CSE241 Sept 11, 2017

Hw#1

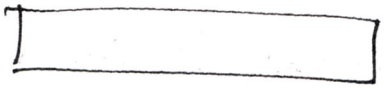
sign-mag +ve | 0 | mag

-ve | 1 | mag

2's complement +ve | 0 | mag

-ve | 2's comply

Assume 8 bits



67

1.00|0011

+67 8 bits sign Magnitude

sign-mag 0|100 0011

-67 8bits 1|100 0011

32

BCD

4-bit code each digit is coded in 4-bit binary

32

0011 0010

32 in BCD 0011|0010

32 in binary value

00100000

Sep 11, 2017

flop: floating point ops/sec

general purpose CPU

Complex data representation

↓ for efficient operations

IEEE 754 standard format
for floating rep.

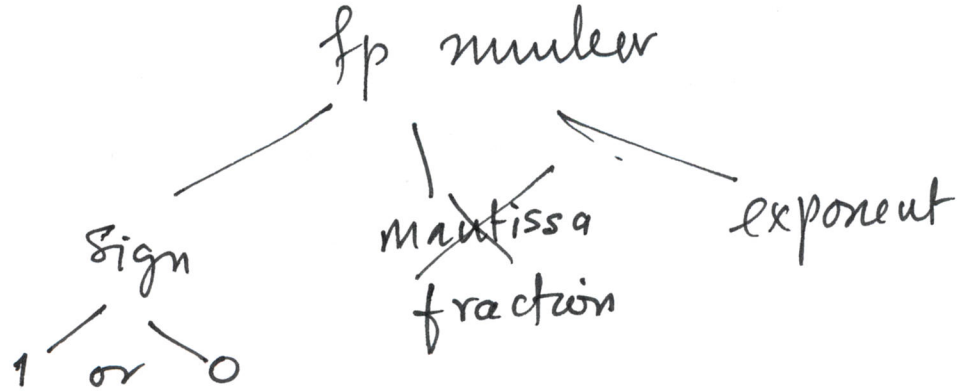
525.125600 4
+ 135245.75679432

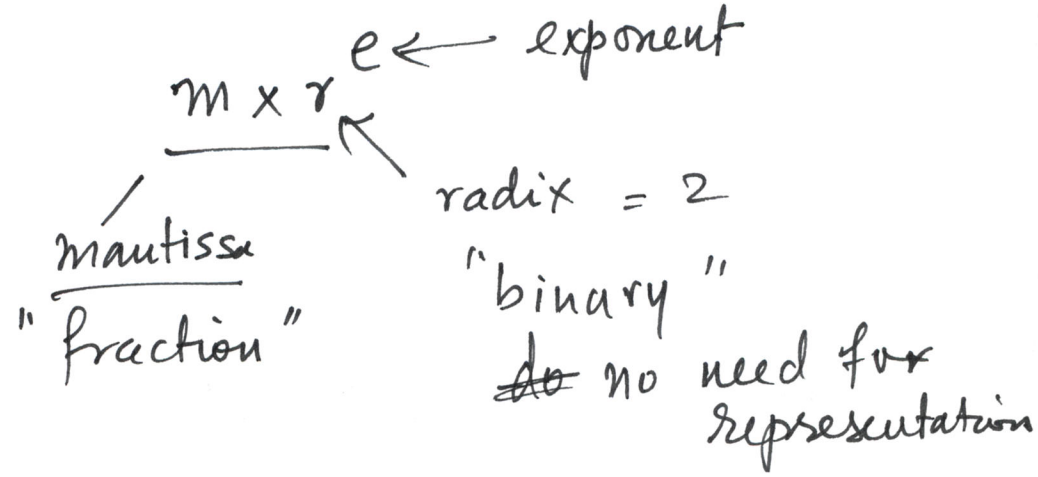
↓ ~~staudfo~~ standard format

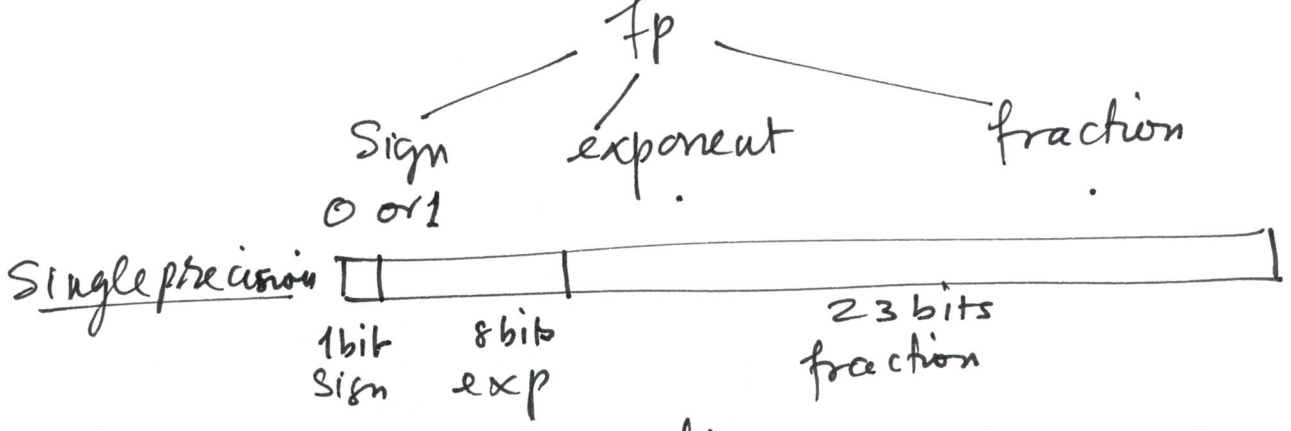
normalized
floating pt.

~~←~~ larger range

Sept 11 2017







fp(sign, exp, fraction, ~~exp~~ radix)

radix

-127 to 127

0 254

$$fp = (sign)\ f \times r^{e}$$

exponent excess -127

$e_{actual} + 127 = e_{rep}$

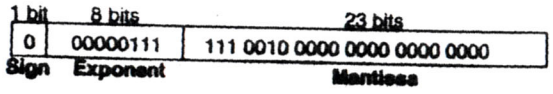
5.3.2 Floating-Point Number Systems

Floating-point numbers are analogous to scientific notation. They circumvent the limitation of having a constant number of integer and fractional bits, allowing the representation of very large and very small numbers. Like scientific notation, floating-point numbers have a *sign*, *mantissa* (M), *base* (B), and *exponent* (E), as shown in Figure 5.25. For example, the number 4.1×10^3 is the decimal scientific notation for 4100. It has a mantissa of 4.1, a base of 10, and an exponent of 3. The decimal point *floats* to the position right after the most significant digit. Floating-point numbers are base 2 with a binary mantissa. 32 bits are used to represent 1 sign bit, 8 exponent bits, and 23 mantissa bits.

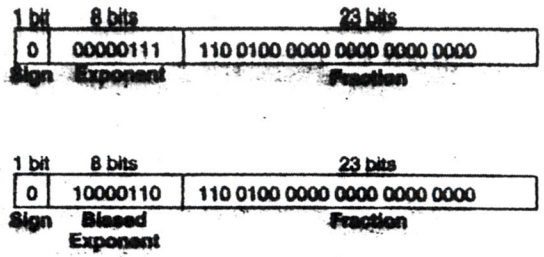
Example 5.5 32-BIT FLOATING-POINT NUMBERS

Show the floating-point representation of the decimal number 228.

Solution: First convert the decimal number into binary: $228_{10} = 11100100_2 = 1.11001_2 \times 2^7$. Figure 5.26 shows the 32-bit encoding, which will be modified later for efficiency. The sign bit is positive (0), the 8 exponent bits give the value 7, and the remaining 23 bits are the mantissa.



In binary floating-point, the first bit of the mantissa (to the left of the binary point) is always 1 and therefore need not be stored. It is called the *implicit leading one*. Figure 5.27 shows the modified floating-point representation of $228_{10} = 11100100_2 \times 2^0 = 1.11001_2 \times 2^7$. The implicit leading one is not included in the 23-bit mantissa for efficiency. Only the fraction bits are stored. This frees up an extra bit for useful data.



We make one final modification to the exponent field. The exponent needs to represent both positive and negative exponents. To do so, floating-point uses a *biased* exponent, which is the original exponent plus a constant bias. 32-bit floating-point uses a bias of 127. For example, for the exponent 7, the biased exponent is $7 + 127 = 134 = 10000110_2$. For the exponent -4, the biased exponent is: $-4 + 127 = 123 = 01111011_2$. Figure 5.28 shows $1.11001_2 \times 2^7$ represented in floating-point notation with an implicit leading one and a biased exponent of 134 (7 + 127). This notation conforms to the IEEE 754 floating-point standard.

Special Cases: 0, $\pm\infty$, and NaN

The IEEE floating-point standard has special cases to represent numbers such as zero, infinity, and illegal results. For example, representing the number zero is problematic in floating-point notation because of the implicit leading one. Special codes with exponents of all 0's or all 1's are reserved for these special cases. Table 5.2 shows the floating-point representations of 0, $\pm\infty$, and NaN. As with sign/magnitude numbers, floating-point has both positive and negative 0. NaN is used for numbers that don't exist, such as $\sqrt{-1}$ or $\log_2(-5)$.

Single- and Double-Precision Formats

So far, we have examined 32-bit floating-point numbers. This format is also called *single-precision*, *single*, or *float*. The IEEE 754 standard also

Table 5.2 IEEE 754 floating-point notations for 0, $\pm\infty$, and NaN

0	X	00000000	00000000000000000000000
∞	0	11111111	00000000000000000000000