

## Counting, symbols, positions, powers, choice, arithmetic, binary, translation, hex, addresses, and gates.

**Counting.** Suppose the concern is a collection of objects. As an example, let the objects be denoted  $U = \{a, b, c\}$ . Consider how it is possible to make a selection from  $U$ . Here are the possibilities

$$P(U) = \{\{\}, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}.$$

The so-called empty selection  $\{\}$  is often denoted  $\emptyset$ . To simplify things, do not distinguish between  $\{b, a\}$  and  $\{a, b\}$  or permutations of any of the selections.

Mathematics is a subject concerned with meaningful and useful abstractions, i.e., the extraction of properties not common to all collections. As an example consider the process of *counting* and the associated concept of number. For the purpose of counting, the selections  $\{a\}$ ,  $\{b\}$ , and  $\{c\}$  are not distinguished, nor are  $\{a, b\}$ ,  $\{a, c\}$ , and  $\{b, c\}$ . Meanwhile, the selections  $\emptyset$ ,  $\{a\}$ ,  $\{a, b\}$ , and  $\{a, b, c\}$ , are all distinguished from each other through counting.

**Symbols.** It is convenient to introduce symbols to facilitate comparisons. Traditionally,  $\{a\}$ ,  $\{b\}$ , and  $\{c\}$  are associated with the counting symbol '1', and  $\{a, b\}$ ,  $\{a, c\}$ , and  $\{b, c\}$  are associated with '2'. In this example there is a concept of 'three'. Within a given context it is assumed that this very concept is fixed and it is referred to as the 'base' of the counting. If the symbol 0 is used for the count of  $\emptyset$ , three symbols 0,1,2 have been introduced, and thus the concept behind the base matches the numbers of symbols. The fundamental notational problem concerns the counting symbol for the selection  $\{a, b, c\}$ .

**Positions.** Observe that there are *more* distinct selections than there are counting symbols. One solution to this problem is the introduction of one more counting symbol. This solution is far from satisfactory since there will always be a need for more counting symbols. A different solution to the problem is based on the clever idea that the existing symbols can be concatenated and that the relative position of the symbol affects the interpretation. When two symbols are concatenated there are the following new choices: 00,01,02,10,11,12,20,21,22. By convention, the leading or leftmost symbol represents how many times three distinct objects are present in the collection. It follows that  $\{a, b, c\}$  is represented by the counting number 10 with respect to the base three. Similarly,  $\{a, b, c, d, e, f\} = \{a, b, c\} \cup \{d, e, f\}$  is represented by 20 with respect to base three, and  $\{a, b, c, d, e\} = \{a, b, c\} \cup \{d\} \cup \{e\}$  by 12. Since 00,01,02 have the same meaning as 0,1,2, it is agreed that leading zeroes are dropped, but trailing zeroes are kept.

**New positions.** To better understand how this system of notation is extended to larger collections observe that  $\{a, b, c\} = \{a\} \cup \{b\} \cup \{c\}$ , so there are not enough symbols and a new position is needed in the notation. Similarly,

$$\{a, b, c, d, e, f, g, h, i\} = \{a, b, c\} \cup \{d, e, f\} \cup \{g, h, i\},$$

so again a new position is needed. This time 100 denotes the count with respect to the base three. As a practice example consider 1012. Here there is 1 collection of three groups of three groups of threes, 0 collections of three groups of threes, 1 collections of threes, and 2 collection of ones

**Power notation.** The previous discussion seems unnecessarily complicated. The core problem is the restriction to only use as many symbols as the base in the counting, which in this case is three. To clarify things, let  $(1012)_{three}$  denote the value in the example. The subscript indicates the base, which does not have its own symbol. A natural task is to find the string of digits in  $(x)_{ten}$  such that  $(x)_{ten} = (1012)_{three}$ . To answer this, use power notation and write

$$(1012)_{three} = (1 \cdot 3^3 + 0 \cdot 3^2 + 1 \cdot 3^1 + 2 \cdot 3^0)_{ten} = (27 + 3 + 2)_{ten} = (32)_{ten}.$$

To be sure, the concept of counting is separate from the ideas behind the notation. Since base ten is the one we are most familiar with,  $(32)_{ten}$  is more meaningful to us than  $(1012)_{three}$  because it relates to our previous experiences. The real difference concerns only the choice of collecting things in tens or threes as we count. Look at

$$\{[a, b, c, d, e, f, g, h, i, j], [k]\}$$

and  $(11)_{ten}$ , and compare with

$$\{[[a, b, c], [d, e, f], [g, h, i]], [j], [k]\}$$

and  $(102)_{three}$ . Similarly, compare

$$\{[[[a, b], [c, d]], [[e, f], [g, h]]], [i, j], [k]\}$$

and  $(1011)_{two}$  with

$$\{[a, b, c, d, e, f, g, h], [i], [j], [k]\}$$

and  $(13)_{eight}$ . This correspondence goes far beyond mere counting, and is very satisfying as arithmetic algorithms are seen to not depend on the choice of base.

**Choice.** With the acceptance of bases other than ten, several questions arise. Why should a base different than ten ever be considered? Which base is the best base? The answer to the first question becomes apparent as one examines the electronic hardware buried deep inside digital computers. The technology of two-state switches has brought the base *two* into prominence. Subsequent software issues forced base 8, or better, base 16 to the forefront. To answer the second question these developments have shown that the best base is the one that serves the purpose the best. Since the purpose varies, the choice of base varies.

**Arithmetic.** Arithmetic is introduced using tables. The addition table in base three is given by

$$\begin{array}{r|rrr}
 + & 0 & 1 & 2 \\
 \hline
 0 & 0 & 1 & 2 \\
 1 & 1 & 2 & 10 \\
 2 & 2 & 10 & 11
 \end{array}$$

Next calculate  $121 + 220$  using the familiar algorithms as in

$$\begin{array}{r}
 \underline{1} \quad \underline{1} \\
 \phantom{1} 1 \quad 2 \quad 1 \\
 \phantom{1} 2 \quad 2 \quad 0 \quad + \\
 \hline
 1 \quad 1 \quad 1 \quad 1
 \end{array}$$

This is checked like this

$$\begin{aligned}
 (121)_{three} + (220)_{three} &= (9 + 6 + 1)_{ten} + (18 + 6)_{ten} = (40)_{ten} \\
 &= (27)_{ten} + (9)_{ten} + (3)_{ten} + (1)_{ten} = (1111)_{three} .
 \end{aligned}$$

The multiplication table is given by

$$\begin{array}{r|rrr}
 \times & 0 & 1 & 2 \\
 \hline
 0 & 0 & 0 & 0 \\
 1 & 0 & 1 & 2 \\
 2 & 0 & 2 & 11
 \end{array}$$

Calculate  $12 \times 21$  using

$$\begin{array}{r}
 \underline{1} \quad \underline{1} \\
 \phantom{1} 1 \quad 2 \\
 \phantom{1} 2 \quad 1 \quad \times \\
 \hline
 \phantom{1} 1 \quad 2 \\
 \phantom{1} 0 \quad 1 \\
 \hline
 1 \quad 0 \quad 2 \quad 2
 \end{array}$$

Verify this by

$$\begin{aligned}
 (12)_{three} \times (21)_{three} &= (3 + 2)_{ten} \times (6 + 1)_{ten} = (5)_{ten} \times (7)_{ten} \\
 &= (35)_{ten} = (27 + 6 + 2)_{ten} = (1022)_{three} .
 \end{aligned}$$

**Binary.** In the history of digital electronics it soon became clear that devices were best built based on two states such as on/off or high/low. This suggests using base two and the digits 1 for on/high and 0 for off/low. The corresponding notation is referred to as binary. The addition table is given by

$$\begin{array}{r|rr}
 + & 0 & 1 \\
 \hline
 0 & 0 & 1 \\
 1 & 1 & 10
 \end{array}$$

and the multiplication table is given by

$$\begin{array}{r|l} \times & 0 & 1 \\ \hline 0 & 0 & 0 \\ 1 & 0 & 1 \end{array}$$

As an example, compute  $1010 \times 1100$

$$\begin{array}{r} 1010 \\ 1100 \times \\ \hline 0000 \\ 0000 \\ 1010 \\ 1010 \\ \hline 1111000 \end{array}$$

This agrees with

$$\begin{aligned} (1010)_{two} \times (1100)_{two} &= (8 + 2)_{ten} \times (8 + 4)_{ten} = (10)_{ten} \times (12)_{ten} = \\ &= (120)_{ten} = (64 + 32 + 16 + 8)_{ten} = (1111000)_{two} . \end{aligned}$$

The simple binary multiplication table in fact simplifies the arithmetic algorithm since no multiplication produces a carry. This is not the case in bases three or higher.

**Translation.** The obvious drawback with binary notation is the long sequences of digits that are difficult to read. A remedy is offered by grouping numbers as illustrated by the following. Leading zeroes are kept to simplify the process. Consider the move from binary to base four in

$$(1001)_{two} = (10 \mid 01)_{two} = (21)_{four}$$

or

$$(1110)_{two} = (11 \mid 10)_{two} = (32)_{four} .$$

Similarly, change from binary to base eight as in

$$(110101)_{two} = (110 \mid 101)_{two} = (65)_{eight} .$$

**Hex.** When this process is taken one step further one reaches eight digit binary numbers. In the context of computer memory each digit is referred to as a *bit* and all eight as a *byte*. This time there is a twist to the story. The next base to use is sixteen. This takes us beyond the familiar ten digits. There is a need for more symbols and the convention is to use *A, B, C, D, E, F* to have a complete set. Instead of referring to base sixteen the term *hex* is used for hexadecimal. Here is an example

$$(11010111)_{two} = (1101 \mid 0111)_{two} = (D7)_{hex} .$$

Here is another

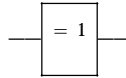
$$(98)_{ten} = (64 + 32 + 2)_{ten} = (01100010)_{two} = (0110 \mid 0010)_{two} = (62)_{hex} .$$

**Addresses.** Eight bit address space runs from  $(00)_{hex}$  to  $(FF)_{hex}$ . Sixteen bit address space runs from  $(0000)_{hex}$  to  $(FFFF)_{hex}$ . The size of this kind of memory is given by

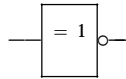
$$(FFFF)_{hex} + (0001)_{hex} = (10000)_{hex} = (2^{16})_{ten} = (65536)_{ten}.$$

The first IBM personal computer had exactly this amount of memory. To confuse things more, the phrase 64k-bytes is coined. Observe how two address bytes are used to distinguish between approximately 64k bytes.

**Gates.** To better understand how binary is used in conjunction with electronic circuits, consider the basic logic gates. The first example is the *relay* with symbol



A relay uses a power source to restore the signal to high if there is sufficiently high signal on the input. The output of a relay can be used as a power source to a different relay. Next there is the *inverter* or NOT-gate with symbol

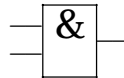


The inverter changes a low signal to high and a high signal to low. The following tables represent the two gates

$x$	= 1
0	0
1	1

$x$	= 1
0	1
1	0

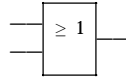
Next, suppose there are two input lines. Pass the first input through a relay and use the output as power source of a second relay with the second line as input. Both lines must have a high signal for the second relay to have a high output. This resulting gate is called an AND-gate. Its symbol is given by



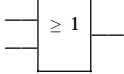
Its table is given by

$x_1$	$x_0$	&
0	0	0
0	1	0
1	0	0
1	1	1

When the signals are interpreted as binary bits, this gate corresponds directly to the multiplication table. If two relays have their output joined to a single output line then the result is an OR-gate. To get a high signal at least one input must be high. The symbol used here is given by



and the table is

$x_1$	$x_0$	
0	0	0
0	1	1
1	0	1
1	1	1