Notes for IT5002

Number System

Conversion

Base-R to Decimal Conversion

$$1101.101_2 = 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-3}$$

$$2A.8_{16} = 2 \times 16^1 + 10 \times 16^0 + 8 \times 16^{-1}$$

Decimal to Base-R Conversion

Repeated Division-by-R Method(For whole numbers)

$$43_{10} = 101011_2$$

$number$		$reminder$
43	$\%2$	
21	$\%2$	$1(LSB)$
10	$\%2$	1
5	$\%2$	0
2	$\%2$	1
1	$\%2$	0
0	$\%2$	$1(MSB)$

Repeated Multiplication-by-R Method(For fractions)

$$0.3125_{10} = .0101_2$$

$number$		$reminder$
0.3125	$\times 2$	
0.625	$\times 2$	$0(MSB)$
1.25		1
0.25	$\times 2$	
0.5	$\times 2$	0
1		$1(LSB)$
0		

Hex Oct Bin

Bin to Hex: **grouping** of binary digits into sets of 4, **Conversion** each group of 4 bits to corresponding hexadecimals for example 1010 to A, **Concatenation** every hex digits.

ASCII Code

American Standard Code for Information Interchange(ASCII), 7 bits, 1 parity bit(for odd or even parity)

mind that digits(0 - 9) < capital letter (A - Z) < lowercase letter (a - z)

Negative Number

- Unsigned Numbers: Only non-negative values
- Signed Numbers: Include all values

For **Signed Binary Number**, there are 3 common representations:

- Sign-and-Magnitude
- 1's Complement
- 2's Complement

Sign-and-Magnitude

the sign is represented by **sign bit**: 0 for +, 1 for -

Example: 1-bit sign and 7-bit magnitude

$$+52_{10} = +110100_2 = 00110100$$
$$-19_{10} = -10011_2 = 10010011$$

For n-bit Sign-and-Magnitude number:

- Largest Value: $2^{(n-1)} - 1$ (n-1 ones)
- Smallest Value: $-2^{n-1} + 1$
- Zeros: +0 and -0

Complement

REMIND: **Positive** number's complement is itself.

A **Negative** number's complement is calculated as:

1's complement

$$-x = 2^n - 1 - x$$

For example: -12 1's complement (8 bit)

$$\begin{aligned} -12 =& 2^8 - 1 - 12 \\ =& 243 \\ =& 1111\ 0011_{1's} \end{aligned}$$

TECHNIQUE: invert all bits.

Largest Value: $2^{n-1} - 1 = 0111\ 1111 = 127$

Smallest Value: $-2^{n-1} - 1 = 1000\ 0000 = -127$

Zeros: $+0 = 0000\ 0000; \quad -0 = 1111\ 1111$

MSB can still represent sign.

2's complement

$$-x = 2^n - x$$

For example: -12 2's complement (8 bit)

$$\begin{aligned}-12_{10} =& 2^8 - 12 \\ =& 244 \\ =& 11110100_{2's}\end{aligned}$$

TECHNIQUE: Convert to binary(0000 1100), **invert** all bits(1111 0011), **add 1**(1111 0100)

Largest Value: $2^{n-1} - 1|_{n=8} = 0111\ 1111 = 127$

Smallest Value: $-2^{n-1}|_{n=8} = 1000\ 0000 = 128$

Zero: $0 = 0000\ 0000$

MSB can still represent sign

Fraction

The main idea of complement is the same. For negative number, 1's complement to invert all bits, 2's complement to invert and add 1. For positive number, complements are itself.

Base-R complement

For a number in Base-R, there are two kinds of complement: **r's complement** and **r-1's complement**. Positive numbers are themselves. **Negative** number with **n-bit integer** part and **m-bit fraction** part, the complement is computed as:

- r's complement

$$-x = r^n - x$$

To obtain the essence, the r's complement of a negative n-bit number is the number that, when added to its corresponding positive counterpart, yields r^n(0000 0000).(MSB 1 is overflowed)

- r-1's complement

$$-x = r^n - r^m - x$$

Similarly, r-1's complement of a negative n-bit integer and m-bit fraction add its positive counterpart yield $r^n - r^m$(which is all ones, 1111.1111).

Addition and Subtraction

Signed numbers are of a **fixed** range, so if go beyond this range, **overflow** occurs.

Check Overflow:

- positive add positive -> negative
- negative add negative -> positive

For 2's complement, if there is a carry out of MSB, ignore.

For 1's complement, if there is a carry out of MSB, **add 1** to the result

$$\begin{aligned}-2 + -5 =& 1101 + 1010 \\ =& 1\ 0111(\text{carry out}) + 1 \\ =& 1000 = -7\end{aligned}$$

Subtraction is add the second operand's complement.

Excess Notation

Excess Notation swifts the binary number.

For example, 000 in BIN is corresponding to 0 in DEC. In Excess-4 representation, 000 in BIN is corresponding to -4 in DEC, other number are swift like this.

Excess-4	DEC value
000	-4
001	-3
010	-2
011	-1
100	0
101	1
110	2
111	3

Real Number

Fixed-Point Representation

Range is limited.

Floating Point Representation

3 component: sign, exponent, mantissa(fraction)

sign	exponent	mantissa

The base (radix) is assumed to be 2.

Two formats (IEEE 754):

- Single-precision (32 bits): 1-bit sign, 8-bit, exponent with bias 127 (excess-127), 23-bit mantissa
- Double-precision (64 bits): 1-bit sign, 11-bit, exponent with bias 1023 (excess-1023), 52-bit mantissa

IMPORTANT

exponent	fraction $= 0$	fraction $\neq 0$	Equation
$00_{\mathrm{H}} = 0000\ 0000_2$	± 0	denomalize number	$(-1)^{\text{sign}} \times 2^{-126} \times 0.\text{fraction}$
$01_{\mathrm{H}}, \ldots, \mathrm{FE}_{\mathrm{H}}$	normal	normal	$(-1)^{\text{sign}} \times 2^{\text{exponent}-127} \times 1.\text{fraction}$
FF_{H}	$\pm\infty$	NaN	