

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



BÀI TẬP LỚN KIẾN TRÚC MÁY TÍNH
Đề 4: Nhân 2 số thực chuẩn IEEE 754 chính xác đơn.
Lớp L01-N059

Họ và tên	MSSV
Hồng Phi Trường	2313703
Nguyễn Công Vũ Hoàng	2211087

Mục lục

1	Giới thiệu đề tài	2
2	Giải pháp hiện thực	2
3	Hiện thực chi tiết và đánh giá chương trình	4
3.1	Hiện thực chương trình	4
3.2	Đánh giá chương trình	5
4	Tài liệu tham khảo	9

1 Giới thiệu đề tài

Đề 4

Nhân 2 số thực chuẩn IEEE 754 chính xác đơn. Viết chương trình thực hiện phép nhân 2 số thực chuẩn IEEE 754 chính xác đơn mà không dùng các lệnh tính toán số thực của MIPS. Dữ liệu đầu vào đọc từ file lưu trữ dạng nhị phân trên đĩa FLOAT2.BIN (2 tri x 4 bytes = 8 bytes).

số thực chuẩn IEEE 754

Số thực dấu chấm động là cách biểu diễn các số thực trong máy tính được chuẩn hóa bởi tổ chức IEEE (Institute of Electrical and Electronic Engineers) vào năm 1985 gọi là chuẩn IEEE 754. Chuẩn IEEE 754 gồm nhiều dạng như 16bit, 32bit, 64bit,... Phổ biến là hai dạng 32bit và 64bit. Biểu diễn số thực dấu chấm động theo dạng chuẩn IEEE 754 dùng 32bit gọi là biểu diễn theo dạng "chính xác đơn" trong khi biểu diễn 64bit được gọi là biểu diễn theo dạng "chính xác kép". Bảng 1 mô tả cách biểu diễn số thực dấu chấm

	Phần dấu (S)	Phần mũ (E)	Phần phân số (F)
Chính xác đơn	1	8	23
Chính xác kép	1	11	52

Bảng 1: Cấu trúc định dạng dấu chấm động IEEE 754

động theo định dạng IEEE 754, gồm 3 phần: Phần S(sign) biểu diễn dấu, phần E biểu diễn phần mũ và F biểu diễn phần phân số. Đối với số thực dấu chấm động chính xác đơn thì ba phần trên lần lượt là 1bit, 8bit và 23bit. Phần số thực dấu chấm động chính xác kép thì con số đó lần lượt là 1, 11 và 52.

$$\text{Giá trị hệ thập phân} = (-1)^S \times (1 + \text{phan so}) \times 2^{so \text{ mu} - \text{bias}}$$

với bias = 127(chính xác đơn)

$$\text{Giá trị phân số}_{\text{chính xác đơn}} = \sum_{i=-1}^{-23} x_i \times 2^i$$

2 Giải pháp hiện thực

Phép nhân hai số thực dấu chấm động theo định dạng IEEE 754 (X_1, X_2) được thực hiện theo công thức như sau:

$$X_3 = X_1 \times X_2 = (-1)^{S_1+S_2} \times (1 + F_1) \times (1 + F_2) \times 2^{E_1+E_2-2\text{bias}}$$

Trong đó:

S_1, S_2 là bit dấu của X_1, X_2

F_1, F_2 là phần phân số

E_1, E_2 là phần số mũ

Chi tiết thực hiện như sau:

1. Tính số mũ của hai thừa số:

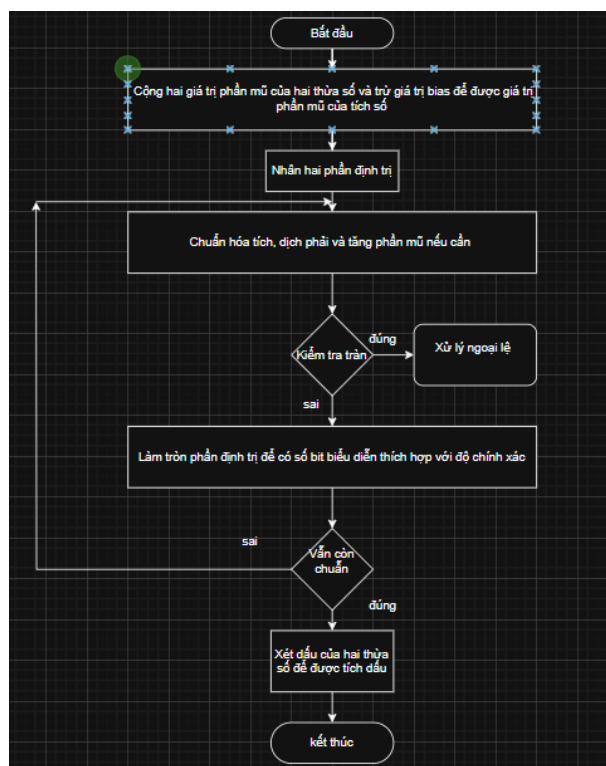
$$E = E_1 + E_2 - bias$$

2. Nhân hai phần định trị (fraction), ta nhân hai số trong hệ nhị phân. Đối với trường hợp độ chính xác đơn thì phần định trị của hai thừa số sẽ có kích thước 24 bit (thêm bit 1 vào trước phần phân số).

3. Chuẩn hóa, nếu kích thước kết quả phép nhân là 47 bit thì không cần hiệu chỉnh, nếu kết quả có kích thước là 48 bit thì cần phải dịch phải kết quả này 1 bit và tăng phần mũ lên 1 đơn vị.

4. Kiểm tra tràn số. Nếu E_3 (đã chuẩn hóa nếu cần) > 255 hoặc < 0 chương trình sẽ dẫn đến ngoại lệ.

5. xác định dấu của phép nhân. bit dấu của phép nhân là 0 khi S1 bằng với S2 và bằng 1 nếu S1 khác S2. 6. Kết quả của phép nhân là một chuỗi dài 32 bit với 3 trường số biểu diễn theo chuẩn IEEE 754, dưới đây là lưu đồ giải thuật.



Hình 1: Giải thuật thực hiện phép nhân

Ví dụ

X1 =	S	E	F
5.5	0	10000001	011000000000000000000000

X2 =	S	E	F
4.25	0	10000001	000100000000000000000000

Thực hiện phép nhân của 5.5 với 4.25, bởi vì hai thừa số trong phép nhân được lưu trữ bằng 32 bit nên cả hai thuộc dạng chính xác đơn.

Giá trị phần mũ được tính như sau.

$$1000\ 0001 + 1000\ 0001 - 0111\ 1111 = 1000\ 0011$$

Hai phần định trị của hai thừa số có kích thước 24 bit lần lượt là:

1011 0000 0000 0000 0000 0000

1000 1000 0000 0000 0000 0000

Kết quả của phép nhân hai thừa số trên là:

101 1101 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000

Kết quả này có kích thước là 47 bit nên không cần thực hiện dịch và điều chỉnh kết quả mũ. Giá trị phần định trị của phép nhân là:

1011 1011 0000 0000 0000 0000

Bởi vì cả hai đều là số dương nên tích số cũng là một số dương, bit dấu của S là 0. Suy ra kết quả phép nhân là:

$$0\ 10000011\ 011101100000000000000000 = 0x41BB0000 = 23.375_{10}$$

3 Hiện thực chi tiết và đánh giá chương trình

3.1 Hiện thực chương trình

Có thể dùng Python để tạo ra một file FLOAT2.BIN với đầu vào mong muốn

1. Mở và đọc file nhị phân chứa 2 số thực dạng IEEE-754 (32-bit), chương trình mở file nhị phân và đọc 4byte vào cho mỗi số thực vào bộ nhớ
2. Tách mỗi số thực thành: Bit dấu (sign), Exponent (8-bit) và Fraction (23-bit)
3. Thực hiện nhân hai số thực theo quy tắc IEEE-754:
 - XOR 2 bit dấu để lấy dấu kết quả.

- Cộng hai số mũ exponent, rồi trừ bias (127), sau đó cộng lại bias.
 - Nhân 2 phần fraction (thêm bit 1 ẩn vào).
 - Chuẩn hóa kết quả: nếu vượt quá 24 bit thì điều chỉnh exponent.
4. Gộp lại thành số thực IEEE-754 (32-bit).
 5. In kết quả ra màn hình dưới dạng số thực và nhị phân.
 6. Xử lý các trường hợp đặc biệt như nhân với 0, overflow và underflow.

3.2 Đánh giá chương trình

Để tính thời gian thực hiện chương trình, ta sử dụng công thức

$$CPU_{time} = \frac{IC \times CPI}{CR}$$

IC là tổng số lệnh

CPI là chu kì thực thi mỗi lệnh, với lệnh chuẩn CPI=1

CR: clock rate, chu kì trên giây (1 CR = 1GHz)

Dưới đây là bảng tổng kết:



Thừa số 1	Thừa số 2	Kết quả	Lệnh R	Lệnh I	Lệnh J	Thời gian chạy (ns)
1.5	-2.25	-3.375	63	59	4	126
0.0	-34.9	0.0	17	38	0	55
-2.88325664E8	-7065.0	2.03702075E12	63	59	4	126
60.0	-5.0	-300.0	66	64	4	134
0.3	-0.3	-0.09	63	59	4	126
1.0E-38	1.0E-38	Xay ra Underflow	31	46	2	79
1.0E38	1.0E26	Xay ra Overflow	30	44	2	76
-6718.0	184900.0	-1.24215808E9	66	64	4	134
34.65	12.5675	435.4639	63	59	4	126
7.0	4.0	28.0	63	59	4	126
-3.78	-67.0	253.26	63	59	4	126
0.0	0.0	0.0	17	38	0	55
12E12	-9876.74	-1.18520877E17	63	59	4	126
200E10	23.987	4.7973996E13	66	64	4	134
800.0	700.0	560000.0	66	64	4	134
-10.0	-10.0	100.0	63	59	4	126
1.4E21	1.4E21	Xay ra Overflow	30	44	2	76
1.0E-37	1.0E-37	Xay ra Underflow	31	46	2	79
123.9875	23.654	2932.8003	66	64	4	134
-764.0	-976.0	745664.0	66	64	4	134

Bảng 2: Kết quả nhân hai số thực và thông số thực thi

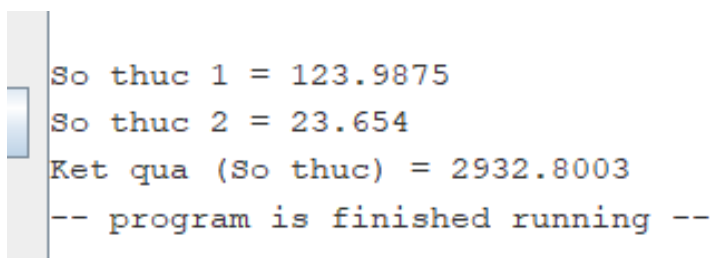
Nhận xét

Các trường hợp chạy nhanh nhất là các trường hợp nhân với 0 với 55ns

Trường hợp chạy lâu nhất là 134ns

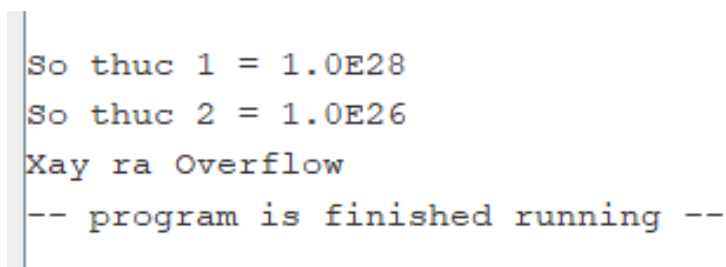
Trường hợp các input quá nhỏ hoặc quá lớn có thể dẫn đến kết quả đầu ra không chính xác, nặng hơn có thể gây ra underflow hoặc overflow

Một số hình ảnh output



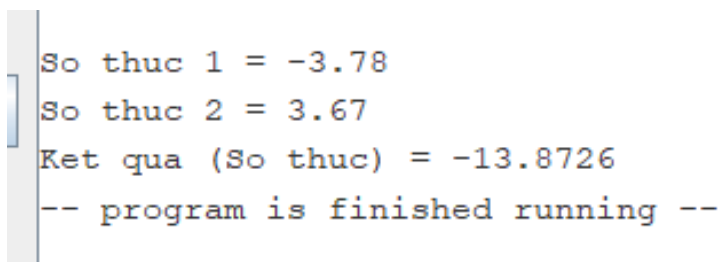
```
So thuc 1 = 123.9875
So thuc 2 = 23.654
Ket qua (So thuc) = 2932.8003
-- program is finished running --
```

Hình 2: output



```
So thuc 1 = 1.0E28
So thuc 2 = 1.0E26
Xay ra Overflow
-- program is finished running --
```

Hình 3: output overflow



```
So thuc 1 = -3.78
So thuc 2 = 3.67
Ket qua (So thuc) = -13.8726
-- program is finished running --
```

Hình 4: output


```
So thuc 1 = 1.0E-37
So thuc 2 = 1.0E-37
Xay ra Underflow
-- program is finished running --
```

Hình 5: output underflow

```
So thuc 1 = 0.0
So thuc 2 = 0.0
Ket qua (So thuc) = 0.0
-- program is finished running --
```

Hình 6: output 0



4 Tài liệu tham khảo

1. *MIPS32™ Architecture For Programmers Volume II: The MIPS32™ Instruction Set*
2. *Computer Organization and Design. THE HARDWARE/SOFTWARE INTERFACE* David A. Patterson