Discrete Math for Computing

Ch 4.2 Integer Representations and Algorithms

- What is an integer?

Subset of real numbers formed by the natural numbers together with the negatives of the non-zero natural numbers

Set **Z** -> {..., −2, −1, 0, 1, 2, ...}

- What is an algorithm?

Procedures for performing arithmetic operations using the decimal, binary representations of integers

Integers and Algorithms

- Representation of Integers – Base b expansion of n

  Let ‘b’ be a positive integer > 1

  If ‘n’ > 0 it can be expressed uniquely in the form

  $$n = a_k b^k + a_{k-1} b^{k-1} + \ldots + a_1 b + a_0$$

  where ‘k’ – nonnegative integer

  $a_0, a_1, \ldots a_k$ are nonnegative integers < b

  $a_k \neq 0$

Integers and Algorithms

- Example: What is the integer representation of 245 in base 8?

$$(245)_8 = 2.8^2 + 4.8 + 5$$

Example: What is the integer representation of 965 in base 10?

$$(965)_{10} = 9.10^2 + 6.10 + 5$$

Integers and Algorithms

- Binary Expansion

  Choosing 2 as Base

  2 digits – 0, 1

- Decimal Expansion

  Choosing 10 as Base

  10 digits – 0, 1,…, 9

- Hexadecimal Expansion

  Choosing 16 as Base

  16 digits – 0, 1,…, 9, A, B, C, D, E, F

Integers and Algorithms

- Example: What is decimal expansion of the decimal expansion of $(2AE0B)_{16}$?

$(2AE0B)_{16} = 2.16^4 + 10.16^3 + 14.16^2 + 0.16 + 11$
$= (175627)_{10}$

Example: What is binary expansion of $(241)_{10}$?

$241 = 2.120 + 1, \quad 120 = 2.60 + 0$

$60 = 2.30 + 0, \quad 30 = 2.15 + 0$

$15 = 2.7 + 1, \quad 7 = 2.3 + 1$

$3 = 2.1 + 1, \quad 1 = 2.0 + 1 = (1111\ 0001)_2$

UTD

Integers and Algorithms

- ALGORITHM: Constructing Base b Expansions

procedure base b expansion(n: positive integer)

q := n

k := 0

while q ≠ 0

begin

a_k := q mod b

q := $\lfloor q/b \rfloor$

k := k + 1

end {the base b expansion of n is $(a_{k-1} \ldots a_1 a_0)_b$}

Integers and Algorithms

- Algorithms for Integer Operations
- Performing operations with integers using their binary expansions

  - Important Applications in Computer Arithmetic
- For any two integers 'a' and 'b' with n bits -

  Binary expansions:

  $a = (a_{n-1}a_{n-2}\ldots a_1a_0)_2$

  $b = (b_{n-1}b_{n-2}\ldots b_1b_0)_2$

UTD

Integers and Algorithms

- Addition of Integers in binary notation
- Based on the addition of numbers
- Start with the rightmost bits – a_0 and b_0

  $a_0 + b_0 = c_0 .2 + s_0$

  s_0 - rightmost bit, c_0 - carry
- Add the next pair of bits with the carry

  $a_1 + b_1 + c_0 = c_1.2 + s_1$
- Last stage add a_{n-1}, b_{n-1}, and c_{n-2} to obtain $c_{n-1}.2 + s_{n-1}$

Leading bit of the sum $s_n = c_{n-1}$

- Binary expansion of sum $a + b = (s_n s_{n-1} s_{n-2} \ldots s_1.s_0)_2$

  O(n) additions

Integers and Algorithms

- Example: Add $a = (1110)_2$ and $b = (1011)_2$
- $a_0 + b_0 = 0 + 1 = 0.2 + 1$

$\Rightarrow s_0 = 1, c_0 = 0$

$a_1 + b_1 + c_0 = 1 + 1 + 0 = 1.2 + 0$

$\Rightarrow s_1 = 0, c_1 = 1$

$a_2 + b_2 + c_1 = 1 + 0 + 1 = 1.2 + 0$

$\Rightarrow s_2 = 0, c_2 = 1$

$a_3 + b_3 + c_2 = 1 + 1 + 1 = 1.2 + 1$

$\Rightarrow s_3 = 1, c_3 = 1 \Rightarrow s_4 = c_3 = 1$ $\therefore s = a + b = (1\,1001)_2$

$$\begin{array}{r} \mathit{1\;1\;1\;\;\;} \\ 1\;1\;1\;0 \\ 1\;0\;1\;1 \\ \hline 1\;1\;0\;0\;1 \end{array}$$

UTD

Integers and Algorithms

- ALGORITHM: Addition of Integers

procedure add(a, b: positive integers)

{the binary expansions of a and b are $(a_{n-1}a_{n-2}\ldots a_1a_0)_2$
and $(b_{n-1}b_{n-2}\ldots b_1b_0)_2$, respectively}

c := 0

for j := 0 to n-1

begin

d := $(a_j + b_j + c)/2$

s_j := $a_j + b_j + c - 2d$

c := d

end

s_n := c

{the binary expansion of the sum is $(s_ns_{n-1}\ldots s_0)_2$}

Integers and Algorithms

- Multiplication of Integers
- If ‘a’ and ‘b’ are two n-bit integers
- ab $= a(b_0 2^0 + b_1 2^1 + \ldots + b_{n-1} 2^{n-1})$

  $= a(b_0 2^0) + a(b_1 2^1) + \ldots + a(b_{n-1} 2^{n-1})$, Using distributive law
- To obtain $(ab_j)2^j$

  - Shift the binary expansion of ab_j, j places to the left

  - Add ‘j’ zero bits at the tail end

  => ab = sum of n integers $(ab_j)2^j$, j = 0, 1, 2,...

  $O(n^2)$ additions

Integers and Algorithms

- Example: Multiply $a = (110)_2$ and $b = (101)_2$

  $ab_0.2^0 = (110)_2.1.2^0 = (110)_2$ (1)

  $ab_1.2^1 = (110)_2.0.2^1 = (0000)_2$ (2)

  $ab_2.2^2 = (110)_2.1.2^2 = (11000)_2$ (3)

Adding (1), (2), and (3)

$$
\begin{array}{r}
1\,1\,0 \\
1\,0\,1 \\
\hline
1\,1\,0 \\
0\,0\,0 \\
1\,1\,0 \\
\hline
1\,1\,1\,1\,0
\end{array}
$$

$1\,1\,1\,1\,0$ => $ab = (1\ \ 1110)_2$

Integers and Algorithms

- ALGORITHM: Multiplying Integers

procedure multiply(a, b: positive integers)

{the binary expansions of a and b are $(a_{n-1}a_{n-2}\ldots a_1a_0)_2$

and $(b_{n-1}b_{n-2}\ldots b_1b_0)_2$, respectively}

for j := 0 to n - 1

begin

if $b_j = 1$ then c_j := a shifted j places

else $c_j := 0$

end

{$c_0, c_1, \ldots, c_{n-1}$ are the partial products}

p := 0

For j := 0 to n – 1

$p := p + c_j$

{p is the value of ab}

Integers and Algorithms

- Computing div and mod
- Given integers ‘a’ and ‘d’, $d > 0$
- $q = a \text{ div } d$, $r = a \bmod d$
- When $a > 0$, subtract ‘d’ from ‘a’ as many times until what is left is $< d$
- Number of times = quotient, Number left = remainder
- When $a < 0$, $|a|$ divided by ‘d’
- When $a < 0$ and $r > 0$, quotient $-(q + 1)$, remainder $d - r$
- $O(n^2)$ bit operations

UTD

Integers and Algorithms

- ALGORITHM: Computing div and mod

```
procedure division(a: integer, d: positive integer)
q := 0
r  :=  a
while r ≥ d
begin
   r := r – d
   q := q + 1
end
if a < 0 and r > 0 then
begin
   r  := d – r
   q  := -(q + 1)
end
{q = a div d is the quotient, r = a mod d is the remainder}
```

Integers and Algorithms

- Modular Exponentiation
- In cryptography, find b^n mod m 'b', 'n', 'm' - large integers
- Binary expansion of the exponent 'n' $(a_{k-1} \ldots a_1 a_0)_2$

=> b^n = $$b^{a_{k-1}.2^{k-1}} \ldots b^{a_1.2}.b^{a_0}$$

= $$b^{a_{k-1}.2^{k-1}+\ldots+a_1.2+a_0}$$

Calculate values b, b^2…

Multiply terms b^{2^j} mod m where $a_j = 1$

O($(\log m)^2 \log n$) bit operations

Integers and Algorithms

- ALGORITHM: Modular Exponentiation

procedure modular exponentiation(b: integer, $n = (a_{k-1}a_{k-2}\ldots a_1a_0)_2$, m: positive integers)

x := 1

power := b mod m

for i := 0 to k – 1

begin

    if a_i = 1 then x := (x.power) mod m

    power := (power.power) mod m

end

{x equals b^n mod m}

Integers and Algorithms

- Example: Find 3^{644} mod 645
- x = 1, power = 3 mod 645 = 3
- Binary expansion of 644 = $(10\ 1000\ 0100)_2$
- 3^{2^j} mod 645 for j = 1, 2, …, 9
- Successively squaring and reducing modulo 645

i = 0, a_0 = 0, x = 1 and power = 3^2 mod 645 = 9 mod 645 = 9

i = 1, a_1 = 0, x = 1 and power = 9^2 mod 645 = 81 mod 645 = 81

i = 2, a_2 = 1, x = 1.81mod 645 = 81 and power = 81^2mod 645 = 111

……………………………………………………………………

I = 7, a_7 = 1, x = (81.396)mod 645 = 471 and power = 396^2mod 645 = 81

i = 8, a_8 = 0, x = 471 and power = 81^2 mod 645 = 6561 mod 645 = 111

i = 9, a_9 = 1, x = (471.111) mod 645 = 36 => 3^{644} mod 645 = 36