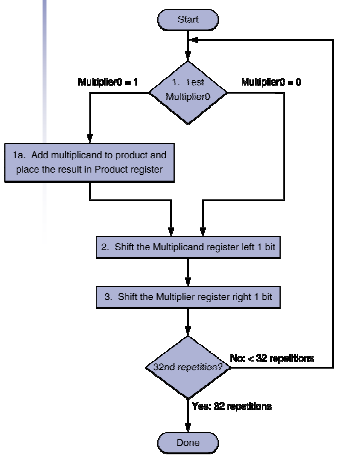
Πολλαπλασιασμός

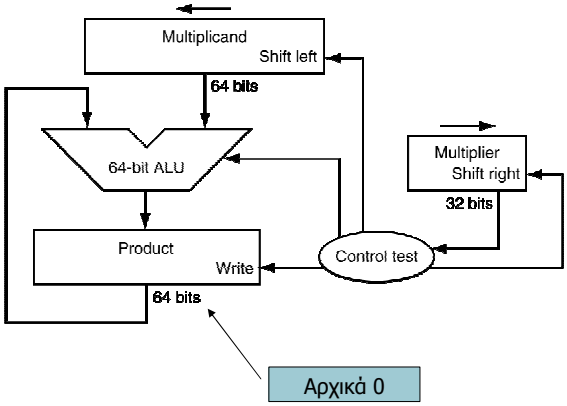
- Αλγόριθμος

πολλαπλασιαστέος → 1000
πολλαπλασιαστής → × 1001
1000
0000
0000
1000
γινόμενο → 1001000

Το μήκος του γινομένου είναι ίσο με το άθροισμα των μηκών των τελεστέων

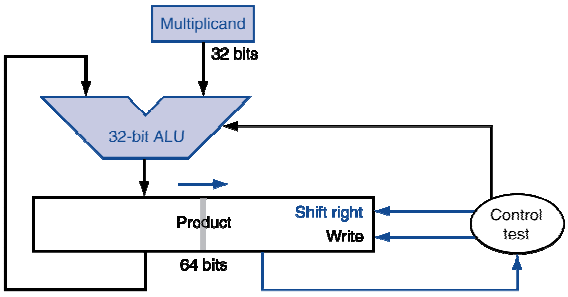
Κύκλωμα πολλαπλασιασμού





Βελτιστοποιημένο κύκλωμα

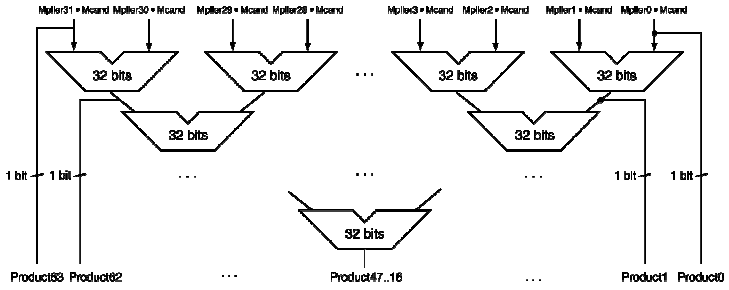
- Εκτελεί τα βήματα παράλληλα: πρόσθεση/ολίσθηση



- Ένας κύκλος ανά πρόσθεση μερικού γινομένου
  - Αποδεκτό, εάν οι πολλαπλασιασμοί εκτελούνται αραιά

Παράλληλος πολλαπλασιαστής

- Χρησιμοποιεί πολλαπλούς αθροιστές
  - Συμβιβασμός κόστους/απόδοσης



- Μπορεί να έχει διοχέτευση (pipeline)
  - Εκτελεί πολλούς πολλαπλασιασμούς παράλληλα

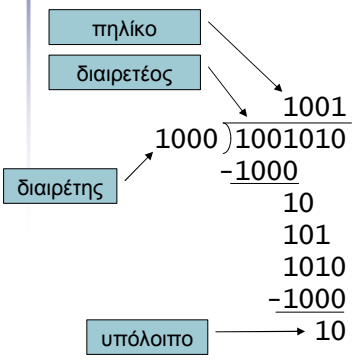
Πολλαπλασιασμός στον MIPS

- 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`
    - 64-bit γινόμενο στους HI/LO
  - `mfhi rd / mflo rd`
    - Move from HI/LO to rd
    - Μπορούμε να ελέγξουμε την τιμή HI για να δούμε εάν το γινόμενο δεν χωράει στα 32 bit
  - `mul rd, rs, rt`
    - Τα λιγότερο σημαντικά 32 bit του γινομένου –> rd

Μείωση δύναμης

- Κάποιοι μεταγλωττιστές αντικαθιστούν τις πράξεις πολλαπλασιασμούς με σταθερές με μια σειρά από ολισθήσεις και προσθέσεις
- Παράδειγμα:
  - Κώδικας MIPS για τις παρακάτω εντολές με μείωση δύναμης (strength reduction), (a στον $s0, b στον $s1)
  - a = 2 * b;
  - a = 65 * b;

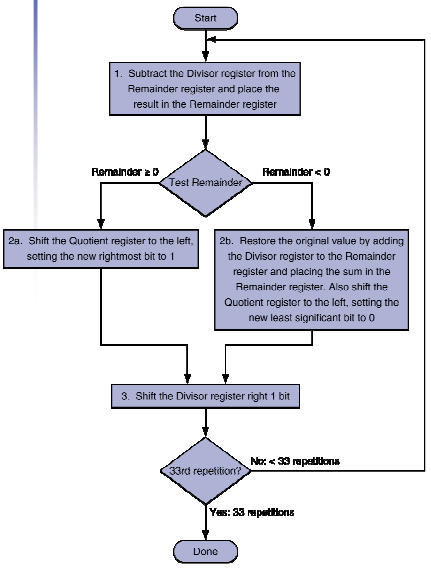
Διαίρεση

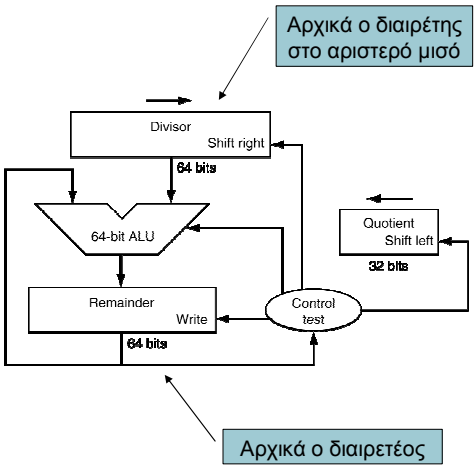


- Έλεγχος για διαίρεση με 0
- Αλγόριθμος διαίρεσης
  - Εάν διαιρέτης ≤ bit διαιρετέου
    - 1 στο πηλίκο, αφαίρεση
  - διαφορετικά
    - 0 στο πηλίκο, κατεβάζουμε το επόμενο bit του διαιρετέου

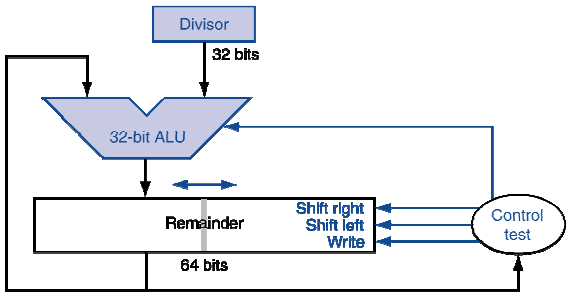
Τελεστέοι των *n*-bit παράγουν πηλίκο των *n*-bit και υπόλοιπο

Κύκλωμα διαίρεσης





Βελτιστοποιημένο κύκλωμα



- Ένας κύκλος ανά αφαίρεση μερικού υπολοίπου
- Μοιάζει πολύ με τον πολλαπλασιαστή!
  - Μπορεί να χρησιμοποιηθεί το ίδιο υλικό και για τις δύο

Ταχύτερη διαίρεση

- Δεν μπορεί να χρησιμοποιήσει παράλληλο υλικό όπως στον πολλαπλασιασμό
  - Το εάν θα γίνει αφαίρεση εξαρτάται από το πρόσημο του υπολοίπου
- Ταχύτεροι διαιρέτες παράγουν πολλαπλά bit πηλίκου σε κάθε βήμα
  - Και πάλι απαιτούν πολλά βήματα

Διαίρεση στον MIPS

- Καταχωρητές HI/LO για το αποτέλεσμα
  - HI: 32-bit υπόλοιπο
  - LO: 32-bit πηλίκο
- Εντολές
  - `div rs, rt` / `divu rs, rt`
  - Κανένας έλεγχος για υπερχείλιση ή διαίρεση με το 0
    - Εάν απαιτείται ο έλεγχος πρέπει να γίνει από το λογισμικό
  - Χρήση των `mfhi`, `mflo` για να πάρουμε το αποτέλεσμα

Δεξιά ολίσθηση και διαίρεση

- Αριστερή ολίσθηση κατά *i* θέσεις ισοδυναμεί με πολλαπλασιασμό με 2^i
- Δεξιά ολίσθηση διαιρεί με 2^i ?
  - Μόνο για απρόσημους αριθμούς
- Για προσημασμένους ακεραίους
  - Αριθμητική δεξιά ολίσθηση: επαναλαμβάνει το bit προσήμου
  - π.χ., –5 / 4
    - 11111011_2 >> 2 = 11111110_2 = –2
    - στρογγυλοποίηση προς το –∞
  - αντί για 11111011_2 >>> 2 = 00111110_2 = +62

Συμβολική γλώσσα MIPS

Κατηγορία	Εντολή	Παράδειγμα	Σημασία	Σχόλια
Αριθμητικές πράξεις	multiply	mult $s2, $s3	Hi, Lo = $s2 x $s3	προσημασμένο γινόμενο 64 bit στους Hi, Lo
	multiply unsigned	multu $s2, $s3	Hi, Lo = $s2 x $s3	απρόσημο γινόμενο 64 bit στους Hi, Lo
	divide	div $s2, $s3	Lo = $s2 / $s3, Hi = $s2 mod $s3	Lo = πηλίκο, Hi = υπόλοιπο
	divide unsigned	divu $s2, $s3	Lo = $s2 / $s3, Hi = $s2 mod $s3	Απρόσημο πηλίκο και υπόλοιπο
	move from Hi	mfhi $s1	$s1 = Hi	Χρησιμοποιείται για να πάρει αντίγραφο του Hi
	move from Lo	mflo $s1	$s1 = Lo	Χρησιμοποιείται για να πάρει αντίγραφο του Lo

Αναπαράσταση πραγματικών

- Επιστημονική σημειογραφία (scientific notation)
  - Μια σημειογραφία που δίνει αριθμούς με ένα μόνο ψηφίο στα αριστερά της υποδιαστολής
  - Υποστηρίζει πολύ μικρούς και πολύ μεγάλους αριθμούς
- Κανονικοποιημένος (normalized)
  - Ένας αριθμός σε επιστημονική σημειογραφία χωρίς αρχικά 0
- Παραδείγματα:
  - $1{,}0_{10} \times 10^{-9}$ [κανονικοποιημένος]
  - $0{,}1_{10} \times 10^{-8}$ και $10{,}0_{10} \times 10^{-10}$ [μη κανονικοποιημένοι]

Δυαδικοί σε επιστημονική σημειογραφία

- Δυαδικός πραγματικός αριθμός:
  $1{,}0_{two} \times 2^{-1} = 0{,}1_{two}$
- Αριθμητική **κινητής υποδιαστολής (floating point - FP)**
  - Υποστηρίζει δυαδικούς αριθμούς σε επιστημονική σημειογραφία
  - Αναπαριστά αριθμούς στους οποίους η υποδιαστολή δεν είναι σε σταθερή θέση
  - Η γλώσσα C χρησιμοποιεί τους τύπους: float, double

Αναπαράσταση FP

- Οι αριθμοί κινητής υποδιαστολής έχουν τη μορφή:

  $$1{,}xxxxxxxxx_{two} \times 2^{yyyy}$$

- **κλάσμα** (fraction):
  Η τιμή που τοποθετείται στο κλασματικό πεδίο
- **εκθέτης** (exponent):
  Η τιμή που τοποθετείται στο πεδίο του εκθέτη

Αναπαράσταση FP

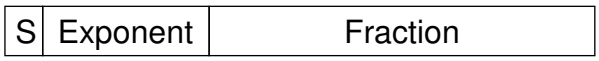
- Συμβιβασμός μεταξύ των μεγεθών του κλάσματος (fraction) και του εκθέτη (exponent)
  - Λέξη σταθερού μεγέθους
- Συμβιβασμός μεταξύ ακρίβειας (precision) και εύρους (range)
  - Η αύξηση του μεγέθους του κλάσματος ενισχύει την ακρίβεια
  - Η αύξηση του μεγέθους του εκθέτη αυξάνει το εύρος των αριθμών

Πρότυπο FP

- Ορίζεται από το IEEE Std 754-1985
- Αναπτύχθηκε ως απάντηση στην εμφάνιση διάφορων αναπαραστάσεων
  - Φορητότητα για επιστημονικό κώδικα
- Πλέον καθολική αποδοχή
- Δύο αναπαραστάσεις
  - Απλή ακρίβεια (single precision): 32-bit
  - Διπλή ακρίβεια (double precision): 64-bit

IEEE Floating-Point

απλή: 8 bit διπλή: 11 bit | απλή: 23 bit διπλή: 52 bit



$$x = (-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

- S: bit προσήμου (0 ⇒ μη αρνητικός, 1 ⇒ αρνητικός)
- Κανονικοποιημένο σημαντικό (significand): 1.0 ≤ |significand| < 2.0
  - Έχει πάντα ένα bit 1 πριν την υποδιαστολή, άρα δεν χρειάζεται να το αναπαραστήσουμε (κρυμμένο bit)
  - Σημαντικό = 1 + Κλάσμα
- Εκθέτης (exponent): πολωμένη σημειογραφία (biased notation): πραγματικός εκθέτης + πόλωση (bias)
  - Εγγυάται ότι ο εκθέτης είναι απρόσημος
  - Απλή: Πόλωση= 127; Διπλή: Πόλωση= 1203

Εύρος απλής ακρίβειας

- Εκθέτες 00000000 και 11111111 δεσμευμένοι
- 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

- Εκθέτες 0000…00 και 1111…11 δεσμευμένοι
- Μικρότερη τιμή
  - Εκθέτης: 00000000001
    ⇒ πραγματικός εκθέτης = 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 FP

- Σχετική ακρίβεια
  - Όλα τα bit του κλάσματος είναι σημαντικά
  - Απλή: περίπου 2^{-23}
    - Ισοδύναμο με $23 \times \log_{10}2 \approx 23 \times 0.3 \approx 6$ 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 FP

- Αναπαράσταση του –0.75
  - $-0.75 = (-1)^1 \times 1.1_2 \times 2^{-1}$
  - S = 1
  - Κλάσμα = $1000\ldots00_2$
  - Εκθέτης = –1 + Πόλωση
    - Απλή: $-1 + 127 = 126 = 01111110_2$
    - Διπλή: $-1 + 1023 = 1022 = 01111111110_2$
- Απλή: 10111111101000…00
- Διπλή: 10111111111101000…00

Παράδειγμα FP

- Ποιος αριθμός αναπαριστάνεται από την μορφή απλής ακρίβειας
  11000000101000…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 αριθμοί

- Εκθέτης = 000...0 ⇒ το κρυμμένο bit 0

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

- Μικρότεροι από τους κανονικοποιημένους
  - επιτρέπουν ανεπάρκεια (underflow) με φθίνουσα ακρίβεια
- Με κλάσμα = 000...0

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

Δύο αναπαράστασεις του 0.0!

Άπειρο και NaN

- Εκθέτης = 111...1, Κλάσμα = 000...0
  - ± άπειρο
  - Μπορεί να χρησιμοποιηθεί για να αποφύγουμε την ανάγκη για έλεγχο υπερχείλισης
- Εκθέτης = 111...1, Κλάσμα ≠ 000...0
  - Not-a-Number (NaN)
  - Υποδηλώνει μη επιτρεπτό ή μη καθορισμένο αποτέλεσμα
    - π.χ., 0.0 / 0.0

Πρόσθεση FP

- Θεωρήστε δεκαδικούς των 4 ψηφίων
  - $9.999 \times 10^{1} + 1.610 \times 10^{-1}$
- 1. Ευθυγράμμιση των υποδιαστολών
  - Ολίσθηση του αριθμού με το μικρότερο εκθέτη
  - $9.999 \times 10^{1} + 0.016 \times 10^{1}$
- 2. Πρόσθεση των σημαντικών
  - $9.999 \times 10^{1} + 0.016 \times 10^{1} = 10.015 \times 10^{1}$
- 3. Κανονικοποίηση αποτελέσματος και έλεγχος για over/underflow
  - 1.0015×10^{2}
- 4. Στρογγυλοποίηση και κανονικοποίηση ξανά (εάν χρειάζεται)
  - 1.002×10^{2}

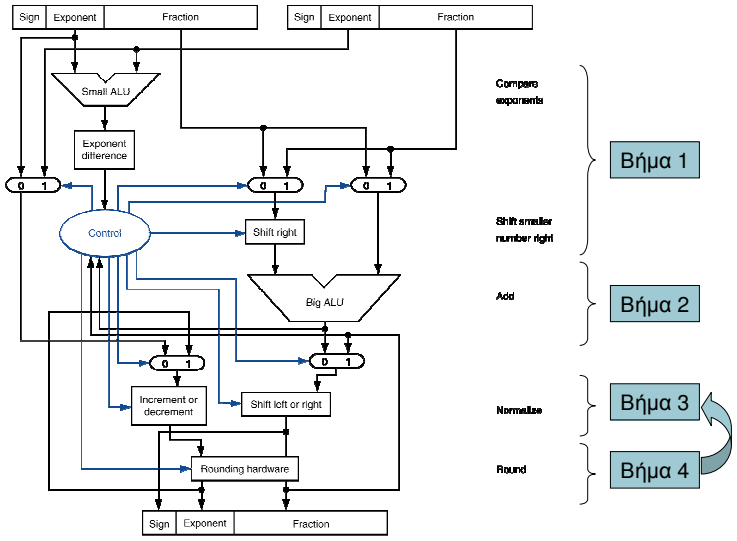
Πρόσθεση FP

- Θεωρήστε δυαδικούς των 4 ψηφίων
  - $1.000_2 \times 2^{-1} + -1.110_2 \times 2^{-2}$ (0.5 + –0.4375)
- 1. Ευθυγράμμιση των δυαδικών υποδιαστολών
  - Ολίσθηση του αριθμού με το μικρότερο εκθέτη
  - $1.000_2 \times 2^{-1} + -0.111_2 \times 2^{-1}$
- 2. Πρόσθεση των σημαντικών
  - $1.000_2 \times 2^{-1} + -0.111_2 \times 2^{-1} = 0.001_2 \times 2^{-1}$
- 3. Κανονικοποίηση αποτελέσματος και έλεγχος για over/underflow
  - $1.000_2 \times 2^{-4}$
- 4. Στρογγυλοποίηση και κανονικοποίηση ξανά (εάν χρειάζεται)
  - $1.000_2 \times 2^{-4} = 0.0625$

Κύκλωμα αθροιστή FP

- Πιο πολύπλοκο από τον αθροιστή ακεραίων
- Εάν εκτελεστεί η πράξη σε έναν κύκλο θα διαρκέσει πολύ
  - Αργό ρολόι θα επιβάρυνε την εκτέλεση και των άλλων εντολών
- Οι αθροιστές FP συνήθως εκτελούν την πράξη σε πολλούς κύκλους
  - Μπορεί να έχουν διοχέτευση (pipeline)

Κύκλωμα αθροιστή FP



Πολλαπλασιασμός FP

- Θεωρήστε δεκαδικούς των 4 ψηφίων
  - $1.110 \times 10^{10} \times 9.200 \times 10^{-5}$
- 1. Πρόσθεση των εκθετών
  - Για πολωμένους εκθέτες, αφαίρεση της πόλωσης από το άθροισμα
  - Νέος εκθέτης = 10 + –5 = 5
- 2. Πολλαπλασιασμός των σημαντικών
  - $1.110 \times 9.200 = 10.212 \Rightarrow 10.212 \times 10^{5}$
- 3. Κανονικοποίηση αποτελέσματος και έλεγχος για over/underflow
  - 1.0212×10^{6}
- 4. Στρογγυλοποίηση και κανονικοποίηση ξανά (εάν χρειάζεται)
  - 1.021×10^{6}
- 5. Καθορισμός του προσήμου του αποτελέσματος
  - $+1.021 \times 10^{6}$

Πολλαπλασιασμός FP

- Θεωρήστε δυαδικούς των 4 ψηφίων
  - $1.000_2 \times 2^{-1} \times -1.110_2 \times 2^{-2}$ (0.5 × –0.4375)
- 1. Πρόσθεση των εκθετών
  - Μη πολωμένοι: –1 + –2 = –3
  - Πολωμένοι: (–1 + 127) + (–2 + 127) = –3 + 254 – 127 = –3 + 127
- 2. Πολλαπλασιασμός των σημαντικών
  - $1.000_2 \times 1.110_2 = 1.1102 \Rightarrow 1.110_2 \times 2^{-3}$
- 3. Κανονικοποίηση αποτελέσματος και έλεγχος για over/underflow
  - $1.110_2 \times 2^{-3}$
- 4. Στρογγυλοποίηση και κανονικοποίηση ξανά (εάν χρειάζεται)
  - $1.110_2 \times 2^{-3}$
- 5. Καθορισμός του προσήμου: $+v \times -v \Rightarrow -v$
  - $-1.110_2 \times 2^{-3} = -0.21875$

Κυκλώματα αριθμητικής FP

- Ο πολλαπλασιαστής FP έχει παρόμοια (ή μικρότερη) πολυπλοκότητα με τον αθροιστή FP
  - Χρησιμοποιεί πολλαπλασιαστή αντί αθροιστή για τα σημαντικά
  - Δεν ευθυγραμμίζει τους εκθέτες
- Τα κυκλώματα αριθμητικής FP συνήθως εκτελούν
  - Πρόσθεση, αφαίρεση, πολλαπλασιασμό, διαίρεση, τετραγωνική ρίζα
  - Μετατροπές FP ↔ integer
- Οι λειτουργίες διαρκούν πολλούς κύκλους
  - Μπορεί να έχουν διοχέτευση (pipeline)

Εντολές FP στον MIPS

- Υλικό FP στον συνεπεξεργαστή 1
  - Πρόσθετος επεξεργαστής που επεκτείνει το σύνολο εντολών
- Ξεχωριστοί καταχωρητές FP
  - 32 απλής ακρίβειας: $f0, $f1, … $f31
  - Σε ζευγάρια για διπλή ακρίβεια: $f0/$f1, $f2/$f3, …
- Οι εντολές FP εκτελούνται μόνο σε καταχωρητές FP
  - Γενικά τα προγράμματα δεν εκτελούν λειτουργίες ακεραίων σε δεδομένα FP data και το ανάποδο
  - Περισσότεροι καταχωρητές με μικρή επίπτωση στο μέγεθος του κώδικα
- Εντολές φόρτωσης και αποθήκευσης FP s
  - `lwc1`, `ldc1`, `swc1`, `sdc1`
    - π.χ., `ldc1 $f8, 32($sp)`

Εντολές FP 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`, …)
  - Θέτει στο 1 ή μηδενίζει το bit συνθήκης FP
    - π.χ. `c.lt.s $f3, $f4`
- Διακλάδωση εάν η συνθήκη FP είναι αληθής (true) ή ψευδής (false)
  - `bc1t`, `bc1f`
    - π.χ., `bc1t TargetLabel`

Συμβολική γλώσσα MIPS

Κατηγορία	Εντολή	Παράδειγμα	Σημασία	Σχόλια
Αριθμητικές πράξεις	FP add single	add.s f2,$f4,$f6	$f2 = $f4 + $f6	Πρόσθεση FP (απλή ακρίβεια)
	FP subtract single	sub.s $f2,$f4,$f6	$f2 = $f4 - $f6	Αφαίρεση FP (απλή ακρίβεια)
	FP multiply single	mul.s $f2,$f4,$f6	$f2 = $f4 x $f6	Πολλαπλασιασμός FP (απλή ακρίβεια)
	FP divide single	div.s $f2,$f4,$f6	$f2 = $f4 / $f6	Διαίρεση FP (απλή ακρίβεια)
	FP add double	add.d $f2,$f4,$f6	$f2 = $f4 + $f6	Πρόσθεση FP (διπλή ακρίβεια)
	FP subtract double	sub.d $f2,$f4,$f6	$f2 = $f4 - $f6	Αφαίρεση FP (διπλή ακρίβεια)
	FP multiply double	mul.d $f2,$f4,$f6	$f2 = $f4 x $f6	Πολλαπλασιασμός FP (διπλή ακρίβεια)
	FP divide double	div.d $f2,$f4,$f6	$f2 = $f4 / $f6	Διαίρεση FP (διπλή ακρίβεια)

Συμβολική γλώσσα MIPS

Κατηγορία	Εντολή	Παράδειγμα	Σημασία	Σχόλια
Μεταφορά δεδομένων	load word copr. 1	lwc1 $f1,100($s2)	$f1 = Memory[$s2 + 100]	Δεδομένα 32 bit σε καταχωρητή FP
	store word copr. 1	swc1 $f1,100($s2)	Memory[$s2 + 100] = $f1	Δεδομένα 32 bit στη μνήμη
Διακλάδωση υπό συνθήκη	branch on FP true	bc1t 25	αν (cond == 1) μετάβαση στο PC + 4 + 100	Σχετική ως προς PC διακλάδωση αν ισχύει συνθήκη FP
	branch on FP false	bc1f 25	αν (cond == 0) μετάβαση στο PC + 4 + 100	Σχετική ως προς PC διακλάδωση αν δεν ισχύει συνθήκη FP
	FP compare single (eq, ne,lt,le,gt,ge)	c.lt.s $f2,$f4	αν ($f2 < $f4) τότε cond = 1· αλλιώς cond = 0	Σύγκριση FP «μικρότερο από», απλής ακρίβειας
	FP compare double (eq, ne,lt,le,gt,ge)	c.lt.d $f2,$f4	αν ($f2 < $f4) τότε cond = 1· αλλιώς cond = 0	Σύγκριση FP «μικρότερο από», διπλής ακρίβειας

Παράδειγμα FP : °F σε °C

- Παράδειγμα: Μετατροπή θερμοκρασίας από Φαρενάιτ σε βαθμούς Κελσίου:

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

  - fahr στον $f12, αποτέλεσμα στον $f0, οι σταθερές 5.0, 9.0 και 32.0 στη μνήμη

Κώδικας MIPS

```
# Φόρτωση των σταθερών σε καταχωρητές ΚΥ:
f2c:
lwc1   $f16,const5($gp)   # $f16 = 5.0
lwc1   $f18,const9($gp)   # $f18 = 9.0
div.s  $f16, $f16, $f18   # $f16 = 5.0 / 9.0
lwc1   $f18,const32($gp)  # $f18 = 32.0
# Μετατροπή
sub.s  $f18, $f12, $f18   # $f18 = fahr – 32.0
mul.s  $f0, $f16, $f18    # $f0 = (5/9)*(fahr - 32.0)
# επιστροφή
jr     $ra
```


Πολλαπλασιασμός πινάκων

- Πολλαπλασιασμός πινάκων **X = X + Y * Z**
  - X, Y, Z : πίνακες διαστάσεων 32 x 32 με FP

```
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 πινάκων στους $a0, $a1, $a2
- Μεταβλητές i, j, k στους $s0, $s1, $s2

Κώδικας MIPS

- Χρήση των ψευδοεντολών της συμβολικής γλώσσας :
  - **`li`** : φορτώνει μια σταθερά σε έναν καταχωρητή)
  - **`l.d`** και **`s.d`** : ζεύγος εντολών μεταφοράς δεδομένων, lwc1 ή swc1, σε ένα ζεύγος καταχωρητών ΚΥ

```
# Αρχικοποίηση των βρόχων
# Η τιμή τερματισμού του βρόχου σε έναν προσωρινό καταχωρητή
# Αρχικές τιμές στις μεταβλητές των τριών βρόχων for:
mm:...
     li $t1, 32  # $t1 = 32 (τέλος μεγέθους γραμμής)
     li $s0, 0   # i = 0· αρχικές τιμές πρώτου βρόχου for
L1:  li $s1, 0   # j = 0· αρχικές τιμές δεύτερου βρόχου for
L2:  li $s2, 0   # k = 0· αρχικές τιμές τρίτου βρόχου for
```


Κώδικας MIPS

```
# Υπολογισμός της διεύθυνση του x[i][j] και φόρτωση στον $f4
sll  $t2, $s0, 5     # $t2 = i * 2^5 (μέγεθος γραμμής του x)
addu $t2, $t2, $s1   # $t2 = i * μέγεθος(γραμμής) + j
sll  $t2, $t2, 3     # $t2 = σχετική διεύθυνση byte του [i][j]
addu $t2, $a0, $t2   # $t2 = διεύθυνση byte του x[i][j]
l.d  $f4, 0($t2)     # $f4 = 8 byte του x[i][j]

# Υπολογισμός της διεύθυνση του z[k][j] και φόρτωση στον $f16
L3:
sll  $t0, $s2, 5     # $t0 = k * 2^5 (μέγεθος γραμμής του z)
addu $t0, $t0, $s1   # $t0 = k * μέγεθος(γραμμής) + j
sll  $t0, $t0, 3     # $t0 = σχετική διεύθυνση byte του [k][j]
addu $t0, $a2, $t0   # $t0 = διεύθυνση byte του z[k][j]
l.d  $f16, 0($t0)    # $f16 = 8 byte του z[k][j]
```

Κώδικας MIPS

```
# Υπολογισμός της διεύθυνση του y[i][k] και φόρτωση στον $f18
sll  $t2, $s0, 5      # $t0 = i * 25 (μέγεθος γραμμής του y)
addu $t0, $t0, $s2    # $t0 = i * μέγεθος(γραμμής) + k
sll  $t0, $t0, 3      # $t0 = σχετική διεύθυνση byte του [i][k]
addu $t0, $a1, $t0    # $t0 = διεύθυνση byte του y[i][k]
l.d  $f18, 0($t0)     # $f18 = 8 byte του y[i][k]

# Πολλαπλασιασμός των y και z και το άθροισμα στον $f4
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]
```


Κώδικας MIPS

```
# Αύξηση του k και έλεγχος για το τέλος του εσωτερικού βρόχου.
# Στο τέλος του βρόχου, αποθηκεύουμε το άθροισμα στο x[i][j]
addiu $s2, $s2, 1      # $k k + 1
bne   $s2, $t1, L3     # αν (k != 32) μετάβαση στην L3
s.d   $f4, 0($t2)      # x[i][j] = $f4

# Αύξηση των j και I του μεσαίου και του εξωτερικού βρόχου,
# και έλεγχος για το τέλος των βρόχων.
addiu $s1, $s1, 1      # j = j + 1
bne   $s1, $t1, L2     # αν (j != 32) μετάβαση στην L2
addiu $s0, $s0, 1      # i = i + 1
bne   $s0, $t1, L1     # αν (i != 32) μετάβαση στην L1
...
```