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

KIẾN TRÚC MÁY TÍNH (CO2008)

Báo cáo Bài tập lớn
CHIA 2 SỐ THỰC CHUẨN IEEE 754 CHÍNH XÁC ĐƠN.

GVHD: Nguyễn Xuân Minh

Nhóm - Lớp: Nhóm 16 - Lớp L03
Sinh viên thực hiện: Vũ Hoàng Thiên An 2110717
Nguyễn Trọng Anh 2110014

TP. Hồ Chí Minh, Tháng 04/2023

Mục lục

1 Đề bài **2**

2 Yêu cầu **2**

3 Giới thiệu đề tài và ý tưởng thực hiện **2**

4 Giải thuật **3**

5 Thống kê số lệnh, thời gian chạy chương trình và kiểm thử **5**
5.1 Trường hợp không denormalize toán hạng 5
5.1.1 Kết quả cần normalize 5
5.1.2 Kết quả không cần normalize 6
5.2 Trường hợp denormalize toán hạng 6
5.2.1 Denormalize số chia hoặc số bị chia 6
5.2.2 Chỉ denormalize thương số 7
5.3 Các phép tính với số 0 7
5.3.1 Chia cho số 0 7
5.3.2 Số 0 chia một số 7
5.3.3 Số 0 chia số 0 8

6 Kết luận **8**

1 Đề bài

Đề 7: Chia 2 số thực chuẩn IEEE 754 chính xác đơn.
Viết chương trình thực hiện phép chia 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 trị x 4 bytes = 8 bytes).

2 Yêu cầu

- Chương trình viết và chạy trên MARS MIPS 4.5.
- Code:
  - Code style phải rõ ràng, có chú thích.
  - Phải có gọi hàm. Truyền tham số và trả kết quả khi gọi hàm theo quy ước của thanh ghi ($Ai chứa tham số, $Vi hoặc $fi chứa giá trị trả về).
  - In kết quả ra màn hình để kiểm tra.
- Nội dung báo cáo:
  - Trình bày giải pháp hiện thực.
  - Giải thuật (nếu có).
  - Thống kê số lệnh, loại lệnh (instruction type) sử dụng trong chương trình.
  - Tính thời gian chạy của chương trình (CR=1GHz).
  - Kết quả kiểm thử.
- Nộp báo cáo : 3 files
  - File báo cáo (không source code) định dạng .PDF (Bc_nhom16.pdf).
  - File mã nguồn (Mn_nhom16.asm).
  - File dữ liệu đầu vào (FLOAT2.BIN).

3 Giới thiệu đề tài và ý tưởng thực hiện

Việc chia 2 số thực chuẩn IEEE 754 chính xác đơn được hỗ trợ bởi lệnh div.s trong Mips. Tuy nhiên, trong đề tài này, chúng ta không dùng các lệnh tính toán số thực của MIPS để thực hiện việc này. Phép chia này có thể được thực hiện bằng cách xử lý 3 phần : Sign, Exponent và Fraction của số bị chia và số chia nhằm tìm ra Sign, Exponent và Fraction của kết quả.

Số thực chuẩn IEEE 754 có giá trị hệ thập phân bằng :

$$(-1)^{\text{Sign}}.(1+\text{fraction}).2^{\text{exponent}-127}$$

Trong đó:

- Sign: giá trị phần sign ở hệ thập phân.
- Fraction: giá trị phần Fraction ở hệ thập phân.
- Exponent: giá trị phần Exponent ở hệ thập phân.

Kết quả phép chia 2 số thực chuẩn IEEE 754 chính xác đơn có thể viết bằng biểu thức sau:

$$\text{Thương số} = \frac{\text{Số bị chia}}{\text{Số chia}} = \frac{(-1)^{\text{Sign}_1}.(1+\text{fraction}_1).2^{\text{exponent}_1-127}}{(-1)^{\text{Sign}_2}.(1+\text{fraction}_2).2^{\text{exponent}_2-127}}$$

Ý tưởng thực hiện: chia kết quả cần tính làm 3 phần:

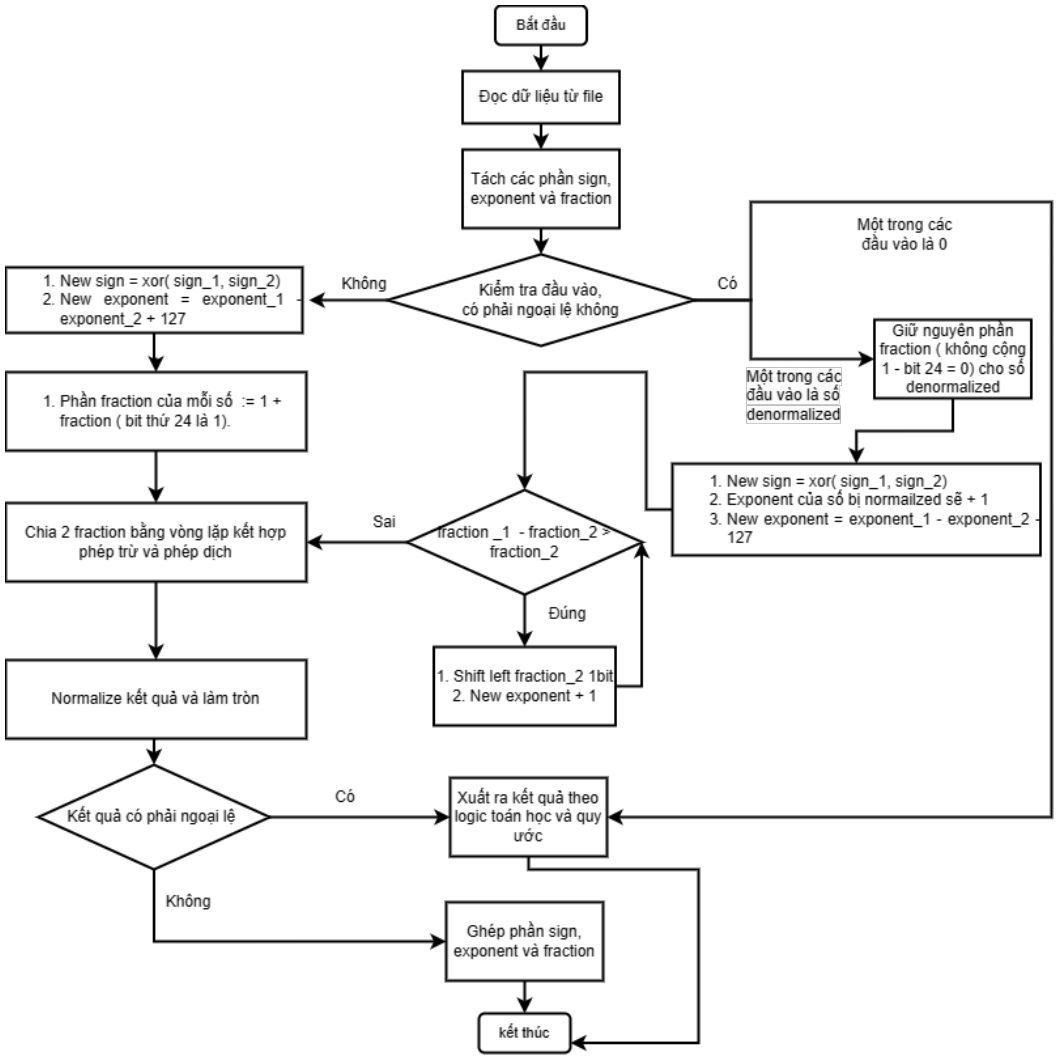
- Đối với phần Sign của kết quả :là kết quả phép xor phần Sign của số bị chia và số chia .

- Đối với phần Exponent của kết quả : là kết quả của phép tính (Exponent số bị chia) - (Exponent số chia) + 127.
- Đối với phần Fraction của kết quả : là kết quả của phép tính 1-((1 + fraction số bị chia) / (1 + fraction số chia)).

Lưu ý: Trong chương trình học, đối với các số thực có phần exponent = 0 (8 bit 0) thì được coi là ngoại lệ do vượt ngoài tầm vực của số thực chính xác đơn 32 bit. Tuy nhiên, trong thuật toán của nhóm thiết kế, vẫn chấp nhận các số này, và các số này, theo IEEE 754, được gọi là các số **Denormalized**. Đối với các số denormalized này, phần fraction của chúng **không** được cộng 1 trong quá trình biểu diễn.

4 Giải thuật

Trong chương trình này, nhóm đã thiết kế giải thuật có thể bao quát được cho cả số thực dạng thường và số thực dạng denormalized. Giải thuật nhìn chung có thể biểu diễn bằng flowchart như sau:



Giải thích chi tiết:

- Đọc file : Dữ liệu đầu vào là 2 số thực đọc từ file lưu trữ dạng nhị phân trên đĩa FLOAT2.BIN. Sử dụng các lệnh syscall 13, 14, 16 để mở file, đọc dữ liệu từ file và đóng file.
- Phép chia fraction: Đầu vào của phép chia 2 fraction, lặp lại 31 lần các bước sau:
  1. Dịch trái thanh ghi kết quả 1 bit.

2. Nếu $\text{fraction}_1 < \text{fraction}_2$, nhảy đến bước 3, không thì thực hiện phép trừ $\text{fraction}_1 = \text{fraction}_1 - \text{fraction}_2$, đồng thời +1 cho thanh ghi kết quả.
3. dịch trái fraction_1 1 bit, nhảy đến bước 1.

- Các ngoại lệ:
  - Các phép tính với số 0:
    - Chia cho số 0 : Trường hợp một số khác 0 chia cho số 0, kết quả xuất ra sẽ là -Infinity hoặc Infinity tùy thuộc vào dấu của số chia.
    - Số 0 chia một số : Trường hợp số 0 chia cho một số khác 0, kết quả xuất ra sẽ là -0.0 hoặc 0.0 tùy thuộc vào dấu của số bị chia.
    - Số 0 chia số 0 : Trường hợp số 0 chia số 0, kết quả xuất ra sẽ là NaN(Not a Number).
  - Tràn số: số thực bị tràn khi phần exponent không nằm trong khoảng giá trị hợp lệ (tràn trên khi phần exponent vượt quá giá trị 254 và tràn dưới khi phần exponent nhỏ hơn -23).
    - Số chia hoặc số bị chia bị tràn : Trong trường hợp này, Mips sẽ thông báo dữ liệu đầu vào bị tràn và không thể thực hiện chương trình.
    - Thương số bị tràn:
      Tràn trên : Trong trường hợp này, kết quả xuất ra sẽ là -Infinity hoặc Infinity tùy thuộc vào dấu của thương số.
      Tràn dưới : Trong trường hợp này, kết quả xuất ra sẽ là 0.

- Normalize thương số: Trong giải thuật chia 2 số thực của nhóm, phần mantissa sau khi chia phải có bit đầu tiên bằng 1, như vậy 23 bit sau bit đầu tiên sẽ là phần fraction của thương số. Tuy nhiên trong nhiều trường hợp, bit đầu tiên của phần mantissa sau khi chia bằng 0. Trong trường hợp này, ta có thể normalize thương số bằng cách:

  1. Dịch trái 1 bit phần mantissa.
  2. Trừ phần exponent đi 1 đơn vị.
  3. Kiểm tra phần exponent có hợp lệ(lớn hơn 0), nếu không thì dừng việc normalize và chuyển tới bước denormalize thương số, nếu có thì chuyển tới bước 4.
  4. Kiểm tra phần mantissa đã hợp lệ chưa, nếu chưa thì chuyển tới bước 1, nếu đã hợp lệ thì dừng việc normalize

- Làm tròn thương số: Kết quả phép chia phần mantissa là 32 bit, tuy nhiên ta chỉ lấy kết quả 24 bit đầu tiên để xử lý, trong trường hợp bit thứ 25 bằng 0, ta tiếp tục xử lý như bình thường, nếu bit thứ 25 bằng 1, ta cần làm tròn thương số bằng cách:

  1. Tăng fraction của thương số lên 1 đơn vị, nếu fraction khác 0 thì kết thúc quá trình làm tròn.
  2. Nếu fraction bằng 0, tăng exponent của thương số lên 1 đơn vị và kiểm tra thương số có bị tràn.

- Denormalize toán hạng: Trong Mips, việc phần Exponent của một số thực chuẩn IEEE 754 chính xác đơn bằng 0 và phần Fraction khác 0 được chấp nhận. Trong trường hợp này, số thực được xử lý theo một cách khác so với số thực chuẩn IEEE 754 thông thường và được gọi là số denormalized. Việc xử lý số denormalized là cần thiết để đảm bảo tính chính xác của phép chia.

  - Số chia hoặc số bị chia là số denormalized : giải thuật đã được trình bày trong flowchart.
  - Thương số cần được denormalize
    1. Kiểm tra thương số có bị tràn dưới, nếu có thì chuyển tới phần giải quyết trường hợp tràn số, nếu không thì chuyển tới bước 2.
    2. Cộng exponent một giá trị x sao cho exponent = 0.
    3. Dịch phải phần mantissa (x+1) bit.

5 Thống kê số lệnh, thời gian chạy chương trình và kiểm thử

Sử dụng công cụ Instruction Counter và Instruction Statistics để thống kê số lệnh, loại lệnh sử dụng trong chương trình đối với từng trường hợp được liệt kê dưới đây.
Ta sẽ tính thời gian chạy của chương trình với các trường hợp đã thống kê ở trên bằng công thức :

$$\text{CPU TIME} = \frac{\text{CPU clock cycles}}{\text{CPU clock rate}} = \frac{\text{Instruction Count . CPI}}{\text{Clock rate}}$$

Trong đó:

- Clock Rate = 1 GHz : tần số xung clock của CPU hay số chu kỳ trong 1 giây.
- CPU clock cycles : tổng số chu kỳ thực thi.
- Instruction Count(IC) : số lệnh của chương trình.
- CPI(Clock cycle per Instruction) : số chu kỳ để thực thi một lệnh (ở đây ta giả sử CPI = 1).

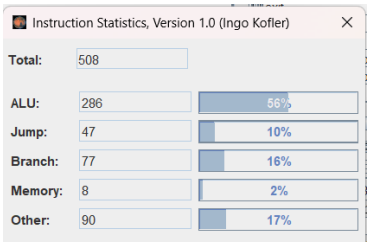
Như vậy, CPU TIME = IC.$10^-9(s)$.
Tiếp theo, nhóm sẽ xuất kết quả ra màn hình, cùng với việc thống kê số lượng lệnh. Trong lúc xuất kết quả, nhóm sẽ cho biết dữ liệu đầu vào, đầu ra, kết quả theo thuật toán tự thiết kế và kết quả do câu lệnh **div.s** của MARS MIPS 4.5.

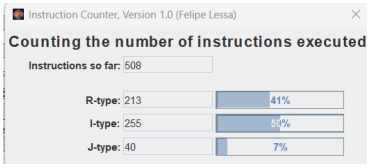
5.1 Trường hợp không denormalize toán hạng

5.1.1 Kết quả cần normalize

Kết quả cần làm tròn

```
Du lieu 1 = 8.0
Du lieu 2 = 3.0
our algorithm: 2.6666667
result from div.s: 2.6666667
```

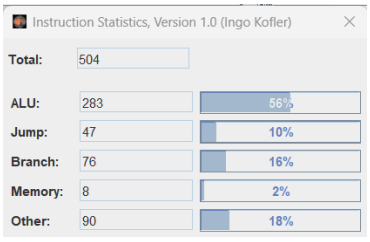


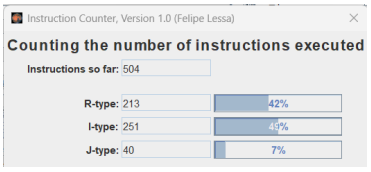


CPU TIME = 508ns

Kết quả không cần làm tròn

```
Du lieu 1 = 5.0
Du lieu 2 = 2.6
our algorithm: 1.923077
result from div.s: 1.923077
```

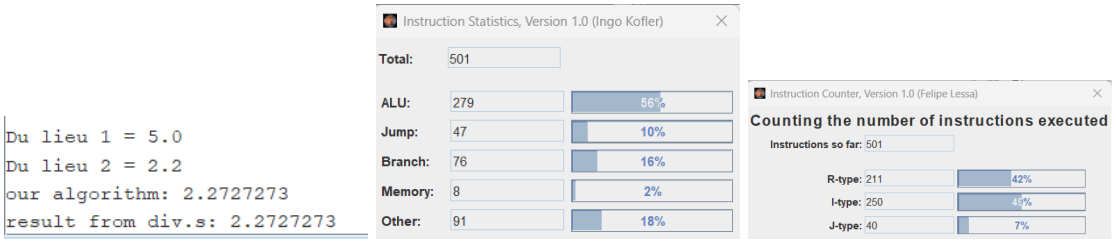




CPU TIME = 504ns

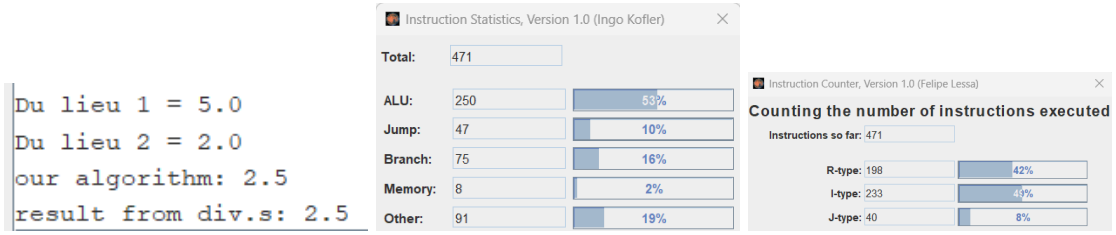
5.1.2 Kết quả không cần normalize

Kết quả cần làm tròn



$$\text{CPU TIME} = 501ns$$

Kết quả không cần làm tròn

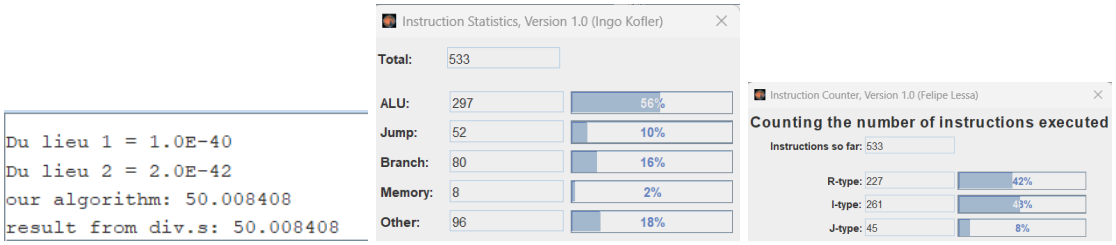


$$\text{CPU TIME} = 471ns$$

5.2 Trường hợp denormalize toán hạng

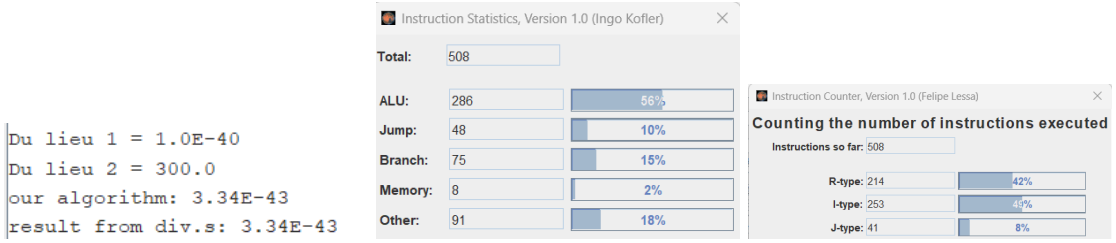
5.2.1 Denormalize số chia hoặc số bị chia

Không cần denormalize thương số



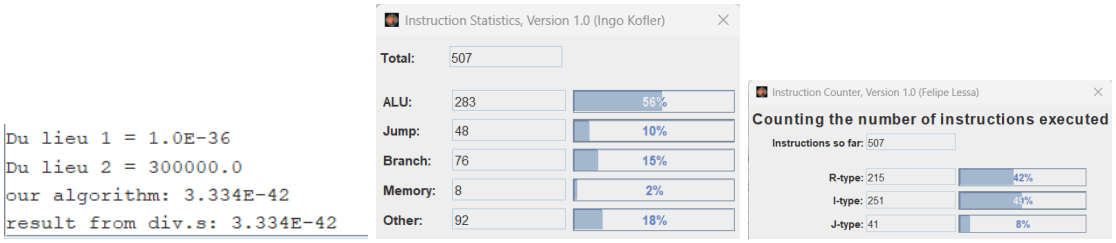
$$\text{CPU TIME} = 533ns$$

Cần denormalize thương số



$$\text{CPU TIME} = 508ns$$

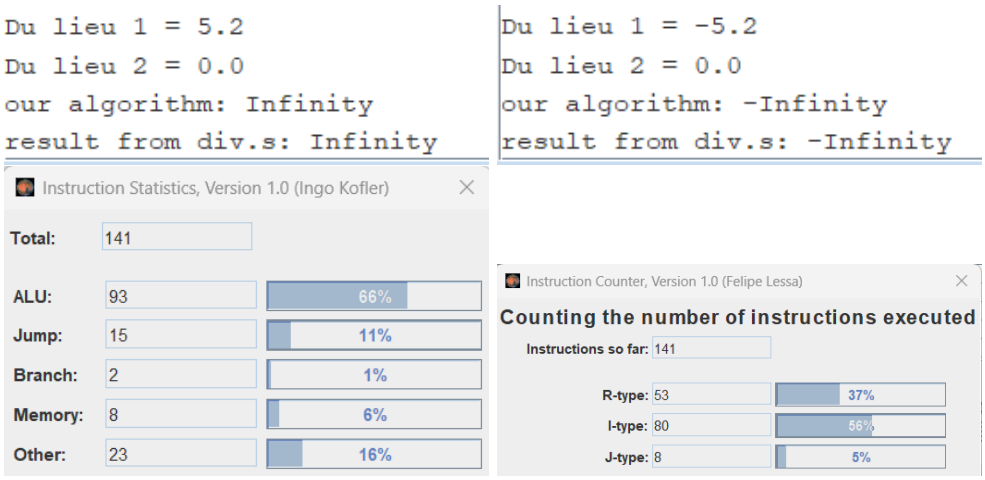
5.2.2 Chỉ denormalize thương số



$$\text{CPU TIME} = 507ns$$

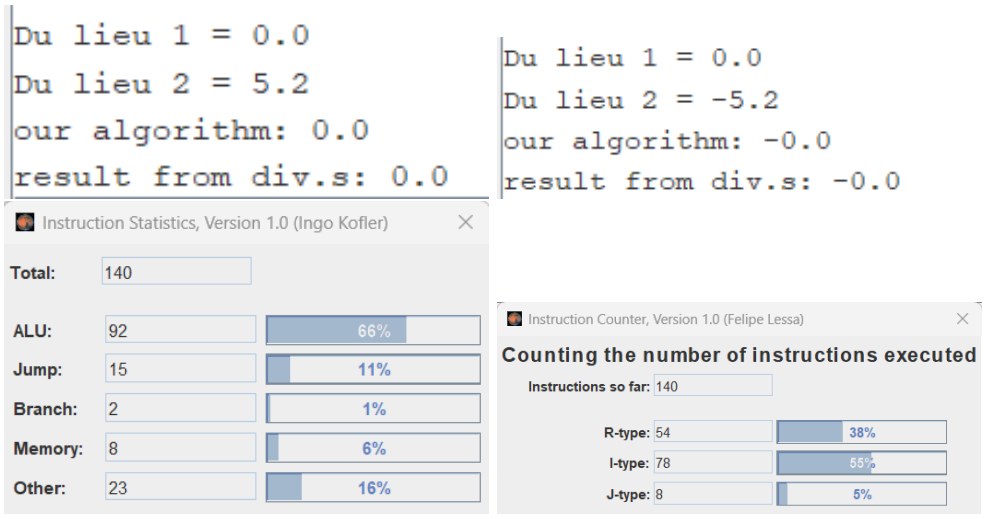
5.3 Các phép tính với số 0

5.3.1 Chia cho số 0



$$\text{CPU TIME} = 141ns$$

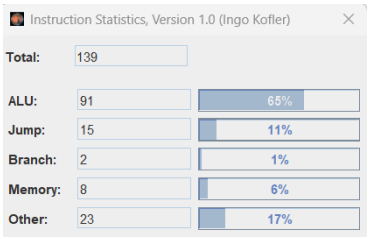
5.3.2 Số 0 chia một số

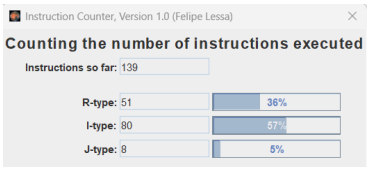


$$\text{CPU TIME} = 140ns$$

5.3.3 Số 0 chia số 0

```
Du lieu 1 = 0.0
Du lieu 2 = 0.0
our algorithm: NaN
result from div.s: NaN
```





$$\text{CPU TIME} = 139ns$$

6 Kết luận

Với kết quả đã trình bày ở trên, bài làm của nhóm nhìn chung đã đạt được những thứ sau đây:

1. Hiện thực được phép chia 2 số thực chính xác đơn trong MARS MIPS mà không dùng câu lệnh liên quan số thực.
2. Nắm được trường hợp denormalized theo chuẩn IEEE 754.
3. Thiết kế thuật toán chạy tốt cho cả trường hợp số normalized và denormalized.
4. Chưa phát hiện sai sót và sự khác biệt (so với lệnh div.s) cho nhiêu trường hợp khác nhau của đầu vào.