ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

BÁO CÁO BÀI TẬP LỚN

KIẾN TRÚC MÁY TÍNH

Đề tài: NHÂN HAI SỐ THỰC DẠNG CHUẨN

GVHD: Trần Thanh Bình

SINH VIÊN THỰC HIỆN: Bùi Đức Huy

MSSV: 1812336

Tp Hồ Chí Minh, Tháng 12/2020

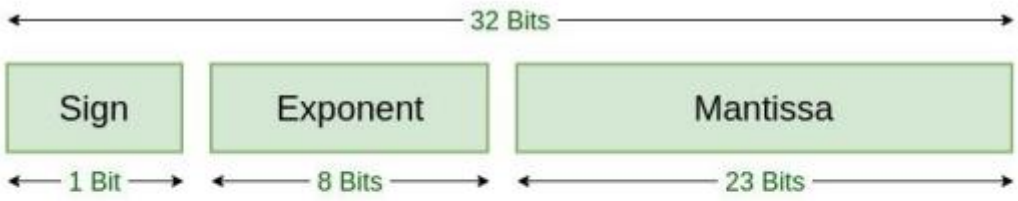
Mục lục

1. Số thực là dấu chấm động

Số thực dấu chấm động là cách biểu diễn các số thực trong máy tính đã được chuẩn hóa bởi tổ chức IEEE (Institute of Electrical and Electronic Engineers) vào năm 1985 và được gọi là chuẩn IEEE 754. Chuẩn IEEE 754 gồm nhiều dạng định dạng số thực với các kích thước khác nhau như 16bit, 32bit, 64bit, … Tuy nhiên, trong máy tính, dạng 32bit và 64bit được sử dụng rộng rãi tương đương với hai kiểu dữ liệu float và double trong ngôn ngữ C chuẩn (ANSI C). Biểu diễn số thực dấu chấm động theo dạng chuẩn IEEE 754 dùng 32bit được gọi là biểu diễn theo dạng "chính xác đơn" (single precision) trong khi 64bit được gọi là biểu diễn theo dạng "chính xác kép" (double precision)[1] .

Theo như yêu cầu bài Assignment, ta sẽ nói qua về Số thực dấu chấm động biểu diễn theo dạng chính xác đơn.

Định dạng số thực dấu chấm động theo chuẩn IEEE 754 32bit gồm 3 phần lần lượt là "S" biểu diễn dấu (sign) – 1bit, "phần mũ" (exponent) – 8bit và "phần phân số" (fraction) – 23bit.



Single Precision
IEEE 754 Floating-Point Standard

2. Giải thuật nhân 2 số thực dạng chuẩn (Standard Floating Point) A và B với độ chính xác đơn (32 bit).

Công thức xác định dạng của số thực X là:

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

Ta sẽ chuyển 2 số A và B về dạng này và sau đó thực hiện phép nhân:

$$A * B = (-1)^{(sA + sB)} \times (1.mA \times 1.mB) \times 2^{(eA + eB - 2*b)}$$

B1: Tách phần bit dấu - bằng cách dịch 32bit của từng thừa số sang phải 31bits. Xong xác định bit dấu của tích bằng cách dùng phép toán XOR 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 (quy ước bit âm là s = 1, dương s = 0).

B2: Tách phần xác định mũ. - bằng cách dịch sang trái 1 bit để loại bỏ bit dấu sau đó dịch 24 bit sang phải. Tiếp tục trừ mỗi số mũ cho 127 và cộng hai kết quả với nhau. Kết quả thu được là phần xác định mũ của kết quả.

B3: Xác định phần định trị. - dịch trái 9 bit sau đó dịch phải 9 bit để loại bỏ 9 bit đầu tiên, kết quả ta thu được là phần 23 bit xác định giá trị số thực.

B4: Nhân 2 phần định trị với nhau: - Sau khi lấy được 2 phần định trị m của 2 số A và B, ta sẽ thêm bit 1 vào đầu và thực hiện phép nhân. - Khi thực hiện phép nhân, do kết quả của phép nhân sẽ > 32bit nên ta sẽ chia và xử lý kết quả bằng cách lưu 32 bit high và

32bit low vào 2 thanh ghi. + Nếu thanh ghi chứa bit high mà có 15 bit (nghĩa là sau khi nhân thì kết quả ở dạng 1,….) thì sẽ lấy 14 bit cuối của high (nghĩa là chỉ lấy phần sau dấu phẩy) và 9 bit đầu của low phần định trị của tích(23bits). + Còn nếu thanh ghi high có 16 bit (nghĩa là sau khi nhân thì kết quả ở dạng 10, ... hoặc 11, ...) thì ta sẽ dịch phải thanh ghi Low 1 bit và lấy bit đầu cuối của thanh ghi High thêm vào thanh ghi Low, còn với thanh ghi High thì ta dịch trái 1 bit để còn 15bit có nghĩa (về dạng 1,…).Sau đó thì lấy 14 bit cuối của high (nghĩa là chỉ lấy phần sau dấu phẩy) và 9 bit đầu của low phần định trị của tích(23bits). Bước này tương tự như việc dịch tất cả các bit kết quả sang trái, do vậy ta phải cộng thêm 1 cho phần số mũ (nhân 2).

B5: Cuối cùng ta ghép các bit theo thứ tự Sign-Exponent-Mantissa ta sẽ được kết quả.

LƯU Ý: Ta thấy vớ cách biểu diễn trên thì không thể biểu diễn chính xác được số 0 nên ta quy ước Nếu tất cả các bit của e và m đều bằng 0 thì X = 0

3. Thống kê số lệnh, loại lệnh.

3.1 R-type:

R là viết tắt của Registers, là định dạng dùng mã hóa các lệnh và các toán hạng của nó đều là thanh ghi (ví dụ lệnh add) và các lệnh dịch trái (sll), dịch phải (srl) và rẽ nhánh thanh ghi(jr). Định dạng R gồm có 6 fields được mô tả như dưới đây:

Op(6)	Rs(5)	Rt(5)	Rd(5)	Sa(5)	Funct(6)

Trong đó:

- Op: mã phép toán(Opcode), cho biết lệnh thực hiện phép toán nào (R-type có opcode bằng 0)
- Funct: function code (mở rộng cho Opcode)
- Ba thanh ghi toán hạn (Register Operand):
    + Rs, Rt: Hai toán hạng nguồn
    + Rd: Toán hạn đích chứa kết quả
    + Sa: Quy định số bit dịch trong các lệnh dịch

3.2 I-Type:

Định dạng lệnh I (Immediate – số nguyên trực tiếp) là định dạng dùng mã hóa các lệnh có một toán hạng là số nguyên (trừ các lệnh dịch) hay có sự tham gia của số nguyên (như các lệnh di chuyển dữ liệu) và các lệnh rẽ nhánh có điều kiện. Định dạng I có 4 fields được mô tả như sau:

Op(6)	Rs(5)	Rt(5)	Immediate(16)

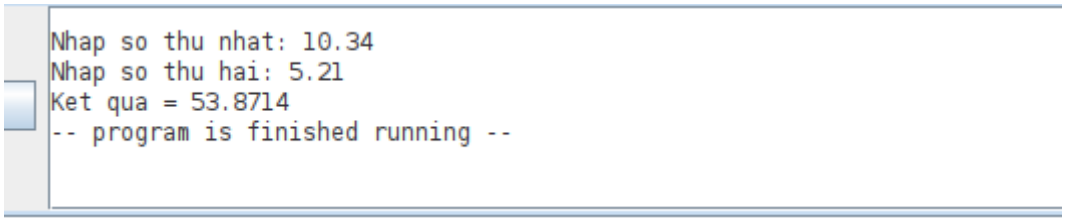
Trong đó:

- Op: mã phép toán (opcode), cho biết lệnh làm gì
- Immediate: hằng số 16 bit được lưu trong lệnh:
    + Rs: thanh ghi toán hạng nguồn
    + Rt: thanh ghi toán hạng đích

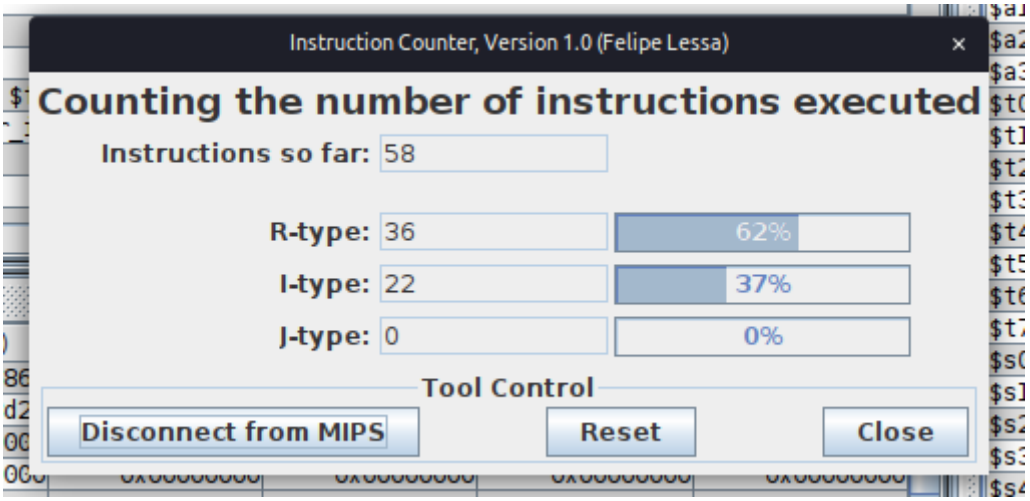
4. Kiểm thử phần mềm

4.1 Test case 1

Kiểm tra nhân hai số thực a = 10.34 và b = 5.21
Nhận được kết quả in ra màn hình :

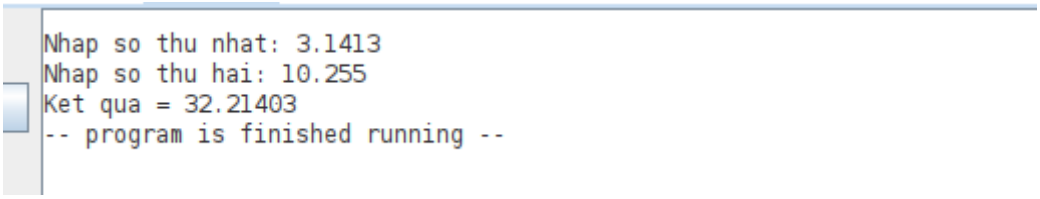


Kiểm tra Instruction count:

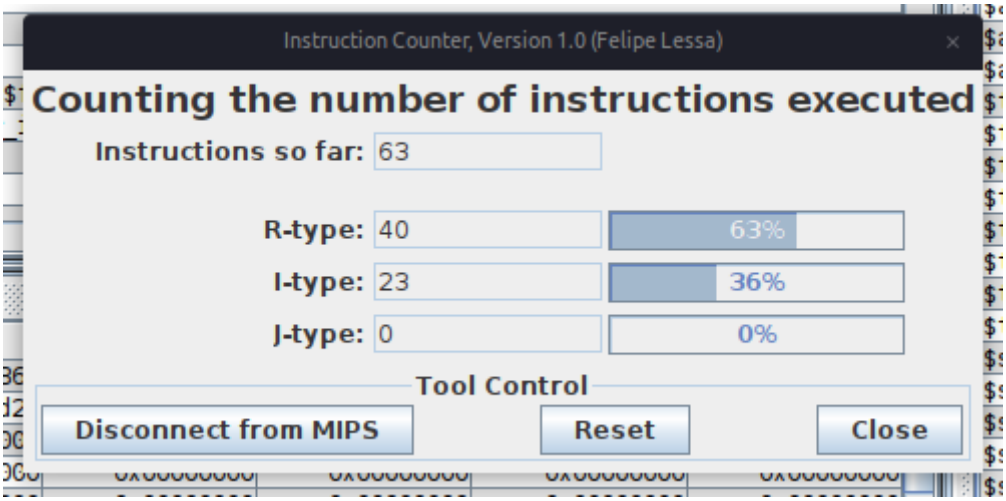


4.2 Test case 2

Kiểm tra nhân hai số thực a = 3.1413 và b = 10.255
Nhận được kết quả in ra màn hình:

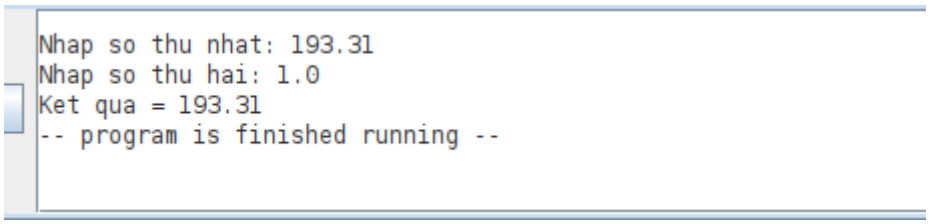


Kiểm tra Instruction count:

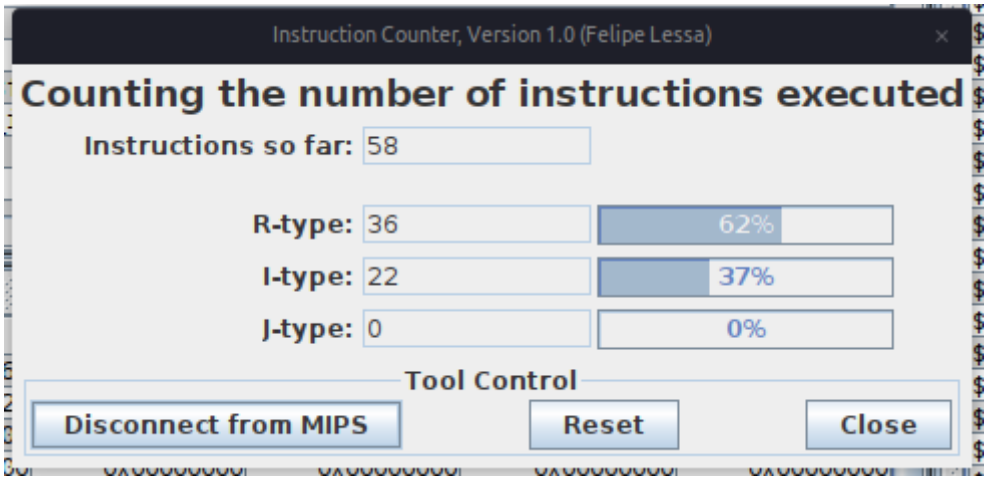


4.3 Test case 3

Kiểm tra nhân hai số thực a = 193.31 và b = 1.0
Nhận được kết quả in ra màn hình:



Kiểm tra Instruction count:

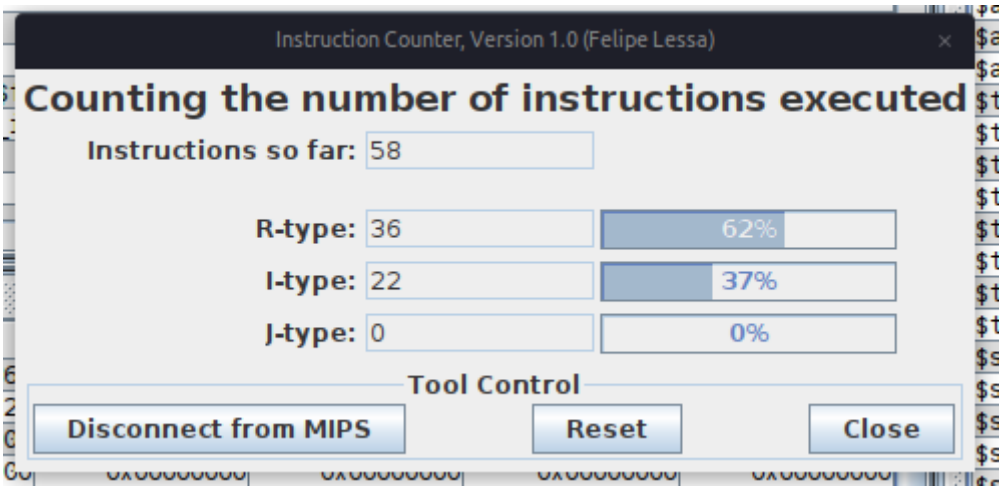


4.4 Test case 4

Kiểm tra nhân hai số thực a = 18.12336 và b = 18.12336
Nhận được kết quả in ra màn hình:

```
Nhap so thu nhat: 18.12336
Nhap so thu hai: 18.12336
Ket qua = 328.45615
-- program is finished running --
```

Kiểm tra Instruction count:

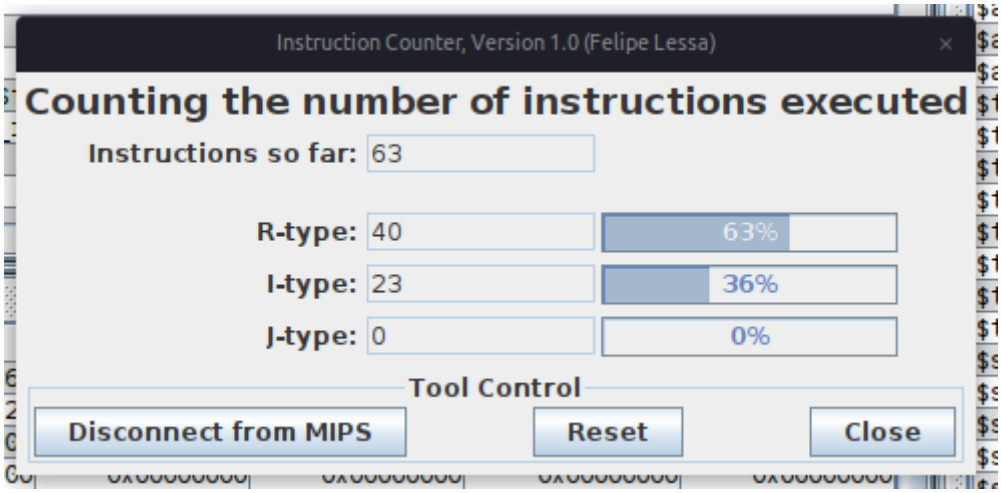


4.5 Test case 5

Kiểm tra nhân hai số thực a = 2020.202 và b = 2000.1999
Nhận được kết quả in ra màn hình:

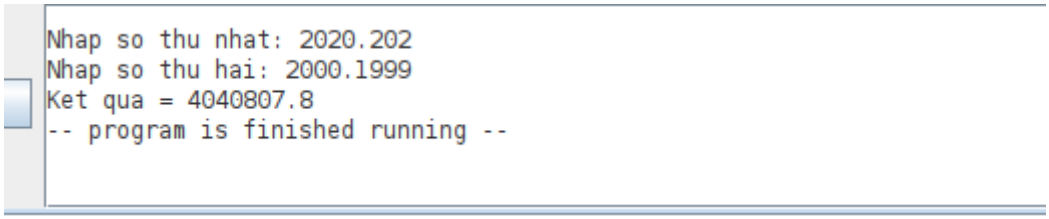
```
Nhap so thu nhat: 2020.202
Nhap so thu hai: 2000.1999
Ket qua = 4040807.8
-- program is finished running --
```

Kiểm tra Instruction count:

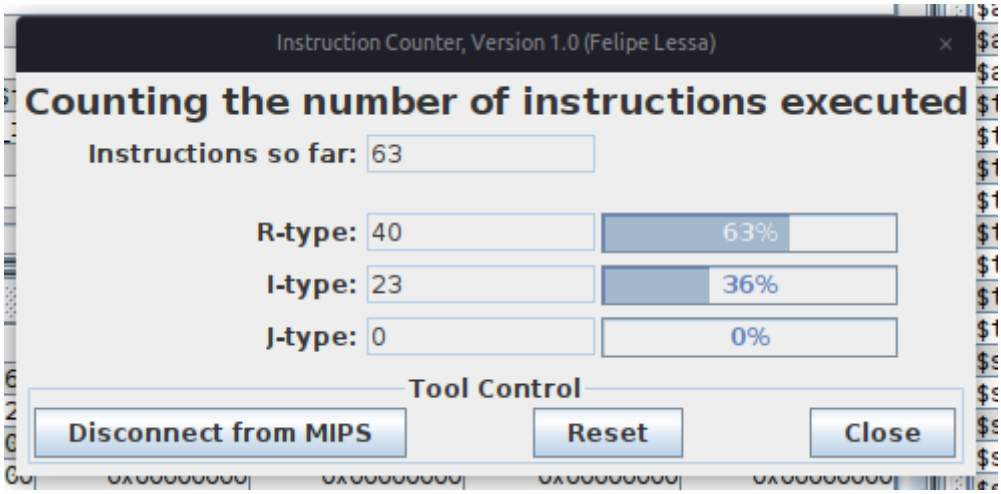


4.5 Test case 5

Kiểm tra nhân hai số thực a = 2020.202 và b = 2000.1999
Nhận được kết quả in ra màn hình:

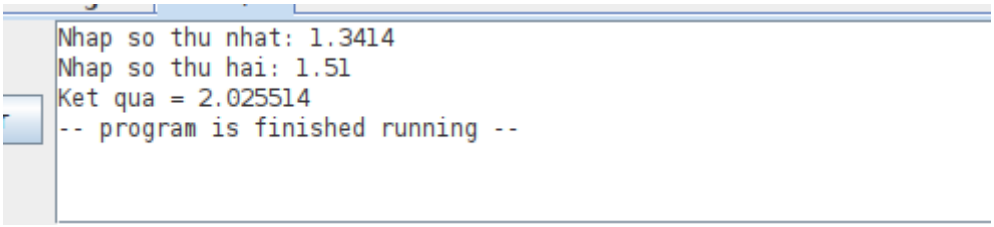


Kiểm tra Instruction count:

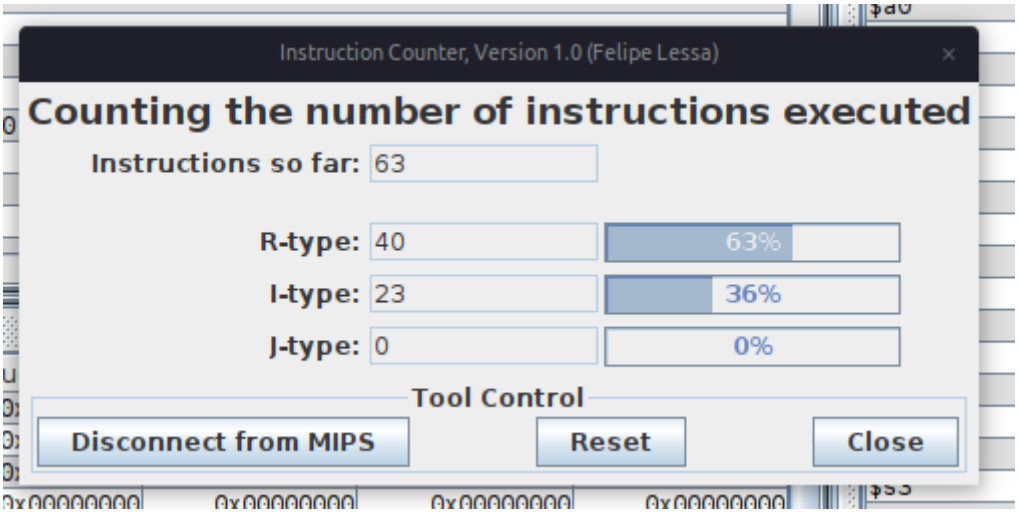


4.6 Test case 6

Kiểm tra nhân hai số thực a = 1.3414 và b = 1.51
Nhận được kết quả in ra màn hình:



Kiểm tra Instruction count:

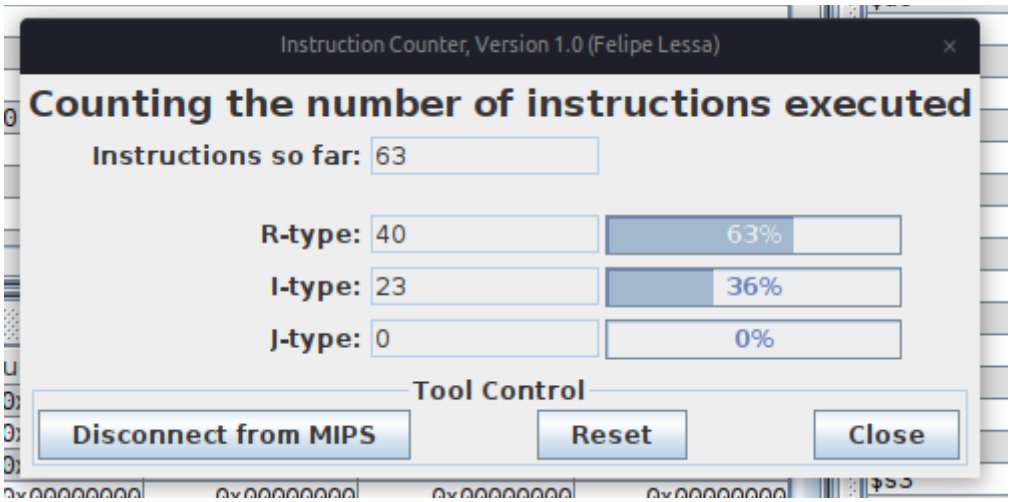


4.7 Test case 7

Kiểm tra nhân hai số thực a = -1.31 và b = -3.1
Nhận được kết quả in ra màn hình:

```
Nhap so thu nhat: -1.31
Nhap so thu hai: -3.1
Ket qua = 4.0609994
-- program is finished running --
```

Kiểm tra Instruction count:

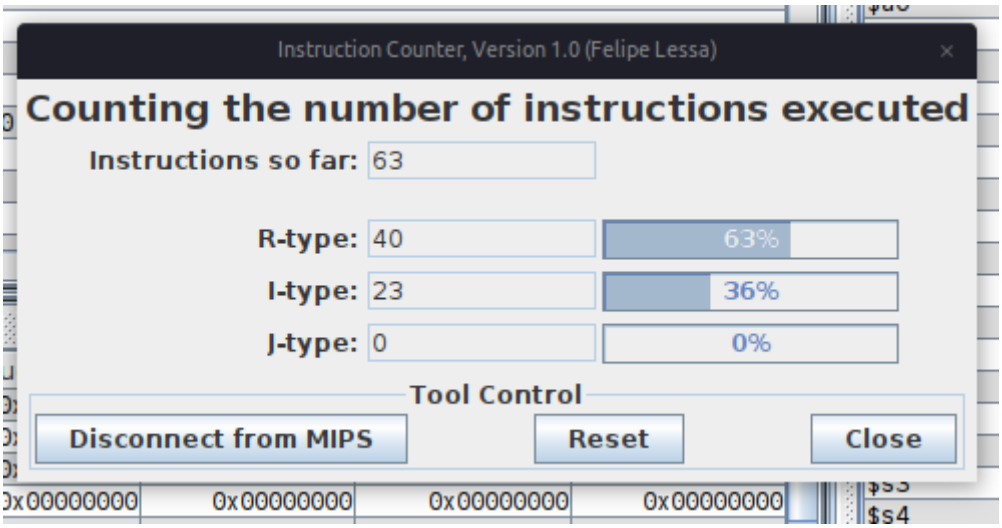


4.8 Test case 8

Kiểm tra nhân hai số thực a = -10.1 và b = 31.5
Nhận được kết quả in ra màn hình:

```
Nhap so thu nhat: -10.1
Nhap so thu hai: 31.5
Ket qua = -318.15
-- program is finished running --
```

Kiểm tra Instruction count:

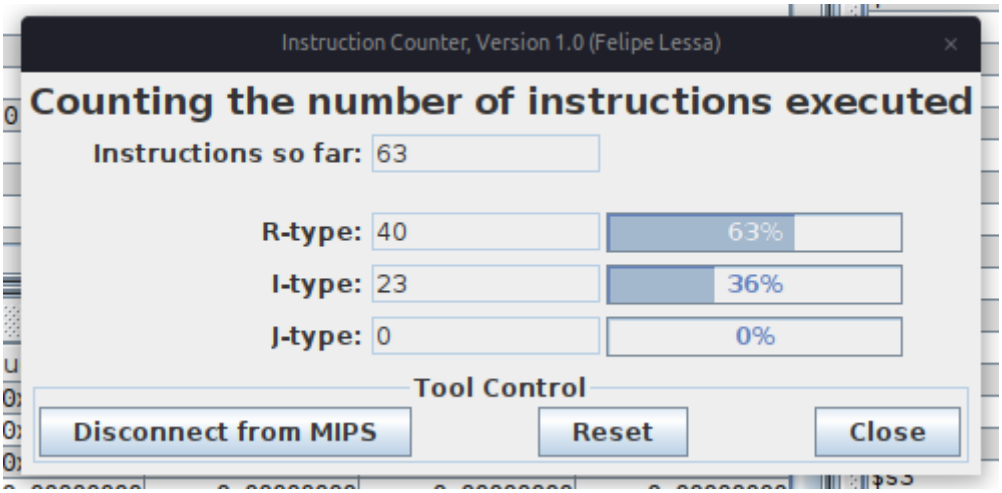


4.9 Test case 9

Kiểm tra nhân hai số thực a = 90.1 và b = 31.3
Nhận được kết quả in ra màn hình:

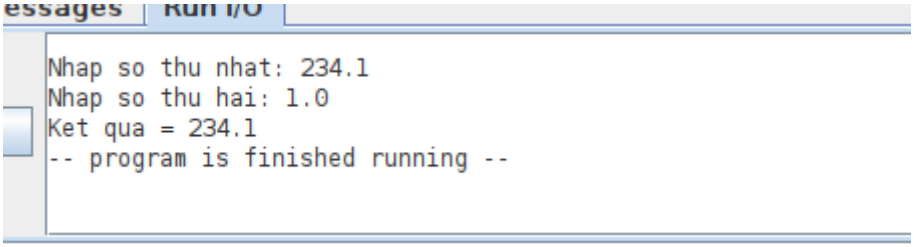
```
Nhap so thu nhat: 90.1
Nhap so thu hai: 31.3
Ket qua = 2820.13
-- program is finished running --
```

Kiểm tra Instruction count:

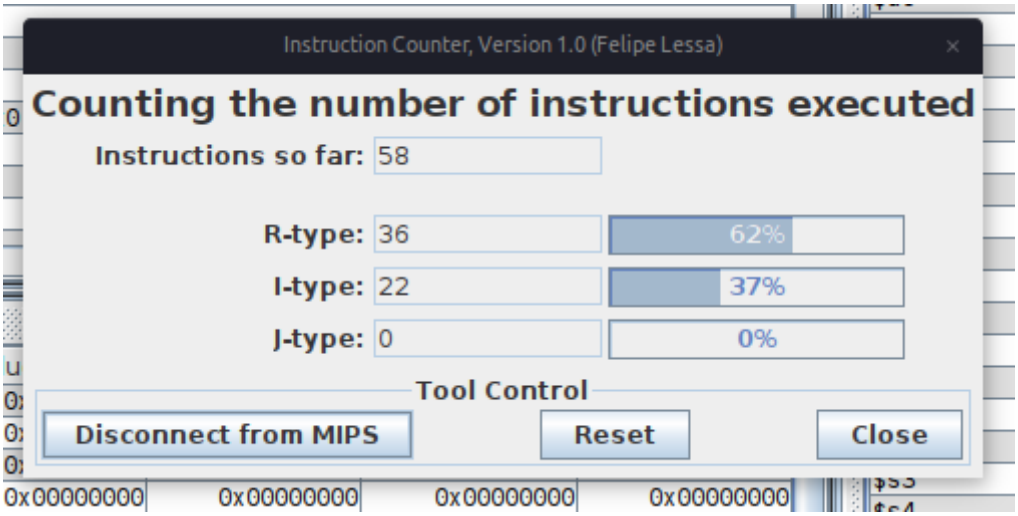


4.10 Test case 10

Kiểm tra nhân hai số thực a = 234.1 và b = 1.0
Nhận được kết quả in ra màn hình:



Kiểm tra Instruction count:



TÀI LIỆU THAM KHẢO

1. https://vietcodes.github.io/algo/mips
2. https://www3.ntu.edu.sg/home/smitha/FYP_Gerald/instruction.html
3. https://stackoverflow.com/questions/10204706/multiplication-of-floating-point-numbers-in-mips