IST 230 Number Systems (Epp/1.5) Fall 2001
9/7/01

- Humans tend to use base 10 → 10 fingers, etc.
- Numbers represented as a concatenation (juxtaposition) of single digit symbols (Base 10 {0, 1, 2, 3, 4, 5, 6, 7, 8, 9})

In base 10, $(X)_{10} = (x_n x_{n-1} \cdots x_2 x_1 x_0)_{10}$

And the value of the number is given by

$$X = (x_n * 10^n) + (x_{n-1} * 10^{n-1}) + \cdots + (x_2 * 10^2) + (x_1 * 10^1) + (x_0 * 10^0)$$

Ex: $5,268 = (5 * 1000) + (2 * 100) + (6 * 10) + (8 * 1)$

The base (in this case 10) can be changed at will (without actually changing the number's value)

- Machines prefer base 2; Computer-oriented people often find it convenient to use powers of 2 bases such as base 8 (octal) or base 16 (Hexadecimal)

Base 2 (Binary): Given a number with value X,

$$X = (x_n * 2^n) + (x_{n-1} * 2^{n-1}) + \cdots + (x_2 * 2^2) + (x_1 * 2^1) + (x_0 * 2^0)$$

Then we write the binary representation of X as

$$(X)_2 = (x_n x_{n-1} \cdots x_2 x_1 x_0)_2$$

Where, for each $i = 0, 1, 2, \ldots, n$, each x_i takes it's value from the binary alphabet {0, 1}

Binary → Decimal

- Just apply value formula for X
  (doing decimal addition for each polynomial term)

Ex: $(X)_2 = (1\ 0\ 1\ 1\ 0\ 0\ 1\ 0)_2$

Then $(X)_{10} = 1*2^7 + 0*2^6 + 1*2^5 + 1*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 0*2^0$

$= 128 + 0 + 32 + 16 + 0 + 0 + 2 + 0$

$= (178)_{10}$

Decimal → Binary

Write decimal number (X_{10}) as a sum of powers of 2
(starting w/ highest power of 2 which is $\leq (X)_{10}$

Ex (Same as last one, but reverse it)

$(178)_{10} = 128 + (178-128) = 2^7 + 50$

$\uparrow = 2^7$ = highest power of 2 ≤ 178

$(50)_{10} = 32 + (50-32) = 2^5 + 18$

$\uparrow = 2^5$ = highest power of 2 ≤ 50

$(18)_{10} = 16 + 2 = 2^4 + 2^1$

$\uparrow = 2^4$ = highest power of 2 ≤ 18

Hence $(178)_{10} = 2^7 + 2^5 + 2^4 + 2^1$

$= (1\ 0\ 1\ 1\ 0\ 0\ 1\ 0)_2$

$2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

Binary Addition / Subtraction

Adding in binary follows exactly the same procedure as in decimal except that there are only 2 symbols instead of $(10)_{dec}$.

Carry 1

1)	2)	3)	4)
0	0	1	1
+0	+1	+0	+1
= 0	= 1	= 1	= 10

This is the same "carry" method as $(9)_{dec} + (1)_{dec} = (10)_{dec}$

Ex:
$$\begin{array}{r} {\scriptstyle 1\ \ \ \ \ } \\ 1101 \\ +\ 0110 \\ \hline = 10011 \end{array}$$

Subtraction is handled exactly inverse, as you should expect.

→ Try Binary Addition / Subtraction Using one of the Binary Arithmetic Links

↳ Make a Binary Addition / Subtraction Table.

Binary Multiplication

Multiplication in binary same as decimal except multiplication table is different

Multiplier \ Multiplicand	0	1
0	0	0
1	0	1

Ex:
$$\begin{array}{r} 1011 \\ 0110 \\ \hline 0000 \\ 1011 \\ 1011 \\ 0000 \\ \hline 10000010 \end{array}$$

← Result

1 1 1 1 ← Carries

Binary Equivalent of Numerals in a Higher Base, n (up to Base 16 = 2^4)

$(0)_n = (0)_2$
$(1)_n = (1)_2$
$(2)_n = (10)_2$
$(3)_n = (11)_2$
$(4)_n = (100)_2$
$(5)_n = (101)_2$
$(6)_n = (110)_2$
$(7)_n = (111)_2$
$(8)_n = (1000)_2$
$(9)_n = (1001)_2$
$(10)_n \equiv A = (1010)_2$
$(11)_n \equiv B = (1011)_2$
$(12)_n \equiv C = (1100)_2$
$(13)_n \equiv D = (1101)_2$
$(14)_n \equiv E = (1110)_2$
$(15)_n \equiv F = (1111)_2$

Hence the base using 3 binary bits and 8 symbols (0-7) } is Octal

The base using 4 binary bits and 16 symbols (0-F) } is Hexadecimal

Hexadecimal is particularly convenient since most computers these days use 16, 32 or 64 bit addresses

Notation: 8 bits = 1 byte
16 bits = 2 bytes = 1 word
32 bits = 2 words = 1 dword

Doing this type of Arithmetic essential for assembly/machine language programming & often needed to Debug any program.

→ Try Bin/octal/Hex/Decimal Calculators

Next level of Abstraction

→ Modulo Arithmetic

Idea is simple. Do arithmetic in various bases, But Do Not Carry, just Truncate

Hence $((5)_{10} * (3)_{10}) \bmod 10 = \begin{array}{r} 5 \\ *\ 3 \\ \hline \end{array}$

$= 5$ (Dont carry 1!)

Another way to look at it:

$((5)_{10} * (3)_{10}) \bmod 10 = \text{Remainder}\left\{\frac{(15)_{10}}{(10)_{10}}\right\} = 5$

In general, for any base b

$$((x)_b * (y)_b) \bmod b = \text{Remainder}\left\{\frac{(x)_b * (y)_b}{b}\right\}$$

→ Try The Modulo Arithmetic Table Applet

→ Finally

→ Play the number guessing Game

→ Read why almost every number has a 3 in it