



Bài Thu Hoạch:

Floating Point

Tìm hiểu về dấu chấm động

Nhóm M3T

Lê Tiến Tài

letientai299@yahoo.com

Nguyễn Hữu Thiện

mario.alpha@yahoo.com.vn

Nguyễn Trương Trung Tín

ka_lick@yahoo.com

V Đức Minh

vdminh24@yahoo.com

I. Giới thiệu về dấu chấm động

Dấu chấm động là hình thức biểu diễn số dùng trong khoa học vì mức độ thực tế và chính xác cao của nó. Trong tin học, dấu chấm động được dùng để chỉ một hệ thống biểu diễn số mà trong đó sử dụng một chuỗi chữ số (hay bit) để biểu diễn một số hữu tỉ.

Trước đây do chưa có bất kì 1 chuẩn chung nào cho việc biểu diễn số dấu chấm động nên xảy ra tính không tương thích giữa các hệ máy khác nhau, điều này chỉ được giải quyết triệt để khi IEEE đưa ra chuẩn dấu chấm động IEEE 754 vào năm 1985. Theo thời gian thì chuẩn này càng được các nhà sản xuất máy tính cũng như các thiết bị phần cứng tương thích chấp nhận rộng rãi.

Thuật ngữ dấu chấm động xuất phát từ chỗ hệ thống dấu phẩy động có dấu phẩy cơ sở (tức là dấu phẩy thập phân trong trường hợp dùng hệ thập phân thường ngày hoặc là dấu phẩy nhị phân trong trường hợp dùng bên trong máy tính) không cố định mà có thể thay đổi vị trí của nó bất kỳ đâu trong các chữ số có nghĩa của số cần được biểu diễn. Vị trí này được mô tả một cách độc lập trong biểu diễn cụ thể của từng số. Đã có nhiều hệ thống dấu chấm động khác nhau được dùng trong máy tính; tuy nhiên, vào khoảng hai mươi năm trở lại đây thì hầu hết các máy tính đều dùng cách biểu diễn tuân thủ theo chuẩn IEEE 754.

Cách biểu diễn

Cách biểu diễn số dấu phẩy động tương tự với cách dùng trong ký hiệu khoa học. Một tả luận lý thì một số dấu phẩy động bao gồm:

Một chuỗi chữ số có dấu với chiều dài cho trước và có cơ sở cho trước. Chuỗi này được gọi là phần định trị. Dấu phẩy cơ sở không được thể hiện tường minh ở phần này, nhưng được qui ước ngầm là luôn luôn nằm tại 1 vị trí cụ thể trong phần định trị - mà thường là ngay sau hoặc ngay trước chữ số có nghĩa lớn nhất. Bài viết này nếu không nói rõ sẽ tuân theo qui ước là dấu phẩy cơ sở luôn ở ngay sau chữ số có nghĩa lớn nhất (tức là chữ số đầu tiên tính từ bên trái qua). Độ dài của phần định trị xác định độ chính xác mà các con số có thể được biểu diễn.

Một số mũ là số nguyên có dấu, nhằm mô tả phần lấy tỉ lệ tức cho phép người đọc xác định được giá trị thực của số từ phần định trị.

Ví dụ: số 152853.5047 ở cơ sở 10 thì được viết với dạng dấu phẩy động như sau: Phần định trị là 1528535047 (với qui ước là vị trí của dấu phẩy cơ sở nằm ngay sau chữ số có nghĩa lớn nhất, tức là chữ số 1). Số mũ là 5. Số đó sẽ là 1.528535047×10^5 .

Ưu điểm của cách biểu diễn dấu phẩy động là nó cho phép biểu diễn một tầm giá trị rộng hơn nhiều so với cách biểu diễn dấu phẩy tĩnh. Lấy ví dụ, nếu cách biểu diễn dấu phẩy tĩnh có bảy chữ số thập phân với quy ước dấu phẩy thập phân luôn nằm cố định ở chữ số thứ năm, thì cách biểu diễn dấu phẩy tĩnh có thể mô tả các số như 12345.67, 8765.43, 123.00 (tức 00123.00) và vân vân. Trong khi đó cách biểu diễn dấu phẩy động (chẳng hạn như định dạng thập phân32 của IEEE 754) với bảy chữ số thập phân ngoài việc mô tả được các số nói trên còn mô tả được nhiều số khác mà dấu phẩy tĩnh không mô tả được như 1.234567, 123456.7, 0.00001234567, 1234567000000000, và nhiều nữa. Tất nhiên, định dạng theo kiểu dấu phẩy động cần thêm bộ nhớ hơn so với dấu phẩy tĩnh (vì cần có thêm bộ nhớ để

mô tả vị trí của dấu phẩy cơ số), nhưng ta có thể nói: với cùng một không gian bộ nhớ, cách biểu diễn dấu phẩy động đạt được tầm mô tả rộng hơn.

II. Chuẩn IEEE 754

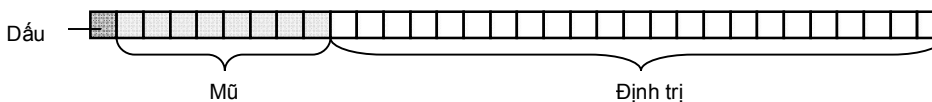
Số IEEE 754 dấu chấm động gồm 3 thành phần chính: Dấu (Sign), Mũ (Exponent), Phần định trị (Mantissa). Số mũ có cơ số là 2 sẽ được ẩn và không cần lưu trữ ($M \times 2^e$).

Chuẩn IEEE 754 đưa ra nhiều định dạng rất gần nhau, chỉ khác nhau ở một ít chi tiết. Năm trong số những định dạng này được gọi là định dạng cơ bản, và hai trong chúng đặc biệt được dùng rộng rãi trong cả phần cứng máy tính và ngôn ngữ lập trình:

Độ chính xác đơn, được gọi bằng tên là "float" trong họ ngôn ngữ lập trình C và tên là "real" hay "real*4" trong ngôn ngữ Fortran. Đây là định dạng nhị phân chiếm 32 bit (4 byte) và phần định trị của nó có độ chính xác 24 bit (tương đương với khoảng 7 chữ số thập phân).

Độ chính xác kép, được gọi bằng tên là "double" trong họ ngôn ngữ lập trình C và tên là "double precision" hay "real*8" trong ngôn ngữ Fortran. Đây là định dạng nhị phân chiếm 64 bit (8 byte) và phần định trị của nó có độ chính xác 53 bit (tương đương với khoảng 16 chữ số thập phân).

	Dấu (Sign)	Mũ (Exponent)	Định Trị (Mantissa)	Bias
Độ chính xác đơn (Single Precision)	1 Bit [Bit thứ 31]	8 Bit [Từ Bit 30 đến Bit 23]	23 Bit [Từ Bit 22 đến Bit 0]	127
Độ chính xác kép (Double Precision)	1 Bit [Bit thứ 63]	11 Bit [Từ Bit 62 đến Bit 52]	52 Bit [Từ Bit 51 đến Bit 0]	1023



Hình minh họa số Float 32 bit

- **Bit Dấu:** Bit 0 biểu diễn số dương và ngược lại Bit 1 để biểu diễn số âm.
- **Số Mũ:** Do không thể lưu trữ số âm của lũy thừa nên người ta sử dụng một giá trị đặc biệt gọi là “thể hiệu dịch” (Bias) để cộng với lũy thừa thật sự tạo nên biểu diễn cuối cùng.
 Nếu lũy thừa là 0 thì phần lũy thừa chứa 01111111b, biểu diễn 127d.
 Nếu lũy thừa là 12 thì phần lũy thừa chứa 10001011b, biểu diễn 139d.
 Nếu lũy thừa là -3 thì phần lũy thừa chứa 01111100b, biểu diễn 124d.
- **Phần định trị:** Có dạng là $a.f$. Với a là bit tuyệt đối, f là những bit thập phân. Để làm rõ cho bit tuyệt đối a ta xét ví dụ sau: 9 có thể được biểu diễn bằng 1 trong những cách sau:
 - 9.000×10^0
 - 0.009×10^3
 - 9000×10^{-3}
 Nhằm biểu diễn được nhiều nhất các số ở phần định trị, người ta thường sử dụng định dạng normalized để lưu. Định dạng này quy định dấu phẩy được đặt sau số

khác không (non-zero) đầu tiên. Như trong ví dụ trên, cách biểu diễn số 9 thứ 1 đã minh họa cho định dạng này.

Có 1 điều thú vị ở đây! Ở hệ cơ số 2 (base 2) số non-zero duy nhất là số 1 nên ta mặc định bit tuyệt đối a là 1 và không cần lưu.

Các giá trị đặc biệt

- ✓ **Zero:** Zero không thể được biểu diễn trực tiếp vì mặc định bit tuyệt đối có giá trị là 1. Zero được biểu diễn bởi 1 dãy các bit không ở phần mũ và 1 dãy các bit không ở phần định trị. Trong chuẩn IEEE 754 xuất hiện số zero có dấu: “+0” và “-0”. Hai giá trị này được xem là bằng nhau về mặt giá trị nhưng một vài phép toán sẽ thông báo kết quả phân biệt giữa +0 và -0. Lấy ví dụ, $a/(-0)$ sẽ cho ra kết quả là vô cực âm còn $a/(+0)$ sẽ cho kết quả là vô cực dương với a là một số dương.
- ✓ **Infinity:** là các giá trị được biểu diễn với số mũ đều là 1 và phần định trị đều là 0. Bit dấu để phân biệt +infinity và -infinity.
- ✓ **Not A Number:** giá trị NaN được dùng để biểu diễn một giá trị không phải là số thực như kết quả của các phép toán $0/0$, $\infty \times 0$, or $\text{sqrt}(-1)$. NaN được biểu diễn với các bit mũ đều bằng 1 và phần định trị biểu diễn 1 số thập phân khác không. Có 2 kiểu NaN: signaling và quiet. qNaN dùng cho những phép toán indeterminate, còn sNaN dùng cho các phép toán invalid.

ví dụ:

```

0 00000000 000000000000000000000000 = 0
1 00000000 000000000000000000000000 = -0

0 11111111 000000000000000000000000 = Infinity
1 11111111 000000000000000000000000 = -Infinity

0 11111111 000001000000000000000000 = NaN
1 11111111 001000100010010101010101 = NaN

0 10000000 000000000000000000000000 = +1 x 2(128-127) x 1.0 = 2
0 10000001 101000000000000000000000 = +1 x 2(129-127) x 1.101 = 6.5
1 10000001 101000000000000000000000 = -1 x 2(129-127) x 1.101 = -6.5

0 00000001 000000000000000000000000 = +1 x 2(1-127) x 1.0 = 2(-126)
0 00000000 100000000000000000000000 = +1 x 2(-126) x 0.1 = 2(-127)
0 00000000 000000000000000000000001 = +1 x 2(-126) x
                                          0.000000000000000000000001 =
                                          2(-149) (Smallest positive value)

```

III. Các phép toán trên dấu chấm động

1. Phép cộng và trừ

Điều chỉnh sao cho số mũ bằng nhau.
Sau đó cộng hoặc trừ phần định trị (Mantissa).

Ví dụ: Trong hệ cơ số 10, ta thực hiện phép cộng $123456.7 + 101.7654$

Như sau:

đây là định dạng normalized:

$e=5; s=1.234567$ (123456.7)

+ $e=2; s=1.017654$ (101.7654)

Thực hiện phép cộng:

$e=5; s=1.234567$

+ $e=5; s=0.001017654$ (sau khi đã dịch chuyển)

 $e=5; s=1.235584654$ (kết quả đúng: 123558.4654)

và sau đó kết quả đúng sẽ được làm tròn là 123558.5 và trả về biểu diễn IEEE
 $e=5; s=1.235585$.

2. Phép nhân và chia

Phép nhân: cộng số mũ và nhân mantissa.
Phép chia : trừ số mũ và chia mantissa.

Ví dụ:

$e=3; s=4.734612$

× $e=5; s=5.417242$

 $e=8; s=25.648538980104$ (kết quả đúng)

$e=8; s=25.64854$ (sau khi làm tròn)

$e=9; s=2.564854$ (sau khi định dạng IEEE)

3. Phép so sánh

Trong mọi phép biểu diễn số học đều có sai số, điều này làm cho phép so sánh trở nên khập khiễng.

Ví dụ: $(a+b)*c$ sẽ khác với $a*c+b*c$ mặc dù về mặt toán học ta nói chúng như nhau.

$$1234.567 \times 3.333333 = 4115.223$$

$$1.234567 \times 3.333333 = 4.115223$$

$$4115.223 + 4.115223 = 4119.338$$

Nhưng

$$1234.567 + 1.234567 = 1235.802$$

$$1235.802 \times 3.333333 = 4119.340$$

Do vậy nếu kiểm tra trực tiếp mối tương quan của 2 số là một điều không nên.

Trong trường này chúng ta đề ra một giá trị esp (một giá trị cực nhỏ) hay còn gọi là sai số.

Ví dụ: thay vì so sánh $a=b$ ta sẽ làm như sau $\text{abs}(a-b) \leq \text{Esp}$ thì xem như $a=b$.