Floating Point Representation

CS 520

Dept. of Computer Science
Univ. of New Hampshire

real numbers often are represented with scientific notation

$$+1.1066 \times 10^{22}$$

normalized form
one non-zero digit to the left of the decimal point

$$11.066 \times 10^{21}$$

$$0.11066 \times 10^{23}$$

no reason we can't show binary numbers in scientific notation

$1.00 \times 2^{-1} = 1/2 = 0.5_{10}$

$1.11 \times 2^{0} = 1 + \frac{1}{2} + \frac{1}{4} = 1.75_{10}$

scientific notation components

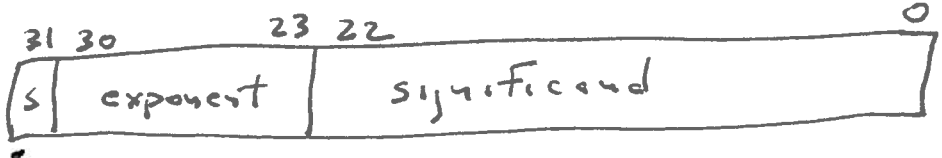
sign
significand
exponent – ~~possibly~~ signed

floating point number is computer support for real numbers that uses binary numbers in scientific notation

IEEE Floating Point

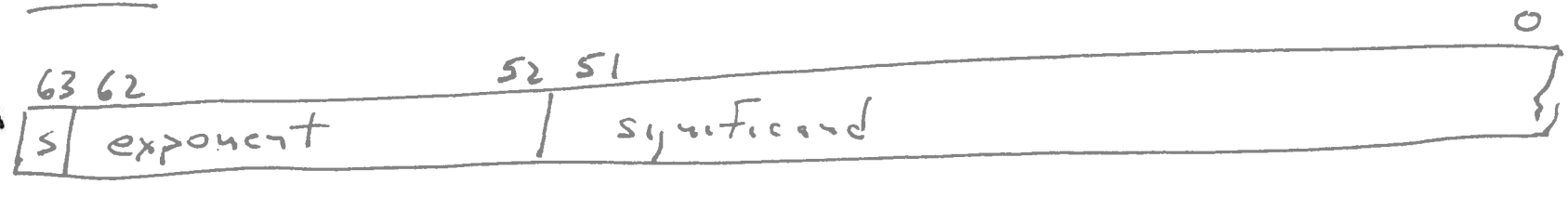
standard established in 1985

single-precision (32 bits)



sign bit

double-precision (64 bits)



IEEE F.p. has normalized form

$$1.XXX\cdots X \times 2^{exp}$$

↑ trick: since this one must be present, don't represent it.

But how do we represent 0.0?

IEEE F.P. exponent

if exponent is all zero bits:

a) significand is all zero bits

then value is ± 0.0

b) significand is not all zero bits

then value is denormalized

$$\pm 0.f \times 2^{-126}$$

↑ i.e. the significand bits

if exponent is all one bits

a) significand is all zero bits

then value is ± Infinity

i.e. divide by zero
overflow

b) significand is not all zero bits

then value is NaN

↳ not a number

i.e. zero divided by zero

$\sqrt{-p}$

otherwise exponent is a biased exponent

actual exponent = stored exponent
minus bias amount

single precision: bias is 127

double precision: bias is 1023

ie treat exponent bits as unsigned integer
and subtract bias amount to
get the actual exponent

example $-0.75_{10} \rightarrow$ IEEE single-precision

$$-\left(\frac{1}{2}+\frac{1}{4}\right) = -.11 \times 2^{0}$$

$$= -1.10 \times 2^{-1}$$

↳ not stored!

actual_exp = stored_exp − 127

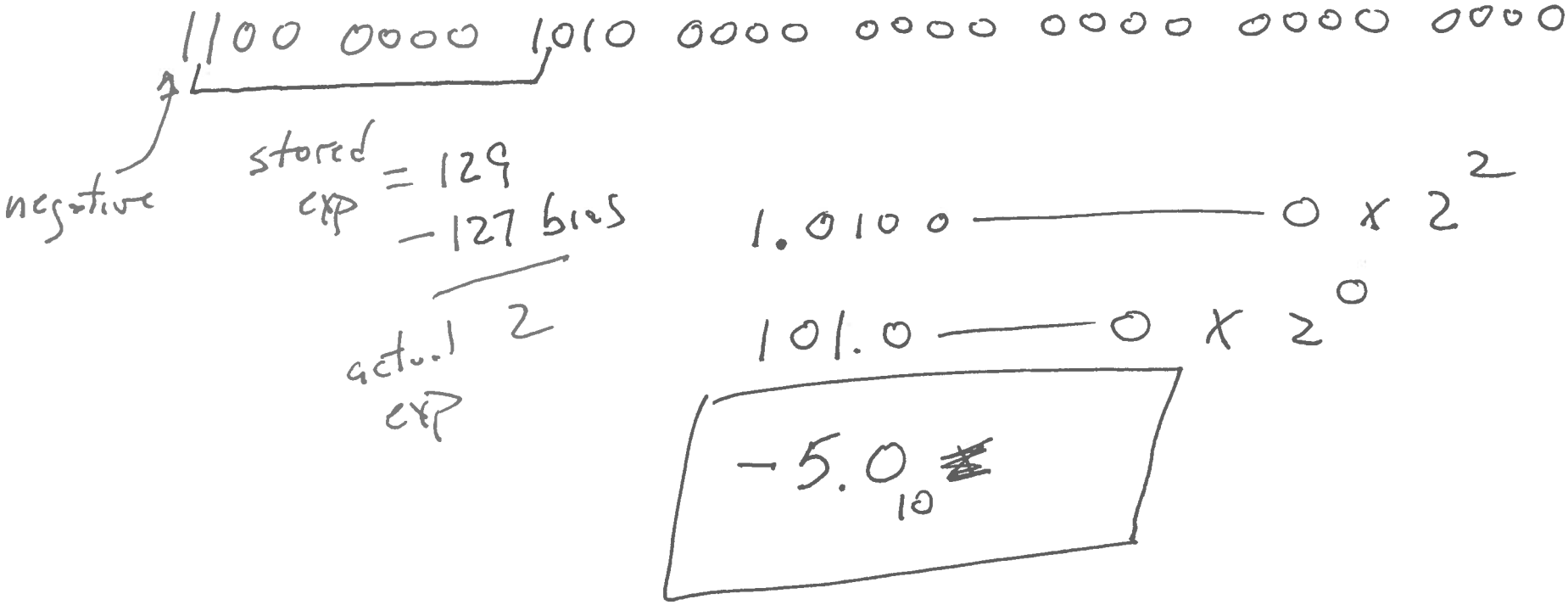
-1 = stored_exp − 127

stored_exp = $+126 = 01111110_2$

Little Endian									Big Endian
00	1011	1111	0100	0000	0000	0000	0000	0000	BF
00	B	F	4	0	0	0	0	0	40
40									00
BF									00

example IEEE F.P. value consisting of these bits

$$C0A00000_{16}$$



example -5.0_{10} → IEEE double-precision

sign = 1

-101.00 —— 0 $\times 2^0$

-1.0100 —— 0 $\times 2^2$

L.E.	B.E.
00	C0
00	14
00	00
00	00
00	00
00	00
14	00
C0	00

actual exp = stored exp − 1023

2 = stored − 1023

stored = 1025 = 10000000001

1100|0000|0001|0100|—— 0

C 0 1 4 00 00 00 00 00 00 00

overflow

like with integers, value can get too big to fit in a fixed-size f.p. container

question: what component of a f.p. number actually overflows?

exponent!

underflow

Floating-point value can get too close to zero to be seen

question: what happens to the exponent in this case?

it gets too negative