Systems of Numeration

numeral — a specific word in a natural language, or a symbol, that represents a specific number

Examples:

English zero one two three four five six seven eight nine

0 1 2 3 4 5 6 7 8 9

Arabic ٠ ١ ٢ ٣ ٤ ٥ ٦ ٧ ٨ ٩

Bengali ০ ১ ২ ৩ ৪ ৫ ৬ ৭ ৮ ৯

numeral system — a system of numeration is a writing system for expressing numerals and a mathematic system for representing numbers of a given set, using symbols in a consistent manner.

©2006-09 McQuain, Feng & Ribbens

Positional Notation

A *positional* or *place-value* notation is a numeral system in which each position is related to the next by a constant multiplier, called the *base* or *radix* of that numeral system.

The value of each digit position is the value of its digit, multiplied by a power of the base; the power is determined by the digit's position.

The value of a positional number is the total of the values of its positions.

So, in positional base-10 notation:

$$73901 = 7\times10^4 + 3\times10^3 + 9\times10^2 + 0\times10^1 + 1\times10^0$$

And, in positional base-2 notation:

$$100100000101011101 = 1\times2^{16} + 1\times2^{13} + 1\times2^7 + 1\times2^5 + 1\times2^3 + 1\times2^2 + 1\times2^0$$

Why is the second example a cheat?

©2006-09 McQuain, Feng & Ribbens

Converting from base-10 to base-2

Given a base-10 representation of an integer value, the base-2 representation can be calculated by successive divisions by 2:

73901	Remainder
36950	1
18475	0
9237	1
4618	1
2309	0
1154	1
577	0
288	1
144	0
72	0
36	0
18	0
9	0
4	1
2	0
1	0
0	1

10010000010101101

©2006-09 McQuain, Feng & Ribbens

Converting from base-2 to base-10

Given a base-2 representation of an integer value, the base-10 representation can be calculated by simply expanding the positional representation:

$$10010000010101101 = 1\times 2^{16} + 1\times 2^{13} + 1\times 2^{7} + 1\times 2^{5} + 1\times 2^{3} + 1\times 2^{2} + 1\times 2^{0}$$

$$= 65536 + 8192 + 127 + 32 + 8 + 4 + 1$$

$$= 73901$$

©2006-09 McQuain, Feng & Ribbens

Other Bases

Are analagous… given a base-10 representation of an integer value, the base-16 representation can be calculated by successive divisions by 16:

```
 73901     Remainder
  4618        13 --> D
   288        10 --> A
    18         0
     1         2
     0         1
```

120AD

The choice of base determines the set of numerals that will be used.

base-16 (hexadecimal or simply hex)

numerals: 0 1 . . . 9 A B C D E F

©2006-09 McQuain, Feng & Ribbens

Converting from base-2 to base-16

Given a base-2 representation of an integer value, the base-16 representation can be calculated by simply converting the nybbles:

```
1 0010 0000 1010 1101
1    2    0    A    D
```

The same basic "trick" works whenever the target base is a power of the source base:

```
10 010 000 010 101 101
 2   2   0  2    5   5  : octal
```


©2006-09 McQuain, Feng & Ribbens

Important Bases in Computing

base-2	binary	0 1
base-8	octal	0 1 2 3 4 5 6 7
base-10	decimal	0 1 2 3 4 5 6 7 8 9
base-16	hex	0 1 2 3 4 5 6 7 8 9 A B C D E F

©2006-09 McQuain, Feng & Ribbens

Hardware and Bases

The simplest sort of hardware storage device would logically resemble a simple on/off switch, and hence be able to represent or "store" two different states.

This would provide for a direct hardware-level representation of integer values in base-2.

Direct representation of other bases is possible, but would require more complex individual storage devices capable of representing a larger number of distinct states.

Indirect hardware-level representation of other bases is also possible. For example:

```
73901 -->  0111 0011 1001 0000 0001
```



However, this often requires more bits than a pure base-2 representation, and more complex hardware for basic arithmetic operations.

©2006-09 McQuain, Feng & Ribbens

Impact of Hardware Limitations

Any storage system will have only a finite number of storage devices.

Whatever scheme we use to represent integer values, we can only allocate a finite number of storage devices to the task.

Put differently, we can only represent a (small) finite number of bits for any integer value.

This means that computations, even those involving only integers, are inherently different on a computer than in mathematics.

©2006-09 McQuain, Feng & Ribbens

Example: 32-bit Integers

As an example, suppose that we decide to provide support for integer values represented by 32 bits.

There are 2^32 or precisely 4,294,967,296 different patterns of 32 bits.

So we can only represent that many different integer values.

Which integer values we actually represent will depend on how we interpret the 32 bits:

```
1 bit for sign, 31 for magnitude (abs value):      -2147483647 to +2147483647

32 bits for magnitude (no negatives):                        0 to +4294967295

2's complement representation:                     -2147483648 to +2147483647
```


©2006-09 McQuain, Feng & Ribbens