COMPUTER ENGINEERING

KIẾN TRÚC MÁY TÍNH

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Tuần 9

PHÉP TOÁN SỐ HỌC TRÊN MÁY TÍNH

(Tiếp theo)

PHÉP TOÁN SỐ HỌC TRÊN MÁY TÍNH

Mục tiêu:

Hiểu các phép toán số học trên số nguyên và số thực dấu chấm động trong máy tính.

- **Với số nguyên:**
  - ✔ **Hiểu các phép toán cộng, trừ, nhân và chia**
  - ✔ **Cách thiết kế mạch nhân và chia**
- **Với số thực dấu chấm động:**
  - ✔ **Hiểu các phép toán cộng, trừ và nhân**
  - ✔ **Cách thiết kế mạch nhân**

Slide được dịch và các hình được lấy từ sách tham khảo:

Computer Organization and Design: The Hardware/Software Interface, Patterson, D. A., and J. L. Hennessy, Morgan Kaufman, Revised Fourth Edition, 2011.

COMPUTER ENGINEERING

PHÉP TOÁN SỐ HỌC TRÊN MÁY TÍNH

1. Giới thiệu
2. Phép cộng & Phép trừ
3. Phép nhân
4. Phép chia
5. **Số thực dấu chấm động**

Số thực dấu chấm động

Định nghĩa:

Biểu diễn số thực:

$3.14159265\ldots_{ten}$ (pi)

$2.71828\ldots_{ten}$ (e)

0.000000001_{ten} or $1.0_{ten} \times 10^{-9}$ (seconds in a nanosecond)

$3{,}155{,}760{,}000_{ten}$ or $3.15576_{ten} \times 10^{9}$ (seconds in a typical century)

- ***Scientific notation:*** Một số thực được gọi là "scientific notation" khi bên trái dấu chấm có đúng 1 chữ số.

- ***Normalized number:*** Một số thực được gọi là "***Normalized number***" (dạng chuẩn) khi số này được viết trong "scientific notation" và chữ số bên trái dấu chấm không phải là 0.

  Ví dụ: $1.0_{ten} \times 10^{-9}$: số thực chuẩn

  $0.1_{ten} \times 10^{-8}$: **không** phải số thực chuẩn

  $10.0_{ten} \times 10^{-10}$: **không** phải số thực chuẩn

Số thực dấu chấm động

Định nghĩa:

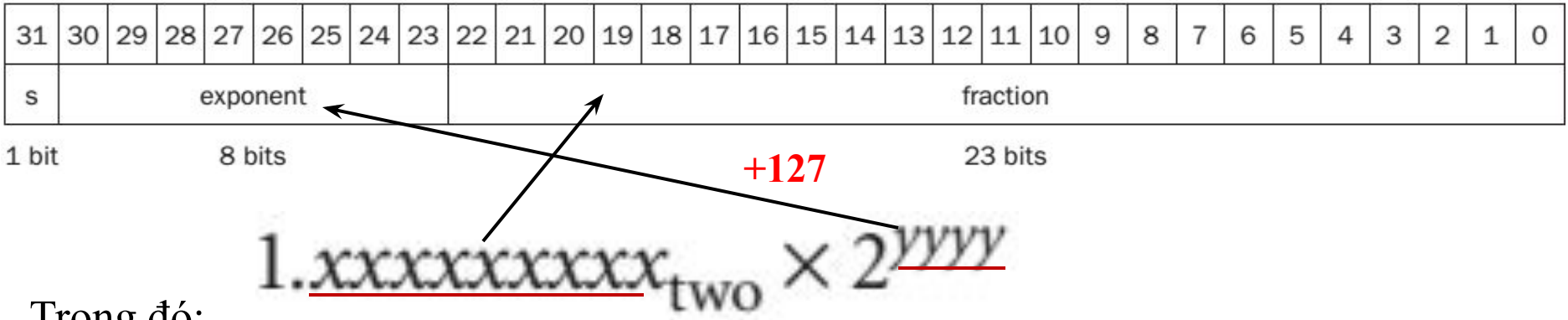
- Trong máy tính, các số nhị phân phải được đưa về dạng chuẩn như sau:

$$1.xxxxxxxxxx_{two} \times 2^{yyyy}$$

Số thực dấu chấm động

Biểu diễn số thực dấu chấm động

Biểu diễn số thực dấu chấm động theo chuẩn **IEEE 754** (với độ chính xác đơn)
(chuẩn này được áp dụng cho hầu hết các máy tính được chế tạo từ năm 1980)



Trong đó:

s biểu diễn dấu của số thực dấu chấm động (1 nghĩa là âm, ngược lại 0 là dương)

Phần mũ (exponent) có kích thước là 8 bit. Exponent là biểu diễn quá 127 của *yyyy* (*excess-127* hoặc *bias of 127*).

Phần lẻ (fraction) dùng 23 bits để biểu diễn cho *xxxxxxxxx*

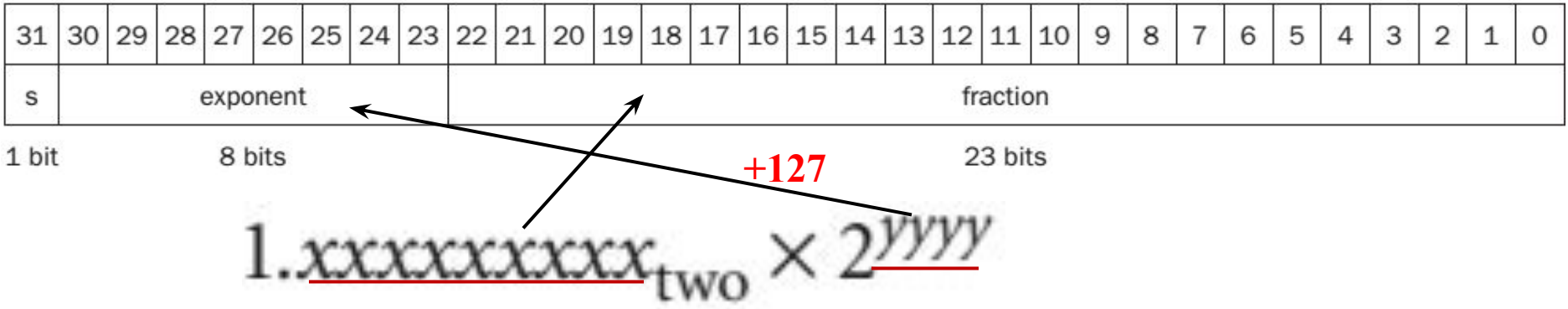
Tổng quát, số th... Bias = 127):

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

Số thực dấu chấm động

Biểu diễn số thực dấu chấm động

Biểu diễn số thực dấu chấm động theo chuẩn IEEE 754 (với độ chính xác đơn)
(chuẩn này được áp dụng cho hầu hết các máy tính được chế tạo từ năm 1980)



Tổng quát, số thực dấu chấm động được tính dựa theo:

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

Hoặc:

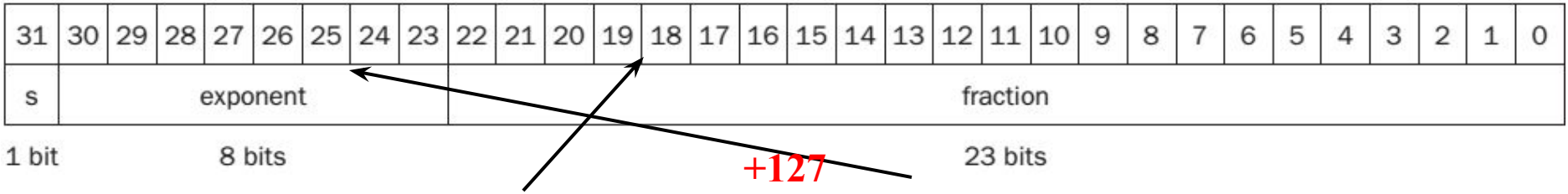
$$(-1)^S \times (1 + (s1 \times 2^{-1}) + (s2 \times 2^{-2}) + (s3 \times 2^{-3}) + (s4 \times 2^{-4}) + \ldots) \times 2^E$$

(với s1, s2, s3 … là các bit lần lượt từ trái sang phải của fraction)

Số thực dấu chấm động

Biểu diễn số thực dấu chấm động

Biểu diễn số thực dấu chấm động theo chuẩn IEEE 754 (với độ chính xác đơn)
(chuẩn này được áp dụng cho hầu hết các máy tính được chế tạo từ năm 1980)



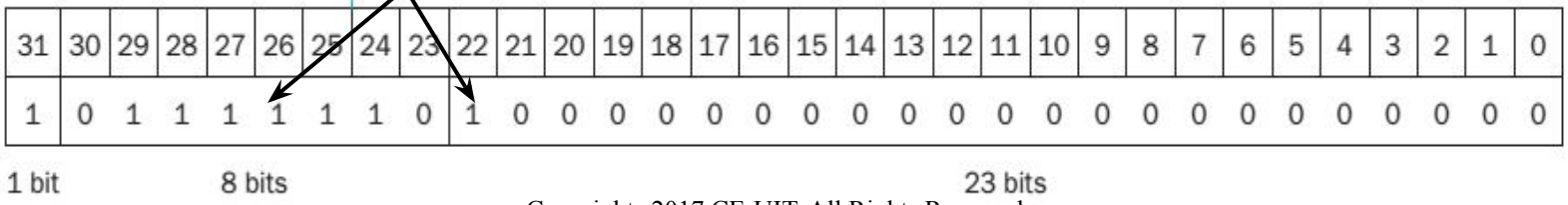
$$1.xxxxxxxxxx_{two} \times 2^{yyyy}$$

Ví dụ: Số -0.75 sẽ được biểu diễn trong máy tính như thế nào nếu dùng chuẩn IEEE 754 với độ chính xác đơn

$-0.75_{ten} = -3/4_{ten} = -3/2^2_{ten} = -11_{two}/2^2_{ten} = -0.11_{two}$

Chuẩn hóa: $0.11_{two} = 1.1_{two} \times 2^{-1}$

-1 +127 = 126



COMPUTER ENGINEERING

Số thực dấu chấm động

Ví dụ: Cho biểu diễn số dấu chấm động với độ chính xác đơn như hình sau, hỏi số tương ứng với biểu diễn này trong hệ thập phân là bao nhiêu?

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.	.	.

Trả lời:

bit dấu s là 1

exponent chứa 129

Số tương ứng: $(-1)^s \times (1+ \text{fraction}) \times 2^{(\text{exponent} - 127)}$

$$= (-1)^1 \times (1+ 0.01) \times 2^{(129 - 127)}$$

$$= (-1.01 \times 2^2)_{two} = -5.0_{ten}$$

COMPUTER ENGINEERING

Số thực dấu chấm động

Biểu diễn số thực dấu chấm động

- **Tràn trên (Overflow):** trường hợp này xảy ra khi kích thước của số mũ lớn hơn kích thước giới hạn trên (số mũ dương).
- **Tràn dưới (Underflow):** trường hợp này xảy ra khi kích thước của số mũ nhỏ hơn kích thước giới hạn dưới (số mũ âm).

Nhằm hạn chế việc tràn trên hoặc tràn dưới về số mũ, IEEE 754 giới thiệu thêm một cách biểu diễn số thực dấu chấm động, vơi trường exponent mở rộng lên tới 11 bits. Cách biểu diễn này gọi là IEEE 754 với độ chính xác kép

- **Độ chính xác đơn (Single precision):** một số thực dấu chấm động được biểu diễn ở dạng 32 bit.
- **Độ chính xác kép (Double precision):** một số thực dấu chấm động được biểu diễn ở dạng 64 bit.

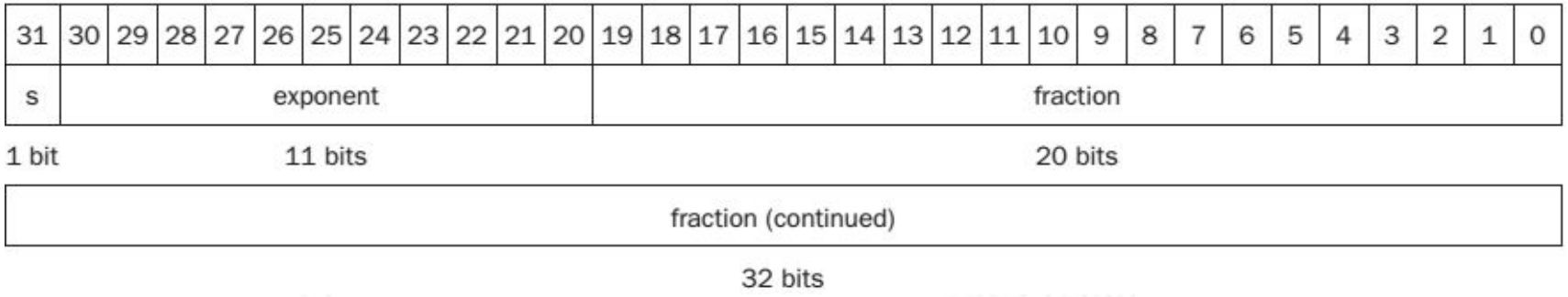
*Chú ý: Trong lập trình ngôn ngữ C, các số thực dạng **float** sẽ được định dạng theo kiểu độ chính xác đơn, còn các số dạng **double** sẽ được định dạng theo kiểu độ chính xác kép*

COMPUTER ENGINEERING

Số thực dấu chấm động

Biểu diễn số thực dấu chấm động

Biểu diễn số thực dấu chấm động theo chuẩn **IEEE 754** (với độ chính xác kép)



$$1.xxxxxxxxxx_{two} \times 2^{yyyy}$$

Trong đó:

s biểu diễn dấu của số thực dấu chấm động (1 nghĩa là âm, ngược lại 0 là dương)

Phần mũ (exponent) có kích thước là 11 bits. Exponent là biểu diễn quá 1023 của yyyy (*excess-1023* hoặc *bias of 1023*).

Phần lẻ (fraction) dùng 52 bits để biểu diễn

Tổng quát, số [illegible] với Bias = 1023):

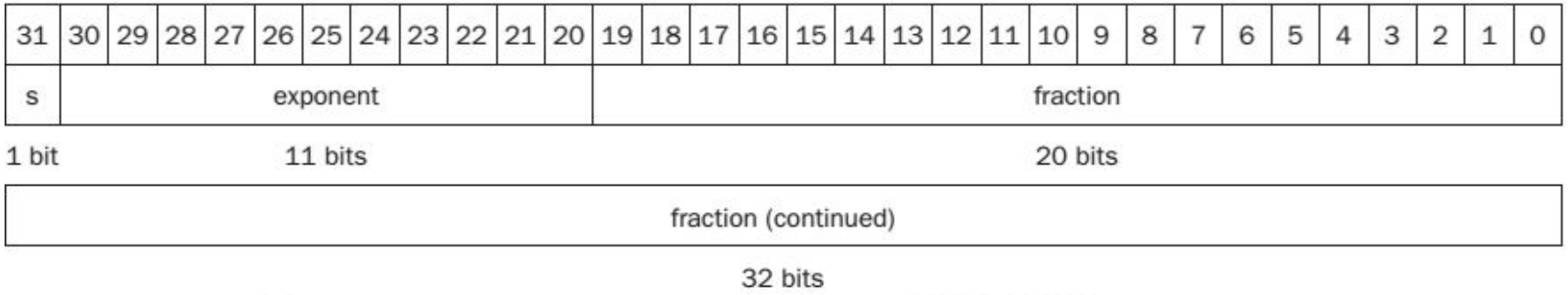
$$(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

COMPUTER ENGINEERING

Số thực dấu chấm động

Biểu diễn số thực dấu chấm động

Biểu diễn số thực dấu chấm động theo chuẩn **IEEE 754** (với độ chính xác kép)

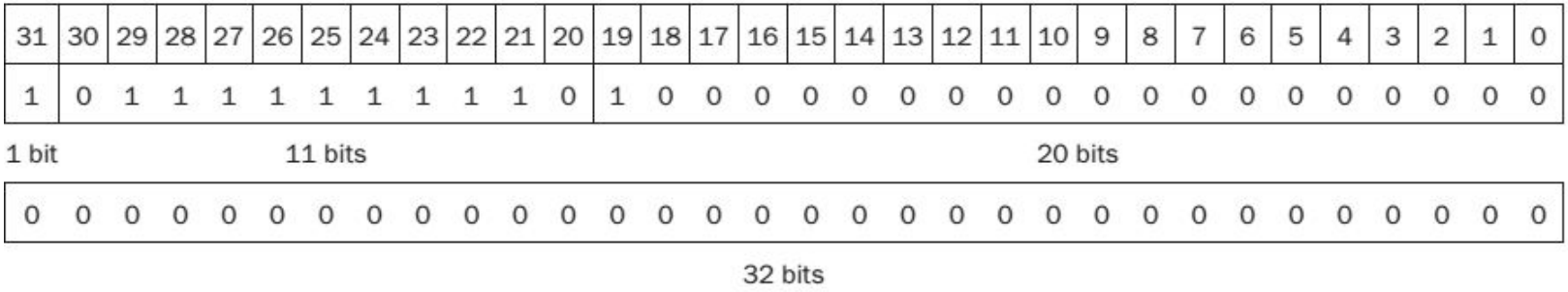


$$1.xxxxxxxxxx_{two} \times 2^{yyyy}$$

Ví dụ: Số -0.75 sẽ được biểu diễn trong máy tính như thế nào nếu dùng chuẩn IEEE 754 với độ chính xác kép

$-0.75_{ten} = -3/4_{ten} = -3/2^2_{ten} = -11_{two}/2^2_{ten} = 0.11_{two}$

Chuẩn hóa: $0.11_{two} = 1.1_{two} \times 2^{-1}$ **(phần exponent = -1 + 1023 = 1022 = 01111111110)**



Số thực dấu chấm động

Biểu diễn số thực dấu chấm động

Tại sao IEEE 754 không sử dụng biểu diễn dạng bù hai cho phần mũ mà dùng dạng bias-of-127 cho độ chính xác đơn và và bias-of-1023 cho độ chính xác kép?

Ví dụ: giả sử dùng bù 2 để biểu diễn phần mũ cho 2 số sau:

$1.0_{two} \times 2^{-1}$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.	.	.

$1.0_{two} \times 2^{+1}$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.	.	.

□ Khi nhìn vào phần mũ của $1.0_{two} \times 2^{-1}$ thì nó lại giống như là số rất lớn (thực chất lại là nhỏ), còn trong khi nhìn vào phần mũ của $1.0_{two} \times 2^{+1}$ thì nó lại giống như là số nhỏ (thực chất lại là lớn) □ vì vậy IEEE 754 chọn cách biểu diễn dùng bias-of-127 cho độ chính xác đơn thay vì bù 2

Số thực dấu chấm động

Biểu diễn số thực dấu chấm động dùng IEEE 754

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

Dãy biểu diễn số độ chính xác đơn có tầm trị từ:

Số nhỏ nhất:

$$\pm 1.0000\ 0000\ 0000\ 0000\ 0000\ 000_{two} \times 2^{-126}$$

Đến số lớn nhất

$$\pm 1.1111\ 1111\ 1111\ 1111\ 1111\ 111_{two} \times 2^{+127}$$

Single precision		Double precision		Object represented
Exponent	Fraction	Exponent	Fraction	
0	0	0	0	0
0	Nonzero	0	Nonzero	± denormalized number
1–254	Anything	1–2046	Anything	± floating-point number
255	0	2047	0	± infinity
255	Nonzero	2047	Nonzero	NaN (Not a Number)

IEEE 754 mã hóa số thực dấu chấm động

Số thực dấu chấm động

Biểu diễn số thực dấu chấm động

Các vấn đề cần lưu ý:

Rõ ràng, trong một biểu diễn số thực dấu chấm động nếu

- Tăng số bit chứa phần fraction thì tăng độ chính xác.

- Tăng kích thước phần exponent là tăng tầm trị biểu diễn.

- Vì vậy, khi thiết kế một biểu diễn/thể hiện cho số dấu chấm động (ví dụ không sử dụng IEEE 754) thì tùy vào mục đích sử dụng mà lựa chọn số giới hạn cho fraction và exponent sao cho phù hợp nhất.

Số thực dấu chấm động

Biểu diễn số thực dấu chấm động

Các vấn đề cần lưu ý:

- Số thực dấu chấm động dạng nhị phân (binary floating-point) dạng chuẩn

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

- Số thực dấu chấm động dạng thập phân (decimal floating-point) dạng chuẩn:

$$1.xxxxxxxxx_{ten} \times 10^{yyyy}$$

***yyyy*: exponent (phần mũ)**
***xxxxxxxxx*: fraction (tạm dịch là phần phân số/lẻ)**
***1.xxxxxxxxx*: significand (tạm dịch là phần trị)**

Số thực dấu chấm động

Phép toán cộng trên số thực dấu chấm động

Ví dụ: Thực hiện cộng hai số thực dấu chấm động chuẩn trong hệ thập phân sau

$$9.999_{ten} \times 10^{1} + 1.610_{ten} \times 10^{-1}.$$

Giả sử số thực dấu chấm động lưu trữ phần trị (significand) dùng 4 chữ số, phần số mũ (exponent) lưu trữ dùng 2 chữ số.

Bước 1.

Điều chỉnh sao cho phần mũ của hai số hạng trở thành bằng nhau
(Lấy số hạng có số mũ nhỏ hơn điều chỉnh theo số hạng có số mũ lớn hơn)

$$1.610_{ten} \times 10^{-1} = 0.01610_{ten} \times 10^{1}$$

Vì significand chỉ cho phép dùng 4 chữ số, nên $0.01610_{ten} \times 10^{1}$ làm tròn thành 0.016×10^{1}

(quy tắc làm tròn tùy vào đề bài yêu cầu. Trong ví dụ này, làm tròn theo quy tắc nếu chữ số bên phải của phần bỏ đi lớn hơn hoặc bằng 5 thì chữ số bên trái nhất của phần còn lại tăng lên 1)

Số thực dấu chấm động

Phép toán cộng trên số thực dấu chấm động

Bước 2.

Thực hiện cộng phần significand của hai số hạng

$$\begin{array}{rr} & 9.999_{ten} \\ + & 0.016_{ten} \\ \hline & 10.015_{ten} \end{array}$$

Tổng là $10.015_{ten} \times 10^{1}$

Bước 3.

- Chuyển tổng về dạng chuẩn hóa: $10.015_{ten} \times 10^{1} = 1.0015 \times 10^{2}$

- Kiểm tra phần mũ có bị tràn trên, tràn dưới ? => không tràn

(Nếu tràn, phép toán sẽ tạo ra một ngoại lệ (exception) và dừng)

Bước 4.

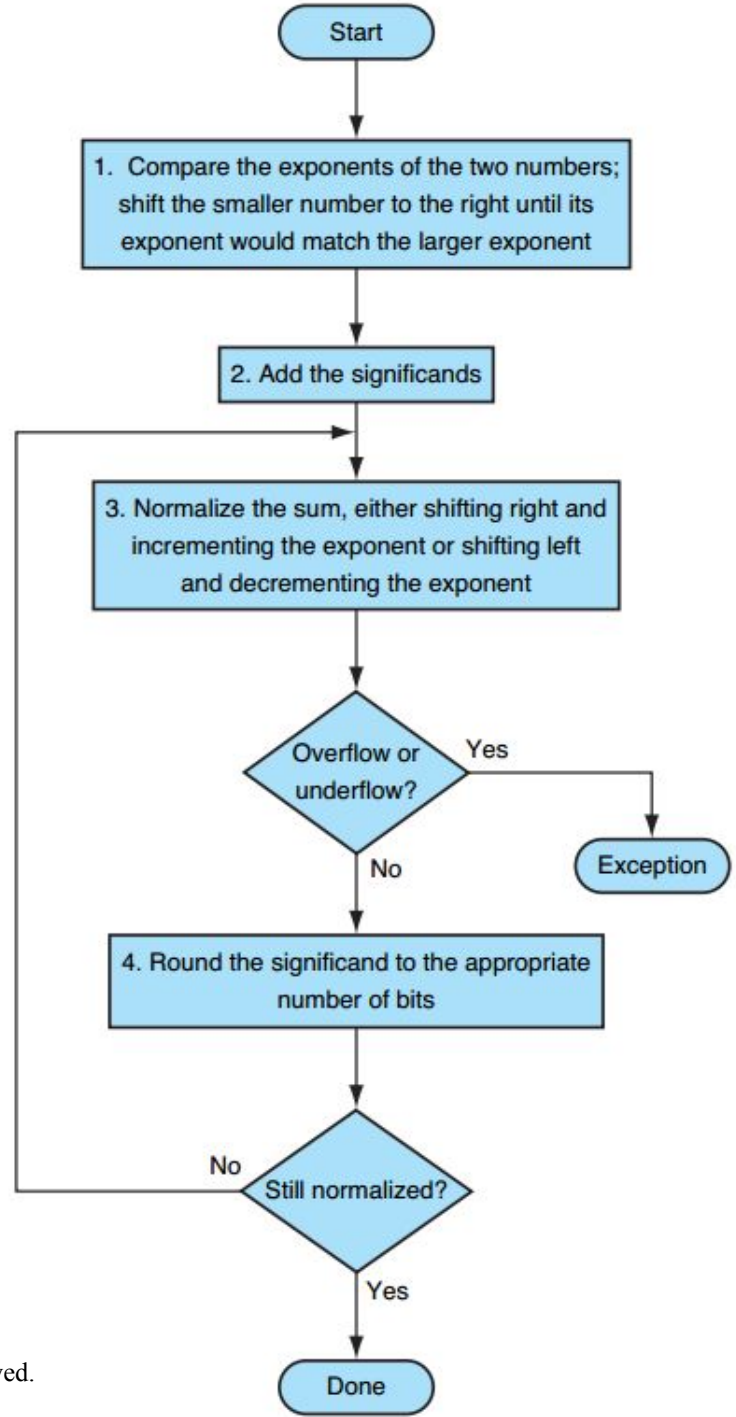
Làm tròn tổng: vì significand chỉ cho phép dùng 4 chữ số, nên 1.0015×10^{2} làm tròn thành 1.002×10^{2}

Lưu ý: Việc làm tròn trong một số trường hợp có thể làm tổng mất đi dạng chuẩn hóa. Vì vậy sau khi làm tròn, phải kiểm tra xem tổng có còn trong dạng chuẩn hóa hay không, nếu không, quay lại bước 3

Số thực dấu chấm động

Phép toán cộng trên số thực dấu chấm động

Giải thuật thực hiện phép cộng trên số thực dấu chấm động trong *hệ nhị phân* tương tự như cho số *hệ thập phân*



Số thực dấu chấm động

Phép toán cộng trên số thực dấu chấm động

Ví dụ: **Cộng 2 số thực dấu chấm động trong hệ nhị phân cho 2 số thập phân sau: 0.5_{10} và -0.4375_{10} theo lưu đồ giải thuật.**

Giả sử phần significant dùng 4 bits lưu trữ, còn phần mũ lưu trữ như IEEE 754 độ chính xác đơn.

Đáp án:

Let's first look at the binary version of the two numbers in normalized scientific notation, assuming that we keep 4 bits of precision:

$$
\begin{aligned}
0.5_{ten} &= 1/2_{ten} &&= 1/2^{1}{}_{ten} \\
&= 0.1_{two} &&= 0.1_{two} \times 2^{0} &&= 1.000_{two} \times 2^{-1} \\
-0.4375_{ten} &= -7/16_{ten} &&= -7/2^{4}{}_{ten} \\
&= -0.0111_{two} &&= -0.0111_{two} \times 2^{0} &&= -1.110_{two} \times 2^{-2}
\end{aligned}
$$

Số thực dấu chấm động

Now we follow the algorithm:

Step 1. The significand of the number with the lesser exponent ($-1.11_{two} \times 2^{-2}$) is shifted right until its exponent matches the larger number:

$$-1.110_{two} \times 2^{-2} = -0.111_{two} \times 2^{-1}$$

Step 2. Add the significands:

$$1.000_{two} \times 2^{-1} + (-0.111_{two} \times 2^{-1}) = 0.001_{two} \times 2^{-1}$$

Step 3. Normalize the sum, checking for overflow or underflow:

$$0.001_{two} \times 2^{-1} = 0.010_{two} \times 2^{-2} = 0.100_{two} \times 2^{-3}$$
$$= 1.000_{two} \times 2^{-4}$$

Since $127 \geq -4 \geq -126$, there is no overflow or underflow. (The biased exponent would be $-4 + 127$, or 123, which is between 1 and 254, the smallest and largest unreserved biased exponents.)

Step 4. Round the sum:

$$1.000_{two} \times 2^{-4}$$

The sum already fits exactly in 4 bits, so there is no change to the bits due to rounding.

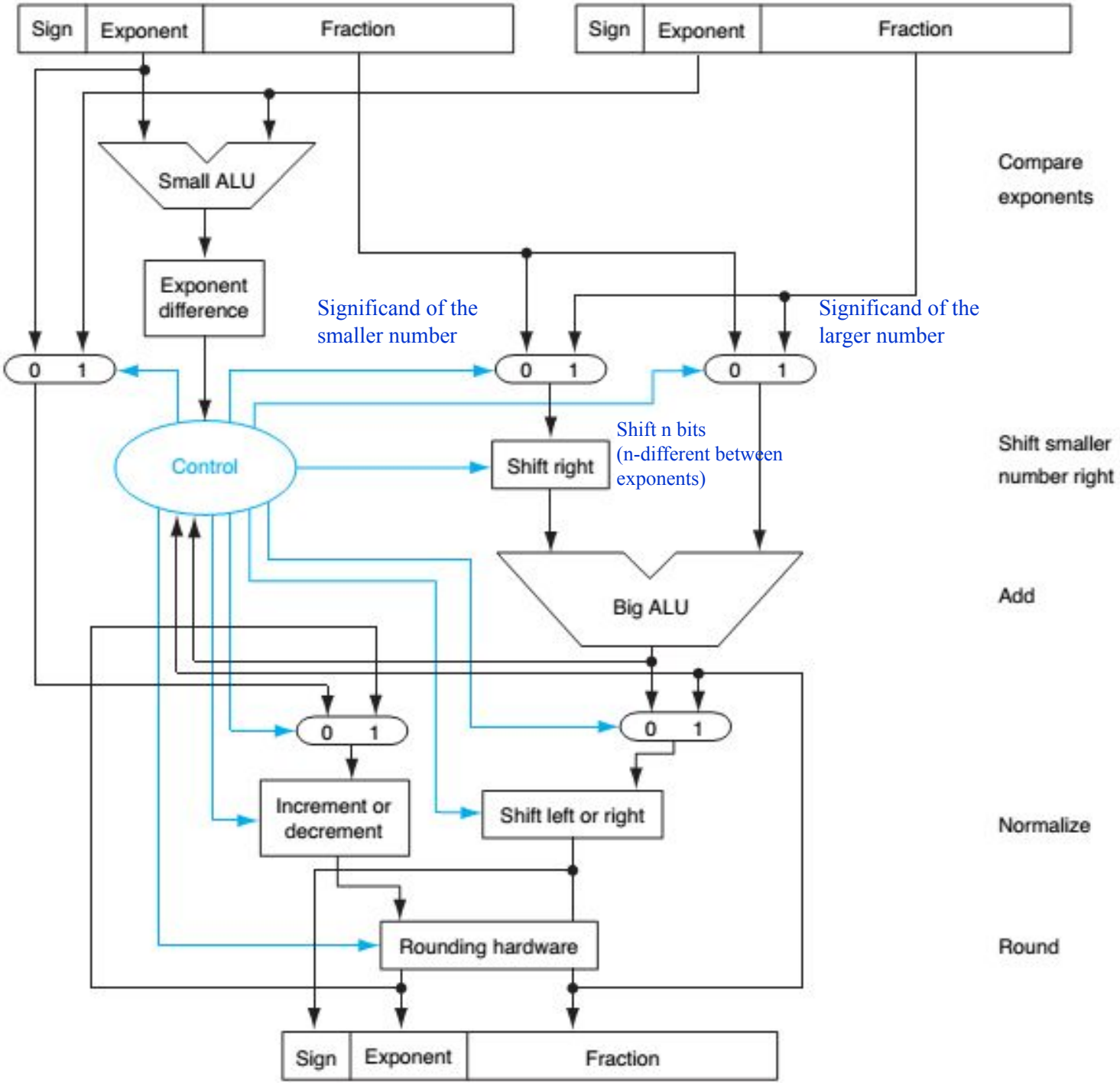
This sum is then

$$1.000_{two} \times 2^{-4} = 0.0001000_{two} = 0.0001_{two}$$
$$= 1/2^{4}_{ten} \qquad = 1/16_{ten} \qquad = 0.0625_{ten}$$

This sum is what we would expect from adding 0.5_{ten} to -0.4375_{ten}.

COMPUTER ENGINEERING

Kiến trúc phần cứng phép nhân hai số floating-point:



Số thực dấu chấm động

Phép nhân trên số thực dấu chấm động

Ví dụ: Thực hiện phép nhân hai số thực dấu chấm động chuẩn trong hệ thập phân sau: $(1.110_{10} \times 10^{10}) \times (9.200_{10} \times 10^{-5})$

Giả sử số thực dấu chấm động lưu trữ phần trị dùng 4 chữ số và phần mũ dùng 2 chữ số.

Đáp án:

Step 1. Unlike addition, we calculate the exponent of the product by simply adding the exponents of the operands together:

$$\text{New exponent} = 10 + (-5) = 5$$

COMPUTER ENGINEERING

Số thực dấu chấm động

Phép nhân trên số thực dấu chấm động

Step 2. Next comes the multiplication of the significands:

$$
\begin{array}{r}
1.110_{ten} \\
\times\ 9.200_{ten} \\
\hline
0000 \\
0000 \\
2220 \\
9990 \\
\hline
10212000_{ten}
\end{array}
$$

There are three digits to the right of the decimal point for each operand, so the decimal point is placed six digits from the right in the product significand:

Assuming that we can keep only three digits to the right of the decimal point, the product is 10.212×10^5.

Step 3. This product is unnormalized, so we need to normalize it:

$$10.212_{ten} \times 10^5 = 1.0212_{ten} \times 10^6$$

Chú ý: kiểm tra số mũ có bị tràn trên, tràn dưới ?

Số thực dấu chấm động

Phép nhân trên số thực dấu chấm động

Step 4. We assumed that the significand is only four digits long (excluding the sign), so we must round the number. The number

$$1.0212_{ten} \times 10^6$$

is rounded to four digits in the significand to

$$1.021_{ten} \times 10^6$$

Step 5. The sign of the product depends on the signs of the original operands. If they are both the same, the sign is positive; otherwise, it's negative. Hence, the product is

$$+1.021_{ten} \times 10^6$$

The sign of the sum in the addition algorithm was determined by addition of the significands, but in multiplication, the sign of the product is determined by the signs of the operands.

Số thực dấu chấm động

Phép nhân trên số thực dấu chấm động

Việc thực hiện phép nhân trên số thực dấu chấm động nhị phân cũng tương tự như ví dụ trên, nhưng lưu ý phần mũ khi được lưu theo định dạng IEEE 754

Ví dụ:

Cần nhân hai số thực dấu chấm động đang được lưu trữ theo IEEE 754 độ chính xác đơn, biết 8 bit phần mũ của số thứ nhất trong lưu trữ có giá trị là 137_{ten} và 8 bit phần mũ của số thứ hai trong lưu trữ có giá trị là 122_{ten}.

Phần mũ của tích khi lưu trữ :

$$137_{ten} + 122_{ten} = 259_{ten}$$

Giá trị 259_{ten} đúng hay sai? □ Sai

Giá trị đúng của tích trong lưu trữ phải là:

$$(137_{ten} + 122_{ten}) - 127_{ten} = 132_{ten}$$

Vì thực chất:

- Số mũ của số thứ nhất là 10. Khi được lưu trữ theo IEEE 754, phần mũ lưu 10 + 127 = 137
- Số mũ của số thứ hai là -5. Khi được lưu trữ theo IEEE 754, phần mũ lưu -5 + 127 = 122

Số mũ của tích phải là 10 + (-5) = 5 Và nếu được lưu trữ theo IEEE 754, phần mũ của tích lưu 5 + 127 = 132

Vì vậy nếu lấy 137 + 122 thì 127 đã được cộng hai lần

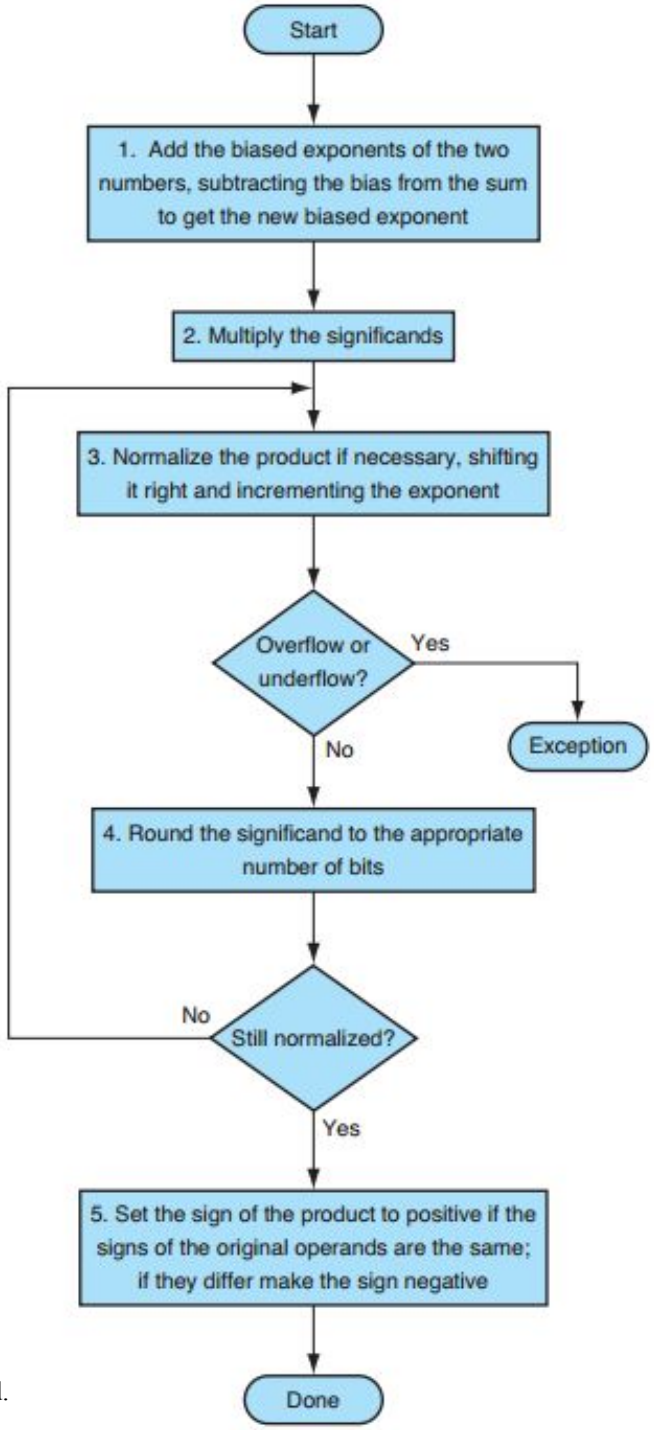
Số thực dấu chấm động

Phép nhân trên số thực dấu chấm động

Giải thuật nhân số thực dấu chấm động trên hệ nhị phân có 5 bước giống như là ví dụ phép nhân số trong hệ thập phân.

Nhưng lưu ý: Bước 1 khi cộng hai exponent của hai số, **nhớ trừ đi số *bias***

- *Nếu IEEE 754 độ chính xác đơn: bias = 127*
- *Nếu IEEE 754 độ chính xác kép: bias = 1023*



Số thực dấu chấm động

Phép nhân trên số thực dấu chấm động

Ví dụ: nhân số thực dấu chấm động trên hệ nhị phân cho 2 số sau:

$$\mathbf{0.5_{10}}\ \textbf{và}\ \mathbf{-0.4375_{10}}.$$

Biết các số dấu chấm động dùng lưu trữ theo IEEE 754 độ chính xác đơn, nhưng phần significant chỉ cho phép 4 bits

COMPUTER ENGINEERING

Số thực dấu chấm động

Phép nhân trên số thực dấu chấm động

In binary, the task is multiplying $1.000_{two} \times 2^{-1}$ by $-1.110_{two} \times 2^{-2}$.

Step 1. Adding the exponents without bias:

$$-1 + (-2) = -3$$

or, using the biased representation:

$$(-1 + 127) + (-2 + 127) - 127 = (-1 - 2) + (127 + 127 - 127)$$
$$= -3 + 127 = 124$$

Step 2. Multiplying the significands:

$$\begin{array}{r} 1.000_{two} \\ \times \quad 1.110_{two} \\ \hline 0000 \\ 1000 \\ 1000 \\ 1000 \\ \hline 1110000_{two} \end{array}$$

The product is $1.110000_{two} \times 2^{-3}$, but we need to keep it to 4 bits, so it is $1.110_{two} \times 2^{-3}$.

Số thực dấu chấm động

Phép nhân trên số thực dấu chấm động

Step 3. Now we check the product to make sure it is normalized, and then check the exponent for overflow or underflow. The product is already normalized and, since $127 \geq -3 \geq -126$, there is no overflow or underflow. (Using the biased representation, $254 \geq 124 \geq 1$, so the exponent fits.)

Step 4. Rounding the product makes no change:

$$1.110_{two} \times 2^{-3}$$

Step 5. Since the signs of the original operands differ, make the sign of the product negative. Hence, the product is

$$-1.110_{two} \times 2^{-3}$$

Converting to decimal to check our results:

$$\begin{aligned} -1.110_{two} \times 2^{-3} &= -0.001110_{two} = -0.00111_{two} \\ &= -7/2^{5}{}_{ten} = -7/32_{ten} = -0.21875_{ten} \end{aligned}$$

The product of 0.5_{ten} and -0.4375_{ten} is indeed -0.21875_{ten}.

COMPUTER ENGINEERING

Số thực dấu chấm động

Sinh viên tìm hiểu:

- **Phép chia với số floating-point trong MIPS**
- **Các lệnh làm việc với số floating-point trong MIPS**
  - Các lệnh liên quan đến số floating-point
  - Phân biệt các lệnh:

    mult, multu, mul.s, mul.d

    div, divu, div.s, div.d

PHÉP TOÁN SỐ HỌC TRÊN MÁY TÍNH

Tổng kết:

- Hiểu cách biểu diễn số thực dấu chấm động theo IEEE 754 trong máy tính theo:
  - ✔ Độ chính xác đơn
  - ✔ Độ chính xác kép
- Hiểu cách máy tính thực hiện cộng, trừ, nhân chia trên số thực dấu chấm động
- Hiểu cách thiết kế một mạch cộng hai số thực dấu chấm động cơ bản

PHÉP TOÁN SỐ HỌC TRÊN MÁY TÍNH

- **Lý thuyết: Đọc sách tham khảo**
  - Mục: 3.5
  - Sách: *Computer Organization and Design: The Hardware/Software Interface*, Patterson, D. A., and J. L. Hennessy, Morgan Kaufman, Revised Fourth Edition, 2011.

- **Bài tập:** file đính kèm