ΗΥ 134
Εισαγωγή στην Οργάνωση και στον Σχεδιασμό Υπολογιστών Ι

Διάλεξη 13

Διαίρεση
Προγράμματα Αριθμητικής Κινητής Υποδιαστολής

Νίκος Μπέλλας
Τμήμα Μηχανικών Η/Υ, Τηλεπικοινωνιών και Δικτύων

Διάιρεση κινητής υποδιαστολής

Παράδειγμα X = A/B,

A = 7,8125, B = 2,5

X = ?

1. Μετατροπή σε κανονικοποιημένη μορφή

$A = 7,8125 = (111,1101)_2 = 1,111101 \times 2^2$

$B = 2,5 = (10,1)_2 = 1,01 \times 2^1$

2. Αφαίρεση εκθετών

Χωρίς πόλωση: 2-1 = 1

Με πόλωση : 1 + 127 = 128

Διάιρεση κινητής υποδιαστολής

2. Διαίρεση σημαντικών

   1,111101 / 1,01

   Απαλοιφή της υποδιαστολής στον διαιρέτη και τον διαιρετέο με εισαγωγή επιπλέον 0 ώστε και οι δύο να έχουν τον ίδιο αριθμό ψηφίων

   1111101 / 1010000 = 1,1001

3. Κανονικοποίηση αποτελέσματος και έλεγχος overflow/underflow

   $1{,}1001_2 \times 2^1$ (καμία αλλαγή) γιατί -126 <= 1 <= 127

4. Στρογγυλοποίηση και (αν ξαναχρειαστεί) κανονικοποίηση

   $1{,}1001_2 \times 2^1$ (καμία αλλαγή)

5. Προσδιορισμός προσήμου

   $+ 1{,}1001_2 \times 2^1 = 3{,}125$

Εντολές κινητής υποδιαστολής στο MIPS

- Οι εντολές κινητής υποδιαστολής λειτουργούν μόνο σε καταχωρητές κινητής υποδιαστολής
  - Το προγράμματα γενικά δεν κάνουν ακέραιες πράξεις σε δεδομένα κινητής υποδιαστολής και αντίστροφα.
- Εντολές φόρτωσης και αποθήκευσης αριθμών κινητής υποδιαστολής

Απλή ακρίβεια

```
lwc1  $f1, 54($s2)      #   $f1 = Memory[$s2+54]
swc1 $f1, 58($s4)       # Memory[$s4+58] = $f1
```

Διπλή ακρίβεια

```
ldc1  $f2, 54($s2)      #   ($f2, $f3) = Memory[$s2+54]
sdc1  $f24, 58($s4)     # Memory[$s4+58] = ($f24, $f25)
```

Εντολές κινητής υποδιαστολής στο MIPS

- Αριθμητική απλής ακρίβειας
  - `add.s, sub.s, mul.s, div.s`
    - `add.s $f0, $f1, $f6`
- Αριθμητική διπλής ακρίβειας
  - `add.d, sub.d, mul.d, div.d`
    - `mul.d $f4, $f4, $f6`
- Σύγκριση απλής & διπλής ακρίβειας
  - `c.`*xx*`.s, c.`*xx*`.d` (*xx* είναι `eq, lt, le`, …)
  - Θέτει ή μηδενίζει το το bit κωδικού συνθήκης FP
    - `c.lt.s $f3, $f4`
- Διακλάδωση ανάλογα με τον κωδικό συνθήκης FP
  - `bc1t, bc1f`
    - `bc1t TargetLabel`
- Μετατροπή μεταξύ κινητής υποδιαστολής και ακεραίων
  - `cvt.w.s  # από κιν. υποδιαστολής απλής ακρίβειας σε ακέραιο`
    - `cvt.w.s $f0, $f1`

Παράδειγμα: Πολλαπλασιασμός πινάκων

- X = Y × Z
  - X, Y, Z: Πίνακες 32 × 32 matrices, με στοιχεία 64-bit διπλής ακρίβειας
- Κώδικας C :

```
void matmul (double x[][],
             double y[][], double z[][]) {
  int i, j, k;
  for (i = 0; i < 32; i++)
    for (j = 0; j < 32; j++) {
      x[i][j] = 0.0;
      for (k = 0; k < 32; k++)
        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: Πολλαπλασιασμός πινάκων

```
    void matmul (double x[][],
                 double y[][], double z[][])
{
  int i, j, k;
  for (i = 0; i < 32; i++)
    for (j = 0; j < 32; j++) {
        x[i][j] = 0.0;
      for (k = 0; k < 32; k++)
        x[i][j] += y[i][k] * z[k][j];
      }
}
```

```
    li   $t1, 32       # $t1=32 (μέγ
    li   $s0, 0        # i = 0; αρχ
L1: li   $s1, 0        # j = 0; επανεκκίνηση 2ου for
L2: li   $s2, 0        # k = 0; επανεκκίνηση 3ου for
    sll  $t2, $s0, 5   # $t2 = i * 32 (μέγεθος γραμμής του X)
    addu $t2, $t2, $s1 # $t2 = i * μέγεθος(γραμμής) + j
    sll  $t2, $t2, 3   # $t2 = byte offset του [i][j]
    addu $t2, $a0, $t2 # $t2 = byte address του x[i][j]
    li   $s5, 0
    mtc1  $s5, $f4     # $f4 = $s0
    cvt.s.w  $f4, $f4  # Convert $f4 to 0.0
L3: sll  $t0, $s2, 5   # $t0 = k * 32 (μέγεθος γραμμής του z)
    addu $t0, $t0, $s1 # $t0 = k * μέγεθος(γραμμής) + j
    sll  $t0, $t0, 3   # $t0 = byte offset του [k][j]
```

Παράδειγμα: Πολλαπλασιασμός πινάκων

```
    void matmul (double x[][],
                 double y[][], double z[][])
{
  int i, j, k;
  for (i = 0; i < 32; i++)
    for (j = 0; j < 32; j++) {
        x[i][j] = 0.0;
      for (k = 0; k < 32; k++)
        x[i][j] += y[i][k] * z[k][j];
```

```
addu $t0, $a2, $t0      # $t0 = byte address του z[k][j]
l.d  $f16, 0($t0)       # $f16 = 8 bytes του z[k][j]
sll  $t0, $s0, 5        # $t0=i*32 (μέγεθος γραμμής του y)
addu  $t0, $t0, $s2     # $t0 = i*μέγεθος(γραμμής) + k
sll   $t0, $t0, 3       # $t0 = byte offset του [i][k]
addu  $t0, $a1, $t0     # $t0 = byte address του y[i][k]
l.d   $f18, 0($t0)      # $f18 = 8 bytes του 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++
blt   $s2, $t1, L3      # if (k < 32) go to L3
s.d   $f4, 0($t2)       # x[i][j] = $f4
addiu $s1, $s1, 1       # j++
blt   $s1, $t1, L2      # if (j < 32) go to L2
addiu $s0, $s0, 1       # i++
blt   $s0, $t1, L1      # if (i < 32) go to L1
```