Week 2.

Pt. 1

Bits (Binary digits) < 1 / 0 1 byte = 8 bits.

10 bits = 1k values. ASCII => 7bit code.

20 1 Mega represent 128 Latin Alphabet.

30 1 Giga. Unicode => 16 bits.

includes non-Latin alphabets.

ASCII codes: 0-31 Non-Printable characters. 127 del.

32-47, 58-64, 91-96, 123-126 Symbols.

48-57 Numbers.

65-90 Capital characters.

97-122 Minor characters.

Base b: $N = \sum_{i=m}^{n-1} a_i b^i$

Some fractions cannot be represented in other base.

Range: Max-Min

n bits binary range from 0 to $2^n - 1$.

Precision: How precise the number is represent. e.g. 3.1415 9

a.bcde => 5 decimal digits

its presicion is 1 in 10^5.

5 significant figures.

Accuracy & Error: The difference between the measure value and the object's actual value.

Accuracy and Precision

e.g. $\pi = 3.14$ is not error.

Measure a 20 cm stick as 19.89 cm.

is not the same. ∥ the error is 0.11 cm.

Pt. 2.

Sign and Magnitude representation.

$(-1)^S \times M$. the range is 2^n. $(-2^{n-1} - 2^{n-1}-1)$

=> e.g. $+13_{10} = 00001101_2$.

$-13_{10} = 10001101_2$.

Which is used in floating point arithmetic.

Two ways to reperesent 0: 00 or 10 (+0/-0)

Biased Reperesent. (excess-k)

Shift the scale to have only non-negative values.

Range: 2^n $(0 - 2^n - 1)$

Two's Complementary Arithmetic.

Range: 2^n $(-2^{n-1} - 2^{n-1}-1)$ zero has only one reperentation.

$P + Q = 2^n$ In 2's complementary, negative starts with 1 while positive numbers start with 0.

e.g. $2_{10} = 00000010$ $-2_{10} = 11111110$

Operation results fall outside the range => Overflow occurs.

Overflow: adding two positive number yieleding negative.

two negative positive.

e.g. $01111_2 + 01000_2 = 10111_2$.

positive positive negative

sign bit of result different from sign bit of A and B

=> Overflow Occurs.

Carry-in bit => sign bit Carry-out bit - next bit of sign bit.

e.g. 1011

```
  0011
+ 0111
------
  1010
```

In this case, carry in bit is not equal to carry out bit, an overflow occurs.

Shifting: Moving all bits left/right, and put zero at the end.

Shifting $\begin{matrix}\text{left} \\ \text{right}\end{matrix}$ equals $\begin{matrix}\times \\ /\end{matrix}$ the base number.