

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



Kiến Trúc Máy Tính

Báo cáo Bài tập lớn 1

Đề tài 3: Nhân hai số thực

Cho 2 số thực dạng chuẩn (Standard Floating Point IEEE 754) A và B với độ chính xác đơn (32 bit). Viết thủ tục cộng (trừ) hai số A, B. Giả sử tập lệnh hợp ngữ MIPS không hỗ trợ phép tính dấu chấm di động.

Giáo viên hướng dẫn: Nguyễn Xuân Minh

Danh sách các thành viên

STT	Họ và tên	MSSV
1	Phạm Tấn Đại	1710929
2	Cao Thành Nhân	1710214
3	Lê Thị Thanh Thảo	1713177

TP. HCM THÁNG 11/2018

Mục lục

I. Cơ sở lý thuyết	2
1. Lý thuyết	2
2. Ý tưởng giải thuật	2
3. Các bước giải thuật	2
4. Flow chart	4
II. Hiện thực bằng hợp ngữ assembly MIPS	5
1. Thống kê tập lệnh, loại lệnh	5
2. Chạy chương trình	6
3. Các Test Case và Thời gian chạy của chương trình	6
3.1 Cách tính thời gian chạy	6
3.2 Các Test Case	6

I. Cơ sở lý thuyết

1. Lý thuyết

Biểu diễn của số thực F dạng chuẩn (Standard Floating Point):

<i>Sign-bit (S)</i> 1 bit	<i>Exponent (E)</i> 8 bits	<i>Mantissa (M)</i> 23 bits
------------------------------	-------------------------------	--------------------------------

Khi đó:

$$F = (-1)^S \times 2^{E-127} \times 1.M$$

Thực hiện phép nhân 2 số thực dạng chuẩn A và B:

$$A = (-1)^{S_A} \times 2^{E_A-127} \times 1.M_A$$

$$B = (-1)^{S_B} \times 2^{E_B-127} \times 1.M_B$$

Thu được kết quả là: $A \times B = (-1)^S \times 2^{E_A+E_B-254} \times (1.M_A \times 1.M_B)$

=> Biểu diễn của kết quả có:

- Bit dấu (*sign-bit*) là kết quả của phép **xor** S_A và S_B .
- *Exponent* $E = E_A + E_B - 127$ và có thể tăng lên 1 tùy thuộc vào kết quả của phép nhân M_A và M_B .
- *Mantissa* lấy ra từ kết quả phép nhân $1.M_A \times 1.M_B$.

2. Ý tưởng giải thuật:

Xử lý riêng trên từng phần (*sign-bit*, *exponent*, *mantissa*) rồi gộp các phần lại thành kết quả biểu diễn số thực hoàn chỉnh.

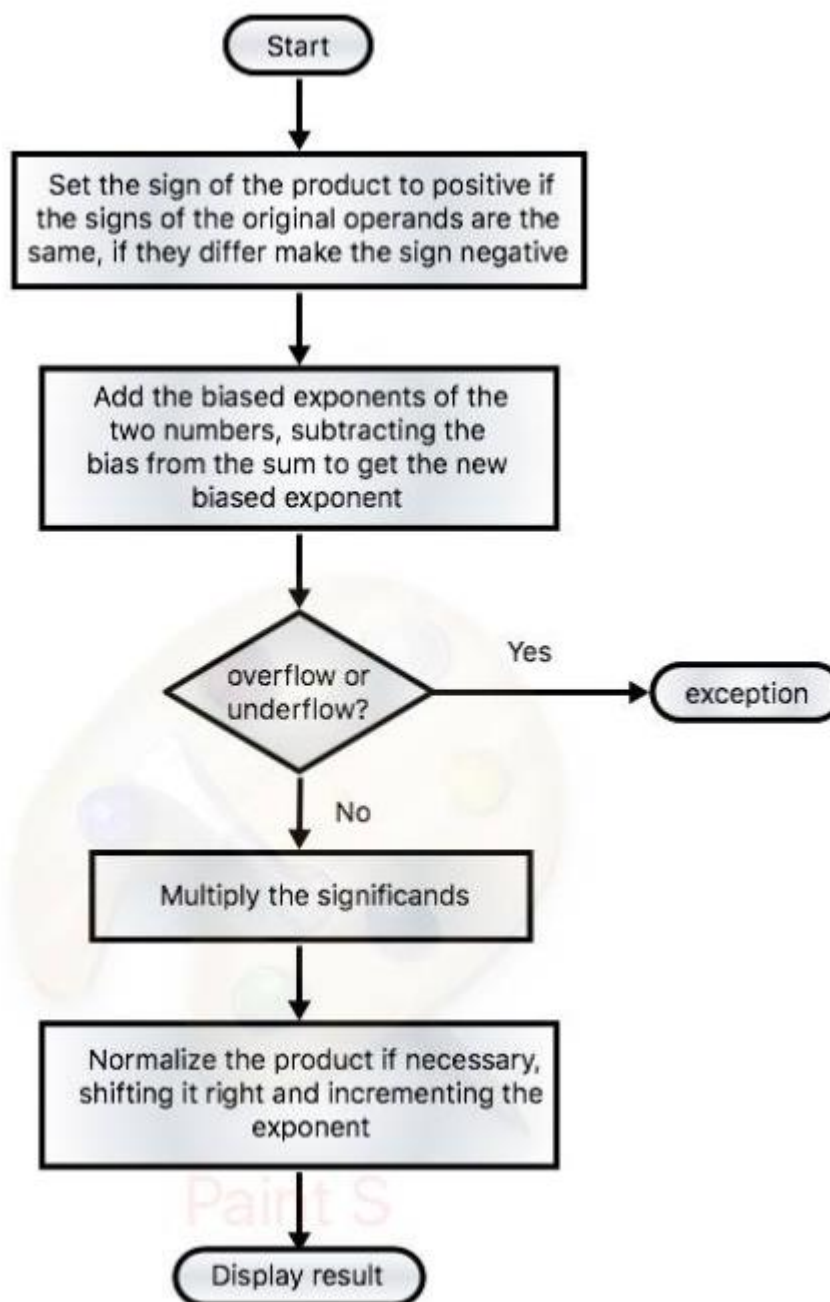
3. Các bước giải thuật:

- ✓ **Bước 1:** Dùng phép **andi** chia số được biểu diễn thành các phần nhỏ để xử lý:
 - *Phần bit dấu:* là kết quả của phép **xor** và được lưu trong \$t0.
 - *Phần exponent:* dùng phép **andi** với **0x7F800000** và **srl** để tách đúng 8 bit và xử lý cộng, lưu vào \$t1. Nếu $\$t1 < 0$ hoặc $\$t1 > 254$ thì đến bước 3 để xử lý tràn.
 - *Phần mantissa:* dùng phép **andi** với **0x007FFFFFFF** để tách ra và sau đó dùng phép **ori** với **0x00800000** để thêm số 1 sau đó nhân lại bằng **multu**, lưu các bit có trọng số cao vào \$t6. Đến bước 2 để xử lý.
- ✓ **Bước 2:** Xét hai trường hợp (phép nhân ở phần mantissa có nhớ và không nhớ):
 - *Có nhớ:* tăng exponent lên 1 và lấy **15 bits HI - 8 bits LO**.
 - *Không nhớ:* giữ nguyên exponent và lấy **14 bits HI - 9 bits LO**.

Lưu kết quả mantissa vào \$t2 và đi đến bước 4.

- ✓ **Bước 3:** Xử lý tràn trên (*overflow*) và tràn dưới (*underflow*):
 - *Overflow*: Gán exponent cho **255** (inf) và đến bước 4.
 - *Underflow*: Đi đến bước 5 – lưu kết quả là 0.
- ✓ **Bước 4:** Merge ghép các thanh ghi \$t0, \$t1, \$t2 bằng 2 phép **or** để được biểu diễn số thực hoàn chỉnh ở \$s2 và lưu kết quả vào biến nhớ.
- ✓ **Bước 5:** Lưu kết quả vào \$f0 và trả về.

4. Sơ đồ khối (Flow chart)



II. Hiện thực bằng hợp ngữ assembly MIPS

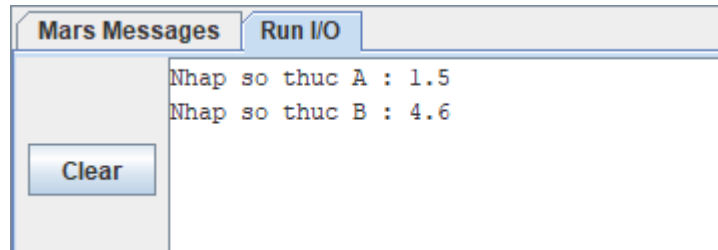
1. Thống kê tập lệnh loại lệnh

Lệnh	Loại Lệnh	Số Lượng
swc1	I type	2
lw	I type	3
andi	I type	10
subi	I type	1
ori	I type	2
addi	I type	1
sw	I type	1
lwc1	I type	1
sll	R type	3
xor	R type	1
addu	R type	1
srl	R type	3
multu	R type	1
mflo	R type	1
mfhi	R type	1
or	R type	3
jr	R type	1
syscall	R type	7
j	J type	4
jal	J type	1
li	Pseudo	8
la	Pseudo	5
mov.s	Pseudo	1
beqz	Pseudo	1
blt	Pseudo	1
bgt	Pseudo	1

Bảng 1: Bảng thống kê lệnh trong hiện thực code.

2. Chạy chương trình

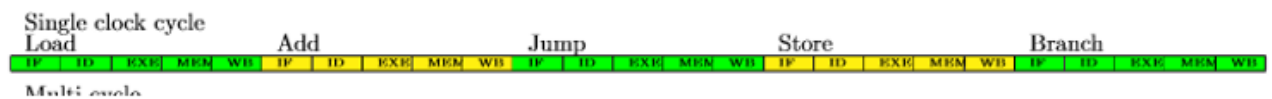
Mở đầu chương trình , máy sẽ yêu cầu bạn nhập 2 số thực A và B.



3. Các Test Case và Thời gian chạy của chương trình

3.1 Cách tính thời gian chạy

- Dùng công cụ Tools → Instruction Counter của ứng dụng MARS để thống kê số câu lệnh.
- Giả sử chương trình chạy trên hệ thống Single clock cycle.
- Mỗi câu lệnh thực thi 5 bước, mỗi câu lệnh thực thi trong 1 chu kỳ đơn.

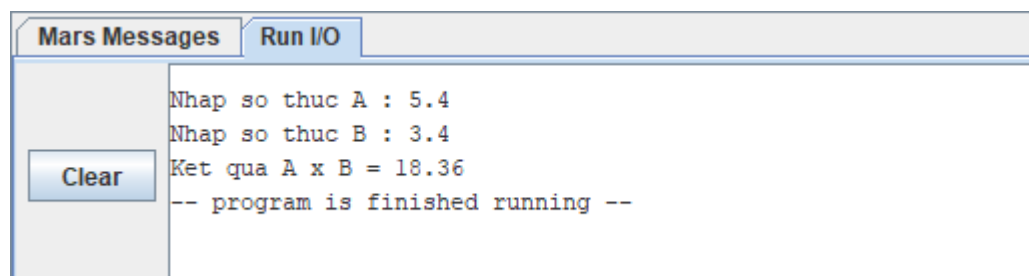


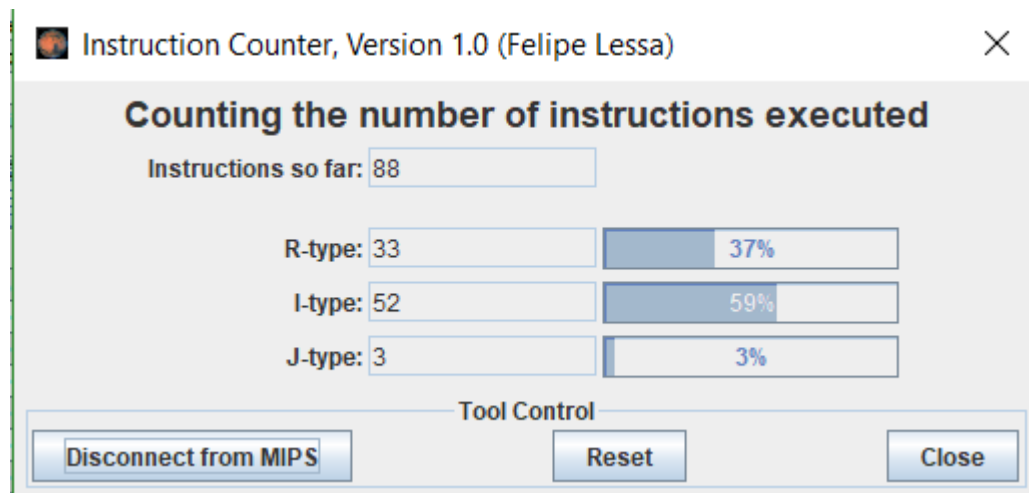
Hình 1: Single clock cycle

3.2 Các Test Case

- ❖ Trường hợp có nhớ sau khi xóa 14 bit chắc chắn thuộc Mantissa và bit đầu tiên → Tăng Exponent lên 1

➤ $5.4 \times 3.4 = 18.36$

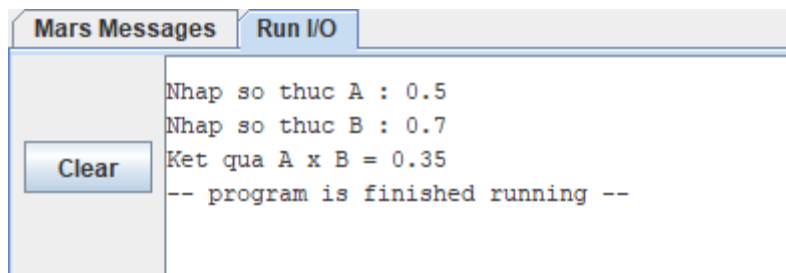




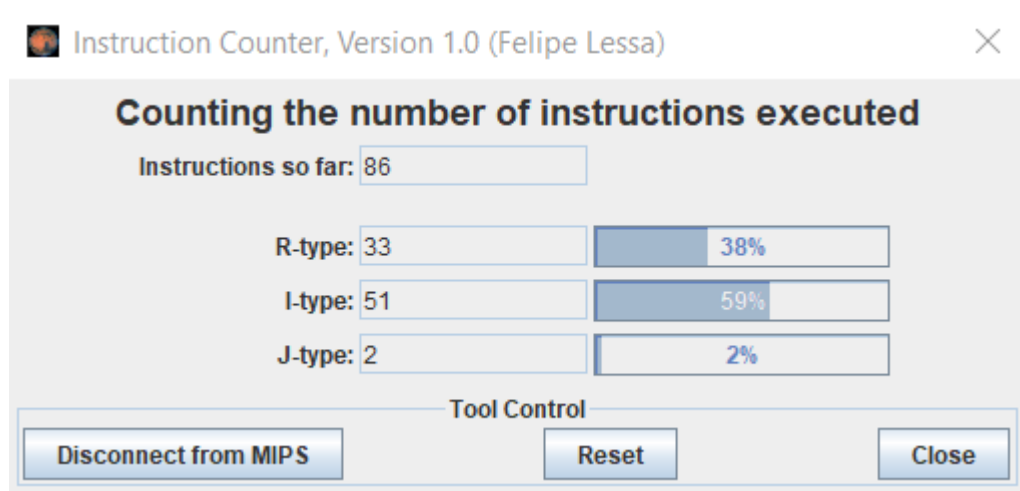
Thời gian thực thi trên máy tính MIPS có chu kỳ T là :
$$\text{ExeTime} = \text{CPI} * \text{IC} * T = 88T$$

❖ Trường hợp “không” có nhớ sau khi xóa 14 bit chắc chắn thuộc Mantissa và bit đầu tiên → Không tăng Exponent.

➤ $0.5 \times 0.7 = 0.35$



```
Nhap so thuc A : 0.5
Nhap so thuc B : 0.7
Ket qua A x B = 0.35
-- program is finished running --
```

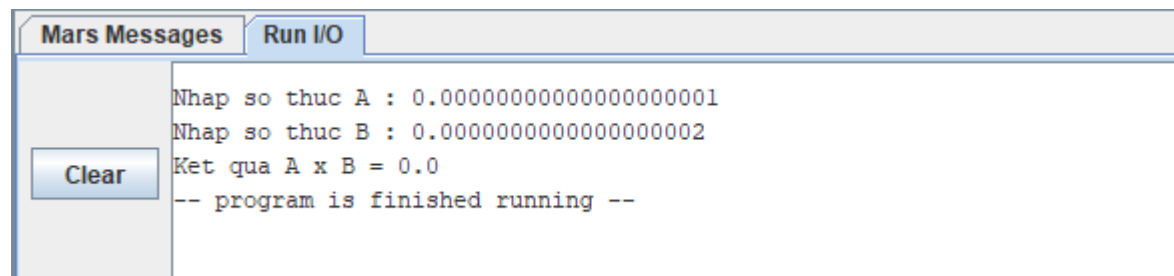
Thời gian thực thi trên máy tính MIPS có chu kỳ T là :

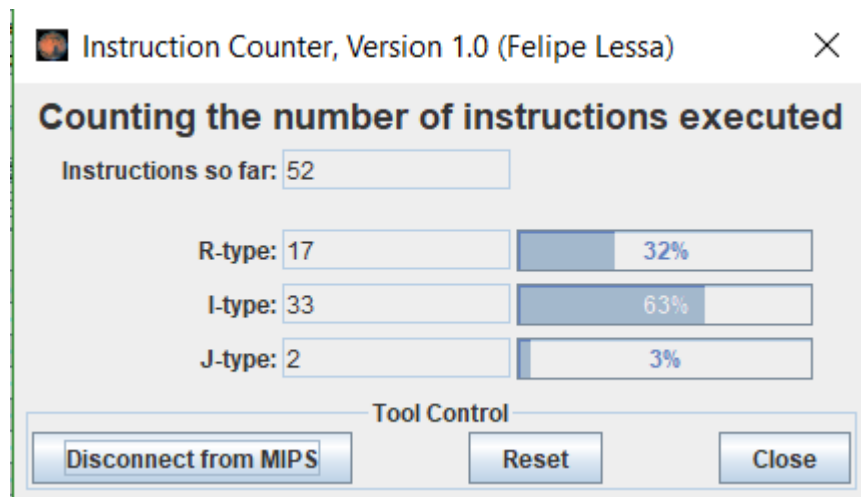
$$\text{ExeTime} = \text{CPI} * \text{IC} * T = 86T$$

❖ Trường hợp UnderFlow

Khi Exponent < 0 thì trả về kết quả 0.

➤ $0.00000000000000000001 \times 0.00000000000000000002 = 0.0$





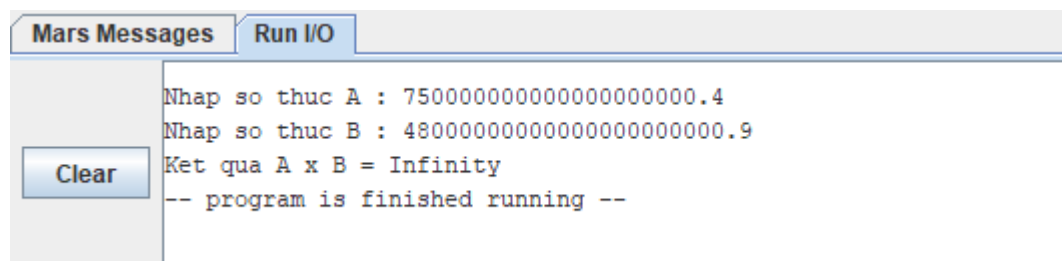
Thời gian thực thi trên máy tính MIPS có chu kỳ T là :

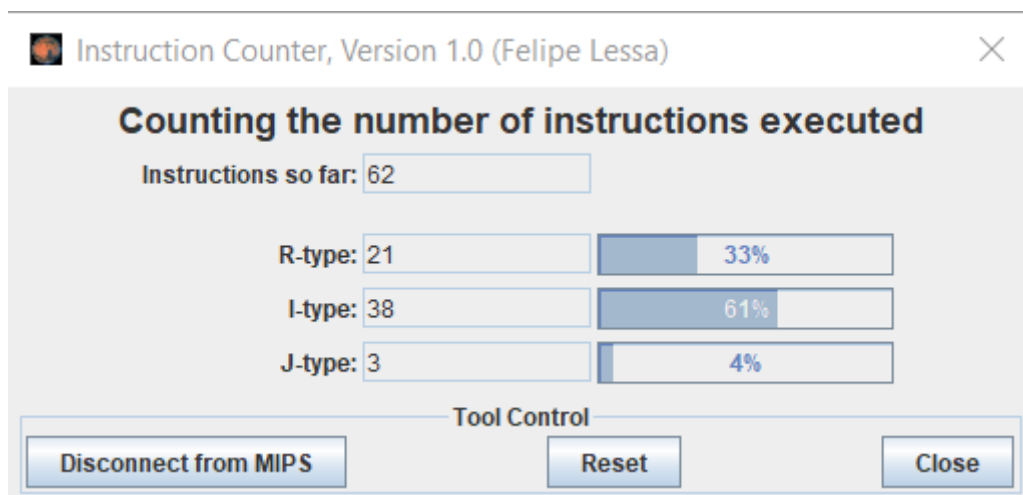
$$\text{ExeTime} = \text{CPI} * \text{IC} * T = 52T$$

❖ Trường hợp OverFlow

Khi Exponent vượt quá 254 (không biểu diễn được dưới số thực 32 bits).
Nên kết quả sẽ bằng Inf (vô cùng)

➤ $7500000000000000000.4 \times 4800000000000000000.9 = \text{Infinity}$



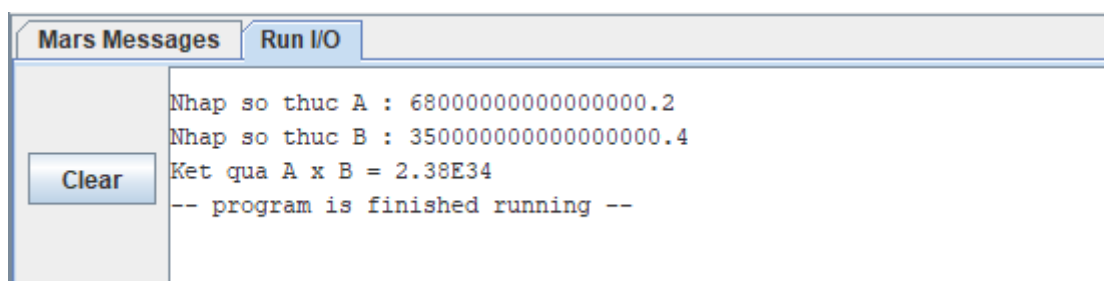


Thời gian thực thi trên máy tính MIPS có chu kỳ T là :

$$\text{ExeTime} = \text{CPI} * \text{IC} * T = 62T$$

❖ Các trường hợp tiềm cận Overflow

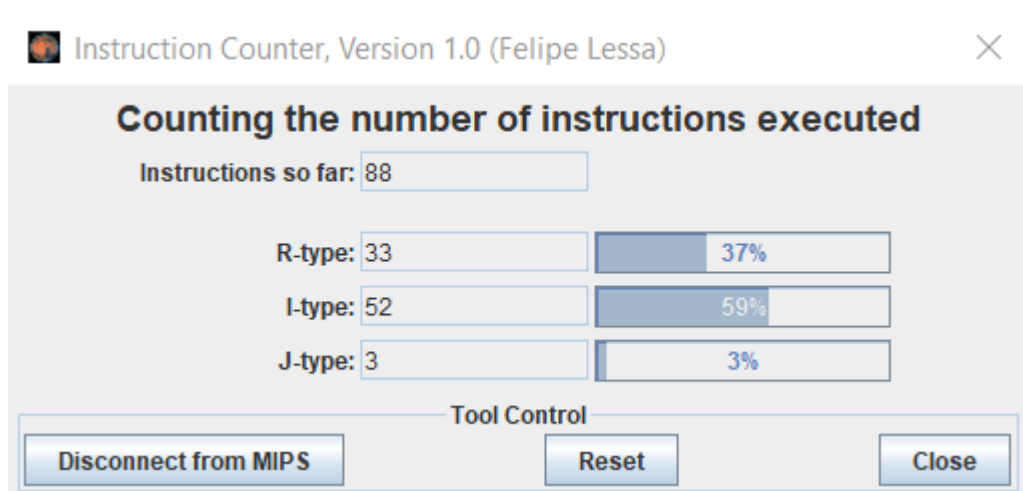
➤ $6800000000000000.2 \times 3500000000000000.4 = 2.38E34$



```

Nhap so thuc A : 6800000000000000.2
Nhap so thuc B : 3500000000000000.4
Ket qua A x B = 2.38E34
-- program is finished running --

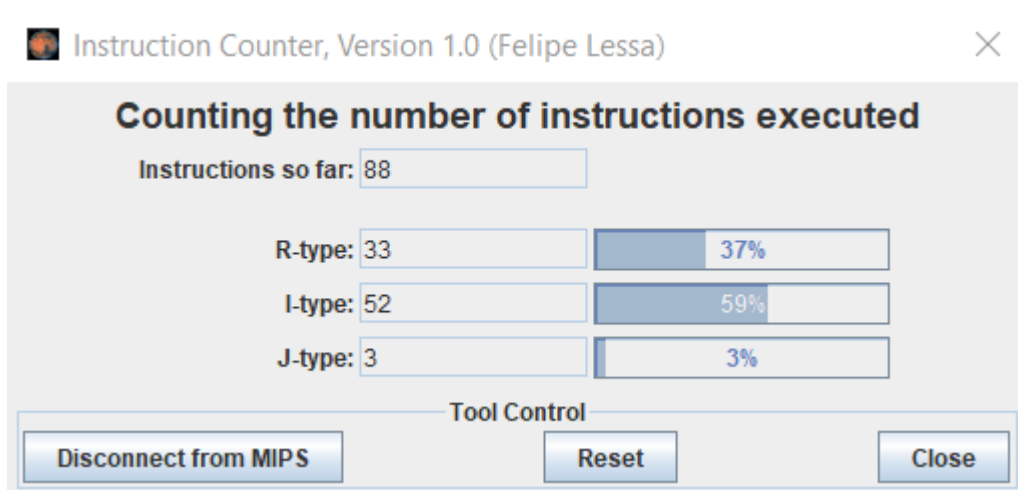
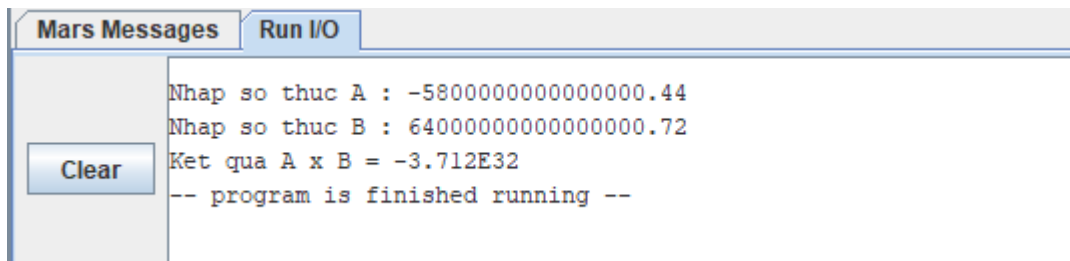
```



Thời gian thực thi trên máy tính MIPS có chu kỳ T:

$$\text{ExeTime} = \text{CPI} * \text{IC} * T = 88T$$

➤ $-5800000000000000.44 \times 6400000000000000.72 = -3.712\text{E}32$

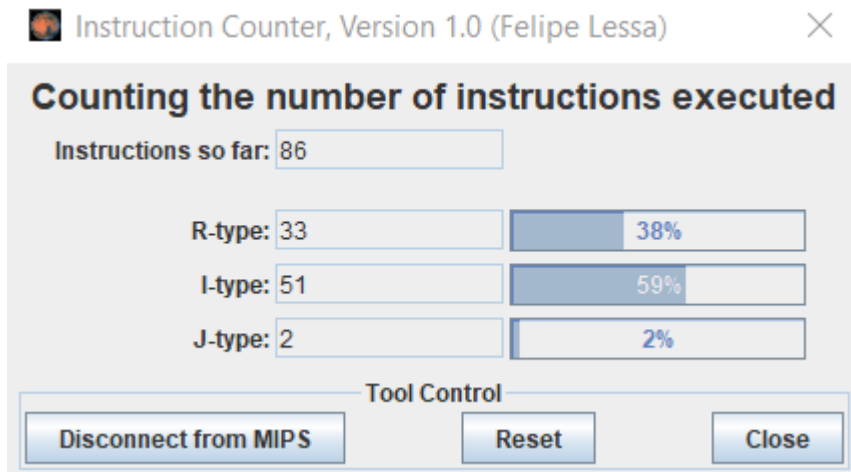
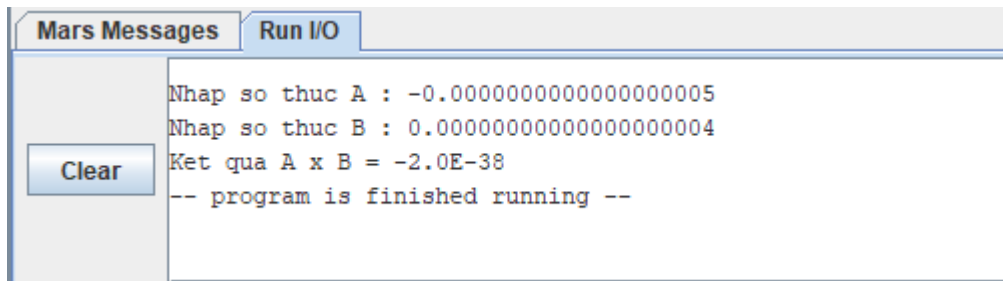


Thời gian thực thi trên máy tính MIPS có chu kỳ T:

$$\text{ExeTime} = \text{CPI} * \text{IC} * T = 88T$$

❖ Trường hợp tiệm cận UnderFlow

➤ $-0.000000000000000005 \times 0.000000000000000004 = -2.0\text{E}-38$



Thời gian thực thi trên máy tính MIPS có chu kỳ T là :

$$\text{ExeTime} = \text{CPI} * \text{IC} * T = 86T$$

Tài liệu

[1]

“http://www4.comp.polyu.edu.hk/~comp2421/ComputerOrganizationAndDesign5thEdition2014.pdf?fbclid=IwAR2T3ceABNoJqeu0escInq_BsxL1FEnFttH2z2j0GhEzKa9NnLaRqUXSZTI”

[2] “<https://stackoverflow.com/search?q=floating+point+multiplication+in+mips>”