

Sept - 7th

Note Title

07-09-2011

Last class: Adders

- Ripple Carry $O(n)$
- Carry Select $O(\sqrt{n})$
- Carry Lookahead Adder
 $O(\log(n))$
(optimal).

This class:

How to add m n -bit numbers?

Why do we need this — for multiply

 } m numbers

 } $m/2$



Time complexity: $\log(m) \times \log(n + \log(m))$
(# of levels)



$\log(m) + \log(n + \log(m))$

how ???

Primitive

Carry Save Addition.

If I give you 3 numbers : a, b, c

Can you give me 2 numbers: d & e .

s.t

$$a + b + c = d + e$$


and, we compute this in $O(1)$ time

Example 1

$$\begin{array}{r} 45 \\ 56 \\ 63 \\ \hline 54 \end{array}$$

$$= \overbrace{54 + 110}$$

$$\begin{array}{r} 110 \\ \hline 164 \end{array}$$


 $(a+b+c = d+e)$

$$\begin{array}{r}
 575 \\
 + 676 \\
 \hline
 429
 \end{array}
 \left. \begin{array}{l} a \\ b \\ c \end{array} \right\} = \underbrace{560 + 1120}$$

$d \rightarrow 560 \leftarrow$ digit by digit sum
 $e \rightarrow 1120 \leftarrow$ digit by digit carry
 $\hline 1680$

A set of digits are being treated independently.
 (parallelly add each set)

$O(1)$ time.

TODO: Do it for binary

$m=30$ add 30 n -bit numbers

step 1: {  (30)

step 2: { 20 $\times 2/3$

step 3: { 14 $\times 2/3$

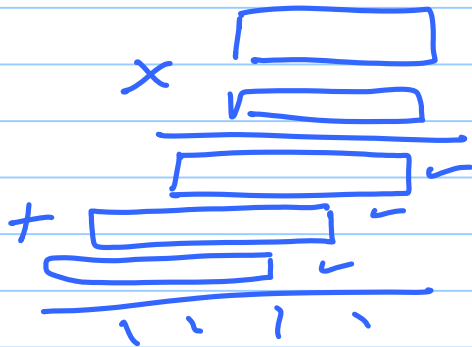
step 4: { 10 $\times 2/3$

⋮
step k : { 2 numbers

$$k = \boxed{\log_{3/2} m} \quad O(\log(m))$$

Time Complexity: $O[\log(m) + \log(n + \log(m))]$

→ Wallace Tree Multiplier
→ Carry Save Multiplier



In the case of multiply: $m=n$
complexity: $O(\log(n))$

Division

We don't have a very efficient algorithm for division.

Most of the common algorithms take $O(n)$ time

Restoring Division Algorithm

Divisor	Dividend	Quotient
7) 10942	1563
	0	
	10	
	7	
	<hr/>	
	39	
	35	
	<hr/>	

$$\begin{array}{r}
 44 \\
 42 \\
 \hline
 22 \\
 21 \\
 \hline
 1
 \end{array}$$

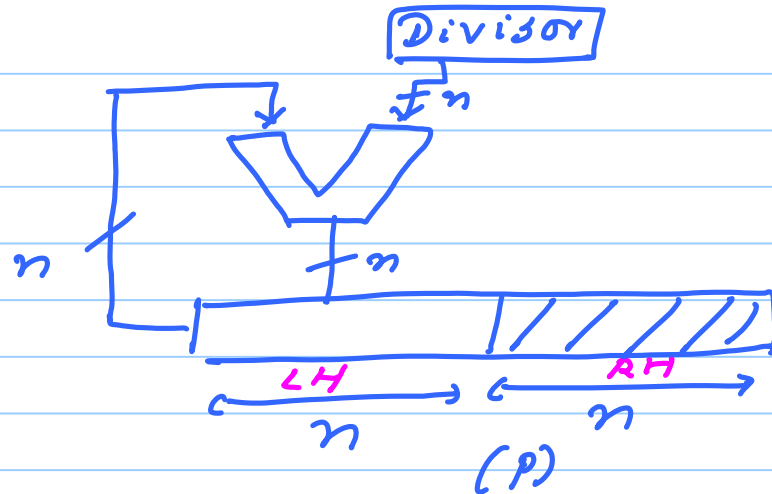
← remainder

$$10 \div 3$$

$$\begin{array}{r}
 11 \overline{) 1010} \quad (0011 \\
 \underline{0} \quad \quad \quad \text{Q} \\
 10 \\
 \underline{0} \\
 101 \\
 - \quad 11 \\
 \hline
 100 \\
 - \quad 11 \\
 \hline
 001 \quad \leftarrow R
 \end{array}$$

$$10 = 3 \times \overset{Q}{(3)} + \overset{R}{(1)}$$

Divide n -bit numbers.



Dividend is initially loaded in RH

LH \rightarrow LEFT HALF
RH \rightarrow RIGHT HALF

$10 \div 3$
 $1010 \div 11$

Step 0:

0000		1010
------	--	------

Step 1: a) Left shift RH

0001		010X
------	--	------

b) Check if the divisor divides

the LH

$\xrightarrow{\text{yes}}$

$x \rightarrow 1$

- subtract divisor from LH

$\xrightarrow{\text{no}}$

$x \rightarrow 0$

Do nothing

Example

0 0 0 1 | 0 1 0 0

Step 2:

0 0 1 0 | 1 0 0 0

Step 3:

0 1 0 1 | 0 0 0
- 0 1 1

0 0 1 0 | 0 0 0 1

step 4 :

0100 | 001
-011

0001 | 0011
R Q

4 bit
division
} done

$$10 \div 3 \\ = 3 \times \underset{Q}{(3)} + \underset{R}{(1)}$$

Time Complexity : (# steps) \times (complexity per step)

$$n \times [\log(n)]$$

1) shift $O(1)$ -

\times 2) Compare [Subtract opn. ^{check} sign bit]

3) Subtract

4) Write quotient bit

$O(1)$ -

Slightly speed it up:

We are doing two subtracts

Can we reduce it to one?

Current time $\approx n \times (2 \log(n))$

reduced $\approx n \times (\log(n))$ [speedup by
a fac. of 2]

To reduce one subtract.

Another Algorithm

Non-restoring Algorithm.

We will eliminate the compare.

Next Class:

- 1) Non-restoring Algorithm
- 2) Floating Point

To do:

Find a new class slot (10 AM - 6 PM)

Homework 2:

Will be released Today

Basic Linux Knowledge.