ARITHMETIC OF INTEGERS

BASIC ARITHMETIC ON 'BIG' INTEGERS

How to store bigger integers 'exactly'?

Represent in some pre-determined base B
Store B-ary digits in an array of built-in integers
Choice of B: power of 2 & as large as possible
e.g. 2^{32} or 2^{64}

multi-precision integers

n: number in base B

$$n = (n_{s-1}, n_{s-2}, \ldots, n_1, n_0)_B = \sum_{i=0}^{s-1} n_i B^i$$

CONVERSION

Decimal $\rightarrow$ Base B

n

$d_0 d_1 \cdots d_{t-1} \rightsquigarrow (n_{s-1}, n_{s-2}, \ldots, n_1, n_0)_B = n$

$d_0 d_1 \cdots d_{t-1} d_t$

$n' = 10n + d_t$

n'

$(n')_B = (10)_B * (n)_B + (d_t)_B$

single digit B-ary numbers

k digits at a time : $10^k < B$

$d_0 d_1 \cdots d_{kt-1} d_{kt} \cdots d_{k(t+1)-1}$

n'

$n' = (10^k)_B * (n)_B + (d_{kt} \cdots d_{k(t+1)-1})_B$

Base $B \to$ Base B'

$B' = 10^{l}$ $\quad 10^{l} \le B < 10^{l+1}$

$B = HB' + L$ $\quad H, L \in \{0, 1, \dots, B'-1\}$

$n = (n_{s-1}, \dots, n_0)_B$

$N_i = (n_{s-1}, \dots, n_i)$ $\quad N_s = 0$ $\quad N_0 = n$

Compute base B' representation of $N_{s-1}, N_{s-2}, \dots, N_0$ iteratively

Initialise $(N_s)_{B'} = (\,)_{B'}$

$n_i = h_i B' + l_i$

$N_i = N_{i+1} B + n_i$

$(N_i)_{B'} = (N_{i+1})_{B'} (HB' + L) + (h_i B' + l_i)$ ← requires base B' arithmetic

$h_i, l_i \in [0, B'-1]$

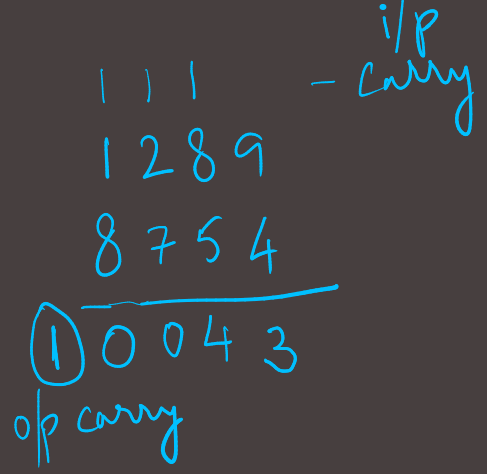
MULTI-PRECISION ARITHMETIC

- ADDITION
- SUBTRACTION
- MULTIPLICATION
- EUCLIDEAN DIVISION

ADDITION

$$a = (a_{s-1}, a_{s-2}, \cdots, a_1, a_0)_B$$

$$b = (b_{t-1}, b_{t-2}, \cdots, b_1, b_0)_B$$



Assume $s = t$

Input carry : c_{in} (initialised to 0)

Output carry : c_{out}

Compute $a_i + b_i + C_{in}$ in the seq $i = 0, 1, 2, \ldots$

$i = 0, 1, 2, \ldots$ $\quad d_i = a_i + b_i + C_{in}$

$$C_{out} = \begin{cases} 1 & \text{if } a_i + b_i + C_{in} \bmod B < a_i \\ 0 & \text{o.w} \end{cases}$$

if $i = s-1$, o/p $(C_{out}, d_{s-1}, d_{s-2}, \ldots, d_1, d_0)_B$

$C_{in} = C_{out}$

SUBTRACTION

$a \geq b$

o.w compute $b - a$

result $= -(b-a)$

C_{out}	↙1	↙1	↙1	
C_{in}	1	1	1	0
	9	0	8	1
	4	6	9	2
	4	3	8	9

C_{in}, C_{out} : borrows initialised to 0

$C_{in} = 0$

$C_{out} = 1$ iff $a_i < b_i$

$C_{in} = 1$

$C_{out} = 1$ iff $a_i \leq b_i$

$i = 0, 1, 2, \ldots$

$$d_i = \begin{cases} a_i - b_i - c_{in} \mod B & \text{if the machine supports 2's complement arithmetic} \\ a_i + (B - b_i - c_{in}) & \text{o.w.} \end{cases}$$

$c_{in} \leftarrow c_{out}$

MULTIPLICATION

$$a = (a_{s-1}, a_{s-2}, \ldots, a_1, a_0)_B$$

$$b = (b_{t-1}, b_{t-2}, \ldots, b_1, b_0)_B$$

$d = ab$

Initialise $d = (\underbrace{0, 0, \ldots, 0}_{s+t})_B$

for $i \leftarrow 0$ to $t-1$

  for $j \leftarrow 0$ to $t-1$

    add $b_i a_j$ to pos $i+j$ of d.

one built-in integer multiplication

affect digits $d_{i+j}, d_{i+j+1}, \ldots$

$$\underset{a}{248} \times \underset{b}{135}$$

```
  000000
     40      5×8 at pos 0+0
    20       5×4 at pos 0+1
   10        5×2 at pos 0+2
  ------
  001240
     24      3×8 at pos 1+0
    12       3×4 at pos 1+1
   6         3×2 at pos 1+2
  ------
  008680
    248      1×8 at pos 2+0
  ------     1×4 at pos 2+1
  033480     1×2 at pos 2+2
```

To compute $a_j b_i$:

- use double precision integer data type if the machine supports it
- else, break each operand into half-size integers

$$a_j = h_j\sqrt{B} + l_j \qquad b_i = h'_i\sqrt{B} + l'_i$$

$$a_j b_i = h_j h'_i B + (h_j l'_i + l_j h'_i)\sqrt{B} + l_j l'_i$$

$h_j h'_i < B$ → $(i+j+1)^{st}$ position of the product

$h_j l'_i < B$, $l_j h'_i < B$ → contribute to both positions

$l_j l'_i < B$ → $(i+j)^{th}$ position of the product

$$= hB + l$$

h → $(i+j+1)^{st}$ position

l → $(i+j)^{th}$ position

EUCLIDEAN DIVISION

a, b : integers with $b \neq 0$

$\exists$ unique integers q, r satisfying

$$a = bq + r \quad \text{and} \quad 0 \leq r \leq |b| - 1$$

$q = a$ quot b $\qquad$ $r = a$ rem b

$$a = (a_{s-1}, \dots, a_1, a_0)_B$$

$$b = (b_{t-1}, \dots, b_1, b_0)_B$$

$a, b > 0$

$a \geq b$

Quotient

$q = (q_{s-t}, \dots, q_0)_B$

initialise q to 0

Normalisation

Multiply b with a suitable power of 2 so that

$$b_{t-1} \geq \frac{B}{2}$$

```
  665
61)40592
  36600   366*10^2
  -----
    399
    366
  -----
     332
     305
  -----
      27
```

40592 quot 61 = 665

40592 rem 61 = 27

① Guessing digits of q

If $a_{s-1} \geq b_{t-1}$ & $a \geq B^{s-t} b$

executed at most once since b is normalised

$$q_{s-t} \leftarrow q_{s-t} + 1$$
$$a \leftarrow a - B^{s-t} b$$

$a_{s-1}, a_{s-2}, \ldots, a_{s-t}, \ldots$

$b_{t-1}, b_{t-2}, \ldots, b_0$

② $\hat{q}_{s-t-1} = \min\left(\left\lfloor \frac{a_{s-1} B + a_{s-2}}{b_{t-1}} \right\rfloor , B-1 \right)$

$q_{s-t-1} \leq \hat{q}_{s-t-1} \leq q_{s-t-1} + 3$ for normalised b

$q_{s-t-1} \leftarrow \hat{q}_{s-t-1}$

(3) Correcting the guess

involves multiplications

while $\left(a_{s-1}B^2 + a_{s-2}B + a_{s-3} < \left(b_{t-1}B + b_{t-2}\right) q_{s-t-1}\right)$

$$q_{s-t-1} \leftarrow q_{s-t-1} - 1$$

no. of iterations ≤ 2 given that b is normalised

At this point, q_{s-t-1} is either correct or 1 more than the correct value.

$$c \leftarrow q_{s-t-1} B^{s-t-1} b$$

If $c > a$, $q_{s-t-1} \leftarrow q_{s-t-1} - 1$; $c \leftarrow c - bB^{s-t-1}$

Replace a with $a - c$ & s with $s-1$

Repeat ①, ② & ③ until a has at most t digits

At this point a has $\leq t$ B-ary digits

either $a \geq b$ → $a \leftarrow a - b$, $q_0 \leftarrow q_0 + 1$

or $a < b$ → nothing to do

a contains the remainder

q contains the quotient.