VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

KIẾN TRÚC MÁY TÍNH (CO2007)

Assignment

CHIA HAI SỐ NGUYÊN 32 BIT

Advisor: Nguyễn Xuân Minh
Students: Bùi Đình Hoàng Hùng - 1913599.
Đào Quốc Khánh - 2013452.

HO CHI MINH CITY, NOVEMBER 2021

Contents

1 Giải pháp hiện thực

- Chương trình hiện thực giải thuật chia 2 số nguyên 32 bit có dấu (từ -2.147.483.648 đến 2.147.483.647). Dữ liệu đầu vào được đọc từ file lưu trữ dạng nhị phân trên đĩa INT2.BIN.

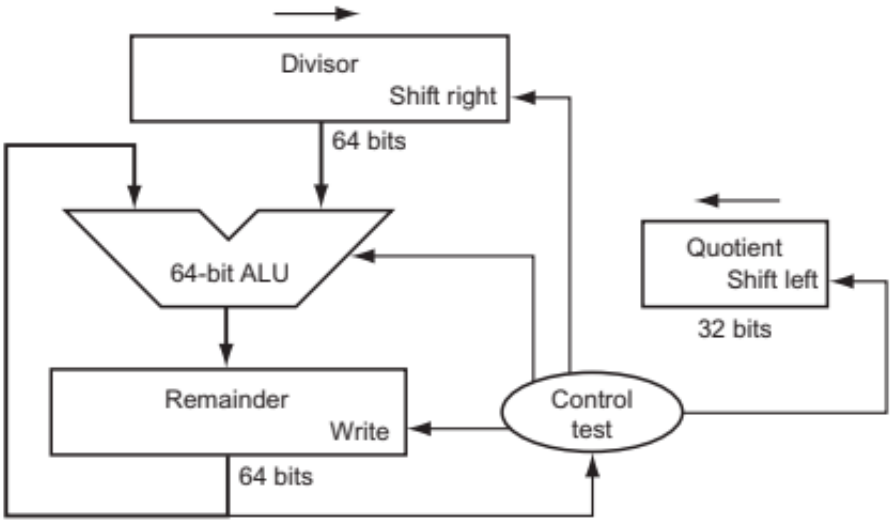
- Chương trình được hiện thực theo lưu đồ hình 3.9 trong Textbook (*Computer Organization and Design: The Hardward/Software Interface* by *David A. Patterson* and *John L. Henessy*)

- Yêu cầu nhập vào 2 số nguyên tương ứng số bị chia và số chia, sau đó xuất ra thương và số dư của chúng.

- Chương trình được viết và chạy trên MARS MIPS 4.5, mã nguồn được lưu trong file Mn_nhom06.asm.

2 Giải thuật

2.1 Quy ước và khởi tạo:



- Số chia và số bị chia sẽ được lưu trữ bởi Divisor và Dividend (ở dạng 64 bits), trong đó số chia sẽ được khởi tạo vào 32 bits cao của Divisor và số bị chia sẽ được khởi tạo vào 32 bits thấp của Dividend.

+ 32 bits cao của Dividend quy ước lưu trữ trong thanh ghi $t0.
+ 32 bits thấp của Dividend quy ước lưu trữ trong thanh ghi $t1.
+ 32 bits cao của Divisor quy ước lưu trữ trong thanh ghi $t2.
+ 32 bits thấp của Divisor quy ước lưu trữ trong thanh ghi $t3.

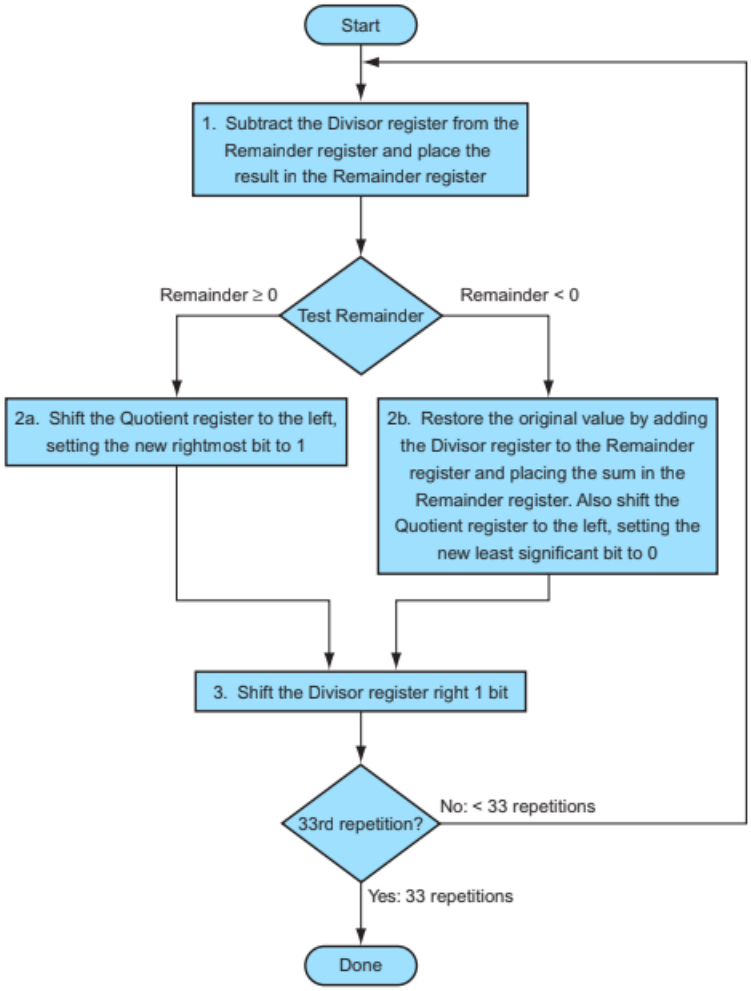
- Giải thuật yêu cầu phải làm việc với các phép toán liên quan đến các con số 64 bits, do đó chúng ta sẽ phải hiện thực thêm các hàm:

+ minus: tính Dividend - Divisor (hiệu 2 số 64 bits) với các tham số truyền vào được quy ước lưu trữ trong các thanh ghi $a0, $a1, $a2, $a3 (tương ứng lần lượt lưu trữ 32 bits cao của Dividend, 32 bits thấp của Dividend, 32 bits cao của Divisor, 32 bits thấp của Divisor). Kết quả trả về sẽ được lưu trong 2 thanh ghi $v0 (32 bits cao) và $v1 (32 bits thấp).

+ shift_right: dịch phải Divisor 1 bit.

- Thương số (Quotient) được khởi tạo bằng 0 và quy ước lưu trữ trong thanh ghi $s2.

2.2 Giải thuật:



- Ta sẽ thực hiện giải thuật chia đối với hai số nguyên không âm bằng cách lấy giá trị tuyệt

đối của số bị chia và số chia. Kết quả sau cùng sẽ dựa vào dấu của dữ liệu nhập vào để xét dấu của thương và số dư.

- Giải thuật được hiện thực theo Textbook của *David A. Patterson* and *John L. Henessy*

+ B1: Nhập vào số bị chia và số chia, khởi tạo Dividend và Divisor.

+ B2: Kiểm tra xem Divisor có bằng 0 hay không, nếu có thì xuất ra cảnh báo và kết thúc chương trình.

+ B3: Tính Dividend - Divisor:

-> Dividend - Divisor < 0: dịch trái Quotient 1 bit, bật bit cuối cùng lên 0.

-> Dividend - Divisor ≥ 0: Dividend = Dividend - Divisor, sau đó dịch trái Quotient 1 bit, bật bit cuối cùng lên 1.

+ B4: Dịch phải Divisor 1 bit, quay lại B3, lặp 33 lần.

+ B5: Xuất kết quả: thương = Quotient, số dư = Dividend.

+ B6: Kết thúc.

3 Thống kê các lệnh đã sử dụng trong chương trình

3.1 Nhóm lệnh R-type

Lệnh	Số câu lệnh	Cú pháp đã sử dụng	Cách thức hoạt động
srl	2	srl $t1, $t2, 10	Shift right logical: Set $t1 to result of shifting $t2 right by number of bits specified by immediate.
sll	3	sll $t1, $t2, 10	Shift left logical: Set $t1 to result of shifting $t2 left by number of bits specified by immediate.
addi	12	addi $t1, $t2, 100000	Addition immediate: set $t1 to ($t2 plus 32-bit immediate).
addiu	1	addiu $t1, $t2, -100	Addition immediate unsigned without overflow : set $t1 to ($t2 plus signed 16-bit immediate), no overflow.
subu	3	subu $t1, $t2, $t3	Subtraction unsigned without overflow: set $t1 to ($t2 minus $t3), no overflow.
or	1	or $t1, $t2, $t3	Bitwise OR: Set $t1 to bitwise OR of $t2 and $t3.
and	1	and $t1, $t2, 100	Bitwise AND: Set $t1 to ($t2 bitwise-AND 16-bit unsigned immediate).
mul	7	mul $t1, $t2, $-100	Multiplication without overflow: Set HI to high-order 32 bits, LO and $t1 to low-order 32 bits of the product of $t2 and 16-bit signed immediate (use mfhi to access HI, mflo to access LO)

-> Tổng số câu lệnh: 30

3.2 Nhóm lệnh Load

Lệnh	Số câu lệnh	Cú pháp đã sử dụng	Cách thức hoạt động
la	13	la $t1, label	Load Address: Set $t1 to label address
lw	7	lw $t1, ;label	Load Word: Set $t1 to contents of memory word at dividend address
li	30	li $t1, 100	Load Immediate: Set $v0 to unsigned 16-bit immediate (zero-extended))

-> Tổng số câu lệnh: 50

3.3 Nhóm lệnh Store

Lệnh	Số câu lệnh	Cú pháp đã sử dụng	Cách thức hoạt động
sw	14	sw $t1, label	Store Word: Store $t1 contents into memory word at label's address.
move	13	move $t1, $t2	MOVE: Set $t1 to contents of $t2.

-> Tổng số câu lệnh: 27

3.4 Nhóm lệnh Branch

Lệnh	Số câu lệnh	Cú pháp đã sử dụng	Hoạt động
beqz	1	beqz $t1, label	Branch if EQual Zero: Branch to statement at zero_divisor if $t1 is equal to zero.
bqez	5	bgez $t1, label	Branch if greater than or equal to zero: Branch to statement at label address if $t1 is greater than or equal to zero
bltz	8	bltz $t1, label	Branch if less than zero: Branch to statement at label address if $t1 is less than zero

- Tổng số câu lệnh: 14

3.5 Nhóm lệnh Jump

Lệnh	Số câu lệnh	Cú pháp đã sử dụng	Cách thức hoạt động
jr	4	jr $ra	Jump register unconditionally: Jump to statement whose address is in $ra
j	10	j target	Jump unconditionally: Jump to statement at target address
jal	7	jal target	Jump and link: Set $ra to Program Counter (return address) then jump to statement at target address

-> Tổng số câu lệnh: 21

4 Thời gian chạy của chương trình

- Clock rate = 1GHz = 1.000.000.000 Hz
- Chương trình có tổng cộng 142 câu lệnh, trong đó:
  + 30 câu lệnh thuộc nhóm lệnh R-type.
  + 50 câu lệnh thuộc nhóm lệnh Load.
  + 27 câu lệnh thuộc nhóm lệnh Store.
  + 14 câu lệnh thuộc nhóm lệnh Branch.
  + 21 câu lệnh thuộc nhóm lệnh Jump.
- Thời gian chạy của chương trình sẽ là:

$$t = \frac{(30 * 4 + 50 * 5 + 27 * 4 + 14 * 3 + 21 * 3)}{1.000.000.000} = 583ns$$

5 Kết quả kiểm thử

5.1 Input1: dividend = 6; divisor= 3

```
dividend = 6
divisor = 3
File Open Successfully
quotient = 2
remainder = 0
-- program is finished running --
```

-> KẾT QUẢ ĐÚNG

5.2 Input2: dividend = 5; divisor= 2

```
dividend = 5
divisor = 2
File Open Successfully
quotient = 2
remainder = 1
-- program is finished running --
```

-> KẾT QUẢ ĐÚNG

5.3 Input3: dividend = 0; divisor= 3

```
dividend = 0
divisor = 3
File Open Successfully
quotient = 0
remainder = 0
-- program is finished running --
```

-> KẾT QUẢ ĐÚNG

5.4 Input4: dividend = 106; divisor= 15

```
dividend = 106
divisor = 15
quotient = 7
remainder = 1
-- program is finished running --
```

-> KẾT QUẢ ĐÚNG

5.5 Input5: dividend = -15; divisor= -3

```
dividend = -15
divisor = -3
File Open Successfully
quotient = 5
remainder = 0
-- program is finished running --
```

-> KẾT QUẢ ĐÚNG

5.6 Input6: dividend = -106; divisor= 15

```
dividend = -106
divisor = 15
File Open Successfully
quotient = -7
remainder = -1
-- program is finished running --
```

-> KẾT QUẢ ĐÚNG

5.7 Input7: dividend = 999; divisor= 4

```
dividend = 9999
divisor = 4
File Open Successfully
quotient = 2499
remainder = 3
-- program is finished running --
```

-> KẾT QUẢ ĐÚNG