Arithmetic Modulo *m*

Definitions: Let $\mathbf{Z}_m$ be the set of nonnegative integers less than *m*: $\{0,1, \ldots, m-1\}$

The operation $+_m$ is defined as $a +_m b = (a + b) \textbf{ mod } m$. This is *addition modulo m*.

The operation $\cdot_m$ is defined as $a \cdot_m b = (a + b) \textbf{ mod } m$. This is *multiplication modulo m*.

Using these operations is said to be doing *arithmetic modulo m*.

Example: Find $7 +_{11} 9$ and $7 \cdot_{11} 9$.

Solution: Using the definitions above:

- $7 +_{11} 9 = (7 + 9) \textbf{ mod } 11 = 16 \textbf{ mod } 11 = 5$
- $7 \cdot_{11} 9 = (7 \cdot 9) \textbf{ mod } 11 = 63 \textbf{ mod } 11 = 8$

Arithmetic Modulo *m*

The operations $+_m$ and $\cdot_m$ satisfy many of the same properties as ordinary addition and multiplication.

- *Closure*: If *a* and *b* belong to $\mathbf{Z}_m$, then $a +_m b$ and $a \cdot_m b$ belong to $\mathbf{Z}_m$.
- *Associativity*: If *a*, *b*, and *c* belong to $\mathbf{Z}_m$, then $(a +_m b) +_m c = a +_m (b +_m c)$ and $(a \cdot_m b) \cdot_m c = a \cdot_m (b \cdot_m c)$.
- *Commutativity*: If *a* and *b* belong to $\mathbf{Z}_m$, then $a +_m b = b +_m a$ and $a \cdot_m b = b \cdot_m a$.
- *Identity elements*: The elements 0 and 1 are identity elements for addition and multiplication modulo *m*, respectively.
  - If *a* belongs to $\mathbf{Z}_m$, then $a +_m 0 = a$ and $a \cdot_m 1 = a$.

Arithmetic Modulo *m*

- *Additive inverses*: If $a \neq 0$ belongs to $\mathbf{Z}_m$, then $m - a$ is the additive inverse of a modulo m and 0 is its own additive inverse.
  - $a +_m (m - a) = 0$ and $0 +_m 0 = 0$
- *Distributivity*: If *a*, *b*, and *c* belong to $\mathbf{Z}_m$, then
  - $a \cdot_m (b +_m c) = (a \cdot_m b) +_m (a \cdot_m c)$ and $(a +_m b) \cdot_m c = (a \cdot_m c) +_m (b \cdot_m c)$.

Exercises 42-44 ask for proofs of these properties.

Multiplicatative inverses have not been included since they do not always exist. For example, there is no multiplicative inverse of 2 modulo 6. But every non zero element of $\mathbf{Z}_m$ will have a multiplicative inverse if m is a prime.

(*optional*) Using the terminology of abstract algebra, $\mathbf{Z}_m$ with $+_m$ is a commutative group and $\mathbf{Z}_m$ with $+_m$ and $\cdot_m$ is a commutative ring.

If m is prime then $\mathbf{Z}_m$ is a field.

Representations of Integers

Let b be a positive integer greater than 1.
Then if n is a positive integer, 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 is a nonnegative integer,
$a_0, a_1, \ldots, a_k$ are nonnegative integers less than b, and $a_k \neq 0$.

Example for b=10:

$859 = 8 \cdot 10^2 + 5 \cdot 10^1 + 9 \cdot 10^0$

Representations of Integers

Example for b=2 (binary expansion):

$(10110)_2 = 1 \cdot 2^4 + 1 \cdot 2^2 + 1 \cdot 2^1 = (22)_{10}$

Example for b=16 (hexadecimal expansion):

(we use letters A to F to indicate numbers 10 to 15)

$(3A0F)_{16} = 3 \cdot 16^3 + 10 \cdot 16^2 + 0 \cdot 16^1 + 15 \cdot 16^0 = (14863)_{10}$

Representations of Integers

How can we construct the base b expansion of an integer n?

First, divide n by b to obtain a quotient q_0 and remainder a_0, that is,

$n = bq_0 + a_0$, where $0 \leq a_0 < b$.

The remainder a_0 is the rightmost digit in the base b expansion of n.

Next, divide q_0 by b to obtain:

$q_0 = bq_1 + a_1$, where $0 \leq a_1 < b$.

a_1 is the second digit from the right in the base b expansion of n. Continue this process until you obtain a quotient equal to zero.

Representations of Integers

Example:
What is the base 8 expansion of $(12345)_{10}$?

First, divide 12345 by 8:
$12345 = 8 \cdot 1543 + 1$

$1543 = 8 \cdot 192 + 7$
$192 = 8 \cdot 24 + 0$
$24 = 8 \cdot 3 + 0$
$3 = 8 \cdot 0 + 3$

The result is: $(12345)_{10} = (30071)_8$.

Representations of Integers

procedure base_b_expansion(n, b: positive integers)
q := n
k := 0
while $q \neq 0$
begin
    a_k := q mod b
    q := $\lfloor q/b \rfloor$
    k := k + 1
end
{the base b expansion of n is $(a_{k-1} \ldots a_1 a_0)_b$ }

Addition of Integers

How do we (humans) add two integers?

```
                 111     carry
Example:          7583
                + 4932
                ------
                 12515
```

```
                          1  1        carry
Binary expansions:        (1011)_2
                        + (1010)_2
                        ----------
                         (10101)_2
```


Addition of Integers

Let $a = (a_{n-1}a_{n-2}\ldots a_1a_0)_2$, $b = (b_{n-1}b_{n-2}\ldots b_1b_0)_2$.
How can we **algorithmically** add these two binary numbers?
First, add their rightmost bits:
$a_0 + b_0 = c_0 \cdot 2 + s_0$,
where s_0 is the **rightmost bit** in the binary expansion of a + b, and c_0 is the **carry**.
Then, add the next pair of bits and the carry:
$a_1 + b_1 + c_0 = c_1 \cdot 2 + s_1$,
where s_1 is the **next bit** in the binary expansion of a + b, and c_1 is the carry.

Addition of Integers

Continue this process until you obtain c_{n-1}.

The leading bit of the sum is $s_n = c_{n-1}$.

The result is:
$a + b = (s_n s_{n-1} \ldots s_1 s_0)_2$

Addition of Integers

Example:
Add $a = (1110)_2$ and $b = (1011)_2$.

$a_0 + b_0 = 0 + 1 = 0 \cdot 2 + 1$, so that $c_0 = 0$ and $s_0 = 1$.
$a_1 + b_1 + c_0 = 1 + 1 + 0 = 1 \cdot 2 + 0$, so $c_1 = 1$ and $s_1 = 0$.
$a_2 + b_2 + c_1 = 1 + 0 + 1 = 1 \cdot 2 + 0$, so $c_2 = 1$ and $s_2 = 0$.
$a_3 + b_3 + c_2 = 1 + 1 + 1 = 1 \cdot 2 + 1$, so $c_3 = 1$ and $s_3 = 1$.
$s_4 = c_3 = 1$.

Therefore, $s = a + b = (11001)_2$.

Addition of Integers

procedure add(a, b: positive integers)
// a_i, b_i are the bits of a and b.
c := 0
for j := 0 to n-1
begin
d := $\lfloor(a_j + b_j + c)/2\rfloor$ // gives the high bit of sum
$s_j := a_j + b_j + c - 2d$ // gives the low bit of sum
c := d
end
s_n := c
{the binary expansion of the sum is $(s_n s_{n-1} \ldots s_1 s_0)_2$}

Multiplication of Integers

procedure multiply(a, b: positive integers)
// a_i, b_i are the bits of a and b.
for j := 0 to n-1
begin
if b_j = 1 then c_j := a shifted left j places
else c_j := 0 // c_j are the partial products
end
p := 0
for i := 0 to n-1
p := p + c_j
{p is the value of the product as an integer.
Note that we haven't computed bits for p}

More Algorithms

Take a look at Algorithms 4 and 5 on pages 253, 254 and be sure you understand them. It's important to be able to read the code and see what it says.

Algorithm 4 gives a way of doing the division algorithm using repeated subtractions instead of division.

Algorithm 5 gives a way of computing b^n using a binary representation of n

Section Summary

Integer Representations
- Base *b* Expansions
- Binary Expansions
- Octal Expansions
- Hexadecimal Expansions

Base Conversion Algorithm

Algorithms for Integer Operations

Representations of Integers

In the modern world, we use *decimal,* or *base* 10, *notation* to represent integers. For example when we write 965, we mean $9{\cdot}10^2 + 6{\cdot}10^1 + 5{\cdot}10^0$.

We can represent numbers using any base *b*, where *b* is a positive integer greater than 1.

The bases *b* = 2 (*binary*), *b* = 8 (*octal*) , and *b*= 16 (*hexadecimal*) are important for computing and communications

The ancient Mayans used base 20 and the ancient Babylonians used base 60.

Base *b* Representations

We can use positive integer *b* greater than 1 as a base, because of this theorem:

Theorem 1: Let *b* be a positive integer greater than 1. Then if *n* is a positive integer, 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* is a nonnegative integer, $a_0, a_1, \ldots a_k$ are nonnegative integers less than *b*, and $a_k \neq 0$. The a_j, $j = 0,\ldots,k$ are called the base-*b* digits of the representation.

(We will prove this using mathematical induction in Section 5.1.)

The representation of n given in Theorem 1 is called the *base b expansion of n* and is denoted by $(a_k a_{k-1} \ldots a_1 a_0)_b$.

We usually omit the subscript 10 for base 10 expansions.

Binary Expansions

Most computers represent integers and do arithmetic with binary (base 2) expansions of integers. In these expansions, the only digits used are 0 and 1.

Example: What is the decimal expansion of the integer that has $(1\ 0101\ 1111)_2$ as its binary expansion?

Solution:

$(1\ 0101\ 1111)_2 = 1\cdot2^8 + 0\cdot2^7 + 1\cdot2^6 + 0\cdot2^5 + 1\cdot2^4 + 1\cdot2^3 + 1\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 = 351.$

Binary Expansions

Example: What is the decimal expansion of the integer that has $(11011)_2$ as its binary expansion?

Solution: $(11011)_2 = 1\cdot2^4 + 1\cdot2^3 + 0\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 = 27.$

Octal Expansions

The octal expansion (base 8) uses the digits $\{0,1,2,3,4,5,6,7\}$.

Example: What is the decimal expansion of the number with octal expansion $(7016)_8$?

Solution: $7\cdot8^3 + 0\cdot8^2 + 1\cdot8^1 + 6\cdot8^0 = 3598$

Octal Expansions

Example: What is the decimal expansion of the number with octal expansion $(111)_8$?

Solution: $1\cdot8^2 + 1\cdot8^1 + 1\cdot8^0 = 64 + 8 + 1 = 73$

Hexadecimal Expansions

The hexadecimal expansion needs 16 digits, but our decimal system provides only 10. So letters are used for the additional symbols. The hexadecimal system uses the digits $\{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$. The letters A through F represent the decimal numbers 10 through 15.

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

Solution:

$2\cdot16^4 + 10\cdot16^3 + 14\cdot16^2 + 0\cdot16^1 + 11\cdot16^0 = 175627$

Example: What is the decimal expansion of the number with hexadecimal expansion $(E5)_{16}$?

Solution: $1\cdot16^2 + 14\cdot16^1 + 5\cdot16^0 = 256 + 224 + 5 = 485$

Base Conversion

To construct the base *b* expansion of an integer *n*:

- Divide *n* by *b* to obtain a quotient and remainder.
  - $n = bq_0 + a_0 \quad 0 \leq a_0 \leq b$
- The remainder, a_0, is the rightmost digit in the base *b* expansion of *n*. Next, divide q_0 by *b*.
  - $q_0 = bq_1 + a_1 \quad 0 \leq a_1 \leq b$
- The remainder, a_1, is the second digit from the right in the base *b* expansion of *n*.
- Continue by successively dividing the quotients by *b*, obtaining the additional base *b* digits as the remainder. The process terminates when the quotient is 0.

continued →

Algorithm: Constructing Base *b* Expansions

procedure *base b expansion*(n, b: positive integers with $b > 1$)
$q := n$
$k := 0$
while ($q \neq 0$)
  $a_k := q$ **mod** b
  $q := q$ **div** b
  $k := k + 1$
return($a_{k-1}, \ldots, a_1, a_0$){($a_{k-1} \ldots a_1 a_0)_b$ is base b expansion of n}

q represents the quotient obtained by successive divisions by *b*, starting with $q = n$.

The digits in the base *b* expansion are the remainders of the division given by *q* **mod** *b*.

The algorithm terminates when $q = 0$ is reached.

Base Conversion

Example: Find the octal expansion of $(12345)_{10}$

Solution: Successively dividing by 8 gives:

- $12345 = 8 \cdot 1543 + 1$
- $1543 = 8 \cdot 192 + 7$
- $192 = 8 \cdot 24 + 0$
- $24 = 8 \cdot 3 + 0$
- $3 = 8 \cdot 0 + 3$

The remainders are the digits from right to left yielding $(30071)_8$.

Comparison of Hexadecimal, Octal, and Binary Representations

TABLE 1 Hexadecimal, Octal, and Binary Representation of the Integers 0 through 15.

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Octal	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17
Binary	0	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111

Initial 0s are not shown

Each octal digit corresponds to a block of 3 binary digits.

Each hexadecimal digit corresponds to a block of 4 binary digits.

So, conversion between binary, octal, and hexadecimal is easy.

Conversion Between Binary, Octal, and Hexadecimal Expansions

Example: Find the octal and hexadecimal expansions of $(11\ 1110\ 1011\ 1100)_2$.

Solution:

- To convert to octal, we group the digits into blocks of three $(011\ 111\ 010\ 111\ 100)_2$, adding initial 0s as needed. The blocks from left to right correspond to the digits 3,7,2,7, and 4. Hence, the solution is $(37274)_8$.
- To convert to hexadecimal, we group the digits into blocks of four $(0011\ 1110\ 1011\ 1100)_2$, adding initial 0s as needed. The blocks from left to right correspond to the digits 3,E,B, and C. Hence, the solution is $(3EBC)_{16}$.

Binary Addition of Integers

Algorithms for performing operations with integers using their binary expansions are important as computer chips work with binary numbers. Each digit is called a *bit*.

The number of additions of bits used by the algorithm to add two *n*-bit integers is $O(n)$.

Binary Addition of Integers

procedure *add*(a, b: positive integers)
{the binary expansions of a and b are $(a_{n-1}, a_{n-2}, \ldots, a_0)_2$ and $(b_{n-1}, b_{n-2}, \ldots, b_0)_2$, respectively}
$c := 0$
for $j := 0$ to $n - 1$
  $d := \lfloor (a_j + b_j + c)/2 \rfloor$
  $s_j := a_j + b_j + c - 2d$
  $c := d$
$s_n := c$
return($s_0, s_1, \ldots, s_n$)
  {the binary expansion of the sum is $(s_n, s_{n-1}, \ldots, s_0)_2$}

Binary Multiplication of Integers

Algorithm for computing the product of two n bit integers.

```
procedure multiply(a, b: positive integers)
{the binary expansions of a and b are (a_{n-1},a_{n-2},…,a_0)_2 and (b_{n-1},b_{n-2},…,b_0)_2,
   respectively}
for j := 0 to n − 1
    if b_j = 1 then c_j = a shifted j places
    else c_j := 0
{c_0,c_1,…, c_{n-1} are the partial products}
 p := 0
for j := 0 to n − 1
   p := p + c_j
return p {p is the value of ab}
```

The number of additions of bits used by the algorithm to multiply two n-bit integers is $O(n^2)$.

Binary Modular Exponentiation

In cryptography, it is important to be able to find b^n **mod** m efficiently, where b, n, and m are large integers.

Use the binary expansion of n, $n = (a_{k-1},…,a_1,a_o)_2$, to compute b^n . Note that:

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

Therefore, to compute b^n, we need only compute the values of b, b^2, $(b^2)^2 = b^4$, $(b^4)^2 = b^8$, …, b^{2^k} and then multiply the terms b^{2^j} in this list, where $a_j = 1$.

Example: Compute 3^{11} using this method.

Solution: Note that $11 = (1011)_2$ so that $3^{11} = 3^8\, 3^2\, 3^1 = ((3^2)^2)^2\, 3^2\, 3^1 = (9^2)^2 \cdot 9 \cdot 3 = (81)^2 \cdot 9 \cdot 3 = 6561 \cdot 9 \cdot 3 = 117{,}147.$

continued →

Binary Modular Exponentiation Algorithm

The algorithm successively finds b **mod** m, b^2 **mod** m, b^4 **mod** m, …, $b^{2^{k-1}}$ **mod** m, and multiplies together the terms b^{2^j} where $a_j = 1$.

```
procedure modular exponentiation(b: integer, n = (a_{k-1}a_{k-2}…a_1a_0)_2 , m: positive
   integers)
 x := 1
power := b mod m
for i := 0 to k − 1
      if a_i= 1 then x := (x·power ) mod m
      power := (power·power ) mod m
return x {x equals b^n mod m }
```

– $O((\log m)^2 \log n)$ bit operations are used to find b^n **mod** m.