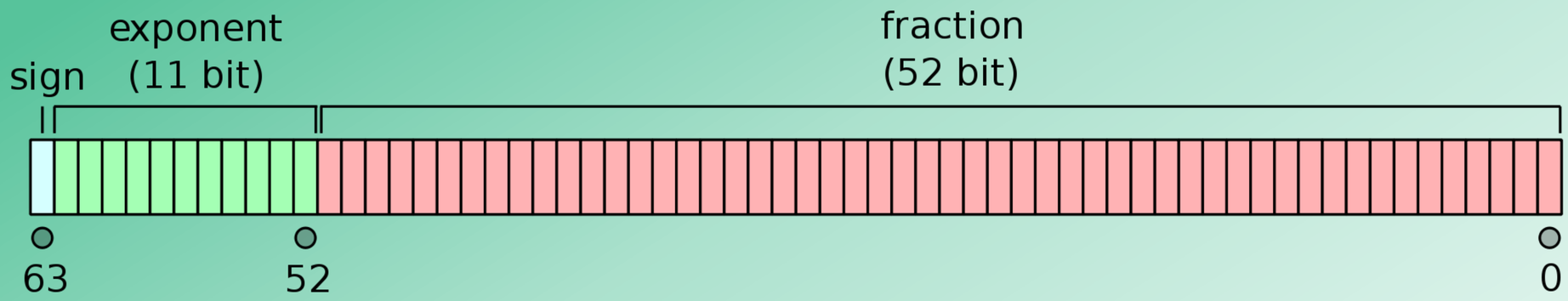


Lecture 7: Data Types

ENAE 380 Flight Software Systems
September 18, 2024

Imprecision

1/10

format(1.0/10 ‘.29f’)

0.10000000000000000555111512313

Data Types

Integer Types

Type	Storage Value	Value Range
signed char	1 byte	-128:127
unsigned char	2 byte	0:255
int	2 or 4 bytes	-32,768:32767 or -2,147,483,648:2,147,483,647
unsigned int	2 or 4 bytes	0:65,535 or 0:4,294,967,295
short	2 bytes	-32,768:32,767
unsigned short	2 bytes	0:65,535
long	4 bytes	-2,147,483,648:2,147,483,647
unsigned long	4 bytes	0:4,294,967,295

UNIVERSITY OF 18 56 MARYLAND

Floating Point Types

Type	Storage Value	Value Range	Precision
float	4 bytes	1.2E-38 to 3.4E+38	6 decimal places
double	8 bytes	2.3E-308 to 1.7E+308	15 decimal places
long double	10 bytes	3.4E-4932 to 1.1E+4932	19 decimal places

UNIVERSITY OF 18 56 MARYLAND

Python's 5 Data Types

- Numbers
  - Int (signed)
  - Long (also in octal and hex) [Python 3 does not differentiate int vs long]
  - Float
  - Complex
- Strings
- Lists []: similar to arrays in C, can be different data types
- Tuples (): "read-only lists"
- Dictionaries {}: no concept of order among elements

UNIVERSITY OF 18 56 MARYLAND

Integer Representation

- Positive numbers only
- Positive and negative numbers
- Ease of human readability
- Speed of computer operations

Integer Representation

- Unsigned
- Sign magnitude
- One’s complement
- Two’s complement

Integer Representation

Virtually all modern computers operate based on 2's complement. Why?

Unsigned

Only positive values
Range: 0 to 2^n - 1, for n bits

Example: 4 bits, values 0 to 15

binary	decimal	hex	binary	decimal	hex
0000	0	0	1000	8	8
0001	1	1	1001	9	9
0010	2	2	1010	10	a
0011	3	3	1011	11	b
0100	4	4	1100	12	c
0101	5	5	1101	13	d
0110	6	6	1110	14	e
0111	7	7	1111	15	f

Sign Magnitude

Human readable way of getting both positive and negative integers.

Hardware that does arithmetic on signed integers is not fast, and it is more complex than the hardware that does arithmetic on 1’s complement and 2’s complement integers.

Sign Magnitude

0101	5
1101	-5

To get inverse of a number, just flip the sign bit

Range: $-2^{(n-1)}+1$ to $2^{(n-1)}-1$

Example: 4 bits

-7 to +7

Sign Magnitude

0000

1000

Ones' Complement

In the past, early computers were based on 1's comp.

Positive integers use the same representation as unsigned

0000	0
0111	7

Ones' Complement

Negation is done by taking a bitwise complement of the positive representation

Find 1's comp of -5

Take +5 in binary: 0101

Take complement of each bit: 1010

That is -5

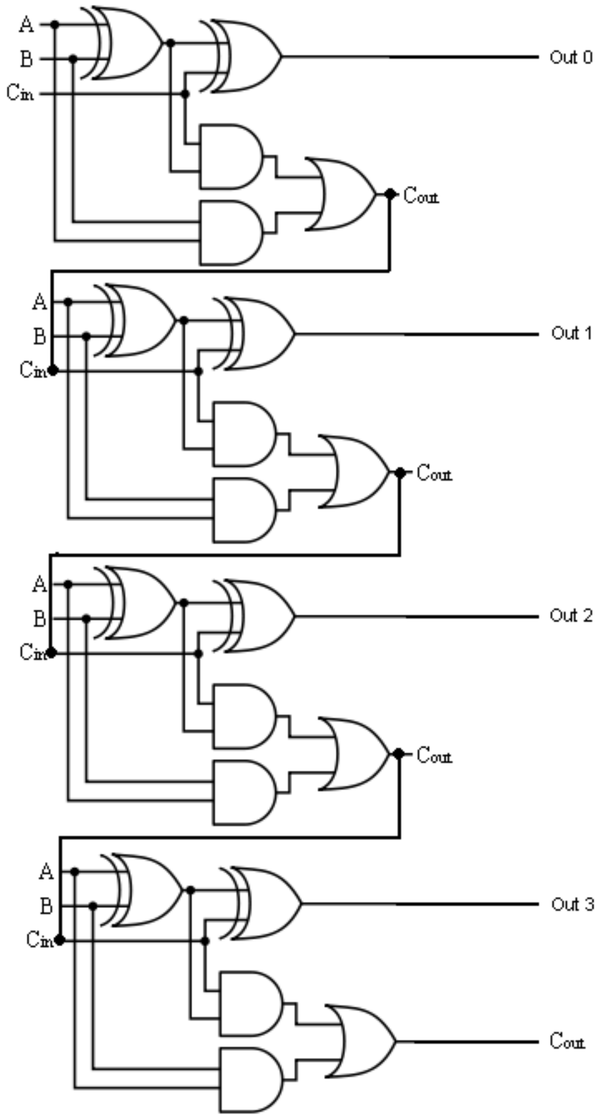
Ones' Complement

Any negative number will have a 1 in the MSB (most significant bit)

There are two representations for 0

Range: $-[2^{(n-1)} - 1]$ to $[2^{(n-1)}-1]$

Two's Complement



A variation on 1's complement that does NOT have two representations of 0. This makes the hardware that does arithmetic faster than for the other representations

Two's Complement

Positive values are the same as before. (They will have a 0 in the MSB)

If it's positive, just write down the value
If it's negative, take the positive value, then take the 1's complement, and then add 1

Range: $-2^{(n-1)}$ to $2^{\wedge(n-1)}-1$

Two’s Complement

Assume 8 bits allocated

What is -5 in 2’s complement?

- Take positive 5 in binary
  - 0000 0101
- Flip the bits
  - 1111 1010
- Add 1 to it
  - 1111 1011

Two’s Complement

Assume 8 bits allocated

What is decimal for 0000 1101 in 2’s complement?

- MSB is 0, so we know it’s a positive, so just take positive value of 0000 1101
- 2^3+2^2+2^0 = 13

Two’s Complement

Assume 8 bits allocated

What is decimal for 1111 1000 in 2’s complement?

- MSB is 1, so we know it’s a negative number. Subtract 1
  - 1111 0111
- Flip the bits
  - 0000 1000
- Convert this to decimal
  - 2^3 = 8
- Remember we started with a negative number so final answer is **-8**

Three Bit Example

a 3 bit example:								
bit pattern:	100	101	110	111	000	001	010	011
1's comp:	-3	-2	-1	0	0	1	2	3
2's comp.:	-4	-3	-2	-1	0	1	2	3

Integer Representation

Virtually all modern computers operate based on 2's complement.

2 + (-1) = 1

Sign and Magnitude

-3

0010
+
1001
=
1011

2's complement

0010
+
1111
=
0001

1

Floating Point Numbers

Computers represent real values in form similar to that of scientific notation. There are standards which define what the representation means so that across computers there will be consistency

This is not the only way to represent floating point numbers, but it is the standard way that IEEE does it.

Floating Point Numbers

Uses scientific notation to encode numbers, with a base and exponent.

Decimal 123.456 = 1.23456 x 10^2

Binary: 10100.110 = 1.0100110 x 2^4

Fixed point has fixed window of representation, limiting range of numbers. Floating point employs a “sliding window” of precision.

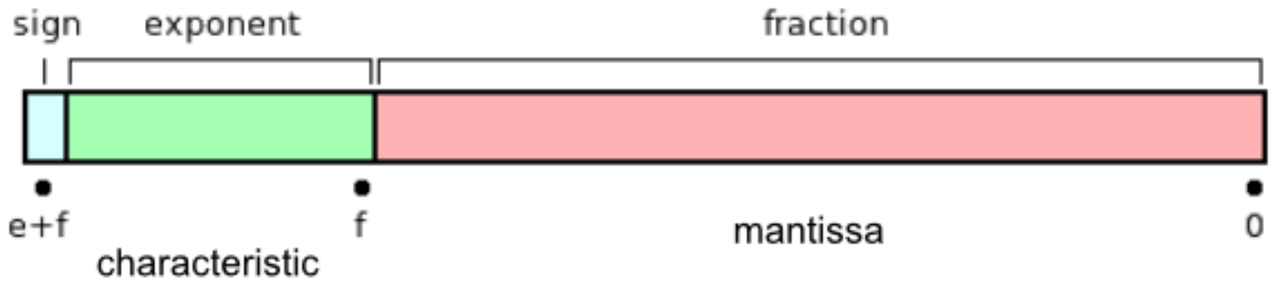
IEEE 754 Standard

IEEE float (32 bits) or double (64 bits)

Floating Point Components

	Sign	Exponent	Fraction
Single Precision	1 [31]	8 [30–23]	23 [22–00]
Double Precision	1 [63]	11 [62–52]	52 [51–00]

Sign Bit



0 denotes a positive number
1 denotes a negative number

Exponent

Need to represent positive and negative exponents. To do this, add a *bias* to actual exponent in order to get the stored exponent. [Bias is $2^{(k-1)}$-1, k bits in exponent field]

For single precision numbers, k=8, bias = 127
For double precision numbers, k=11, bias = 1023

Stored value of 200 indicates exponent of (200-127) = 73

Mantissa

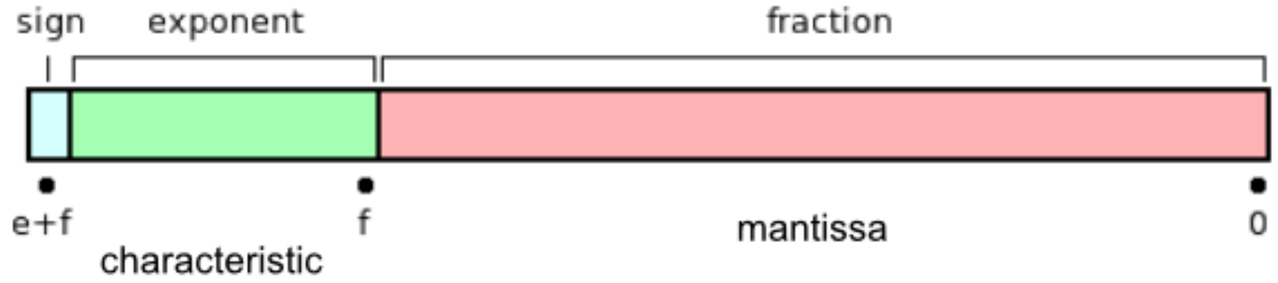
Aka *significand*, represents the precision bits of the number. Composed of an implicit leading bit and fraction bits.

$$.5000 \times 10^{2}$$

$$0.050 \times 10^{3}$$

$$5000. \times 10^{-2}$$

Summary



1. Sign bit is 0 for positive, 1 for negative
2. Exponent base is two
3. Exponent field contains 127 plus true exponent for single-precision, 1023 plus true exponent for double precision
4. First bit of mantissa is typically assumed to be 1.f, where f is the field of fraction bits

Examples: Binary to Decimal

- Separate into sign, exponent, and mantissa
- Extract mantissa from mantissa field and restore leading one (omit trailing zeros)
- Extract exponent from exponent field, subtract bias [2^(k-1)-1], where k is number of bits in exponent field.
  - for 8 bit format, bias = 3, for 32 bit, bias is 127
- De-normalize the number: move binary point so exponent is zero
- Convert binary value to decimal
- Get sign of decimal number according to sign bit

11100111 **-11.5**

separate	**1 110 0111**
mantissa	**1.0111**
exponent	**$110_2 = 6_{10}$; 6-3 = 3**
de-normalize	**$1.0111_2 \times 2^3 = 1011.1$**
convert	**$1011.1_2 = 11.5_{10}$**
sign	**1 —> negative**

Bias = 2^(k-1) - 1
k = 3

00100110 **0.6875**

separate	0 010 0110
mantissa	1.0110
exponent	$010_2 = 2_{10}$; 2-3 = -1
de-normalize	$1.0110_2 \times 2^{-1} = 0.1011$
convert	$0.1011_2 = 0.6875_{10}$
sign	0 —> positive

Examples: Decimal to Binary

2.635_{10} 01000101_2

convert integer	$2_{10} = 10_2$
convert fraction	$0.625_{10} = 0.101_2$
add exponent	10.101×2^0
normalize	1.0101×2^1
exponent and mantissa	$1 + 3 = 4_{10} = 100_2; 0101_2$
sign	positive —> 0

2^-1 = 0.5
2^-2 = 0.25
2^-3 = 0.125

-4.75_{10} 11010011

convert integer	$4_{10} = 100_2$
convert fraction	$0.75_{10} = 0.11_2$
add exponent	100.11×2^0
normalize	1.0011×2^2
exponent and mantissa	$2 + 3 = 5_{10} = 101_2; 0011_2$
sign	negative —> 1

Back to Imprecision

How do you represent 0.1 in binary?

Ariane 5: June 4, 1996

https://www.youtube.com/watch?v=5tJPXYA0Nec

Unmanned rocket launched by ESA exploded 40 seconds after lift-off. $7 billion and decade of development. Destroyed $500 million.

Software error: 64 bit floating point number converted to a 16 bit signed integer.

Overflow

engadget

3 related articles

Latest in Boeing

Boeing simulates worst-case scenarios for space taxi landing

08.25.16

Watch astronauts install a space taxi dock on the ISS

08.19.16

To keep a Boeing Dreamliner flying, reboot once every 248 days

Edgar Alvarez , @abcdedgar
05.01.15

Comments

557
Shares

Sponsc

Th
Wa
Boomerang for Gi

No
Yo
Trackr Bravo

Th

https://www.engadget.com/2015/05/01/boeing-787-dreamliner-software-bug/

unsigned int 32 bytes

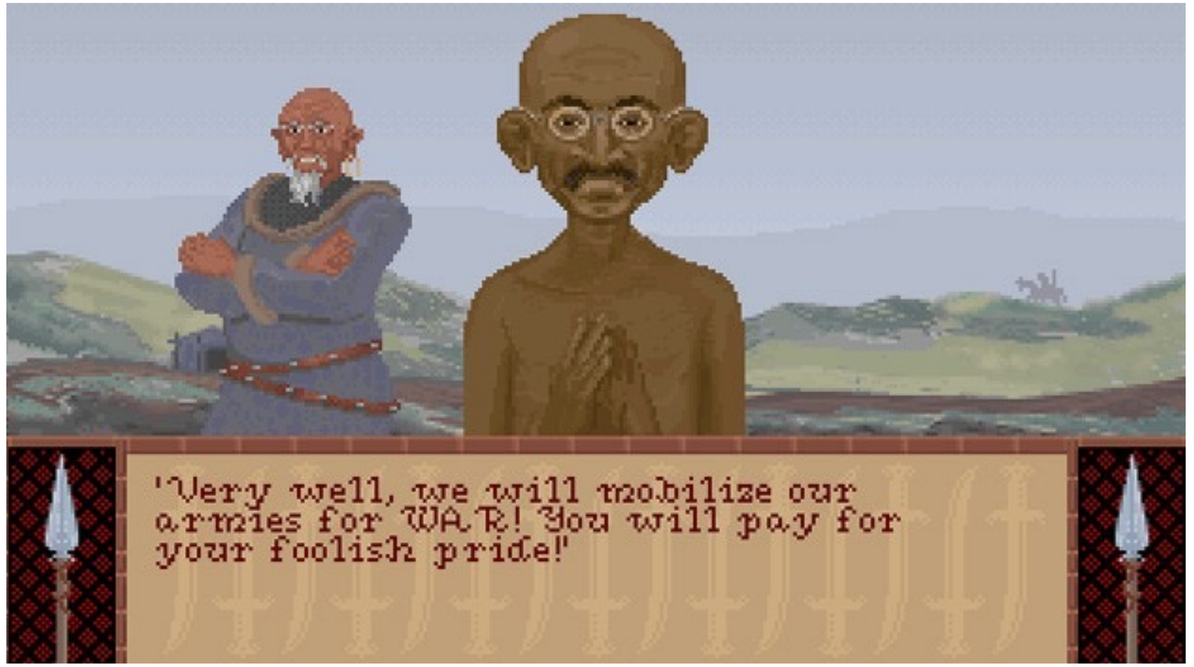
var = $2^{(32-1)}$ - 1 = 2,147,483,647

1 min/60 sec x 1 hour/60 min x 1 day/24 hour = 24855days

So 248.55 days for var in 10 ms increments


LEGO
STAR WARS
THE VIDEO GAME
3+
www.pegi.info


4,000,000,000



http://kotaku.com/why-gandhi-is-such-an-asshole-in-civilization-1653818245

Crowdstrike

https://www.youtube.com/watch?v=wAzEJxOo1ts

https://www.youtube.com/shorts/F2yv30kqSg8