Note Title 30-08-2012

Aug - 30

Division

- → Restoring
- → Non-restoring. {unsigned}

(slow operation)

$11/3 = 3 \times 3 + 2$

11 → Dividend (D)
3 → Divisor (N)
3 → Quotient (Q)
2 → Remainder (R)

$D = NQ + R$

$11/3$: $1011 \div 0011$

decimal

$$
\begin{array}{r}
11\,\overline{)\,950\,}(\,086 \\
\underline{0} \\
\underline{88} \\
70 \\
\underline{66} \\
4
\end{array}
$$

$950 = 11 \times 86 + 4$

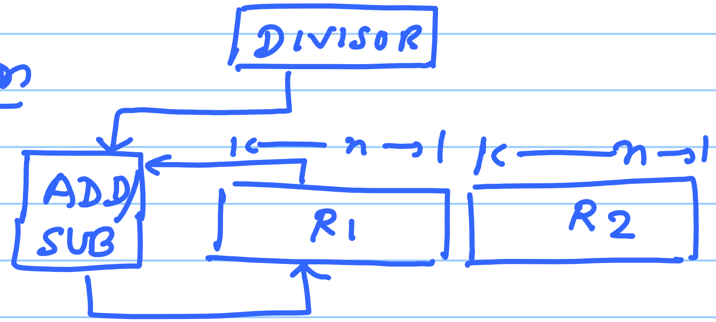
$$
\begin{array}{r}
11\,\overline{)\,1011\,}(\,0011 \\
\underline{0} \\
10 \\
\underline{0} \\
101 \\
\underline{-11} \\
101
\end{array}
$$

Q

binary

$$\begin{array}{r} -\ 11 \\ \hline 010 \\ R \end{array}$$

Restoring Division



Start:

R_2 = Dividend

R_1 = 00 0

Repeat n times.

1) Left shift $R_1 R_2$

2) R_1 -= N // N ← Divisor

3) If $R_1 < 0$

$R_1 \mathrel{+}= N$

$LSB(R_2) = 0$

ELSE

$LSB(R_2) = 1$

4) R_1 (Remainder)

R_2 (Quotient)

$11 \div 3 \rightarrow 1011 \div 0011$

R_1	R_2
0000	1011

1. $\boxed{0001}$ $\boxed{0110}$

2. $\boxed{0010}$ $\boxed{1100}$

× -11

3. $\boxed{0101}$ $\boxed{100}$

-11

$\boxed{0010}$ $\boxed{1001}$

4. $\boxed{0101}$ $\boxed{001}$

-11

$\boxed{0010}$ $\boxed{0011}$

R Q

Time Complexity:

$\{ n \times \log(n) \}$

Restoring Algo.

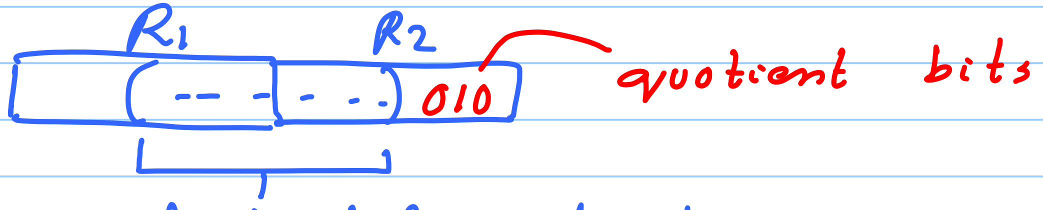
+ simple
+ same as basic version of division
− slow ($n \log(n)$)
− every iteration (worst case 1 SUB + 1 ADD)
[Restore original value if subtract fails]

Non-restoring algorithm [$n \log(n)$]

+ does not restore original value

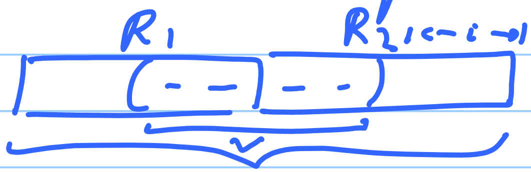
How does it work.

$\boxed{1}$ In the restoring algorithm, consider iteration (i)



part of the dividend left after repeated subtractions.

For the purposes of analysis let us ignore the quotient bits



One large binary number.

$R_1 R_2 \longleftrightarrow V \times 2^i$

$\boxed{R_1}\ \boxed{R_2}$

$- N \quad 0 \cdots 0$

Restoring algo.

$V \times 2^i - N \times 2^n \geq 0$

I $\quad V \times 2^i - N \times 2^n \geq 0$

$\text{new}(R_1 R_2) \leftarrow V \times 2^i - N \times 2^n$

Non-restoring algo does the same

II $\quad V \times 2^i - N \times 2^n < 0$

Rest: $\text{new}(R_1 R_2) \leftarrow V \times 2^i$

Non-Rest: $\text{new}(R_1 R_2) \leftarrow V \times 2^i - N \times 2^n$

$\boxed{2}$ iteration $(i+1)$ (case II)

Rest: $R_1R_2 \leftarrow V\times 2^{i+1}$

Non-rest: $R_1R_2 \leftarrow V\times 2^{i+1} - N\times 2^{n+1}$

Rest: Subtract again

new $(R_1R_2) \leftarrow \{V\times 2^{i+1} - N\times 2^{n}\}$ ✓

case (I) and case (II)

Non-Rest:

new $(R_1R_2) \leftarrow$ old $(R_1R_2) + N\times 2^{n}$

$\leftarrow V\times 2^{i+1} - N\times 2^{n+1} + N\times 2^{n}$

$= [V\times 2^{i+1} - N\times 2^{n}]$ ✓

convergence

Case I (Subtraction successful)

(R_1R_2 same for rest and non-rest algorithms)

It is possible in iteration $(i+1)$

Case II (subtraction not successful)

$i+2$

subtraction successful : rest == non-rest

" not successful :

⋮

[mathematical induction]

Some point :

if subtraction is successful (resto. algo).

The partial dividend (R, R_2)

same for both algos.

you reach the end without a successful subtract.

$$\boxed{<0}\quad \boxed{0\,\cdots\,0}$$

R_1 R_2

$+N$

$R = R_1 + N$

Remainder.

In the last step, we will have a number of the form:

$$\{V \times 2^n - N \times 2^{n+1}\}$$

$$+ \; N \times 2^n$$

$$= (V \times 2^n - N \times 2^n)$$

Leave it as an exercise: Prove that this is actually the remainder.

R_1	R_2

Start

$R_2 \leftarrow$ Dividend

Repeat n times.

1) Left shift R_1, R_2
2) If $(R_1 < 0)$
   $R_1 \mathrel{+}= N$
   ELSE
   $R_1 \mathrel{-}= N$
3) IF $(R_1 < 0)$ $LSB(R_2) \leftarrow 0$ ELSE $LSB(R_2) \leftarrow 1$

4) IF $(R_1 < 0)$
   $R_1 \mathrel{+}= N$

Result:

R_1 — Remainder R_2 — Quotient

7/3

0111 / 0011

−3 : 1101

0000	0111

①

0000	111

(−3) + 1101

1101	1110

②

1011	110

+ 11

1110	1100

③

1101	100

+ 11

0000	1001

($R_1 \geq 0$)

④

0001	001

+ 1101

1110	0010

R_1 Q

$R_1 < 0$

R_1 += 11

→ R_1 = 0001

Remainder

$7 = 3 \times Q + R$

(2)   (1)

Claim 1:

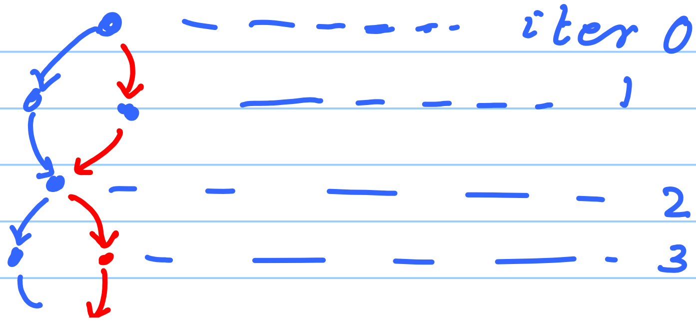
You put 1 in the quotient iff ($R_i \geq 0$) at the end of the iteration.

Claim 2:

The rest and non-rest algorithms converge only when you

$$\left\{ \begin{array}{l} \text{put a 1 in the quotient} \\ \text{OR} \quad R_i \geq 0 \text{ at the end of an iteration} \end{array} \right.$$

Conceptual View.



— restoring algo.
— non-restoring algo.

Non-restoring is a (tricky) algorithm [not very easy to understand]

take a look in the notes.