TRƯỜNG ĐẠI HỌC TÂY BẮC

BÁO CÁO TỔNG KẾT
ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN

LẬP TRÌNH MÔ PHỎNG CÁC GIẢI THUẬT XỬ LÝ CÁC PHÉP TOÁN SỐ NGUYÊN CỦA ALU

Thuộc nhóm ngành: Tự nhiên

Sơn La, tháng năm 2024

TRƯỜNG ĐẠI HỌC TÂY BẮC

BÁO CÁO TỔNG KẾT
ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN

LẬP TRÌNH MÔ PHỎNG CÁC GIẢI THUẬT XỬ LÝ CÁC PHÉP TOÁN SỐ NGUYÊN CỦA ALU

Thuộc nhóm ngành: Tự nhiên

Sinh viên thực hiện: **Phạm Trần Hoàng Anh** Lớp K63 ĐH CNTT A

Người hướng dẫn: **ThS. Phan Trung Kiên**

Sơn La, tháng 7 năm 2023

CHƯƠNG 1. Biểu diễn và xử lý số nguyên trong máy tính điện tử

1.1. Các hệ đếm

Về mặt toán học, chúng ta có thể biểu diễn số theo hệ đếm cơ số bất kỳ, song khi nghiên cứu về máy tính chúng ta chỉ quan tâm đến các hệ đếm sau đây:

+ Hệ thập phân (Decimal System): Con người sử dụng

+ Hệ nhị phân (Binary System): Máy tính sử dụng

+ Hệ thập lục phân (Hexadecimal System): Dùng để viết gọn cho số nhị phân, ta thường thấy máy báo lỗi ở dạng hexa là chính, cụ thể là trong thết kế Flash thì có chế độ màu...

1.1.1. Hệ thập phân.

- Sử dụng 10 chữ số 0, 1... 9 để biểu diễn số

- Dùng n chữ số thập phân để biểu diễn được 10^n giá trị khác nhau, là hệ đếm quen thuộc của nhân loại

- Giả sử một số A được biểu diễn dưới dạng

$A = a_n a_{n-1} a_{n-2} ... a_1 a_0 . a_{-1} ... a_{-m}$

-> Giá trị của A được hiểu như sau:

$A = a_n 10^n + a_{n-1} 10^{n-1} + ... + a_1 10^1 + a_0 10^0 + a_{-1} 10^{-1} + ... + a_{-m} 10^{-m}$

$$A = \sum_{i=-m}^{n} a_i 10^i$$

- Với số nguyên, ta lấy số đó lần lượt chia cho 10 thì ta được số dư chính là các chữ số từ phải sang trái của số nguyên đó. Với phần lẻ ta thực hiện nhân lần lượt phần lẻ cho 10 và lấy phần nguyên.

- Ta thấy rằng một chuỗi n chữ số thập phân sẽ biểu diễn được 10^n giá trị khác nhau.

1.1.2. Hệ nhị phân.

- Sử dụng 2 chữ số: 0 và 1 để biểu dễn

- Nó được hình thành từ cơ sở đại số lôgic Boole, xuất hiện từ cuối thế kỷ 19, hệ đếm này và các môn toán liên quan đến nó thực sự phát huy được sức mạnh từ khi có mạch điện logic hai trạng thái (đóng/mở).

- Chữ số nhị phân được gọi là Bit (Binary digit).

- Bit là đơn vị đo thông tin nhỏ nhất.

- Dùng n bit có thể biểu diễn được 2^n giá trị khác nhau

  00... 000 = 0

  11 ... 111 = $2^n - 1$

- Giả sử có một số A được biểu diễn dưới dạng như sau

  $A = a_n a_{n-1} a_{n-2} \dots a_0 . a_{-1} a_{-2} \dots a_{-m}$ thì giá trị của A là:

  $A = a_n 2^n + a_{n-1} 2^{n-1} + \dots + a_0 2^0 + a_{-1} 2^{-1} + \dots + a_{-m} 2^{-m}$

  $= \sum_{i=-m}^{n} a_i 2^i$

-> Áp dụng các tính ở trên để đổi từ số nhị phân sang thập phân

*** Đổi từ thập phân sang nhị phân**

- Thực hiện chuyển phần nguyên và phần lẻ riêng biệt

  + Chuyển phần nguyên

  - Cách 1: Chia dần số đó cho 2, xác định các phần dư rồi viết các số dư đó theo chiều ngược lại.

  - Cách 2: Phân tích số đó thành tổng các lũy thừa của 2, dựa vào số mũ để xác định dạng nhị phân.

  + Chuyển đổi phần lẻ: Nhân phần lẻ với 2 rồi lấy phần nguyên theo chiều thuận.

- Nếu sau một số lần nhân vẫn không hết phần lẻ, thì ta lấy giá trị gần đúng (xấp xỉ).

1.1.3. Hệ mười sáu (Hexa).

- Vì số nhị phân quá dài và bất tiện khi viết và tính toán, vậy 4 chữ số nhị phân được gộp lại thành 1 chữ số thập lục phân. Như vậy có số của nó là 16, vậy ta cũng cần sử dụng 16 ký tự để biểu diễn đó là: 0, 1, ..., 9, A, B, C, D, E, F.

- Dùng để viết gọn cho hệ nhị phân: Cứ một nhóm 4 bit sẽ được thay bằng một số hexa

Hệ thập phân	Hệ nhị phân	Hệ hexa
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Bảng mô tả so sánh 3 hệ đếm cơ bản

1.2. Biểu diễn số nguyên.

- Đối với số nguyên ta cần chú ý đến 2 bit của byte 8 bit hoặc 1 Word 16 bit. Bit có trọng lượng cao nhất hay bit nặng nhất (Msb- Most Significant Bit) là bit tận

cùng bên trái. Trong 1 Word thì bit nặng nhất là bit 15, tương ứng với byte là 7. Và bit nhẹ nhất Lsb – Last Significant Bit là bit tận cùng bên phải hay bit 0.

- Nếu Lsb = 1 thì giá trị của số đó là lẻ và ngược lại Lsb = 0 thì giá trị của nó là chẵn.

- Số nguyên có dấu thì có thể là số âm hoặc dương, nên nó dùng một bit để biểu diễn dấu của số nguyên đó.

- Các số âm được lưu trữ trong máy tính dưới dạng số bù 2.

1.2.1. Số nguyên không dấu.

- Là số được định nghĩa bởi các số dương nên không cần dùng bit nào để biểu diễn dấu.

- Dạng tổng quát: Giả sử dùng n bit để biểu diễn cho một số nguyên không dấu A

$A = a_n a_{n-1} a_{n-2} \ldots a_0 \rightarrow$ giá trị A được tính như sau:

$A = a_n 2^n + a_{n-1} 2^{n-1} + \ldots + a_0 2^0$

$= \sum_{i=0}^{n-1} a_i 2^i$

- Dải biểu diễn của A là từ: $[0 \div 2^n - 1]$

- Xét các trường hợp cụ thể của n. Với n = 8 bit, dải biểu diễn là [0 ÷ 255]

0000 0000 = 0

1111 1111 = 255

+ Kiểu dữ liệu tương ứng trong Turbo Pascal là kiểu byte

```
Program Kieubyte;
Uses Crt;
Var
    a: Byte;
Begin
    a:=255;
    a:=a+1;
```

```
        Writeln(a);
        Readln;
    End.
```

- Với n = 16 bit, dải biểu diễn là [0 ÷ 65535]

0000 0000 0000 0000 = 0

0000 0000 0000 0001 = 1

0000 0001 0000 0000 = 255

1111 1111 1111 1111 = 65535

+ Kiểu dữ liệu tương ứng trong Turbo Pascal là kiểu Word

```
Program Kieuword;
Uses Crt;
Var
    a:Word;
    Begin
        a:=65535;
        a:=a+1;
        Writeln(a);
        Readln;
    End.
```

- Với n = 32 bit, dải biểu diễn là $[0, 2^{32} – 1]$
- Với n = 64 bit, dải biểu diễn là $[0, 2^{64} – 1]$

1.2.2. Số nguyên có dấu.

- Số bù

* Số bù 9 và bù 10 của hệ thập phân.

- Giả sử có một số nguyên thập phân A được biểu diễn bởi n chữ số thập phân khi đó ta có

Số bù 9 của A là: = $(10^n-1) – A$

Số bù 10 của A là: = $10^n – A$

-> Số bù mười := số bù 9 + 1

* Số bù 1 và bù 2 của hệ nhị phân.

- Giả sử có một số nguyên nhị phân A được biểu diễn bởi n bit khi đó ta có

Số bù 1 của A là:= $(2^n – 1) – A$

Số bù 2 của A là:= $2^n – A$

-> Số bù 2:= (số bù 1) +1

**** Nhận xét:***

+ Có thể tìm số bù một của A bằng cách đảo tất cả các bit của A

+ Số bù hai của A bằng số bù một của A + 1

+ A + số bù 2 của A = 0, nếu bỏ qua bit nhớ ra khỏi bit cao nhất

- Biểu diễn số nguyên có dấu thông qua số bù 2.

- Dùng n bit để biểu diễn số nguyên có dấu

$a_{n-1}a_{n-2} ... a_0$

- Với số dương

+ Bit $a_{n-1}= 0$

+ Các bit còn lại biểu diễn độ lớn của số dương đó

+ Dãy tổng quát của số dương có dạng: $0a_{n-2} ... a_0$

+ Giá trị của số dương: $A = \sum_{i=0}^{n-2} a_i 2^i$

+ Dải biểu diễn của số dương là: $[0, 2^n-1]$

- Với số âm: Được biểu diễn bằng số bù 2 của số dương tương ứng

+ Bit $a_{n-1}= 1$

+ Dãy tổng quát của số âm có dạng: $1a_{n-2} ... a_0$

+ Giá trị của số âm: $A = -2^n + \sum_{i=0}^{n-2} a_i 2^i$

+ Dải biểu diễn của số âm là: $[2^n-1, -1]$

-> Kết hợp lại dải biểu diễn của số nguyên có dấu n bit là: $[-2^n-1 \div 2^n-1]$

1.2.3. Mã hóa số nguyên thập phân bằng nhị phân BCD (Binary Coded Decimal)

- Dùng 4 bit để mã hóa từng chữ số thập phân từ 0 → 9 như sau

0 → 0000 3 → 0011 6 → 0110 9 → 1001

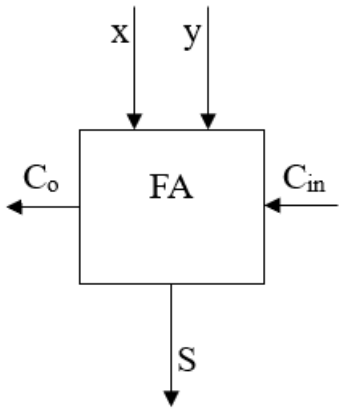
1 → 0001 4 → 0100 7 → 0111

- Có 6 tổ hợp không sử dụng: (1010, 1011, 1100, 1101, 1110, 1111)

1.3. Thực hiện các phép toán số học đối với số nguyên.

1.3.1. Bộ cộng.

- Bộ cộng 1 bit toàn phần (Full Adder)



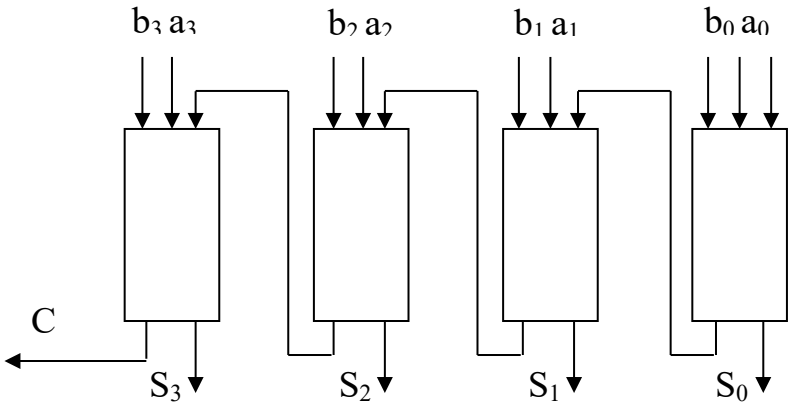
x	y	C_{in}	S	C_{out}
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

$S = x + y + C_{in}$

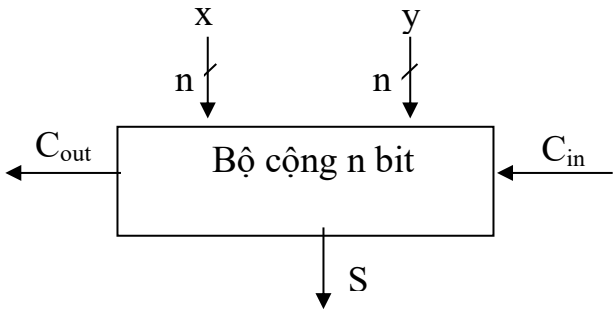
$C_{out} = x.y \oplus x.C_{in} \oplus y.C_{in}$

- Bộ cộng n bit

+ Bộ cộng 4 bit



+ Bộ cộng n bit



$S = S_{n-1}S_{n-2} \dots S_0$

$X = x_{n-1}x_{n-2} \dots x_0$

$Y = y_{n-1}y_{n-2} \dots y_0$

1.3.2. Cộng số nguyên không dấu.

- Nguyên tắc tổng quát: Để cộng 2 số nguyên không dấu n bit ta sử dụng 1 bộ cộng n bit.

+ Nếu tổng của 2 số ≤ $2^n - 1$ thì kết quả đầu ra S là đúng, khi đó $C_{out} = 0$

+ Nếu tổng của 2 số > $2^n - 1$ thì kết quả đầu ra S là sai, khi đó $C_{out} = 1$.

-> Đó là hiện tượng tràn nhớ ra ngoài (Carry Out)

```
Var
    x,y,s: Byte;
Begin
    X: = 197;
    Y: = 71;
    S: = x+y;
    Writeln(s);
    Readln(s);
End.
```

1.3.3. Cộng trừ số nguyên có dấu.

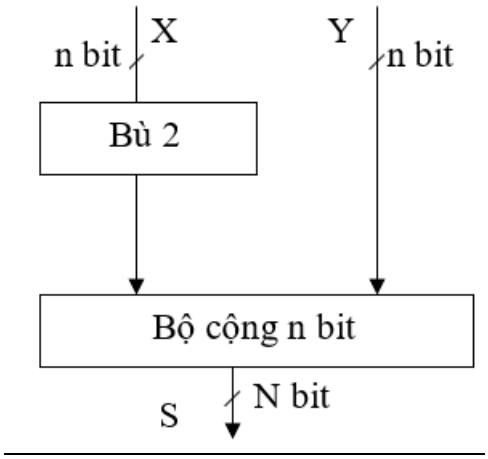
- Nguyên tắc tổng quát

+ Để cộng 2 số nguyên có dấu n bit, ta sử dụng bộ cộng n bit.

+ Để trừ 2 số nguyên có dấu n bit, ta sử dụng một bộ cộng n bit và một bộ lấy số bù 2 n bit.

X – Y = X + (-Y)

-> Phép đảo dấu trong máy tính thực chất là lấy bù 2.



- Trong cả 2 trường hợp, ta bỏ qua bit nhớ C_{out} ở bộ cộng (C_{out} không ảnh hưởng đến kết quả)

- Nếu tổng hoặc hiệu của 2 số nằm trong dải biểu diễn của số nguyên có dấu n bit $[-2^n, 2^{n-1}-1]$ thì kết quả ở đầu ra là đúng.

- Nếu tổng hoặc hiệu của 2 số nằm ngoài dải biểu diễn của số nguyên có dấu n bit $[-2^{n-1}, 2^{n-1}-1]$ thì kết quả ở đầu ra ở đầu ra S là sai $\rightarrow$ có xảy ra tràn số (Overflow)

Kết luận: Khi cộng 2 số nguyên có cùng dấu, kết quả có dấu ngược lại thì xảy ra hiện tượng tràn (Overflow)

1.3.4. Nhân số nguyên.

a) Nhân số nguyên không dấu

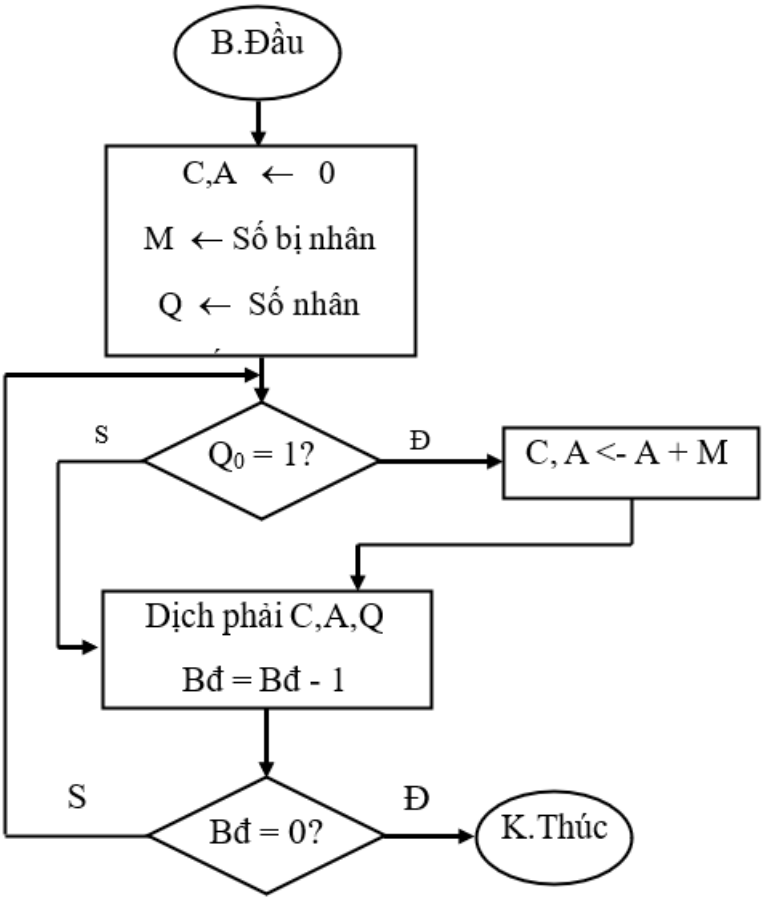
+ Các tích riêng phần được xác đinh như sau

+ Nếu bit của số nhân = 0, $\rightarrow$ tích riêng phần = 0

+ Nếu bit của số nhân = 1, $\rightarrow$ tích riêng phần bằng số bị nhân

+ Tích riêng phần tiếp theo được dịch trái 1 bit so với tích riêng phần trước đó

*** Lưu đồ thực hiện phép nhân số nguyên không dấu**



b) Nhân số nguyên có dấu.

- Sử dụng thuật giải nhân số nguyên không dấu

B1: Chuyển đổi số nhân và số bị nhân thành số nguyên không dấu tương ứng (coi là số không dấu)

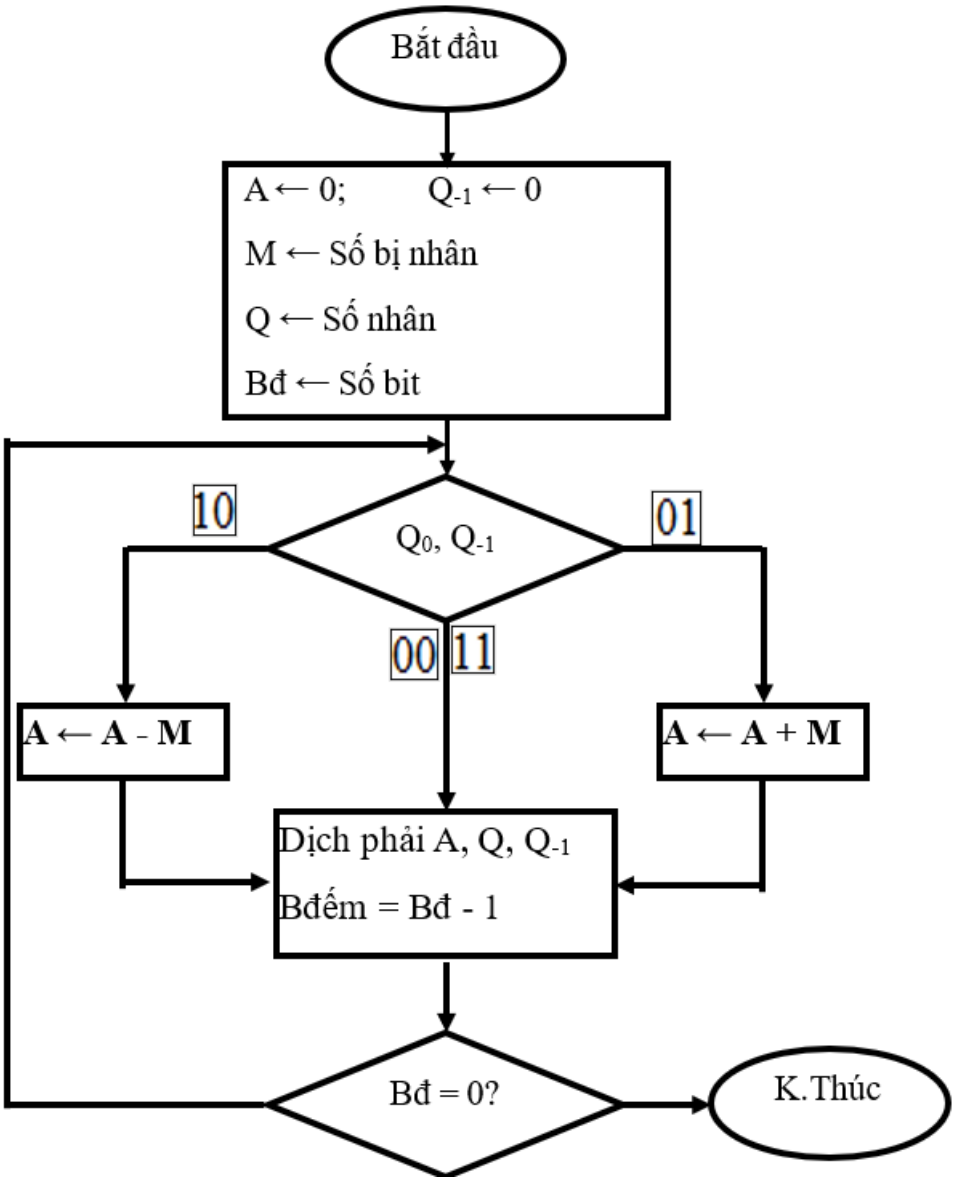
B2: Nhân 2 số bằng thuật giải nhân số nguyên không dấu

B3: Hiệu chỉnh dấu của tích

+ Nếu 2 thừa số ban đầu cùng dấu thì tích nhận được ở bước 2 là kết quả cần tính.

+ Nếu 2 thừa số ban đầu khác dấu nhau thì kết quả là số bù 2 của tích nhận được ở bước 2.

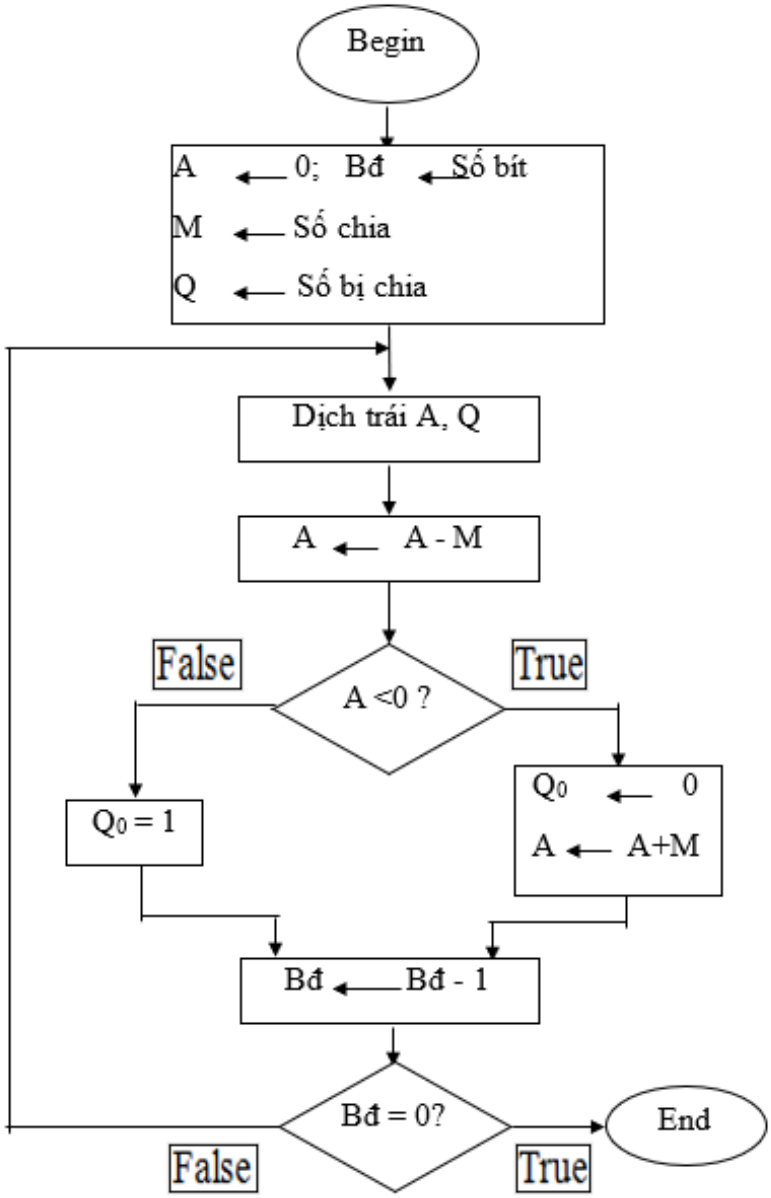
*** Nhân số nguyên có dấu theo thuật toán Booth**.



1.3.5 Chia số nguyên

a) Chia số nguyên không dấu.

* Lưu đồ thực hiện



b) Chia số nguyên có dấu.

B1: Coi số bị chia và số chia là số dương tương ứng

B2: Sử dụng thuật giải chia số nguyên không dấu để chia 2 số dương, kết quả nhận được là thương Q, số dư R đều dương.

B3: Hiệu chỉnh dấu kết quả theo nguyên tắc

Dấu của Số bị chia Q	Dấu của Số chia M	Dấu của Thương Q nhận được	Dấu của Số dư R
+	+	Giữ nguyên	Giữ nguyên
+	-	Đảo dấu	Giữ nguyên
-	+	Đảo dấu	Đảo dấu
-	-	Giữ nguyên	Đảo dấu
Q	**M**	**Q**	**R**

M: Là phần định trị (Mantissa)

R: Là cơ số tự nhiên, là hằng số (2, 10, 16)- (Radix)

E: Là phần mũ (Exponent)

Cơ số thường được sử dụng là cơ số 2 hoặc cơ số 10, M và E thường được biểu diễn theo kiểu số nguyên. Như vậy, với cơ số R xác định, thay vì phải lưu trữ giá trị của X người ta chỉ cần lưu trữ hai giá trị M và E.

Khi ta thực hiện phép toán với số dấu chấm động sẽ được tiến hành trên cơ sở các giá trị của phần định trị và phần mũ. Giả sử có hai số thực được biểu diễn bởi số dấu chấm động như sau:

$X1 = M1 \times R^{E1}$

$X2 = M2 \times R^{E2}$

Khi đó với phép toán số học đối với X1 và X2 ta được

$X1 + X2 = (M1 \times R^{E1-E2} + M2) \times R^{E2}$

$X1 - X2 = (M1 \times R^{E1-E2} - M2) \times R^{E2}$

$X1 \times X2 = (M1 \times M2) \times R^{E1+E2}$

$X1 / X2 = (M1 / M2) \times R^{E1-E2}$

→ Có nhiều chuẩn để biểu diễn khác nhau, nhưng chuẩn thông dụng gồm hai chuẩn định dạng dấu chấm trượt quan trọng đối với người lập trình là: Chuẩn

MSBIN (Microsoft Binary Format) của Microsoft và chuẩn IEEE (Institute of Electrical and Electronic Engineering). Cả hai chuẩn này đều dùng hệ đếm R = 2.

Chuẩn MSBIN

Được dùng trong phần mềm bản hãng, trong một bộ vi xử lý 16 bit, một số thực dấu chấm động ngắn cần có 32 bit để biểu diễn bản thân nó. Từ Msb → Lsb ta có cách phân bố như sau:

1 bit dấu phần mũ (bit 31), 7 bit giá trị phần mũ (bit 30 → 24), 1bit dấu phần định trị (bit23) và 23 bit giá trị phần định trị (bit 22 đến bit 0).

Giá trị của số thực MSBIN được tính như sau

$Y = (-1)^s.(1+M_1.2^{-1} + ... +M_n.2^{-n}).2^{E7...E0 - 129}$

Trong đó n là giá trị định trí cuối cùng của số thực (n =23 cho số thực ngắn –

Chú ý: Giá trị đầu tiên M_0 luôn mặc định là 1.

- Dùng 32 bit để biểu diễn số thực, được số thực ngắn: $-3,4.10^{38} < R < 3,4.10^{38}$

31	30 23	22 0
S	E7 - E0	Định trị (M1 - M23)

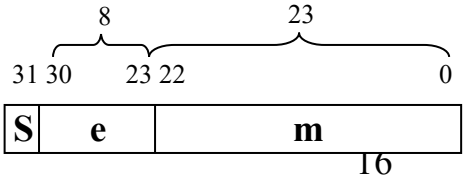
- Dùng 64 bit để biểu diễn số thực, được số thực dài: $-1,7.10^{308} < R < 1,7.10^{308}$

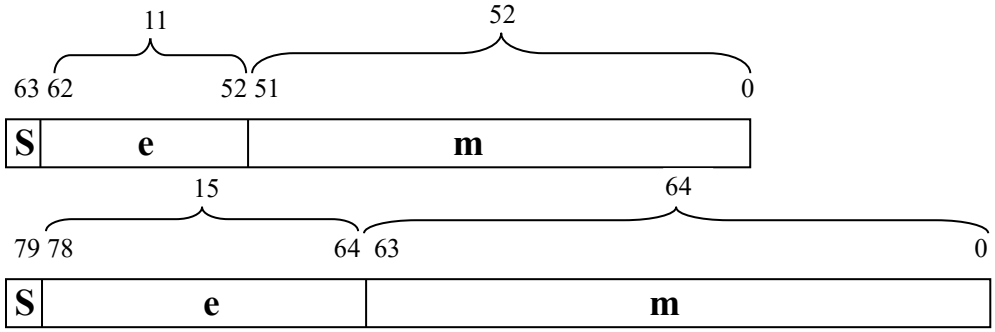
63	62 52	51 0
S	E10 - E0	Định trị (M1 - M52)

Chuẩn IEEE 754/85

Là chuẩn được mọi hãng chấp nhận và được dùng trong bộ xử lý toán học của Intel. Bit dấu nằm tại vị trí cao nhất. Kích thước phần mũ và khuôn dạng phần định trị thay đổi theo từng loại số thực và có ba dạng chính sau đây

- Dạng có độ chính xác đơn, 32 bit
- Dạng có độ chính xác kép, 64 bit
- Dạng có độ chính xác kép mở rộng, 80 bit





Hình mô hình chính của dữ liệu số dấu chấm động theo chuẩn IEEE 754/85

- Với mô hình như trên ta có

- S: Là bit dấu, S = 0 là số dương, ngược lại S = 1 là số âm.

- e: Là mã lệch (excess) của phần mũ E, hay phần mũ E được xác định từ e với công thức sau:

E = e – b với (b là độ lệch – bias：Nghiêng, độ lệch)

- Dạng 32 bit: b = 127, $\rightarrow$ E = e - 127

- Dạng 64 bit: b = 1023, $\rightarrow$ E = e – 1023

- Dạng 80 bit: b = 16383, $\rightarrow$ E = e – 16383

- m: Là các bit phần lẻ của phần định trị, với phần định trị được ngầm định như sau:

M = 1.m

=> Giá trị của số thực tương ứng được xác định bởi công thức sau đây:

$$\mathbf{X = (-1)^S \times 1.m \times 2^{e-b}}$$

* Một số quy ước như sau:

- Nếu tất cả các bit của e đều bằng 0, các bit của m đều bằng 0, thì N = ± 0

- Nếu tất cả các bit của e đều bằng 1, các bit của m đều bằng 0, thì N = ± ∞

- Nếu tất cả các bit của e đều bằng 1, các bit của m đều bằng 0, thì N không phải là số (Not a number - NaN).