CS2100
Computer Organization

AY2022/23 Semester 1

Notes by Jonathan Tay

Last updated on August 9, 2022

Contents

---

1 Number Systems

Any base-R number system is **weighted-positional**.

The **decimal** number system has a **base** (or **radix**) of 10.

Every digit going leftward from the decimal point has a weighted power of 10, starting from 10^0 and increasing.

Every digit going rightward from the decimal point has a weighted power of 10, starting from 10^{-1} and decreasing.

This generalizes to any base-R number system, with R^i in place of 10^i.

1.1 Conversions between Bases

We use the decimal system as an intermediary to convert between bases.

For example, to convert between base-2 (**binary**) to base-7, we first convert to base-10, then convert to base-7.

We can use grouping and/or ungrouping to convert between bases which are powers of 2.

For example, to convert between base-2 and base-16 (**hexadecimal**), we partition the digits of the binary number in groups of 4, starting from the binary point, and partitioning outward.

1.1.1 Base-R ⇒ Decimal

The decimal equivalent of a base-R number is the sum of its weighted digits.

1.1.2 Decimal ⇒ Base-R

This conversion performs **division-by-R** for the whole number portion, and **multiplication-by-R** for the fractional portion.

Division-by-R divides the whole number by R until the quotient is 0. The result is produced from the remainders from **right to left**.

Multiplication-by-R multiplies the fractional portion by R until the product is 0, or until the desired number of decimal places. The result is produced from the **carries left to right**.

1.2 Negative Numbers

Signed numbers include all positive and negative values, unlike **unsigned numbers** which only include positive values.

There are 4 representations for signed binary numbers covered in this module.

1.2.1 Sign-and-Magnitude

Negate a number by inverting the sign bit.

This representation uses the leftmost (most-significant bit) as the **sign bit** — 1 for negative, 0 for positive.

This results in **duplicate zeros** (positive and negative), and a range of values of $-\left(2^{n-1}-1\right)$ to $2^{n-1}-1$.

1.2.2 1s Complement

Negate a number by inverting each bit.

An n-bit number x has a negated value $-x = 2^n - x - 1$.

This also results in **duplicate zeros** (positive and negative), and a range of values of $-\left(2^{n-1}-1\right)$ to $2^{n-1}-1$.

The weight of the leftmost bit is $-\left(2^{n-1}-1\right)$.

1.2.3 2s Complement

Negate a number by inverting each bit, then adding 1.

Alternatively, preserve each bit up to and the rightmost 1, then inverting every bit to the left of the rightmost 1.

An n-bit number x has a negated value $-x = 2^n - x$.

This also results in **no duplicate zeros**, and a range of values of -2^{n-1} to $2^{n-1}-1$.

The weight of the leftmost bit is -2^{n-1}.

1.2.4 Excess Representation

Distribute positive and negative values by a simple addition or subtraction.

Excess-k subtracts k from the decimal value of the number, while preserving its base-2 representation.

To convert from a decimal value to its excess-k representation, we add k to the decimal value and convert that to base-2.

Typically, for an n-bit number, $k = 2^n - 1$.

1.3 Real Numbers

Fixed-point representation allocates a fixed number of bits for the whole number and fractional parts.

This results in a limited range of numbers that can be represented.

The IEEE 754 **floating point representation** resolves this by allocating a fixed number of bits to the **sign**, **exponent**, and **mantissa**.

precision	bits	sign	exponent	mantissa	bias
single	32	1	8	23	127
double	64	1	11	52	1023

The **sign bit** is 0 for positive numbers, and 1 for negative numbers.

The **mantissa** is **normalized** with an implicit leading 1 bit, e.g. 110.1_2 is normalized to $1.101_2 \times 2^2$, and only 101 is stored in the mantissa.