ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Đinh Đức Tài

23TNT1 - 23122013

BÁO CÁO BTVN1

Đề tài: Các cơ chế biểu diễn số thực trên máy tính

Môn: Hệ thống máy tính

GV: Vũ Thị Mỹ Hằng

TP. Hồ Chí Minh - 2025

MỤC LỤC

Chương 1

Giới thiệu

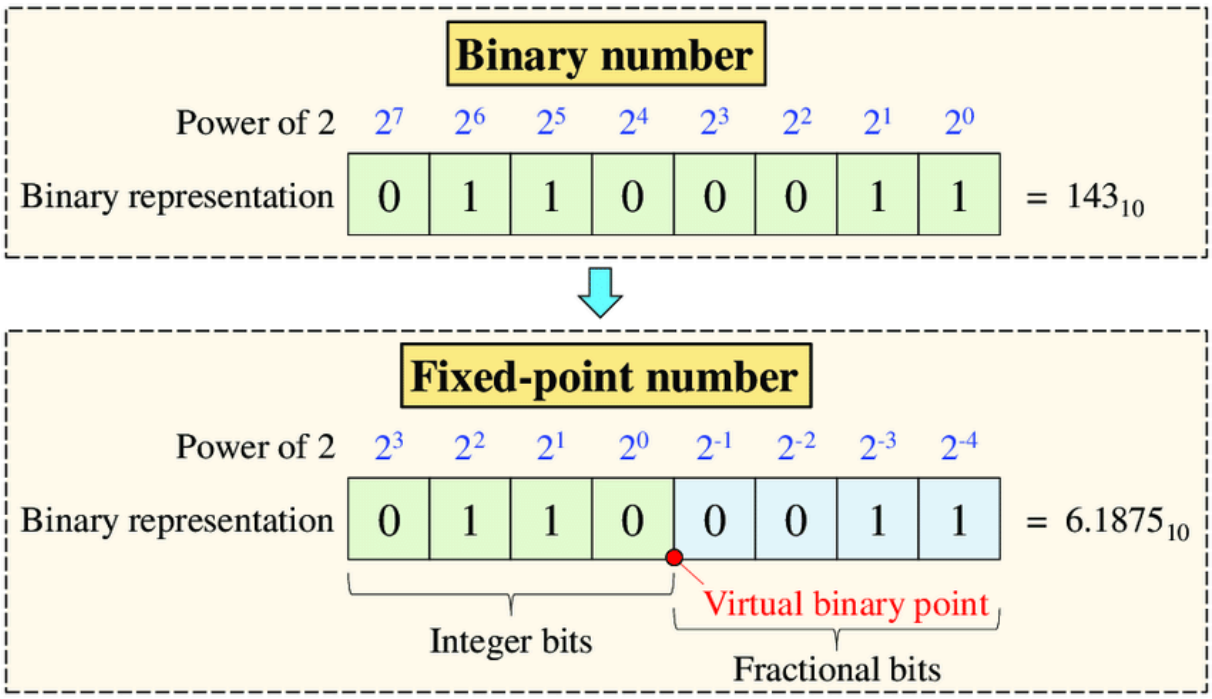
1.1. Thách thức trong việc biểu diễn số thực bằng kỹ thuật số

Số thực, theo định nghĩa toán học, tạo thành một tập hợp liên tục với độ chính xác vô hạn. Ngược lại, máy tính hoạt động trên các đơn vị thông tin rời rạc (bit) trong bộ nhớ hữu hạn. Sự khác biệt cơ bản này đòi hỏi các kỹ thuật xấp xỉ để biểu diễn số thực bằng kỹ thuật số. [1] Thách thức cốt lõi nằm ở việc ánh xạ một miền vô hạn, liên tục sang một miền hữu hạn, rời rạc. Sự ánh xạ này chắc chắn liên quan đến một số hình thức xấp xỉ hoặc cắt bớt.

Hai phương pháp chính được sử dụng để biểu diễn số thực trong máy tính là biểu diễn dấu chấm tĩnh (fixed-point) và biểu diễn dấu chấm động (floating-point). [2] Mỗi phương pháp đưa ra các đánh đổi khác nhau về phạm vi, độ chính xác và độ phức tạp. Việc lựa chọn phương pháp biểu diễn phụ thuộc nhiều vào các yêu cầu cụ thể của ứng dụng, cân bằng giữa nhu cầu về độ chính xác với các ràng buộc về tài nguyên tính toán.

1.2. Biểu diễn dấu chấm tĩnh

Biểu diễn dấu chấm tĩnh sử dụng một số lượng bit cố định cho phần nguyên và một số lượng bit cố định cho phần thập phân. [3] Vị trí của dấu chấm nhị phân (tương tự như dấu chấm thập phân) được xác định ngầm và không thay đổi. Bằng cách cố định dấu chấm nhị phân, việc biểu diễn trở nên đơn giản hơn để thực hiện ở cấp độ phần cứng, vì nó chủ yếu dựa vào số học số nguyên. Với dấu chấm nhị phân cố định, các phép toán như cộng và trừ có thể được thực hiện bằng cách sử dụng các đơn vị số học số nguyên tiêu chuẩn, có khả năng dẫn đến việc thực hiện nhanh hơn.



Hình 1.1 — Một ví dụ về biểu diễn số bằng dấu chấm tĩnh

Một số thực được chia thành hai phần: phần nguyên (số nguyên) và phần thập phân. Trong biểu diễn dấu chấm tĩnh, một số lượng bit được xác định trước được phân bổ cho mỗi phần. Vì số thực có thể là số âm, biểu diễn dấu chấm tĩnh cũng phải đáp ứng dấu. Các phương pháp phổ biến bao gồm:

- **Dấu-Độ lớn (Sign-Magnitude):** Bit quan trọng nhất (MSB) biểu thị dấu (0 cho dương, 1 cho âm) và các bit còn lại biểu thị độ lớn.
- **Bù 1 (One's Complement):** Số âm được hình thành bằng cách đảo ngược tất cả các bit của số dương tương ứng.
- **Bù 2 (Two's Complement):** Số âm được hình thành bằng cách đảo ngược tất cả các bit của số dương tương ứng và cộng thêm 1. Đây là phương pháp được sử dụng rộng rãi nhất để biểu diễn số nguyên có dấu và cũng có thể áp dụng cho dấu chấm tĩnh . Ví dụ cung cấp các ví dụ sử dụng bù 2 cho phần định trị (mantissa) trong ngữ cảnh dấu chấm động, minh họa việc sử dụng nó cho số nhị phân có dấu. và đề cập rõ ràng đến các phương pháp biểu diễn có dấu này. Việc lựa chọn biểu diễn có dấu ảnh hưởng đến độ phức tạp của các phép toán số học, đặc biệt là phép trừ và phép so sánh. Bù 2 đơn giản hóa các phép toán này. Bù 2 có ưu điểm là biểu diễn duy nhất cho số không và đơn giản hóa các mạch cộng và trừ, khiến nó trở thành lựa chọn ưu tiên trong các hệ thống hiện đại.

Ưu điểm:

- **Đơn giản:** Số học dấu chấm tĩnh thường đơn giản hơn để thực hiện trong phần cứng so với dấu chấm động.
- **Tốc độ:** Các phép toán có thể nhanh hơn vì chúng thường tận dụng các đơn vị số học số nguyên.
- **Tính dự đoán:** Độ chính xác là hằng số, giúp phân tích lỗi dễ dàng hơn. Bản chất cố định của dấu chấm nhị phân dẫn đến việc chia tỷ lệ và độ chính xác có thể dự đoán được, điều này có thể rất quan trọng trong một số ứng dụng nhất định.

Nhược điểm:

- **Phạm vi giới hạn:** Phạm vi của các số có thể biểu diễn bị giới hạn bởi số lượng bit cố định được phân bổ cho phần nguyên và phần thập phân .
- **Độ chính xác giới hạn:** Độ chính xác cũng cố định và có thể không đủ cho các ứng dụng đòi hỏi dải động rộng hoặc độ chi tiết rất cao. Sự đánh đổi giữa kích thước của phần nguyên và phần thập phân phải được xem xét cẩn thận dựa trên phạm vi dự kiến và độ chính xác cần thiết của các số.

Trường hợp sử dụng: Biểu diễn dấu chấm tĩnh thường được sử dụng trong các ứng dụng mà hiệu quả tính toán và tính dự đoán là rất quan trọng, chẳng hạn như:

- Hệ thống nhúng.
- Xử lý tín hiệu số (DSP).
- Hệ thống điều khiển.

Ví dụ về biểu diễn nhị phân dấu chấm tĩnh:

- Một số dấu chấm tĩnh 16 bit với 8 bit cho phần nguyên và 8 bit cho phần thập phân. Điều này cho phép biểu diễn các số từ -128 đến $+127$ với độ chính xác là $1/256$. Việc phân bổ bit trực tiếp xác định phạm vi và độ chính xác. Nhiều bit hơn cho phần thập phân làm tăng độ chính xác nhưng làm giảm phạm vi của phần nguyên, và ngược lại.

1.3. Biểu diễn dấu chấm động

Biểu diễn dấu chấm động giải quyết các hạn chế của dấu chấm tĩnh bằng cách sử dụng một dạng ký hiệu khoa học. [4] Một số được biểu diễn bằng một phần định trị (significand hoặc mantissa) và một số mũ (exponent), số mũ này

xác định vị trí của dấu chấm nhị phân. Nguyên tắc này cho phép biểu diễn một phạm vi số rộng hơn nhiều so với dấu chấm tĩnh, bao gồm cả các giá trị rất nhỏ và rất lớn. Bằng cách chia tỷ lệ phần định trị bằng số mũ, dấu chấm động có thể biểu diễn các số trên nhiều bậc độ lớn mà không làm mất quá nhiều độ chính xác cho các số trong một phạm vi cụ thể.

Mặc dù khái niệm này tương tự như ký hiệu khoa học cơ số 10, máy tính chủ yếu sử dụng cơ số 2 (nhị phân) cho biểu diễn dấu chấm động. Các cơ số khác như 10 và 16 cũng đã được sử dụng trong lịch sử hoặc trong các ngữ cảnh cụ thể. Việc sử dụng cơ số 2 phù hợp với bản chất nhị phân của phần cứng máy tính, cho phép thực hiện hiệu quả các phép toán số học.

Các số dấu chấm động thường được lưu trữ ở dạng chuẩn hóa, trong đó chữ số có nghĩa nhất của phần định trị khác không (thường là 1 trong hệ nhị phân) [5]. Điều này tối đa hóa số lượng chữ số có nghĩa có thể được biểu diễn cho một số lượng bit nhất định. Chuẩn hóa đảm bảo rằng biểu diễn là duy nhất và các bit có sẵn cho phần định trị được sử dụng hết tiềm năng cho độ chính xác. Bằng cách đảm bảo bit đầu tiên là 1 (và thường được giả định ngầm), chúng ta có được thêm một bit độ chính xác miễn phí.

Một số dấu chấm động thường bao gồm ba phần:

- **Bit dấu (Sign Bit):** Cho biết số là dương (0) hay âm (1).
- **Số mũ (Exponent):** Biểu thị lũy thừa mà cơ số được nâng lên để chia tỷ lệ phần định trị. Số này thường được lưu trữ với một độ lệch (bias) để cho phép cả số mũ dương và âm.
- **Phần định trị (Mantissa/Significand):** Chứa các chữ số có nghĩa của số. Độ chính xác của biểu diễn được xác định bởi số lượng bit được phân bổ cho phần định trị.

Việc phân chia bit giữa các thành phần này quyết định phạm vi (chủ yếu bởi số mũ) và độ chính xác (chủ yếu bởi phần định trị) của các số dấu chấm động có thể được biểu diễn.

Sự đánh đổi giữa phạm vi và độ chính xác: Tăng số lượng bit cho số mũ làm tăng phạm vi của các số có thể biểu diễn, trong khi tăng số lượng bit cho phần định trị làm tăng độ chính xác. Có một sự đánh đổi vốn có giữa hai điều này. Việc thiết kế một định dạng dấu chấm động liên quan đến việc cân bằng phạm vi và độ chính xác mong muốn cho các ứng dụng dự kiến.

Ưu điểm:

- **Phạm vi giá trị rộng:** Dấu chấm động có thể biểu diễn một phạm vi số rộng hơn đáng kể, từ rất nhỏ đến rất lớn, so với dấu chấm tĩnh.
- **Cân bằng tốt giữa phạm vi và độ chính xác:** Nó cung cấp một cách linh hoạt để xử lý các số có độ lớn khác nhau trong khi vẫn duy trì mức độ chính xác hợp lý.

Nhược điểm:

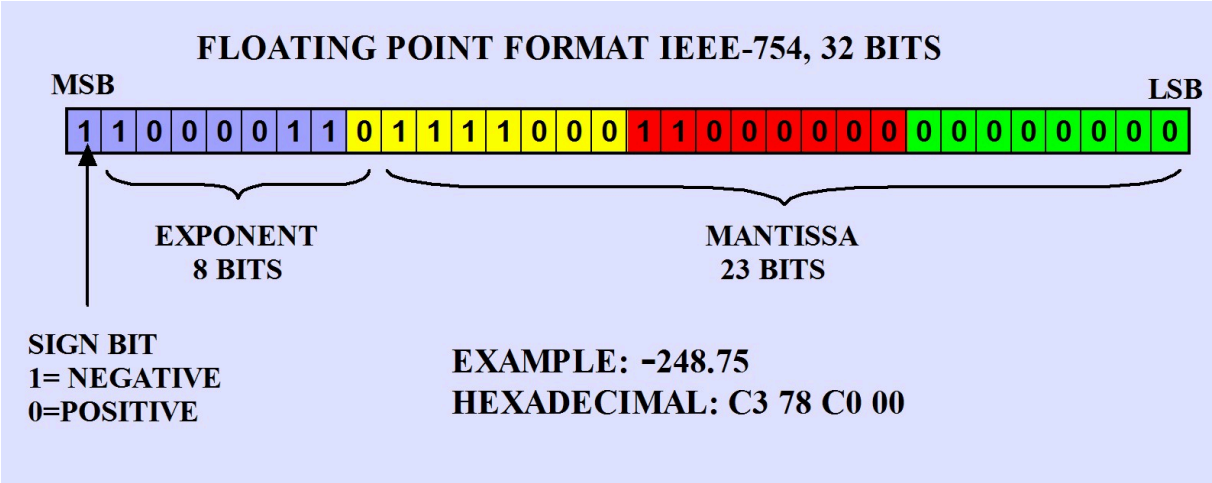
- **Độ phức tạp:** Số học dấu chấm động phức tạp hơn để thực hiện trong phần cứng so với số học dấu chấm tĩnh.
- **Khả năng xảy ra lỗi làm tròn:** Do độ chính xác hữu hạn và cách một số phân số thập phân được biểu diễn trong hệ nhị phân, số học dấu chấm động có thể gây ra lỗi làm tròn. Các xấp xỉ vốn có trong biểu diễn dấu chấm động có thể dẫn đến các lỗi tinh vi nhưng có khả năng nghiêm trọng trong tính toán, đặc biệt là qua một chuỗi các phép toán.

Chương 2

Tiêu chuẩn IEEE 754

2.1. Tiêu chuẩn IEEE 754 cho số học dấu chấm động

Tiêu chuẩn IEEE 754, được thiết lập vào năm 1985, cung cấp một cách thống nhất để biểu diễn số dấu chấm động và thực hiện các phép toán số học trên các hệ thống máy tính khác nhau. [6] Nó giải quyết những mâu thuẫn trong các triển khai trước đó, cải thiện độ tin cậy và khả năng di chuyển. Tiêu chuẩn hóa đảm bảo rằng các số dấu chấm động được diễn giải và xử lý nhất quán trên các nền tảng phần cứng và phần mềm khác nhau, điều này rất quan trọng cho khả năng tương tác.



Hình 2.1 — Tiêu chuẩn IEEE-754, 32 bit

2.1.1. Các định dạng cơ bản

Tiêu chuẩn xác định một số định dạng cơ bản, bao gồm:

- **Độ chính xác đơn (Single-Precision - binary32):** Sử dụng 32 bit.
- **Độ chính xác kép (Double-Precision - binary64):** Sử dụng 64 bit.
- **Độ chính xác nửa (Half-Precision - binary16):** Sử dụng 16 bit.

- **Độ chính xác gấp bốn (Quadruple-Precision - binary128):** Sử dụng 128 bit.
- **Độ chính xác mở rộng (Extended Precision):** Các định dạng có nhiều bit hơn độ chính xác kép, thường là dành riêng cho từng triển khai.

Tiêu chuẩn cung cấp một loạt các định dạng để phù hợp với các yêu cầu khác nhau về độ chính xác và bộ nhớ. Độ chính xác đơn và kép là phổ biến nhất trong tính toán đa năng.

2.1.2. Cấu trúc của định dạng độ chính xác đơn và kép

- **Độ chính xác đơn:** 1 bit dấu, 8 bit số mũ, 23 bit phần định trị.
- **Độ chính xác kép:** 1 bit dấu, 11 bit số mũ, 52 bit phần định trị.

Số lượng bit tăng lên trong độ chính xác kép cho phép phạm vi số mũ rộng hơn và độ chính xác cao hơn đáng kể do phần định trị lớn hơn.

2.1.3. Độ lệch số mũ:

Số mũ được lưu trữ với một độ lệch (offset) để biểu diễn cả số mũ dương và âm dưới dạng số nguyên không dấu. Đối với độ chính xác đơn, độ lệch là 127 và đối với độ chính xác kép, nó là 1023. Việc sử dụng độ lệch giúp đơn giản hóa việc so sánh các số dấu chấm động.

2.1.4. Số chuẩn hóa và số phi chuẩn hóa (Subnormal):

Số chuẩn hóa có một bit '1' ngầm ở đầu phần định trị, cung cấp độ chính xác tối đa. Số phi chuẩn hóa (còn gọi là subnormal) được sử dụng để biểu diễn các số rất nhỏ gần bằng không, trong đó bit đầu tiên là '0' và số mũ ở giá trị tối thiểu. Số phi chuẩn hóa giúp giảm thiểu vấn đề "underflow dần dần", cho phép biểu diễn các số nhỏ hơn số chuẩn hóa nhỏ nhất, mặc dù với độ chính xác giảm.

2.1.5. Các giá trị đặc biệt

Tiêu chuẩn IEEE 754 xác định các mẫu bit đặc biệt để biểu diễn các giá trị như:

- **Số không:** Số không dương (+0) và số không âm (-0).
- **Vô cực:** Vô cực dương ($+\infty$) và vô cực âm ($-\infty$).
- **Không phải là số (NaN - Not-a-Number):** Biểu thị kết quả của các phép toán không hợp lệ (ví dụ: 0/0, ∞/∞).

Các giá trị đặc biệt này cung cấp một cách để xử lý các trường hợp ngoại lệ trong tính toán dấu chấm động theo một tiêu chuẩn hóa.

2.2. Hạn chế và lỗi trong biểu diễn dấu chấm động

Do số lượng bit hữu hạn, máy tính chỉ có thể biểu diễn một tập hợp con hữu hạn của tập hợp vô hạn các số thực. Hầu hết các số thực chỉ có thể được xấp xỉ. Hạn chế này là không thể tránh khỏi và là nguyên nhân gốc rễ của nhiều vấn đề liên quan đến số học dấu chấm động.

2.2.1. Lỗi làm tròn

Xảy ra khi một số thực không thể được biểu diễn chính xác và phải được làm tròn đến số dấu chấm động có thể biểu diễn gần nhất.

- **Nguyên nhân**
  - Biểu diễn nhị phân không kết thúc của số thập phân (ví dụ: 0.1 trong hệ thập phân có biểu diễn nhị phân lặp lại).
  - Độ chính xác hữu hạn của phần định trị.
- **Các kiểu làm tròn (được định nghĩa bởi IEEE 754)**
  - Làm tròn đến số gần nhất (hòa chẵn, hòa xa không).
  - Làm tròn về phía dương vô cực (lên).
  - Làm tròn về phía âm vô cực (xuống).
  - Làm tròn về phía số không (cắt bớt).

Việc lựa chọn kiểu làm tròn có thể ảnh hưởng đến độ chính xác của các phép tính và việc hiểu rõ các kiểu này rất quan trọng đối với phân tích số.

2.2.2. Mất độ chính xác và triệt tiêu thảm khốc

- **Mất độ chính xác:** Xảy ra khi các phép toán số học tạo ra kết quả đòi hỏi nhiều chữ số có nghĩa hơn mức có thể biểu diễn, dẫn đến việc làm tròn hoặc cắt bớt.
- **Triệt tiêu thảm khốc:** Sự mất mát độ chính xác đáng kể có thể xảy ra khi trừ hai số dấu chấm động gần bằng nhau, vì các chữ số có nghĩa hàng đầu bị triệt tiêu, chỉ còn lại các chữ số ít có nghĩa hơn và có khả năng kém chính xác hơn.

Những vấn đề này có thể đặc biệt nghiêm trọng trong các thuật toán lặp hoặc chuỗi các phép tính mà lỗi có thể tích lũy.

2.2.3. Tràn số (Overflow) và thiếu số (Underflow)

- Tràn số: Xảy ra khi kết quả của một phép tính quá lớn để có thể biểu diễn trong phạm vi của định dạng dấu chấm động, thường dẫn đến vô cực.
- Thiếu số: Xảy ra khi kết quả quá nhỏ để có thể biểu diễn dưới dạng một số dấu chấm động chuẩn hóa, thường dẫn đến số không hoặc một số phi chuẩn hóa.

Việc hiểu rõ giới hạn của phạm vi có thể biểu diễn là rất quan trọng để tránh những vấn đề này.

2.2.4. Độ chính xác (Accuracy) so với độ chụm (Precision)

- **Độ chụm:** Đề cập đến số lượng bit được sử dụng để biểu diễn một giá trị, cho biết mức độ chi tiết.
- **Độ chính xác:** Đề cập đến mức độ gần của giá trị được biểu diễn so với giá trị thực.

Các số dấu chấm động có thể có độ chụm cao (nhiều bit trong phần định trị) nhưng vẫn bị sai số do làm tròn. Điều quan trọng là phải phân biệt giữa số lượng chữ số được lưu trữ (độ chụm) và số lượng chữ số đó thực sự chính xác so với giá trị thực (độ chính xác).

2.2.5. Epsilon máy (Machine Epsilon)

Biểu thị số dương nhỏ nhất mà khi cộng với 1 sẽ cho kết quả khác 1 trong định dạng dấu chấm động đã cho . Nó cung cấp thước đo độ chính xác tương đối. Epsilon máy là một khái niệm quan trọng trong phân tích số để hiểu giới hạn của độ chính xác dấu chấm động.

2.2.6. Các chiến lược để giảm thiểu lỗi làm tròn

- Sử dụng các kiểu dữ liệu có độ chính xác cao hơn (ví dụ: double thay vì float).
- Sử dụng các thuật toán số ổn định hơn và ít bị tích lũy lỗi hơn.
- Thận trọng với việc so sánh bằng nhau của các số dấu chấm động; sử dụng một dung sai nhỏ (epsilon).
- Cân nhắc các biểu diễn thay thế như số thập phân cho các phép tính tài chính, nơi biểu diễn thập phân chính xác là rất quan trọng.
- Thực hiện phép nhân trước phép cộng khi xử lý các số có độ lớn khác nhau.
- Làm tròn kết quả đến một số lượng chữ số thập phân cụ thể khi thích hợp.

2.3. So sánh giữa biểu diễn dấu chấm tĩnh và dấu chấm động

Biểu diễn dấu chấm động cung cấp một phạm vi số có thể biểu diễn rộng hơn nhiều, trong khi dấu chấm tĩnh cung cấp độ chính xác nhất quán và có thể dự đoán hơn trong phạm vi giới hạn của nó. Việc lựa chọn phụ thuộc vào việc ứng dụng yêu cầu xử lý một phổ độ lớn rộng hay cần một mức độ chính xác đảm bảo cho các số trong một phạm vi cụ thể, đã biết.

Số học dấu chấm tĩnh thường nhanh hơn và hiệu quả hơn số học dấu chấm động, đặc biệt là trên phần cứng không có đơn vị dấu chấm động chuyên dụng. Đối với các hệ thống thời gian thực và các ứng dụng nhúng có tài nguyên hạn chế, dấu chấm tĩnh có thể là một lựa chọn phù hợp hơn do chi phí tính toán thấp hơn.

Biểu diễn dấu chấm tĩnh đôi khi có thể yêu cầu ít bộ nhớ hơn dấu chấm động cho một phạm vi tương đương trong một số trường hợp nhất định, vì chúng không cần lưu trữ số mũ. Tuy nhiên, khả năng biểu diễn một phạm vi rộng hơn nhiều một cách hiệu quả của dấu chấm động thường biện minh cho việc sử dụng bộ nhớ của nó.

Số học dấu chấm tĩnh đơn giản hơn để triển khai trong phần cứng và phần mềm so với các thuật toán phức tạp hơn cần thiết cho các phép toán dấu chấm động.

Độ phù hợp cho các ứng dụng khác nhau:

- Dấu chấm tĩnh: Hệ thống nhúng, xử lý tín hiệu số, hệ thống điều khiển nơi tốc độ, tính dự đoán và tiêu thụ điện năng thấp là rất quan trọng.
- Dấu chấm động: Tính toán khoa học, mô phỏng kỹ thuật, xử lý đồ họa, học máy, các ứng dụng mục đích chung đòi hỏi dải động rộng và sự cân bằng giữa phạm vi và độ chính xác.

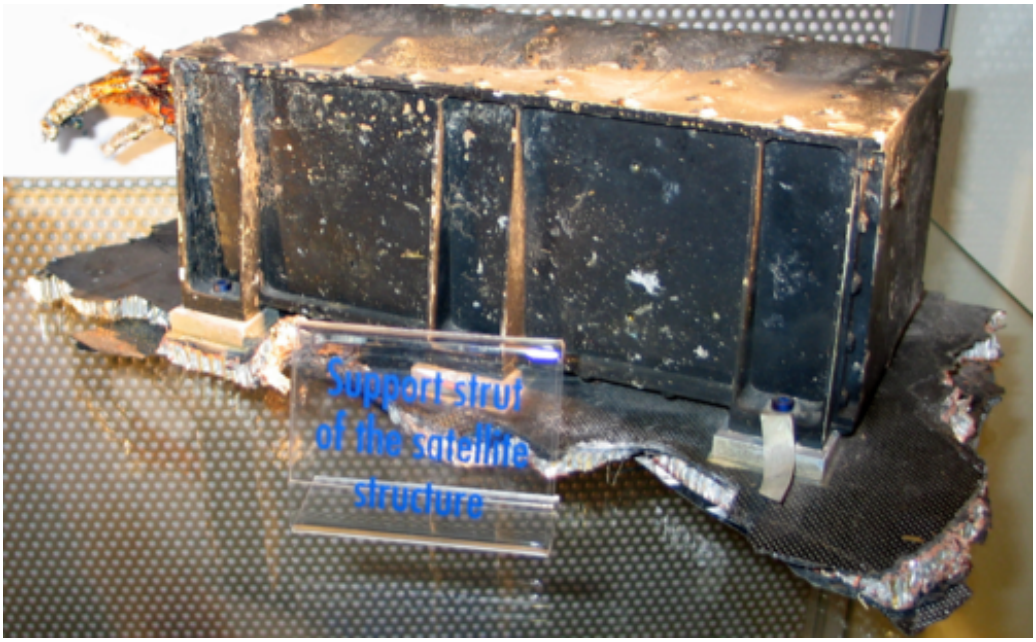
2.4. Tác động thực tế của lỗi dấu chấm động

Một số sự cố lịch sử làm nổi bật những hậu quả nghiêm trọng tiềm ẩn của lỗi dấu chấm động: [7]

- **Thất bại tên lửa Patriot (1991):** Một lỗi làm tròn nhỏ trong tính toán đồng hồ của hệ thống tích lũy theo thời gian, khiến tên lửa đánh giá sai quỹ đạo của một tên lửa Scud đang bay tới, dẫn đến thương vong.

Hình 2.2 — Tên lửa Patriot

- **Vụ nổ tên lửa Ariane 5 (1996):** Một lỗi tràn số xảy ra khi một số dấu chấm động 64 bit biểu diễn vận tốc ngang được chuyển đổi thành một số nguyên có dấu 16 bit, dẫn đến việc tên lửa bị phá hủy.



Hình 2.3 — Mảnh vỡ được thu hồi của tên lửa Ariane 5

- **Lỗi chỉ số Sở giao dịch chứng khoán Vancouver (1982-1984):** Lỗi cắt bớt lặp đi lặp lại trong tính toán chỉ số chứng khoán dẫn đến việc định giá thấp đáng kể giá trị của nó theo thời gian.
- **Nguy cơ ngừng hoạt động của Boeing 787 (2015):** Một lỗi tràn bộ đếm phần mềm sau 248 ngày hoạt động liên tục có thể dẫn đến mất điện.

Những ví dụ này nhấn mạnh tầm quan trọng của việc hiểu các hạn chế của số học dấu chấm động và sự cần thiết của việc thiết kế và thử nghiệm cẩn thận trong các ứng dụng quan trọng.

Lỗi dấu chấm động có thể có những tác động đáng kể trong nhiều lĩnh vực:

- **Tính toán khoa học và kỹ thuật:** Mô phỏng không chính xác, tính toán sai trong vật lý, hóa học và kỹ thuật.
- **Tài chính:** Lỗi trong tính toán tài chính có thể dẫn đến định giá, tính toán chi phí và dự báo không chính xác.
- **Hệ thống điều khiển:** Tính toán sai trong các thuật toán điều khiển có thể dẫn đến sự cố hệ thống, như đã thấy trong vụ tên lửa Patriot và vụ nổ Ariane 5.
- **Học máy và AI:** Mặc dù các định dạng độ chính xác thấp hơn ngày càng được sử dụng để tăng hiệu quả, nhưng việc hiểu rõ sự đánh đổi về độ chính xác là rất quan trọng.
- **Kết xuất đồ họa và trò chơi:** Mặc dù thường chịu được những sai sót nhỏ, nhưng những lỗi đáng kể có thể dẫn đến các hình ảnh hoặc hành vi không mong muốn.

Chương 3

Kết luận

3.1. Kết quả đạt được

Báo cáo này đã khám phá hai cơ chế chính để biểu diễn số thực trên máy tính: dấu chấm tĩnh và dấu chấm động. Dấu chấm tĩnh mang lại sự đơn giản và tốc độ cho các ứng dụng có phạm vi và yêu cầu độ chính xác đã biết. Dấu chấm động, đặc biệt là tiêu chuẩn IEEE 754, cung cấp một phạm vi rộng hơn nhiều và sự cân bằng giữa phạm vi và độ chính xác, khiến nó phù hợp với nhiều ứng dụng hơn.

Việc lựa chọn phương pháp biểu diễn thích hợp đòi hỏi sự cân nhắc cẩn thận về sự đánh đổi giữa phạm vi, độ chính xác, chi phí tính toán và việc sử dụng bộ nhớ. Hơn nữa, việc hiểu rõ các hạn chế vốn có và khả năng xảy ra lỗi, đặc biệt là trong số học dấu chấm động, là rất quan trọng để phát triển phần mềm đáng tin cậy và chính xác.

3.2. Khuyến nghị

Các khuyến nghị để chọn biểu diễn phù hợp:

- Đối với các ứng dụng có yêu cầu nghiêm ngặt về hiệu suất, độ chính xác có thể dự đoán và phạm vi giá trị đã biết, giới hạn, biểu diễn dấu chấm tĩnh có thể là lựa chọn hiệu quả nhất.
- Đối với các ứng dụng đòi hỏi dải động rộng và sự cân bằng tốt giữa phạm vi và độ chính xác, biểu diễn dấu chấm động, tuân thủ tiêu chuẩn IEEE 754, thường được ưu tiên hơn.
- Trong các ứng dụng quan trọng liên quan đến số học dấu chấm động, các nhà phát triển phải nhận thức được các lỗi làm tròn tiềm ẩn, mất độ chính xác và các điều kiện tràn số/thiếu số và sử dụng các chiến lược thích hợp để giảm thiểu tác động của chúng.

- Đối với các ứng dụng tài chính hoặc bất kỳ tình huống nào đòi hỏi biểu diễn thập phân chính xác, hãy cân nhắc sử dụng các kiểu dữ liệu thập phân hoặc các thư viện số học có độ chính xác tùy ý để tránh các hạn chế của dấu chấm động nhị phân.

Tài liệu tham khảo

[1] Number representation in computers 2025. https://fabienmaussion.info/scientific_programming/week_04/02-Numbers.html.

[2] Computer Engineering Concepts - 2.7 Real Number Representation 2025. https://www.computerengineeringconcepts.org/2.7-Real-Number-Representation.

[3] Fixed Point Representation 2025. https://www.geeksforgeeks.org/fixed-point-representation/.

[4] Representing Numbers in Computers 2025. https://www.technologyuk.net/mathematics/about-numbers/representing-numbers-in-computers.shtml.

[5] Computer Science: Real Numbers - IEEE Standard For Floating Point 2025. http://www.multiwingspan.co.uk/a23.php?page=real.

[6] Floating Point Representation – Basics 2025. https://www.geeksforgeeks.org/floating-point-representation-basics/.

[7] True North Floating Point 2025. https://www.truenorthfloatingpoint.com/problem.