# Data Storage

A) Storing data  Base 10 -> Base 2

1) Every integer Base 10 has an exact Base 2 representation

$712d \dashrightarrow 7x10^2 + 7x10^1 + 2x10^0$

```
 ___
2|712
  356 + 0
  178 + 0
   89 + 0
   44 + 1
   22 + 0
   11 + 0
    5 + 1
    2 + 1
  ->1 + 0
```

Read Backwords -> 712d = 1011001000b

$1x2^9 + 0x2^8 + 1x2^7 + 1x2^6 + 0x2^5 + 0x2^4 + 1x2^3 + 0x2^2 + 0x2^1 + 0x2^0$

512 + 128 + 64 + 8 = 712

2) Octal and Hexadecimal

Octal-base 8

1022002000

3) To store negatives

use “2’s complement”

i.e. complement then add 1

So  -10d -> -(0000/1010b)
  complement-change each
  11110101b
  add 1
  -10d = 11110110b

4) Characters are stored in ASCII code

i.e. each char is given a numerical representation

‘A’ <-> 65d
‘B’ <-> 66d
‘a’ <-> 97d
‘b’ <-> 98d
  etc.

letters and digits are contiguous

the rest is also fairly standard

B) Addition and Subtraction (i.e. negative numbers)

Two methods –

1) "signed" -- use the most significant bit (MSB is left-most) as a "sign bit"

ex. 6 = 0000110

then -6 = 1000110 --> easy to use but difficult to add/subtract

3) "2's complement" --- to represent negatives, first flip all bits (i.e. complement all bits) then add 1

*Note: If # is positive, then 2's complement leaves the number alone*

ex. $15_{10} - 17_{10} = 15_{10} + {}^{-}(17_{10})$

15 = 00001111

17 = 00010001 --> complement of 17 = 11101110 then add 1 = 11101111

-17 = 11101111 --> note that a MSB for negatives is always "1"

so

| | | |
|---|---|---|
| $15_{10}$ | = | 00001111 <--2's complement leaves binary alone |
| $+ -17_{10}$ | = | 11101111 <-- now use 2's comp. since negative |
| -2 | = | 11111110 --> MSB is a "1" we have a negative |

Since MSB is "1" this is a negative number and we must "reverse" the process

11111110 --> subtract 1 = 11111101

--> complement = 00000010 = 2 (recall this is a negative)

C) Errors

There are 2 kinds of errors because of storage from Base 10 -> Base 2

1) In general decimals do NOT have an exact value

Consider

1010.0110 would mean

$1x2^3 + 0x2^2 + 1x2^1 + 0x2^0 + 0x2^{-1} + 1x2^{-2} + 1x2^{-3} + 0x2^{-4}$

8 + 2 + 1/4 + 1/8

10 + 3/8 <- heres the problem

i.e. the nearest to 1/10 is 0.1 -> .1000000192 using 32 bits to store a real

This is called a ***Representational Error***

2) This leads to an ***Accumulation Error***

adding repeated decimals

i.e. add 10 billion 0.1's = 1,000,000,000

(should actually have 1,000,000,192)