BIỂU DIỄN SỐ CHẤM ĐỘNG

Floating Point Number

Biểu diễn số thực

- ***Khái niệm dấu chấm động:***
  - Số thực được biểu diễn dưới dạng $\pm$ M $\times 10^{\pm K}$
- Trong đó:
  - M là phần trị (0.1 <= M < 1)
  - K là bậc (K nguyên)
- Ví dụ: 9358,214 = 0.9358214 x 10^4

Biểu diễn số thực

- ***Phương pháp biểu diễn nhị phân:*** Đối với phần lẻ của số thập phân, số lẻ được nhân với 2. Phần nguyên của kết quả sẽ là bit nhị phân, phần lẻ của kết quả lại tiếp tục nhân 2 cho đến khi phần lẻ của kết quả bằng 0.
- ***Ví dụ áp dụng:***
  - Chuyển số 0.625_{10} sang hệ nhị phân
  - Chuyển số 9.625_{10} sang hệ nhị phân

Biểu diễn số thực

- Cách chuyển số 0.625_{10} sang hệ nhị phân

Phép tính	Kết quả	Bit lấy được	Phần dư
0.625 x 2	1.25	1	0.25
0.25 x 2	0.5	0	0.5
0.5 x 2	1.0	1	0

- Cách chuyển 9.625_{10} sang hệ nhị phân
  - Phần nguyên 9 đổi sang hệ nhị phân là 1001
  - Phần lẻ 0.625 đổi sang hệ nhị phân là 0.101
  - Vậy số $9.625_{10}=1001.101_2$

Đặt vấn đề

- Biểu diễn số 123.375_{10} sang hệ nhị phân?
- Ý tưởng đơn giản: Biểu diễn phần nguyên và phần thập phân riêng lẻ
  - Với phần nguyên: Dùng 8 bit ($[0_{10}, 255_{10}]$)

    $123_{10} = 64 + 32 + 16 + 8 + 2 + 1 = 0111\ 1011_2$
  - Với phần thập phân: Tương tự dùng 8 bit

    $0.375 = 0.25 + 0.125 = 2^{-2} + 2^{-3} = 0110\ 0000_2$

→ $123.375_{10} = 0111\ 1011.0110\ 0000_2$

- Tổng quát công thức khai triển của số thập phân hệ nhị phân:

$$x_{n-1}x_{n-2}...x_0.x_{-1}x_{-2}...x_{-m} = x_{n-1}.2^{n-1} + x_{n-2}.2^{n-2}... + x_0.2^0 + x_{-1}.2^{-1} + x_{-2}.2^{-2} + ... + x_{-m}2^{-m}$$

Phần nguyên | Phần thập phân | Phần nguyên | Phần thập phân

Đặt vấn đề

- Tuy nhiên...với 8 bit:
  - Phần nguyên lớn nhất có thể biểu diễn: 255
  - Phần thập phân nhỏ nhất có thể biểu diễn: $2^{-8} \sim 10^{-3} = 0.001$

→ Biểu diễn số nhỏ như 0.0001 (10^{-4}) hay 0.000001 (10^{-5})?

→ Một giải pháp: Tăng số bit phần thập phân

  - Với 16 bit cho phần thập phân: min = $2^{-16} \sim 10^{-5}$
  - Có vẻ không hiệu quả...Cách tốt hơn ?

→ Floating Point Number (Số thực dấu chấm động)

Floating Point Number?

- Giả sử ta có số (ở dạng nhị phân)

  $X = 0.\underbrace{00000000000000}_{\text{14 số 0}}11_2$

→ $X = 0.11_2 * (2^{-14})_2$

→ Thay vì dùng 16 bit để lưu trữ phần thập phân, ta có thể chỉ cần 6 bit:

  X = 0.11 1110

→ Cách làm: Di chuyển vị trí dấu chấm sang phải 14 vị trí, dùng 4 bit để lưu trữ số 14 này

→ Đây là ý tưởng **cơ bản** của số thực dấu chấm động (floating point number)

Chuẩn hóa số thập phân

- Trước khi các số được biểu diễn dưới dạng số chấm động, chúng cần được chuẩn hóa về dạng:

  $\pm 1.M * 2^E$

  - *M: Phần thập phân không dấu (định trị)*
  - *E: Phần số mũ (Exponent)*
- Ví dụ:
  - $+0.09375_{10} = +0.00011_2 = +1.1 * 2^{-4}$
  - $-5.25_{10} = -101.01_2 = -1.0101 * 2^2$

Biểu diễn số chấm động

- Có nhiều chuẩn nhưng hiện nay chuẩn IEEE 754 được dùng nhiều nhất để lưu trữ số thập phân theo dấu chấm động trong máy tính, gồm 2 dạng:

  (slide sau)

Biểu diễn số chấm động

- *Số chấm động chính xác đơn (32 bits):*

- *Số chấm động chính xác kép (64 bits):*

- **Sign:** Bit dấu (1: Số âm, 0: Số dương)
- **Exponent:** Số mũ (Biểu diễn dưới dạng số quá K (Biased)) với
  - *Chính xác đơn:* K = 127 ($2^{n-1} - 1 = 2^{8-1} - 1$) với n là số bit lưu trữ Exponent
  - *Chính xác kép:* K = 1023 ($2^{n-1} - 1 = 2^{11-1} - 1$)
- **Mantissa (Fraction):** Phần định trị (phần lẻ sau dấu chấm)

Ví dụ

Biểu diễn số thực sau theo dạng số chấm động chính xác đơn (32 bit): **X = -5.25**

- **Bước 1:** Đổi X sang hệ nhị phân

  $X = -5.25_{10} = -101.01_2$

- **Bước 2:** Chuẩn hóa theo dạng $\pm 1.M * 2^E$

  $X = -5.25 = -101.01 = -1.0101 * 2^2$

- **Bước 3:** Biểu diễn Floating Point
  - Số âm: bit dấu Sign = 1
  - Số mũ E = 2 → Phần mũ Exponent với số thừa K=127 được biểu diễn:

    → Exponent (bias 127) = E + 127 = 2 + 127 = $129_{10} = 1000\ 0001_2$
  - Phần định trị = 0101 0000 0000 0000 0000 000 (Thêm 19 số 0 cho đủ 23 bit)

→ Kết quả nhận được: 1 1000 0001 0101 0000 0000 0000 0000 000

Ví dụ

Biểu diễn số floating point theo dạng số chấm động chính xác đơn (32 bit) sau thành số thập phân: **X = 11000001010101100000000000000000**

- **Bước 1:** Xác định bit dấu **S**

  S = 1 → Số âm (-1^1) (S là bit trái cùng).

- **Bước 2:** Xác định phần mũ **E**

  e (bias 127) = 1000 0010_2 = 130_{10} -> E = 130 - 127 = 3 (e là 8 bits tiếp theo).

- **Bước 3:** Xác định phần định trị **M**

  m = 101011 -> M = 1.101011 (m là 23 bits còn lại).

  (ở đây không cần quan tâm đến các bit 0 ở cuối vì khi ghép M = 1.m thì các số 0 này không cần viết vào)

→ Kết quả nhận được: X = - 1.101011 * 2^3 = - 1101.011 = - 13.375