# Báo cáo Bài tập lớn

Bộ môn : Kiến trúc máy tính

Nhóm: 02

Lớp: L06 – L12

Đề tài: Đề 3: (2sv) Nhân 2 thanh ghi. Cho 2 thanh ghi A (64 bit) và thanh ghi B (64 bit). Sử dụng hợp ngữ assembly MIPS để hiện thực phép nhân 2 thanh ghi đó. Kết quả được xuất ra console(hiển thị ở dạng HEX và dạng thập phân), bit cao của thanh ghi là bit dấu.

| STT | Họ và tên | MSSV |
|---|---|---|
| 1 | Nguyễn Cao Cường | 2210429 |
| 2 | Nguyễn Hoàng Tú | 2213846 |

# I. GIẢI PHÁP HIỆN THỰC

Trong báo cáo này, chúng tôi sẽ trình bày giải pháp hiện thực phép nhân hai thanh ghi A và B (mỗi thanh ghi 64 bit) bằng hợp ngữ Assembly MIPS:

Ý tưởng thực hiện giải pháp:

1. Các thanh ghi sử dụng để lưu giá trị cho biến
   Ở bài toán này, chúng tôi thực hiện việc shift left các bit của A để thực hiện phép nhân, vì vậy có thể sẽ dẫn đến việc tràn bit, do đó chúng tôi sử dụng thêm 2 thanh ghi $t0, $t1 để xử lí trường hợp này.

- Chúng ta sẽ dùng thanh ghi từ $t0 - $t3 để lưu trữ giá trị của A, thực hiện gán giá trị của $t0, $t1 là 0 và loading 64 bit giá trị của A vào $t2, $t3.
- 64 bit giá trị của B được loading vào $t4, $t5
- Đối với C (C= A*B), chúng ta dùng 4 thanh ghi $s0 - $s3 để biểu diễn 128 bit của C, và thanh ghi $t6 tượng trưng cho bit dấu.

2. Các hàm sử dụng và chức năng của chúng

- Multiply_64_64_unsigned
  Chức năng: Nhân 64 bit A với $t2-$t3 và 64 bit B trong $t4-$t5

- Add_128_bit_unsigned
  Chức năng: Hỗ trợ cho hàm Multiply_64_64_unsigned

- Div_128_64_unsigned
  Chức năng:

  + Hỗ trợ cho hàm print_result trong việc in giá trị của C dưới dạng thập phân

  + Chia 128 bit A (a0-a3) thành 64 bit B (t2-t3), kết quả được ghi trong $v0,$v1 và phần còn lại trong t2-t3

- Sub_128_bit_unsigned
  Chức năng: Hỗ trợ cho hàm Div_128_64_unsigned

- Main:
  Chức năng: luồng thực thi chương trình chính
  Các bước tiến hành:
  1. Lấy giá trị A, B từ .data.
  2. Kiểm tra dấu và chuyển về unsigned nếu cần.
  3. Tính C = A * B (dùng multiply_64_64_unsigned).
  4. Biểu diễn dưới dạng thập phân:

Lấy giá trị của C chia cho 10^19 để tách thành 2 phần, mỗi phần chia lần lượt cho 10^18, và 10^9, sau đó lưu kết quả và phần dư vào các biến int_dec_result_0 đến int_dec_result_5.

Xuất giá trị dưới dạng thập phân, kiểm tra dấu.

5. Biểu diễn dưới dạng hex:

Nếu dấu là âm, đảo ngược bit từ $s0 đến $s3.

Xuất giá trị dưới dạng hex từ $s0 đến $s3.

3. Kết quả

Kết quả sẽ được xuất ra console ở dạng HEX và DEC, trong đó bit cao của thanh ghi là bit dấu.

- Dạng HEX: Kết quả sau khi tính toán sẽ được trình bày trên 4 dòng, với mỗi dòng tương ứng là giá trị 32 bit từ cao đến thấp.
- Dạng DEC: Kết quả được trình bày dưới dạng biểu thức số học, với mỗi thành phần có dạng a x $10^n$, với a là hệ số và n là hệ số mũ.

# II. GIẢI THUẬT

**1. Add_128_bit_unsigned : C <- C+A C($a0-$a3), A($t0-$t3), C>0 & A>0**

+ Thanh ghi $t8 dùng để giữ giá trị bit tràn ra trong trường hợp cộng có nhớ.

+ Cộng $t3 và $a3 :

- Nếu cả 2 bit đầu của $t3 và $a3 đều là 0, trường hợp cộng có nhớ sẽ không xảy ra. Khi này chỉ việc cho thanh ghi $a3 nhận giá trị tổng của $t3 và $a3(unsigned).
- TH ít nhất 1 trong 2 thanh ghi có bit đầu là 1, vẫn cho $a3 nhận giá trị tổng của $t3 và $a3 (unsigned), tiếp theo so sánh $a3 mới và $t3, nếu $a3 nhỏ hơn $t3 (unsigned) thì có nhớ, khi này cho $t8 = 1.

+ Cộng $t2 và $a2 :

- Nếu cả 2 bit đầu của $t2 và $a2 đều là 0, trường hợp có nhớ sẽ không xảy ra. Cho thanh ghi $a2 nhận giá trị tổng của $a2,$t2,$t8. Đưa $t8 = 0.
- Trường hợp ít nhất 1 trong 2 thanh ghi có bit đầu là 1, vẫn cho $a2 nhận giá trị tổng của $t2 và $a2 (unsigned), tiếp theo so sánh $a2 mới và $t2, nếu $a2 nhỏ hơn $t2

(unsigned) thì có nhớ, sau đó tiếp tục cho $a2 bằng tổng của $a2 và $t8, khi này kiểm tra xem $a2 = 0 hay không, nếu có thì có nhớ => Nếu có nhớ ở 1 trong 2 lần kiểm tra $a2, cho $t8 = 1, còn không thì để $t8 về 0.

+ Cộng $t1 và $a1 : Tương tự $t2 và $a2

+ Cộng $t0 và $a0 : Khi này đảm bảo không có nhớ. Chỉ việc cho $a2 nhận giá trị tổng của $a2,$t2,$t8.

**2.multiply_64_64_unsigned : C = A*B, C($a0-$a3), A($t2-$t3),B($t4-$t5)**

+ Đầu tiên gán cho C = 0, vùng để A dịch bit trái là từ $a0-$a3

Bước 1 : Kiểm tra bit cuối của B, nếu = 1 thì C <- C+A (dùng add_128_bit_unsigned).

Bước 2 : Dịch trái A và dịch phải B.

+ Dịch trái A :

- Dịch trái $a0, nếu bit đầu của $a1 là 1 thì cho $a0 <-$a0+1.
- Tương tự cho $a1,$a2.
- Dịch trái $a3.

+ Dịch phải B :

- Dịch phải $t5, nếu bit cuối của $t4 là 1 thì cho $t5 <- $t5 + 0x80000000.
- Dịch phải $t4.

Bước 3 : Kiểm tra B, nếu B = 0 thì kết thúc, không thì quay lại bước 1.

**3.Sub_128_bit_unsigned : C <- C-A, C($a0-$a3), A($t0-$t3), C > A > 0**

So sánh $a3 và $t3 (unsigned), nếu $a3<$t3 thì $a2 <- $a2-1. Sau đó cho $a3 nhận giá trị bằng $a3-$t3. Tương tự cho $a2-$a0.

**4. Div_128_64_unsigned : Chia C($a0-$a3) thành A($t2-$t3), kết quả được lưu trong $v0-$v1, và phần còn lại được lưu trong $a2-$a3.**

$t0 <- $t2, $t1 <- $t3. $t2 = 0, $t3 = 0. Một thanh ghi đếm $s5 = 65.

Vùng để A dịch phải là từ $t0-$t3

Bước 1 : So sánh C và A, nếu A<=C thì C <- C-A.

Bước 2 : Shift left $v0-$v1, nếu A<=C thì cộng $v1 cho 1.

Bước 3 : $s5 = $s5-1, nếu $s5 thì dừng, nếu không thì quay lại bước 1.

**5.Main :**

Lấy giá trị A, B từ .data

Xét dấu A,B sau đó chuyển về unsigned. Một biến lưu dấu của A*B là $t6, nếu $t6 = 1 thì A*B sẽ mang giá trị âm.

Tính C = A*B (dùng multiply_64_64_unsigned). Lưu hết các giá trị đó vào từ $s0 đến $s3.

+ Biểu diễn dưới dạng thập phân :

- Set A = 10^19. Sau đó cho C/A (dùng div_128_64_unsigned), lưu giá trị $v0-$v1 vào .data (int_dec_result_0 và 1), để nguyên remainder.
- Set A = 10^18. Tiếp tục chia C cho A, lưu $v1 vào int_dec_result_3, vẫn giữ remainder và set A = 10^9. Chia C cho A, lưu $v1 vào int_dec_result_4, lưu phần dư $a3 vào int_dec_result_5.
- Lấy lại giá trị từ int_dec_result_0 và 1 vào $a2 và $a3.
- Set A = 10^18. Tiếp tục chia C cho A, lưu $v1 vào int_dec_result_0, vẫn giữ remainder và set A = 10^9. Chia C cho A, lưu $v1 vào int_dec_result_1, lưu phần dư $a3 vào int_dec_result_2.
- Khi này có thể xuất giá trị dưới dạng thập phân. Nếu $t6 = 1 thì dấu của A*B là âm.

+ Biểu diễn dưới dạng hex :

Nếu $t6 = 1, lần lượt đảo bit từ $s0 đến $s3, nếu không thì không làm gì.

Sau đó, xuất toàn bộ giá trị từ $s0 đến $s3 dưới dạng Hex.

# III. CÁC TRƯỜNG HỢP TEST

Các trường hợp test đã được trình bày trong file excel đính kèm !