COMPUTER ENGINEERING

KIẾN TRÚC MÁY TÍNH

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Tuần 8

PHÉP TOÁN SỐ HỌC TRÊN MÁY TÍNH

(Tiếp theo)

COMPUTER ENGINEERING

PHÉP TOÁN SỐ HỌC TRÊN MÁY TÍNH

Mục tiêu:

Hiểu các phép toán số học trên số nguyên trong máy tính:

- ✓**Hiểu các phép toán cộng, trừ, nhân và chia**
- ✓**Cách thiết kế mạch nhân và chia**

Slide được dịch và các hình được lấy từ sách tham khảo:

Computer Organization and Design: The Hardware/Software Interface, Patterson, D. A., and J. L. Hennessy, Morgan Kaufman, Revised Fourth Edition, 2011.

COMPUTER ENGINEERING

PHÉP TOÁN SỐ HỌC TRÊN MÁY TÍNH

1. **Giới thiệu**
2. **Phép cộng & Phép trừ**
3. **Phép Nhân**
4. **Phép chia**

Phép Chia

- Ngược lại của phép nhân là phép chia.
- Trường hợp ngoại lệ – chia 0.

Ví dụ:

$$
\begin{array}{rll}
 & 1001_{ten} & \text{Quotient} \\
\text{Divisor } 1000_{ten} & \overline{)\,1001010_{ten}} & \text{Dividend} \\
 & \underline{-1000} & \\
 & 10 & \\
 & 101 & \\
 & 1010 & \\
 & \underline{-1000} & \\
 & 10_{ten} & \text{Remainder}
\end{array}
$$

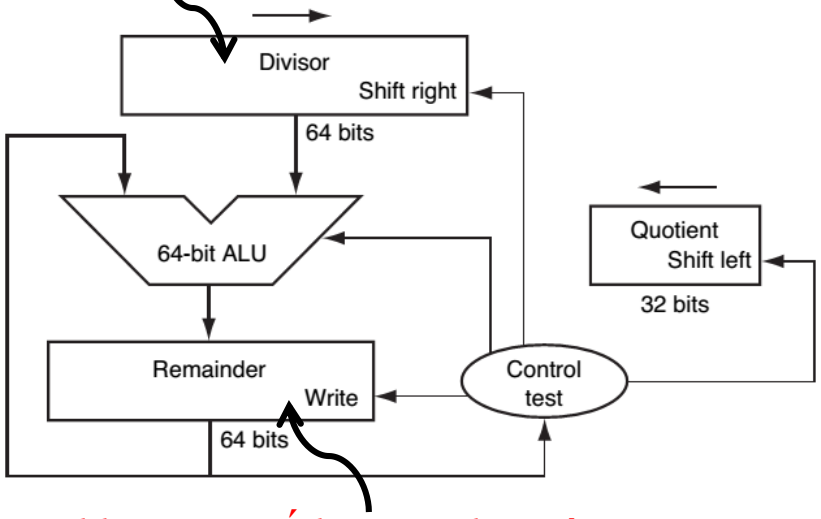
Divisor: số chia
Dividend: số bị chia
Quotient: thương số
Remainder: số dư

COMPUTER ENGINEERING

Phép Chia

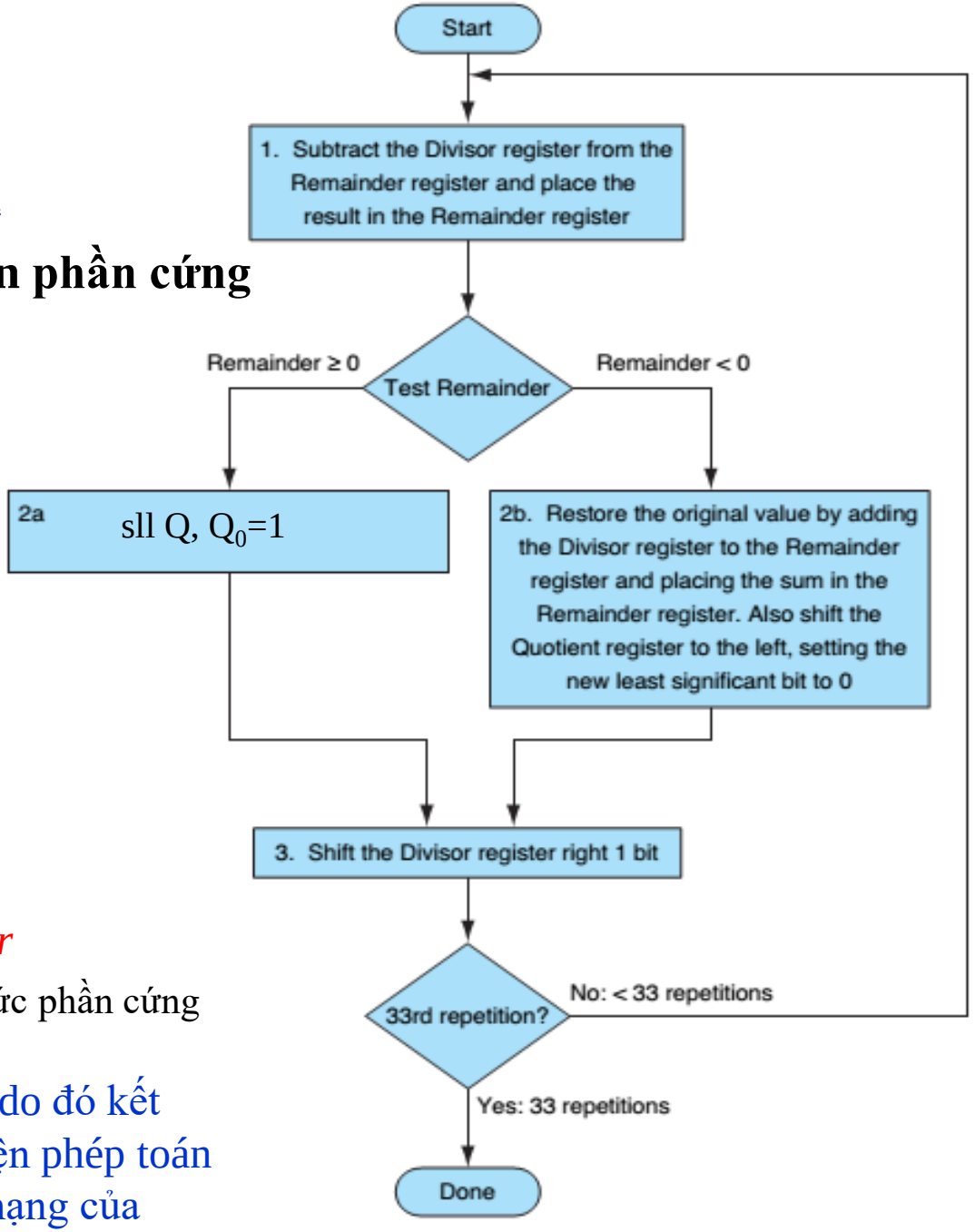
Giải thuật thực hiện phép chia trên phần cứng

Khi khởi tạo, số chia đưa vào nữa cao Divisor



Khi khởi tạo, số bị chia đưa vào Remainder

Hình 1. Sơ đồ các khối hiện thực phép chia ở mức phần cứng



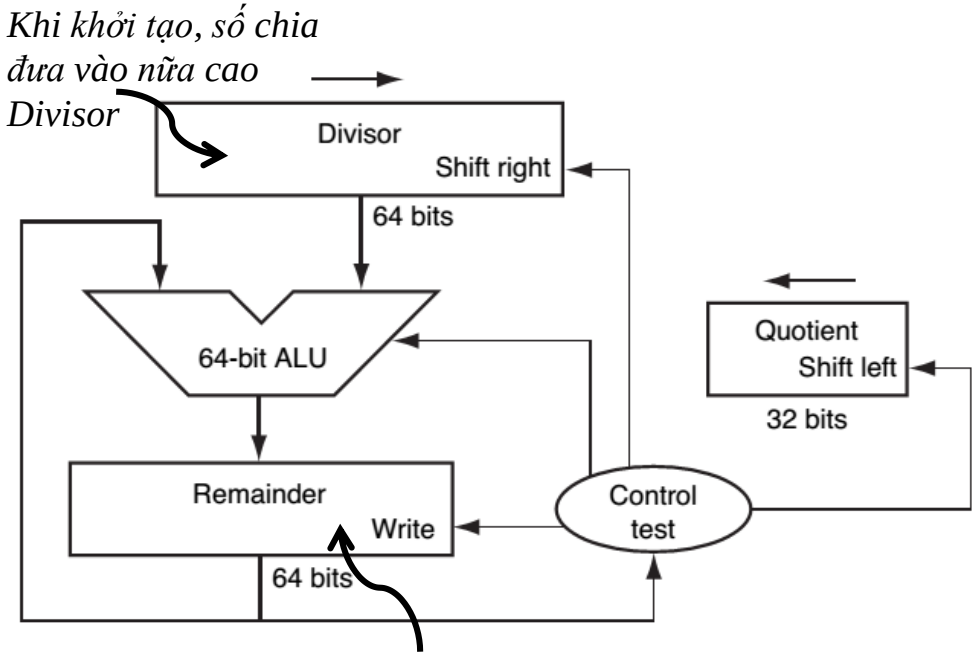
Hình 2. Lưu đồ giải thuật của phép chia

Chú ý: Hai số chia và bị chia là số dương, do đó kết quả thương và số dư là không âm. Thực hiện phép toán trên số dương, do đó, thương và các toán hạng của phép chia có giá trị là 32 bit, bỏ qua các số có dấu.

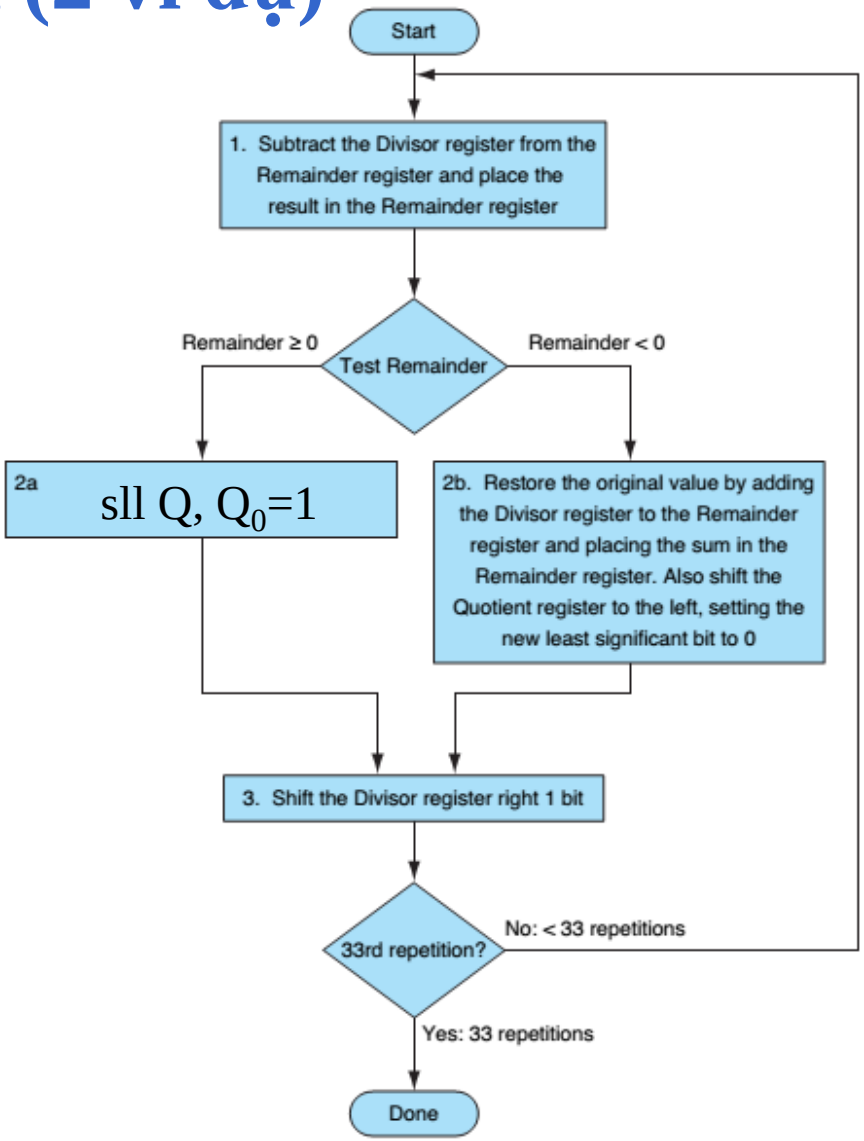
Ví dụ cho phép chia (2 ví dụ)

Ví dụ 1:

Thực hiện phép chia $50_{(8)}/23_{(8)}$ (sử dụng số 6 bit không dấu) theo cấu trúc phần cứng như hình

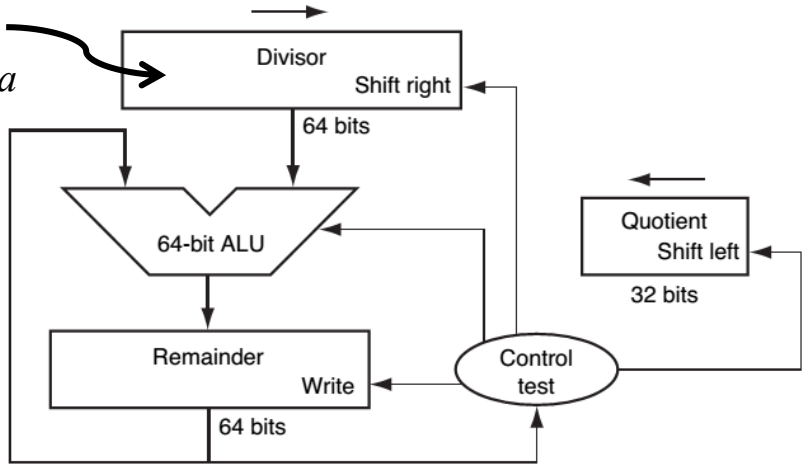


Khi khởi tạo, số bị chia đưa vào Remainder



Lưu đồ giải thuật đi kèm cho cấu trúc phần cứng

Khi khởi tạo, số chia đưa vào nữa cao Divisor



Ví dụ 1:

$50_{(8)}/23_{(8)}$ = ?

Dividend = 50_8 = $101\ 000_2$

Divisor = 23_8 = $010\ 011_2$

Cấu trúc phần cứng như hình vẽ là đang làm việc trên phép chia số 32 bits

Có: thanh ghi divisor 64 bits

thanh ghi quotient là 32 bits

thanh ghi remainder là 64 bits

Ví dụ 1 yêu cầu phép chia dùng số 6 bits không dấu, sử dụng cấu trúc phần cứng tương tự như hình, vậy các thanh ghi trong ví dụ cần được khởi tao với số bit tương ứng:

=> thanh ghi divisor 12 bits (giá trị khởi tao **010011**000000 – **6 bits cao là giá trị của divisor, 6 bits thấp đưa 0 vào**)

thanh ghi quotient là 6 bits (giá trị khởi tạo 000000)

thanh ghi remainder là 12 bits (giá trị khởi tạo 000000**101000** - 6 bits cao đưa 0 vào, 6 bits thấp đưa dividend vào)

-Sau khi khởi tạo xong. Mỗi vòng lặp (interation) sẽ gồm 3 bước:

- B1. Lấy toàn bộ remainder trừ divisor (hiệu lưu đè lên giá trị remainder hiện đang có)
- B2. Kiểm tra hiệu vừa tính ở trên là âm hay dương (kiểm tra bit trọng số cao nhất, nếu 1 là âm, nếu 0 là dương):

  Nếu âm:

  - Lấy giá trị hiện tại của remainder cộng với divisor, tổng lưu lại vào remainder
  - Dich trái quotient 1 bit
  - Thêm 0 vào bit 0 của quotient (thật ra thao tác này không cần, vì dịch trái 1 bit mặc định đã thêm 0 vào bit 0 của nó)

  Nếu dương:

  - Dich trái quotient 1 bit
  - Chuyển bit 0 của quotient thành 1
- B3. Dịch phải Divisor 1 bit

- Số vòng lặp cho giải thuật này đúng bằng số bit dùng biểu diễn + 1 (ví dụ 1 yêu cầu dùng số 6 bit, thì có 7 vòng lặp)

- Sau khi kết thúc số vòng lặp, giá trị trong thanh ghi quotient chính là kết quả phép chia, giá trị trong remainder là phần dư

Step	Action	Quotient	Divisor	Remainder
0	Initial Vals (Giá trị khởi tạo)	000 000	010 011 000000	000000 101000

Ví dụ 1:

$50_{(8)}/23_{(8)}$ = ?

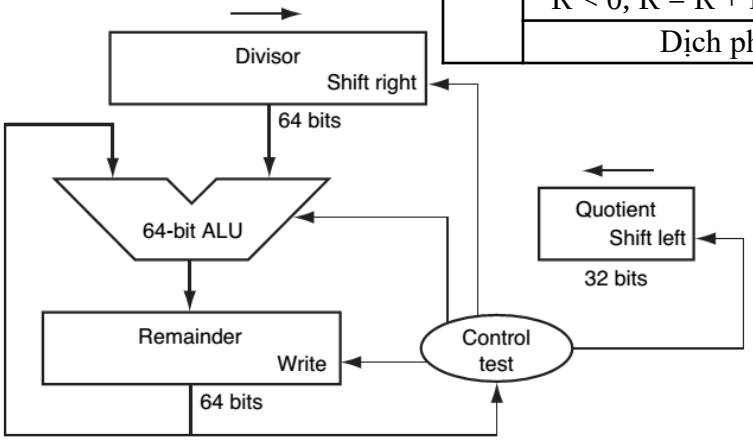
Dividend = 50_8 = $101\ 000_2$

Divisor = 23_8 = $010\ 011_2$

Step	Action	Quotient	Divisor	Remainder
0	Initial Vals	000 000	010 011 000 000	000 000 101 000
1	R = R – D	000 000	010 011 000 000	**1**01 101 101 000
	R < 0, R = R + D, dịch trái Q 1 bit	000 000	010 011 000 000	000 000 101 000
	Dịch phải D 1 bit	000 000	001 001 100 000	000 000 101 000
2	R = R – D	000 000	001 001 100 000	**1**10 111 001 000
	R < 0, R = R + D, dịch trái Q 1 bit	000 000	001 001 100 000	000 000 101 000
	Dịch phải D 1 bit	000 000	000 100 110 000	000 000 101 000
3	R = R – D	000 000	000 100 110 000	**1**11 011 111 000
	R < 0, R = R + D, dịch trái Q 1 bit	000 000	000 100 110 000	000 000 101 000
	Dịch phải D 1 bit	000 000	000 010 011 000	000 000 101 000
4	R = R – D	000 000	000 010 011 000	**1**11 110 010 000
	R < 0, R = R + D, dịch trái Q 1 bit	000 000	000 010 011 000	000 000 101 000
	Dịch phải D 1 bit	000 000	000 001 001 100	000 000 101 000
5	R = R – D	000 000	000 001 001 100	**1**11 110 111 100
	R < 0, R = R + D, dịch trái Q 1 bit	000 000	000 001 001 100	000 000 101 000
	Dịch phải D 1 bit	000 000	000 000 100 110	000 000 101 000
6	R = R – D	000 000	000 000 100 110	**0**00 000 000 010
	R> 0, dịch trái Q 1 bit, Q_0 = 1	000 001	000 000 100 110	000 000 000 010
	Dịch phải D 1 bit	000 001	000 000 010 011	000 000 000 010
7	R = R – D	000 001	000 000 010 011	**1**11 111 101 111
	R < 0, R = R + D, dịch trái Q 1 bit	000 010	000 000 010 011	000 000 000 010
	Dịch phải D 1 bit	000 010	000 000 001 001	000 000 000 010

Thương số

Phần dư



Ký hiệu: Q, D và R lần lượt là viết tắt của Quotion, Divisor và Remainder

Phép Chia

COMPUTER ENGINEERING

Giải thuật thực hiện phép chia trên phần cứng

Ví dụ 2: thực hiệp phép chia cho 2 số 4 bit sau:

$7_{10} : 2_{10}$ hay $0111_2 : 0010_2$

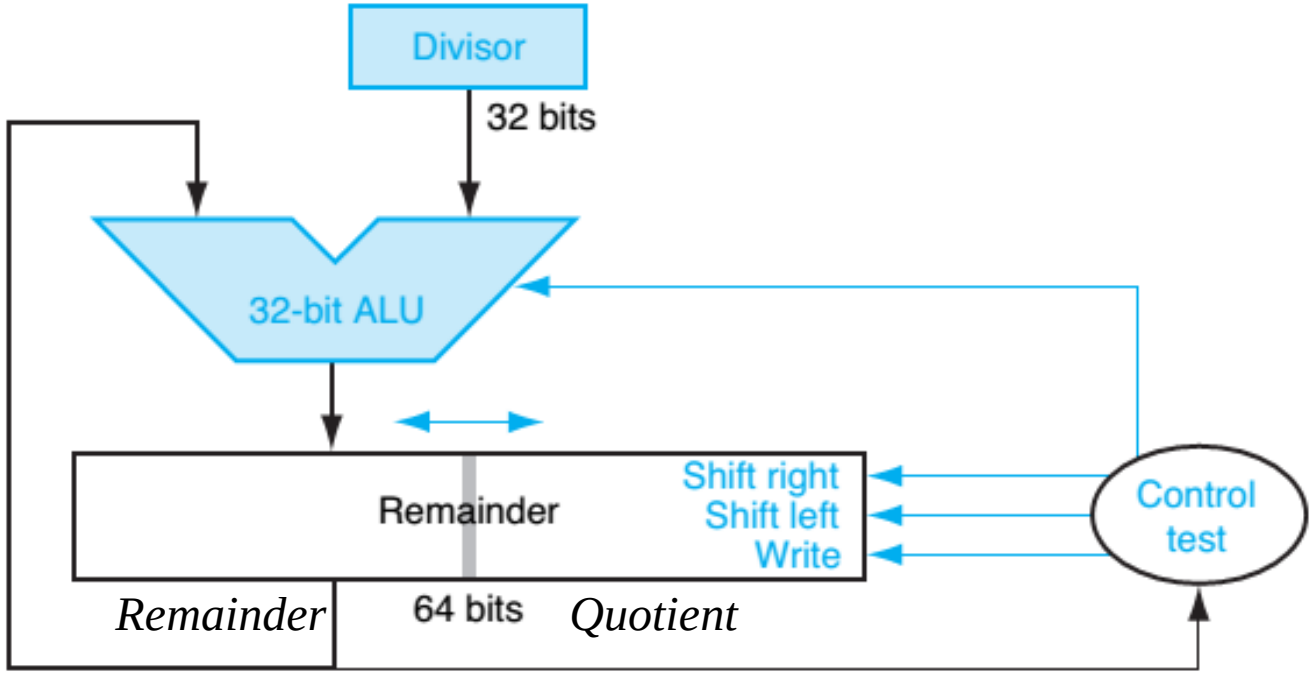
tức Remainder = Remaider + Divisor

Bảng thực hiện giải thuật phép chia theo từng bước

Iteration	Step	Quotient	Divisor	Remainder
0	Initial values	0000	0010 0000	0000 0111
1	1: Rem = Rem – Div	0000	0010 0000	①110 0111
	2b: Rem < 0 ⟹ +Div, sll Q, Q0 = 0	0000	0010 0000	0000 0111
	3: Shift Div right	0000	0001 0000	0000 0111
2	1: Rem = Rem – Div	0000	0001 0000	①111 0111
	2b: Rem < 0 ⟹ +Div, sll Q, Q0 = 0	0000	0001 0000	0000 0111
	3: Shift Div right	0000	0000 1000	0000 0111
3	1: Rem = Rem – Div	0000	0000 1000	①111 1111
	2b: Rem < 0 ⟹ +Div, sll Q, Q0 = 0	0000	0000 1000	0000 0111
	3: Shift Div right	0000	0000 0100	0000 0111
4	1: Rem = Rem – Div	0000	0000 0100	⓪000 0011
	2a: Rem ≥ 0 ⟹ sll Q, Q0 = 1	0001	0000 0100	0000 0011
	3: Shift Div right	0001	0000 0010	0000 0011
5	1: Rem = Rem – Div	0001	0000 0010	⓪000 0001
	2a: Rem ≥ 0 ⟹ sll Q, Q0 = 1	0011	0000 0010	0000 0001
	3: Shift Div right	0011	0000 0001	0000 0001

Phép Chia

Giải thuật thực hiện phép chia trên phần cứng có cải tiến
(Sinh viên tự tham khảo thêm)



Cấu trúc phần cứng phép chia có cải tiến

Phép Chia

Phép chia có dấu

Nếu phép chia có dấu

- Bước 1. Bỏ qua dấu, thực hiện phép chia thông thường
- Bước 2. Xét dấu
  - ✓ Dấu của thương sẽ trái với dấu hiện tại nếu dấu của số chia và số bị chia trái ngược nhau
  - ✓ Dấu của số dư:

Các xác định bit dấu cho số dư bằng công thức sau:

Số bị chia = Thương × Số chia + Số dư

➔ **Số dư = Số bị chia – (Thương × Số chia)**

Ví dụ: – 7 : +2 thì thương = – 3, dư = –1

Kiểm tra kết quả:

–7 = –3 × 2 + (–1) = –6 – 1

Phép Chia

Phép chia trong MIPS

- Trong cấu trúc phần cứng cho phép nhân có cải tiến, hai thanh ghi ***Hi*** và ***Lo*** được ghép lại để hoạt động như thanh ghi 64 bit của Product/Multiplier

  Quan sát cấu trúc phần cứng cho phép nhân có cải tiến và phép chia có cải tiến, rõ ràng hai cấu trúc này tương tự nhau.

  Từ đó, MIPS cũng sử dụng hai thanh ghi ***Hi*** và ***Lo*** cho cả phép nhân và chia.

- Sau khi phép chia thực hiện xong:
  - ✓ ***Hi*** chứa phần dư
  - ✓ ***Lo*** chứa thương số
- Để xử lý cho các số có dấu và số không dấu, MIPS có 2 lệnh: phép chia có dấu (***div***), và phép chia không dấu (***divu***).

Tổng kết:

- Hiểu quy tắc thực hiện các phép toán số học (cộng, trừ, nhân và chia) trên số nguyên trong máy tính
- Hiểu cách thiết kế mạch nhân và chia cơ bản cho số nguyên trong máy tính

PHÉP TOÁN SỐ HỌC TRÊN MÁY TÍNH

- **Lý thuyết: Đọc sách tham khảo**
  - Mục: 3.1, 3.2, 3.3, 3.4
  - Sách: *Computer Organization and Design: The Hardware/Software Interface*, Patterson, D. A., and J. L. Hennessy, Morgan Kaufman, Revised Fourth Edition, 2011.

- **Bài tập:** file đính kèm