ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA KHOA HỌC VÀ KĨ THUẬT MÁY TÍNH

KIẾN TRÚC MÁY TÍNH

BÁO CÁO BÀI TẬP LỚN

ĐỀ TÀI 3: NHÂN 2 SỐ THỰC

GVHD: Thầy Nguyễn Thanh Bình

Thầy Võ Tấn Phương

NHÓM: Lớp: L08

SINH VIÊN THỰC HIỆN:

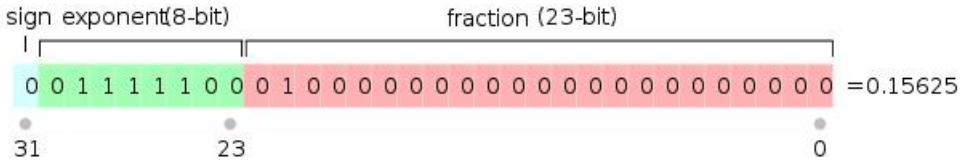
Huỳnh Ngọc Tấn - 1813952

TP. HỒ CHÍ MINH, NĂM 2020

Mục lục

1. Số thực dạng chuẩn IEEE 754 dạng chính xác đơn

Số thực được lưu trong máy tính dưới dạng 32 bit gồm: 1 bit dấu, 8 bit xác định mũ, 23 bit phần định trị.
Bit dấu: 0 biểu thị số dương, 1 biểu thị số âm.
Bit xác định số mũ: 8 bits xác định giá trị mũ (số mũ +127).
Bit định trị: số thực sẽ chuyển về dạng 1.m với m sẽ được chuyển sang nhị phân và lưu bằng 23 bits.



Hình 1: Biểu diễn số thực dạng chính xác đơn

2. Giải thuật nhân hai số thực

• Tách phần bit dấu: bằng cách dịch 31 bits sang phải. Xong xác định bit dấu kết quả bằng cách or hai kết quả của bit dấu sau khi dịch bit, hai số âm hoặc hai số dương thì kết quả là số dương, một số âm một số dương kết quả là số âm.

• Tách phần xác định mũ: bằng cách dịch 1 bit sang trái sau đó dịch 24 bit sang phải. Sau đó trừ mỗi phần số mũ cho 127 và cộng hai số mũ sau khi trừ. Kết quả thu được sau khi cộng thêm cho 127 là phần xác định mũ của kết quả.

• Tách phần định trị: dịch phải 9 bit sau đó dịch trái 9 bit để loại bỏ 9 bit đầu tiên, kết quả ta thu được là phần 23 bit xác định giá trị của số Thực.

3 Thống kê số lệnh, các loại lệnh mỗi nhóm

3.1 R-type

*Op*6	*Rs*5	*Rt*5	*Rd*5	*Sa*5	*Funct*6

Bảng 1: R-type format

• Op: mã phép toán(opcode):

– Cho biết lệnh làm phép toán gì.

• Funct: function code – mở rộng opcode:

– Có thể có 26 = 64 functions có thể mở rộng cho một opcode.

– MIPS sử dụng opcode 0 để định nghĩa lệnh loại R-type.

• Ba thanh ghi toán hạn (Register Operand):

– Rs, Rt: Hai toán hạn nguồn.

– Rd: Toán hạn đích chứa kết quả.

– Sa: Quy định số bit dịch trong các lệnh dịch.

Các lệnh R-type đã sử dụng:

TT	**Tên lệnh**	**Chức Năng**
1	mfc1 $t0, $f0	Gán giá trị thực từ thanh ghi $f0 vào $t0
2	mfc1 $t1, $f0	Gán giá trị thực từ thanh ghi $f0 vào $t1
3	srl $s0, $t0, 31	Dịch bit để lấy phần dấu số thực
4	sll $s2, $s2, 31	Dịch bit sau khi lấy phần dấu
5	or $s2, $s0, $s1	Or hai thanh ghi
6	mult $t4, $t5	Nhân thanh ghi $t4 với thanh ghi $t5, ghi vào thanh ghi Hi-Lo
7	mfhi $s3	copy giá trị thanh ghi Hi vào $s3
8	mflo $s4	copy thanh ghi Lo vào $s4
9	srl $s5, $s3, 15	Dịch thanh ghi $s3 15 bit rồi gán vào thanh ghi $s5
10	sll $s6, $s3, 31	Dịch $s3 31 bit rồi gán vào $s6
11	srl $s4, $s4, 1	Dịch phải $s4 1bit rồi gán lại vào $s4
12	add $s4, $s4, $s6	Cộng $s4 với $s6 rồi gán vào lại $s4
13	srl $s3, $s3, 1	Dịch phải $s3 1 bit rồi gán vào lại $s3
14	mtc1 $t7, $f2	Chuyển nội dung thanh ghi $t0 vào thanh ghi $f0
15	mov.s $f12, $f2	Di chuyển giá trị trong thanh ghi $f12 sang thanh ghi $f2

Bảng 2: Bảng liệt kê các lệnh R-type

3.2 I-type

*Op*6	*Rs*5	*Rt*5	*Immediate*16

Bảng 3: I-type format

• Op: mã phép toán (opcode)
• Hằng số 16-bit được lưu trong lệnh:
– Rs thanh ghi toán hạn nguồn
– Rt thanh ghi đích (destination)

Các lệnh I-type đã sử dụng:

TT	Tên lệnh	Chức năng
1	la $a0, str_in	load địa chỉ label str_in vào thanh ghi $a0
2	addi $v0,$zero, 6	gán giá trị 6 vào thanh ghi $v0
3	beq $s5, 0, man_pro	Rẽ nhánh

Bảng 4: Bảng liệt kê các lệnh I-type

3.3 J-type

- Định dạng J-type áp dụng cho các lệnh nhảy không điều kiện (uncon ditional jump,giống như lệnh goto):
  j label # jump to label

. . .

*Op*6	*Immediate*26

Bảng 5: J-type format

4 Hiện thực nhân hai số thực chuẩn IEEE 754 trên MIPS

4.1 Hiện thực code MIPS

```
.data
str_in: .asciiz "Enter the multiplicant: "
str_in1: .asciiz "Enter the multiplier: "
str_out: .asciiz "Product: "
.text
Enter:
# Enter the multiplicant
la $a0, str_in
add $v0, $0, 4
syscall
li $v0, 6
syscall
mfc1 $t0, $f0 # $t0 =multiplicant

#Enter the multiplier
la $a0,str_in1
add $v0, $0, 4
syscall
li $v0, 6
syscall
mfc1 $t1, $f0 #$t1 s= multiplier

# Sign of product
Get_sign:
srl $s0, $t0, 31 #sign of multiplicant
srl $s1, $t1, 31 #sign of multiplier
xor $s2, $s0, $s1 #sign of product
sll $s2, $s2, 31 # sign is MSB

# function get exponent of multiplicant, multiplier
Get_exponent:
sll $t2, $t0, 1    #shift left $t0 1 bit
srl $t2, $t2, 24 #shift right $t2 24bit
sll $t3, $t1, 1    #shift left $t1 1 bit
srl $t3, $t3, 24 #shift right $t3 24bit
#add 2 exponent
Add_exponent:
add $t6, $t2, $t3
addi $t6, $t6,-127 # t6= exponent of product
```

```
#function get mantissa of multiplicant, multiplier
Get_mantissa:
sll $t4,$t0, 9 #shift left $t0 9 bit
srl $t4, $t4 9 #shift right $t4 bit
addi $t4, $t4, 8388608 #add bit 1 to front of $t4
sll $t5,$t1, 9 #shift left $t1 9 bit
srl $t5, $t5, 9 #shift right $t5 bit
addi $t5, $t5, 8388608 #add bit 1 to front of $t5

#add 2 mantissa
Mul_mantissa:
mult $t4, $t5 # mul 2 mantissa and store to Hi-Lo
mfhi $s3 #copy Hi to $s3
mflo $s4 #copy Lo to $s4
srl $s5, $s3, 15 #shift right $s3 15 bit
beq $s5, 0, man_pro #if $s3 is 14 bit then man_pro
sll $s6, $s3, 31 #else $s3 is 15 bit then process
srl $s4, $s4, 1 #
add $s4, $s4, $s6 #$s4 is Lo register
srl $s3, $s3, 1 #$s3 is Hi register
addi $t6, $t6, 1 #add exponent 1

#process mantissa for product
man_pro:    #get 23 bit MSB for mantissa
sll $s3, $s3, 18
srl $s3, $s3, 9 # get 14bit of Hi
srl $s4, $s4, 23 #get 9 bit for Lo
add $s4, $s3, $s4 #mantissa of product

#connect 3 element and return product $f2
Return_product:
sll $t6, $t6, 23 #
add $t7, $s2, $t6
add $t7, $t7, $s4
mtc1 $t7, $f2
#print result
Print_out:
la $a0, str_out
add $v0, $0, 4
syscall
mov.s $f12, $f2
li $v0, 2
syscall
```

4.2 Kiểm tra các testcase

4.2.1 Xét test case 1 (61 lệnh)

Xem xét nhân 2 số thực: 135.14 và 15.59

```
Enter the multiplicant: 135.14
Enter the multiplier: 15.59
Product: 2106.8325
-- program is finished running (dropped off bottom) --
```

4.2.2 Xét test case 2 (56 lệnh)

Xem xét nhân 2 số thực: 16 và 168.87

```
Enter the multiplicant: 16
Enter the multiplier: 168.87
Product: 2701.92
-- program is finished running (dropped off bottom) --
```

4.2.3 Xét test case 3 (56 lệnh)

Xem xét nhân 2 số thực: 132.52 và 13

```
Enter the multiplicant: 132.52
Enter the multiplier: 13
Product: 1722.76
-- program is finished running (dropped off bottom) --
```

4.2.4 Xét test case 4 (61 lệnh)

Xem xét nhân 2 số thực: 175 và 120

```
Enter the multiplicant: 175
Enter the multiplier: 120
Product: 21000.0
-- program is finished running (dropped off bottom) --
```

4.2.5 Xét test case 5 (61 lệnh)

Xem xét nhân 2 số thực: (bị vượt giới hạn)

```
Enter the multiplicant: 12344445262222
Enter the multiplier: 4115111222
Product: 5.079876E22
-- program is finished running (dropped off bottom) --
```

4.2.6 Xét test case 6 (61 lệnh)

Xem xét nhân 2 số thực:-14.15 và -215.82

```
Enter the multiplicant: -14.15
Enter the multiplier: -215.82
Product: 3053.8528
-- program is finished running (dropped off bottom) --
```

4.2.7 Xét test case 7 (56 lệnh)

Xem xét nhân 2 số thực: -17.52 và 158.95

```
Enter the multiplicant: -17.52
Enter the multiplier: 158.95
Product: -2784.804
-- program is finished running (dropped off bottom) --
```

4.2.8 Xét test case 8 (61 lệnh)

Xem xét nhân 2 số thực: 15.96 và -289.32

```
Enter the multiplicant: 15.96
Enter the multiplier: -289.32
Product: -4617.547
-- program is finished running (dropped off bottom) --
```

4.2.9 Xét test case 9 (61 lệnh)

Xem xét nhân 2 số thực: -198 và -1998

```
Enter the multiplicant: -198
Enter the multiplier: -1998
Product: 395604.0
-- program is finished running (dropped off bottom) --
```

4.2.10 Xét test case 10 (56 lệnh)

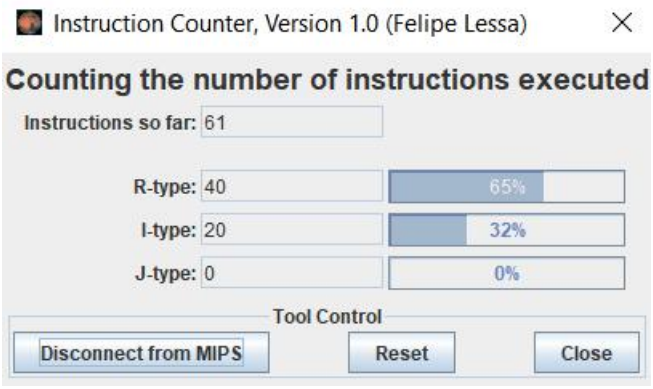
Xem xét nhân 2 số thực: -553555 và - 35447

```
Enter the multiplicant: -553555
Enter the multiplier: -35447
Product: 1.96218634E10
-- program is finished running (dropped off bottom) --
```

4.3 Thời gian hiện thực chương trình

Thời gian chạy của chương trình trên máy tính MIPS có tần số 2 GHz:

Những case có 61 lệnh:

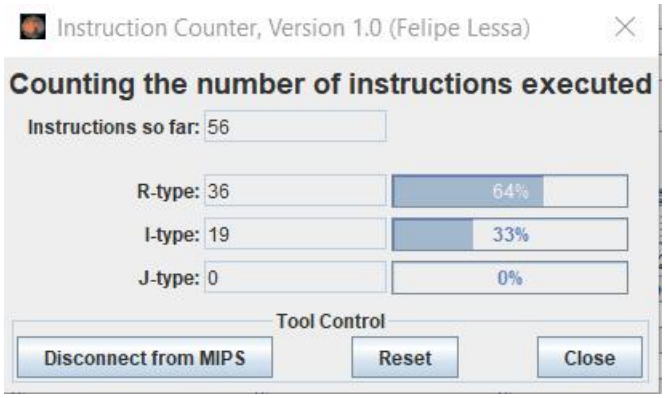


$$Time = CPI * IC * CycleTime$$

$$= 1 * 61 * \frac{1}{2*10^9}$$

$$= 3.05*10^{-8} \text{ s}$$

Những case có 56 lệnh:



$$
\begin{aligned}
Time &= CPI * IC * CycleTime \\
&= 1 * 56 * \frac{1}{2*10^9} \\
&= 2.8*10^{-8} \text{ s}
\end{aligned}
$$