

### 3. Number Bases and Powers

#### Number Bases

Nowadays, almost everyone writes integers in base 10 (*decimal*) notation, that is as a string of digits in the range 0, ...,9. For example the string 37294 represents the integer

$$3 \cdot 10^4 + 7 \cdot 10^3 + 2 \cdot 10^2 + 9 \cdot 10^1 + 4 \cdot 10^0 = 3 \cdot 10^4 + 7 \cdot 10^3 + 2 \cdot 10^2 + 9 \cdot 10 + 4$$

But we can use other bases. For example, if we use base 8 (*octal*) notation, some calculation shows that this number can be written as

$$1 \cdot 8^5 + 1 \cdot 8^4 + 0 \cdot 8^3 + 6 \cdot 8^2 + 5 \cdot 8 + 6$$

and hence is represented by  $(110656)_8$ .

#### Notes

- (1) When no base is indicated, we *assume* that the number is written in base 10.
- (2) Leading zeros can be omitted. 00123 and 123 give the same integer.

Any integer  $B > 1$  can be used as a number base. The positive integer  $A$  can be written

$$A = a_{n-1}B^{n-1} + a_{n-2}B^{n-2} + \dots + a_1B + a_0,$$

with  $0 \leq a_i < B$ , for each  $i$ , and  $a_{n-1} \neq 0$ . Then we write  $A = (a_{n-1}a_{n-2} \dots a_1a_0)_B$  as the representation of  $A$  in base  $B$  – it is an “ $n$ -digit” number.

Note that, as  $a_{n-1} \geq 1$ , and  $a_{n-1} < B$ , we have  $B^{n-1} \leq A < B^n$ . This identifies  $n$ .

Of course, we need symbols for each value in the range 0,1,...,  $B - 1$ . For  $B$  up to 10, we use the ordinary digits. For larger  $B$ , we need to allocate new symbols.

For  $B = 16$ , we get hexadecimal notation (used in computing). The coefficients 10, 11, 12, 13, 14, 15 are denoted by  $A, B, C, D, E, F$ .

For example, the hexadecimal number  $(FADE)_{16}$  is the integer

$$\begin{aligned} F \cdot 16^3 + A \cdot 16^2 + D \cdot 16 + E &= 15 \cdot 16^3 + 10 \cdot 16^2 + 13 \cdot 16 + 14 \\ &= 64222 \end{aligned}$$

This can be calculated more efficiently by writing it in *nested form* :

$$\begin{aligned} (FADE)_{16} &= F \cdot 16^3 + A \cdot 16^2 + D \cdot 16 + E \\ &= ((F \cdot 16 + A) \cdot 16 + D) \cdot 16 + E \\ &= ((15 \cdot 16 + 10) \cdot 16 + 13) \cdot 16 + 14 \\ &= (250 \cdot 16 + 13) \cdot 16 + 14 \\ &= 4013 \cdot 16 + 14 \\ &= 64222. \end{aligned}$$

In this version, each step is an addition or a multiplication by 16. We do not need to compute the successive powers of 16 which were needed in the first version.

To express an integer given in base 10 in another base, we use the Division Theorem repeatedly. We illustrate the technique by an example.

**Example** Express the integer 943 in base 9.

**Solution**

$$(1) 943 = 104.9 + \underline{7}$$

$$(2) 104 = 11.9 + \underline{5}$$

$$(3) 11 = 1.9 + \underline{2}$$

$$(4) 1 = 0.9 + \underline{1}$$

The remainders give us the “digits” of the base 9 representation in reverse order

$$943 = (1257)_9$$

To see why this happens, look at the following calculation :

$$\begin{aligned} 943 &= 104.9 + 7 && \text{line (1)} \\ &= (11.9 + 5).9 + 7 && \text{line (2)} \\ &= ((1.9 + 2).9 + 5).9 + 7 && \text{line (3)} \\ &= 1.9^3 + 2.9^2 + 5.9 + 7 && \text{line (4)} \\ &= (1257)_9 \end{aligned}$$

## Powers

As we shall see later, modern cryptography involves calculating high powers of an integer. Typically, the exponent will be a 300-digit number (in base 10).

The *obvious* way to compute  $x^N$  involves  $N-1$  multiplications. For *large*  $N$ , this is not a practical method!

Thinking about the base 2 (binary) representation of  $N$  gives a *much* better method. For the general case, the notation is rather tiresome. We illustrate the method with a couple of examples.

**Example 1**  $7 = (111)_2 = (1 \cdot 2 + 1) \cdot 2 + 1$

$$\begin{aligned}x^7 &= (x^3)^2 \cdot x \\ &= (x^2 \cdot x)^2 \cdot x\end{aligned}$$

This involves just *four* multiplications – two squarings and two “multiply by  $x$ ”.

**Example 2**  $13 = (1101)_2 = ((1 \cdot 2 + 1) \cdot 2 + 0) \cdot 2 + 1$

$$\begin{aligned}x^{13} &= (x^6)^2 \cdot x \\ &= ((x^3)^2)^2 \cdot x \\ &= ((x^2 \cdot x)^2)^2 \cdot x\end{aligned}$$

This involves just *five* multiplications – three squarings and two “multiply by  $x$ ”.

We now state *without proof* what happens in general.

### Result 1

Suppose that the integer  $N$  has base 2 representation involving  $b$  bits. Then  $x^N$  can be evaluated using at most  $2(b-1)$  multiplications.

The base 10 version is based on the observation that  $2^{10}$  is 1024. It follows that a  $d$ -digit number in base 10 will have at most  $10d/3 + 1$  bits in base 2.

### Result 2

Suppose that the integer  $N$  has base 10 representation involving  $d$  bits. Then  $x^N$  can be evaluated using at most  $20d/3$  multiplications.

To show the efficiency of this method, we observe that, if  $N$  is one billion, then  $x^N$  can be evaluated with just 60 multiplications.