CSAPP Notes

Leonardo

Number representations

Byte = 8 bits

- Decimal
- Binary
- Octal
- Hexadecimal

In C, numeric constants starting with “0x” or “0X” are interpreted as being in hexadecimal. The letters A to F is not Case-sensitive.

Hex digit	0	1	2	3	4	5	6	7
Decimal value	0	1	2	3	4	5	6	7
Binary value	0000	0001	0010	0011	0100	0101	0110	0111
Hex digit	8	9	A	B	C	D	E	F
Decimal value	8	9	10	11	12	13	14	15
Binary value	1000	1001	1010	1011	1100	1101	1110	1111

Figure 2.2 **Hexadecimal notation.** Each Hex digit encodes one of 16 values.

Unsigned and signed

signed

The most common computer representation of signed numbers is known as two’s-complement form. This is defined by interpreting the most significant bit of the word to have negative weight.

$-x = \sim x + 1$

When the sign bit is set to 1, the represented value is negative, and when set to 0 the value is nonnegative.

Consider the range of values that can be represented as a w-bit two’scomplement number. The least representable value is given by bit vector 10 … 0 (set the bit with negative weight, but clear all others), having integer value TMin(w bits) $= -2^{w-1}$. The greatest value is given by bit vector 01 … 1 (clear the bit with negative weight, but set all others as “1”), having integer value TMax(w bits) $= 2^{w-1} - 1$.

So you can see:

- The two’s-complement range is asymmetric: |TMin| = |TMax| + 1, that is, there is no positive counterpart to TMin.
- The maximum unsigned value is just over twice the maximum two’s-complement value: UMax = 2TMax + 1.
- –1 has the same bit representation as UMax—a string of all ones.

$$T2U_w(x) = \begin{cases} x + 2^w, & x < 0 \\ x, & x \geq 0 \end{cases}$$

$$U2T_w(u) = \begin{cases} u, & u < 2^{w-1} \\ u - 2^w, & u \geq 2^{w-1} \end{cases}$$

When do calculation or comparison between unsigned and signed values, signed values implicitly cast to unsigned.

To convert an unsigned number to a larger data type, we can simply add leading zeros to the representation; this operation is known as **zero extension**. For converting a two's- complement number to a larger data type, the rule is to perform a **sign extension**, adding copies of the most significant bit to the representation.

Floating-point

The IEEE floating-point standard represents a number in a form $V = (-1)^s \times M \times 2^E$:

- The *sign* s determines whether the number is negative ($s = 1$) or positive ($s = 0$), where the interpretation of the sign bit for numeric value 0 is handled as a special case.
- The *significand* M is a fractional binary number that ranges either between 1 and $2 - \epsilon$ or between 0 and $1 - \epsilon$.
- The *exponent* E weights the value by a (possibly negative) power of 2.

The bit representation of a floating-point number is divided into three fields to encode these values:

- The single sign bit `s` directly encodes the sign s.
- The k-bit exponent field $\texttt{exp} = e_{k-1} \cdots e_1 e_0$ encodes the exponent E.
- The n-bit fraction field $\texttt{frac} = f_{n-1} \cdots f_1 f_0$ encodes the significand M, but the value encoded also depends on whether or not the exponent field equals 0.

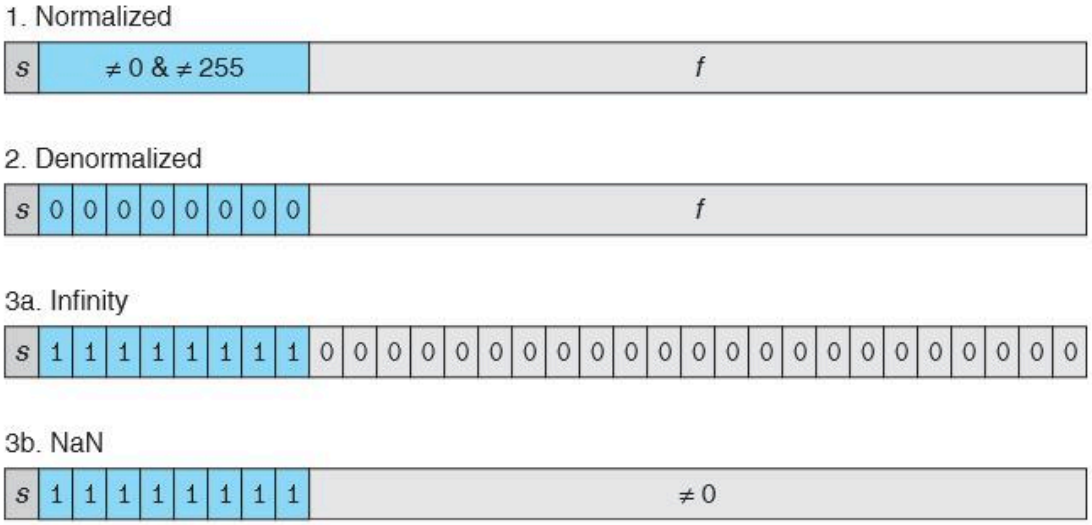

Figure 2.32 **Categories of single-precision, floating-point values.** The value of the exponent determines whether the number is (1) normalized, (2) denormalized, or a (3) special value.

		Exponent			Fraction		Value		
Description	Bit representation	e	E	2^E	f	M	$2^E \times M$	V	Decimal
Zero	0 0000 000	0	−6	$\frac{1}{64}$	$\frac{0}{8}$	$\frac{0}{8}$	$\frac{0}{512}$	0	0.0
Smallest pos.	0 0000 001	0	−6	$\frac{1}{64}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{512}$	$\frac{1}{512}$	0.001953
	0 0000 010	0	−6	$\frac{1}{64}$	$\frac{2}{8}$	$\frac{2}{8}$	$\frac{2}{512}$	$\frac{1}{256}$	0.003906
	0 0000 011	0	−6	$\frac{1}{64}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{3}{512}$	$\frac{3}{512}$	0.005859
	⋮								
Largest denorm.	0 0000 111	0	−6	$\frac{1}{64}$	$\frac{7}{8}$	$\frac{7}{8}$	$\frac{7}{512}$	$\frac{7}{512}$	0.013672
Smallest norm.	0 0001 000	1	−6	$\frac{1}{64}$	$\frac{0}{8}$	$\frac{8}{8}$	$\frac{8}{512}$	$\frac{1}{64}$	0.015625
	0 0001 001	1	−6	$\frac{1}{64}$	$\frac{1}{8}$	$\frac{9}{8}$	$\frac{9}{512}$	$\frac{9}{512}$	0.017578
	⋮								
	0 0110 110	6	−1	$\frac{1}{2}$	$\frac{6}{8}$	$\frac{14}{8}$	$\frac{14}{16}$	$\frac{7}{8}$	0.875
	0 0110 111	6	−1	$\frac{1}{2}$	$\frac{7}{8}$	$\frac{15}{8}$	$\frac{15}{16}$	$\frac{15}{16}$	0.9375
One	0 0111 000	7	0	1	$\frac{0}{8}$	$\frac{8}{8}$	$\frac{8}{8}$	1	1.0
	0 0111 001	7	0	1	$\frac{1}{8}$	$\frac{9}{8}$	$\frac{9}{8}$	$\frac{9}{8}$	1.125
	0 0111 010	7	0	1	$\frac{2}{8}$	$\frac{10}{8}$	$\frac{10}{8}$	$\frac{5}{4}$	1.25
	⋮								
	0 1110 110	14	7	128	$\frac{6}{8}$	$\frac{14}{8}$	$\frac{1792}{8}$	224	224.0
Largest norm.	0 1110 111	14	7	128	$\frac{7}{8}$	$\frac{15}{8}$	$\frac{1920}{8}$	240	240.0
Infinity	0 1111 000	—	—	—	—	—	—	∞	—

Figure 2.34 **Example nonnegative values for 8-bit floating-point format.** There are $k = 4$ exponent bits and $n = 3$ fraction bits. The bias is 7.

Shift operations

- Left shift
- Logical right shift: zero-extend
- Arithmetic right shift: sign-extend

Different effects for signed (2's complement) and unsigned numbers when simply throwing away the MSB in the overflow results

Array

- Multi-Dimensional Arrays
  - Row-major ordering in C
    - Each row is allocated contiguously, and all rows are allocated contiguously
- Multi-Level Arrays
  - The second-level arrays are not necessarily contiguous in memory

Satisfy alignment in structures with padding

- Within structure: each field satisfies its own alignment requirement
- Overall structure: align to the largest alignment requirement of all fields

Big and little endian

Array not affected by byte ordering. Big Endian: Least significant byte has highest address. Little Endian: Least significant byte has lowest address.