Data Representation

Interpreting bits to give them meaning

Part 1: Numbers, Bases, and Binary

Notes for CSC 100 - The Beauty and Joy of Computing
The University of North Carolina at Greensboro

Class Reminders

For this week:

- Homework 1 due Friday (Sept. 12)
- Review Lab 3 solutions (in Blackboard)
- ***Do the Pre-Lab reading for Lab 4*** (really!)

For the not-so-distant future:

- *Blown to Bits* Chapter 2 - on-line discussion
  - Remember: Try to read and respond to others - it should be a discussion (back and forth) not just making a comment and leaving!

What is a number?

Question: You've been working with numbers (almost) all your life - what *are* they?

Example: What is the number 6?

6

六

lyo

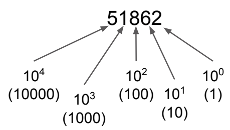
six

seis

110_2

Decimal Representation

Most common written representation of numbers is "decimal notation":



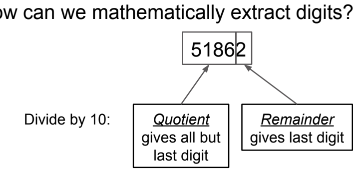
"Representation" is the converse of "Abstraction"

Makes abstractions concrete

<u>Question</u>: Why powers of ten?
Equivalently, why are there 10 different digits?

Decimal Representation

How can we mathematically extract digits?

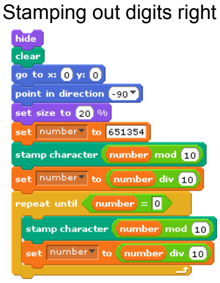


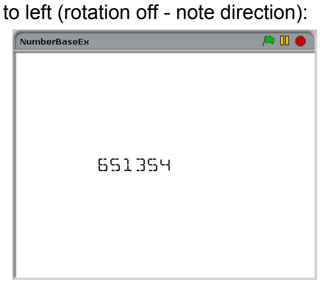
This is like a division operation, but throws away any remainder or fractional part

"mod" gives the remainder after a division

Stamping out decimal representation

Stamping out digits right to left (rotation off - note direction):



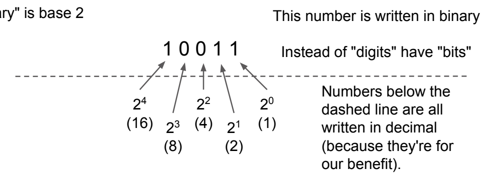


<u>Now let's see this in action...</u>

Binary Representation

The powers used in the representation (also, number of different "digits") is called the ***base***.

- "Decimal" is base 10
- "Binary" is base 2



$1 * 2^4 + 0 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0 = 16 + 2 + 1 =$ ***19***

Converting decimal to binary

Just like the BYOB code, we keep dividing by the base (2), recording remainders and keeping quotients.

Operation	*Quotient*	*Remainder*
43 / 2	21	1
21 / 2	10	1
10 / 2	5	0
5 / 2	2	1
2 / 2	1	0
1 / 2	0	1

First bit found is last bit in binary representation.

1 0 1 0 1 1

Using subscripts to denote base:
$43_{10} = 101011_2$

Converting decimal to binary

Just like the BYOB code, we keep dividing by the base (2), recording remainders and keeping quotients.

Operation	*Quotient*	*Remainder*
43 / 2	21	1
21 / 2	10	1
10 / 2	5	0
5 / 2	2	1
2 / 2	1	0
1 / 2	0	1

Practice problems:

1_{10} = ________$_2$
6_{10} = ________$_2$
8_{10} = ________$_2$
12_{10} = ________$_2$
23_{10} = ________$_2$
31_{10} = ________$_2$

1 0 1 0 1 1

Using subscripts to denote base:
$43_{10} = 101011_2$

Converting binary to decimal

Keep a position and a value, and at each step move position to right, multiply value by 2 and add the new bit.

Start position: Leftmost bit *Start value: 1*

101101	1
101101	1*2 + 0 = 2
101101	2*2 + 1 = 5
101101	5*2 + 1 = 11
101101	11*2 + 0 = 22
101101	22*2 + 1 = 45

So $101101_2 = 45_{10}$

Some terminology:

Leftmost bit is "most significant bit" or "msb"

Rightmost bit is "least significant bit" or "lsb"

Converting binary to decimal

Keep a position and a value, and at each step move position to right, multiply value by 2 and add the new bit.

Start position: Leftmost bit *Start value: 1*

101101	1
101101	1*2 + 0 = 2
101101	2*2 + 1 = 5
101101	5*2 + 1 = 11
101101	11*2 + 0 = 22
101101	22*2 + 1 = 45

So $101101_2 = 45_{10}$

Practice problems:

11_2 = ______$_{10}$
1001_2 = ______$_{10}$
11011_2 = ______$_{10}$
10001_2 = ______$_{10}$
11111_2 = ______$_{10}$
101011_2 = ______$_{10}$

Counting in binary without converting

Picture an odometer with only two values, 0 and 1

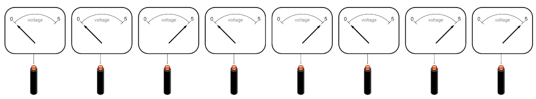
When any wheel goes from 1 to 0, turn the one to the left

0000	$= 0_{10}$	1000	$= 8_{10}$
0001	$= 1_{10}$	1001	$= 9_{10}$
0010	$= 2_{10}$	1010	$= 10_{10}$
0011	$= 3_{10}$	1011	$= 11_{10}$
0100	$= 4_{10}$	1100	$= 12_{10}$
0101	$= 5_{10}$	1101	$= 13_{10}$
0110	$= 6_{10}$	1110	$= 14_{10}$
0111	$= 7_{10}$	1111	$= 15_{10}$

Why binary?

In electronics, you can measure voltages on wires

- Consider 8 wires
- Each with at either 0 volts or 5 volts



Interpreting 0V as 0, and 5V as 1, get: 00101011_2 (= 43_{10})

Voltages can turn on/off switches to create logic circuits

For Future Classes

Some questions for later classes:

Are there useful bases other than binary?

How are pictures or sound clips represented?

Until then:

Practice with this! Binary is the basic language of electronic computers, so if you want to understand modern computers you must be comfortable with their language.

And to answer students' favorite question:

Yes, this will be on the test.