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

ĐỀ BÀI TẬP LỚN
Chia 2 số thực chuẩn IEEE 754 chính xác đơn

GV hướng dẫn: Nguyễn Xuân Minh

Nhóm thực hiện: L08_07

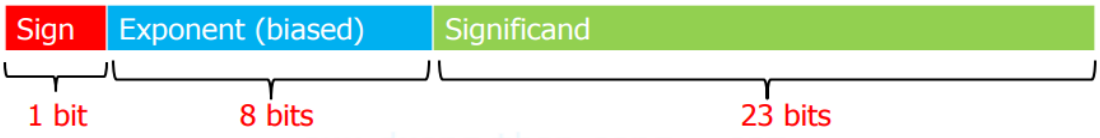
SV thực hiện: Phạm Hoàng Đức Huy – 2011286
Nguyễn Tiến Nam – 2011652

Tp. Hồ Chí Minh, Tháng 11/2021

Mục lục

1 Cơ sở lý thuyết

Số chấm động theo chuẩn IEEE 754 có độ chính xác đơn được biểu diễn như hình sau:



- Sign: Bit dấu (1: Số âm, 0: Số âm).
- Exponent: Phần mũ.
- Significand (Fraction): Phần định trị (Phần lẻ sau dấu chấm).

Tổng quát, số thực dấu chấm động chuẩn IEEE 754 chính xác đơn tính như sau (với Bias = 127):

$$(-1)^S \times (1 + Fraction) \times 2^{Exponent-Bias}$$

2 Giải pháp hiện thực

Giải pháp hiện thực 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):

- B1. Sử dụng chức năng open file (syscall 13) để mở file FLOAT2.BIN.
- B2. Sử dụng chức năng read from file (syscall 14) để đọc file trên.
- B3. Sử dụng các lệnh chuyển dữ liệu lưu 2 số thực cần tính toán vào thanh ghi.
- B4: Viết các hàm để tính thương (kết quả phép chia):
  - Hàm valid: Kiểm tra các trường hợp đặc biệt.
  - Hàm new_exp: Tính Exponent (Phần mũ) của thương.
  - Hàm divide: Tính Mantissa (Phần định trị) của thương.
  - Hàm normalize: Đưa về dạng chuẩn
  - Hàm rounding: Làm tròn phần định trị của thương
- B5: Sử dụng chức năng print string (syscall 4) và chức năng print float (syscall 2) để in ra thương.
- B6: Sử dụng chức năng close file (syscall 16) đóng file và chức năng exit (terminate execution) (syscall 10) kết thúc chương trình.

3 Giải thuật

Giải thuật thực hiện chia 2 số thực chuẩn IEEE 754 chính xác đơn:

- B1. Xét trường hợp đặc biệt (kiểm tra số bị chia (Dividend) và số chia (Divisor)). Nếu:

  - **Divisor bằng 0**, nếu Dividend:
    * bằng 0: In ra NaN. Kết thúc chương trình.
    * lớn hơn 0: In ra Infinity. Kết thúc chương trình.
    * nhỏ hơn 0: In ra -Infinity. Kết thúc chương trình.
  - **Divisor khác 0**, nếu:
    * Dividend bằng 0: In ra 0.0. Kết thúc chương trình.
    * Divisor bằng dương vô cùng, nếu Dividend:
      · bằng dương vô cùng: In ra NaN. Kết thúc chương trình.
      · bằng âm vô cùng: In ra NaN. Kết thúc chương trình.
      · In ra 0.0. Kết thúc chương trình.
    * Divisor bằng âm vô cùng, nếu Dividend:
      · bằng âm vô cùng: In ra NaN. Kết thúc chương trình.
      · bằng dương vô cùng: In ra NaN. Kết thúc chương trình.
      · In ra 0.0. Kết thúc chương trình.
    * Dividend bằng dương vô cùng:
      · Divisor lớn hơn 0: In ra Infinity. Kết thúc chương trình.
      · Divisor nhỏ hơn 0: In ra -Infinity. Kết thúc chương trình.
    * Dividend bằng âm vô cùng:
      · Divisor lớn hơn 0: In ra -Infinity. Kết thúc chương trình.
      · Divisor nhỏ hơn 0: In ra Infinity. Kết thúc chương trình.

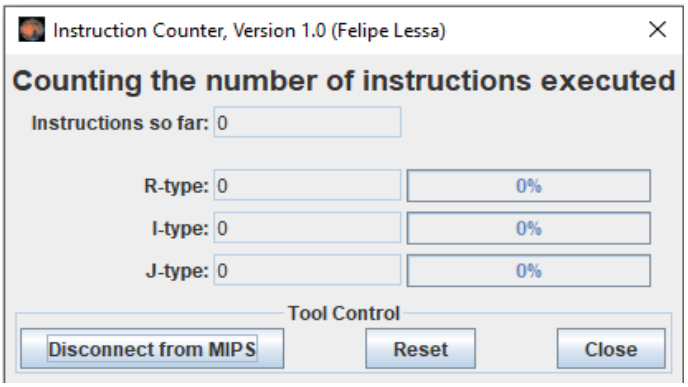
- B2. Tính phần mũ (Exponent) của thương (Quotient):
  Exponent(Quotient) = Exponent(Dividend) - Exponent(Divisor) + 127
- B3. Chia hai phần định trị, tính định trị của thương
  - Thanh ghi $s2 chứa kết quả chia hai phần định trị của thương.
  - Thêm bit 1 vào trước 23 bit phần định trị của Dividend và Divisor, lưu lần lượt vào thanh ghi $t4 và $t6.
  - Nếu $t4 nhỏ hơn $t6:
    * Dịch trái $t4 1 bit
  - Nếu $t4 lớn hơn hoặc bằng $t6:
    * Chuyển bit 0 của $s2 thành 1
    * Lấy $t4 trừ $t6 (hiệu lưu đè lên $t4)
  - Ta thực hiện 31 vòng lặp:
  - Mỗi vòng lặp gồm các bước:
    * Nếu $t4 nhỏ hơn $t6:
      · Dịch trái $s2 1 bit
      · Thêm 0 vào bit 0 của $s2 (Thao tác này không cần vì dịch trái 1 bit mặc định thêm 0 vào bit 0 của nó)
      · Dịch trái $t4 1 bit
    * Nếu $t4 lớn hơn hoặc bằng $t6:
      · Dịch trái $s2 1 bit
      · Chuyển bit 0 của $s2 thành 1

· Lấy $t4 trừ $t6 (hiệu lưu đè lên $t4)

- B4: Chuẩn hóa (dịch phải, tăng số mũ)
- B5: Làm tròn phần định trị để số bit biểu diễn thích hợp với độ chính xác đơn.
- B6: Xét bit dấu (Sign) của thương.
- B7: Kết hợp các phần Sign, Exponent, và Fraction ta được kết quả phép chia cần tính. In ra màn hình. Kết thúc chương trình.

4 Thống kê số lệnh, loại lệnh sử dụng trong chương trình

Ta sử dụng công cụ Instruction Counter trong MARS MIPS 4.5 để thống kê số lệnh trong chương trình:



- Tổng số lệnh thực thi (Instructions so far)
- Lệnh R (R-type)
- Lệnh I (I-type)
- Lệnh J (J-type)

5 Thời gian thực thi

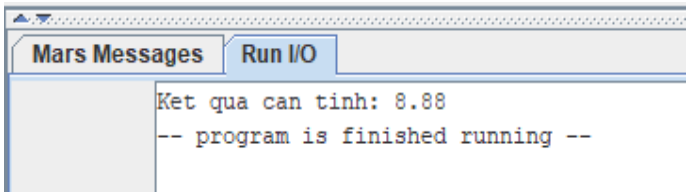
Để tính thời gian thực thi chương trình, ta sử dụng công thức:

$$CPUTime = \frac{CPUClockCycles}{ClockRate} = \frac{InstructionCount \times CPI}{ClockRate}$$

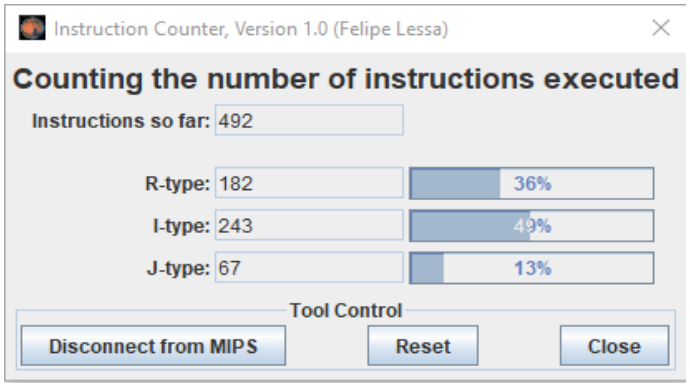
- CPU Time: Thời gian CPU xử lý chương trình.
- CPU Clock Cycles: Tổng chu kỳ thực thi.
- Clock Rate: Tần số (CR = 1GHz).
- Instruction Count: Tổng số lệnh thực thi của chương trình.
- CPI: Số chu kỳ thực hiện 1 lệnh (CPI = 1).

6 Kiểm tra chương trình

- **Kiểm tra 1:** Dữ liệu đầu vào: Số bị chia: 59.1408, số chia: 6.66:
  - Kết quả thực thi:



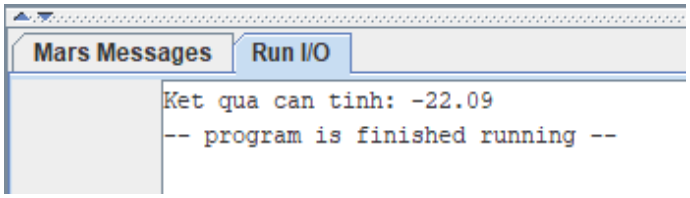
  - Số lệnh thực thi:



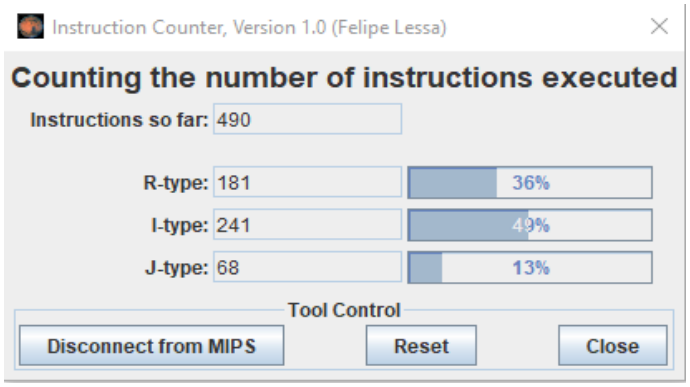
  - Thời gian thực thi:

$$CPUTime = \frac{InstructionCount \times CPI}{ClockRate} = \frac{492 \times 1}{10^9} = 4.92{\times}10^{-7}\ (s) = 492\ (ns)$$

- **Kiểm tra 2:** Dữ liệu đầu vào: Số bị chia: -69.9208, số chia: 3.12:
  - Kết quả thực thi:



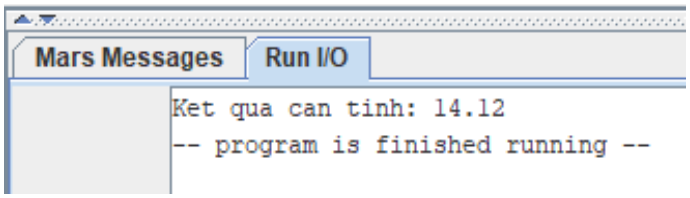
  - Số lệnh thực thi:



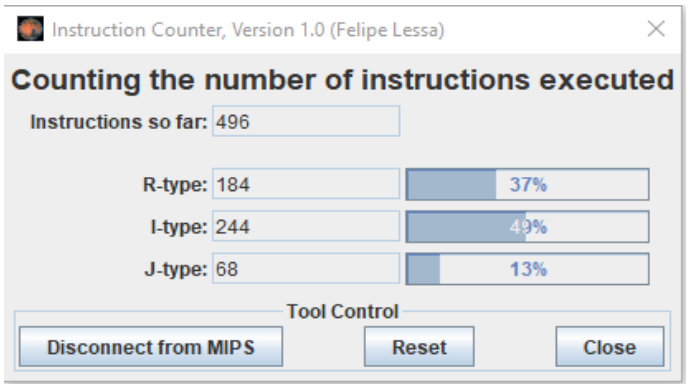
– Thời gian thực thi:

$$CPUTime = \frac{InstructionCount \times CPI}{ClockRate} = \frac{490 \times 1}{10^9} = 4.9 \times 10^{-7}\ (s) = 490\ (ns)$$

- **Kiểm tra 3:** Dữ liệu đầu vào: Số bị chia: -64.3872, số chia: -4.56:
  – Kết quả thực thi:



  – Số lệnh thực thi:



  – Thời gian thực thi:

$$CPUTime = \frac{InstructionCount \times CPI}{ClockRate} = \frac{496 \times 1}{10^9} = 4.96 \times 10^{-7}\ (s) = 496\ (ns)$$

7 Kết luận

Trong bài báo cáo trên, chúng em đã hoàn thành việc thực hiện phép chia 2 số thực chuẩn IEEE 754 chính xác đơn. Kết quả thu được của việc chia 2 số thực chuẩn IEEE 754 chưa hoàn toàn chính xác do độ chính xác của số thực IEEE 754 chính xác đơn bị giới hạn bởi khả năng lưu trữ các bit sau dấu thập phân. Ta có thể cải thiện được độ chính xác bằng cách: thay vì chính xác đơn (32 bit) ta sử dụng chính xác kép (64 bit) hoặc chính xác bậc bốn (128 bit).

Tài liệu

[1] David A. Patterson & John L. Hennessy – *Computer Organization and Design: The Hardware/ Software Interface*