Number System and Data Representation

Byte	**8 bits**
Nibble	4 bits
Words	Multiple(1,2,4 bytes etc.)

- N bits represent up to 2^N values
- $\lceil log_2 M \rceil$ bits for M values

Conversion

Whole Number	**Fraction Number**

2	43	
2	21 rem 1	← LSB
2	10 rem 1	
2	5 rem 0	
2	2 rem 1	
2	1 rem 0	
	0 rem 1	← MSB

	Carry	
0.3125×2=0.625	0	←MSB
0.625×2=1.25	1	
0.25×2=0.50	0	
0.5×2=1.00	1	←LSB

Generally, Decimal to base-*R*

- Whole numbers: repeated division-by-*R*
- Fractions: repeated multiplication-by-*R*

ASCII: American Standard Code for Information Interchange

- 7 bits, plus 1 parity bit (odd or even parity)
- Integers (0 to 127) and characters are interchangeable

01000110	As an 'int', it is 70
	As a 'char', it is 'F'

```
int num = 65;
char ch = 'F';
printf("num (in %%d) = %d\n", num); // 65
printf("num (in %%c) = %c\n", num); // A
printf("ch (in %%c) = %c\n", ch);   // F
printf("ch (in %%d) = %d\n", ch);   // 70
```

Unsigned numbers: only non-negative values

Signed numbers: include all values (positive and negative)

Negative Numbers

(1) Sign-and-Magnitude: The sign is represented by a 'sign bit', 0 for + and 1 for -

sign	Magnitude

Largest value: 01111111 = $+127_{10}$

Smallest value: 10000000 = -127_{10}

Zeros: 00000000 = $+0_{10}$

11111111 = -0_{10}

Range (for 8 bits): -127_{10} to $+127_{10}$

Range (for *n* bits): $-(2^{n-1} – 1)$ to $2^{n-1} – 1$

Negate: just invert the sign bit.

Disadvantage:

- One of the bit patterns is wasted.
- Addition doesn't work the way we want it to.

(2) 1s Complement: Given a number ***x*** which can be expressed as an ***n*-bit** binary number, its negated value can be obtained in 1s-complement representation using:

$$-x = 2^n – x – 1$$

Largest value: 01111111 = $+127_{10}$

Smallest value: 10000000 = -127_{10}

Zeros: 00000000 = $+0_{10}$

11111111 = -0_{10}

Range (for 8 bits): -127_{10} to $+127_{10}$

Range (for *n* bits): $-(2^{n-1} – 1)$ to $2^{n-1} – 1$

Negate: invert all the bits.

MSB: -2^{n-1}

Addition:

1. Perform binary addition on the two numbers.
2. If there is a carry out of the MSB, add 1 to the result.
3. Check for overflow. Overflow occurs if result is opposite sign of A and B.
   [Whether the result have the same sign with A and B]

(3) 2s Complement: Given a number *x* which can be expressed as an *n*-bit binary number, its negated value can be obtained in 2s-complement representation using:

$$-x = 2^n – x = \text{1s complement} + 1$$

Largest value: 01111111 = $+127_{10}$

Smallest value: 10000000 = -128_{10}

Zero: 00000000 = $+0_{10}$

Range (for 8 bits): -128_{10} to $+127_{10}$

Range (for *n* bits): -2^{n-1} to $2^{n-1} – 1$

Negate: invert all the bits, then add 1

MSB: -2^{n-1}

Sign Extension: copy the MSB (most significant bit) of the n-bit number $m - n$ times to the **left** of the n-bit number to create the m-bit number.

[T] sign extension is value-preserving.

Addition:

1. Perform binary addition on the two numbers.
2. Ignore the carry out of the MSB.
3. Check for overflow. Overflow occurs if the 'carry in' and 'carry out' of the MSB are different, or if result is opposite sign of A and B.
   [If the result of addition/subtraction goes beyond this range, an overflow occurs.]

(4) (r-1)'s Complement:

$$(r – 1)\text{'s complement of } N = r^n – r^{-m} – N$$

where

- *n* is the number of integer digits
- *m* the number of fractional digits. (If there are no fractional digits, then *m* = 0 and the formula becomes $r^n – 1 – N$ as given in class.)

Excess Representation: It allows the range of values to be distributed evenly between the positive and negative values, by a simple translation (addition/subtraction).

Range: for N-bit excess-X

- Find the maximum binary number of N bits (111…11), which and convert to decimal, **A**.
- Minimum Number is 0-X = **-X**
- Then maximum positive number is **A-X**
- Therefore, N-bits system ranging from -X to A-X, represent range from 0 to A

[E] Convert excess-128 to decimal (10010110**)**

- Step 1: Begin with the binary number 10010110.
- Step 2: Convert it to decimal: 10010110=150.
- Step 3: **Subtract the bias** (128) from the decimal representation: 150−128=22.
- The original decimal number before applying excess-128 encoding was 22_{10}.

[E] Convert decimal number 25 to 8-bit excess-128 form.

- Step 1: Start with the decimal number 25.
- Step 2: **Add the bias** (128) to the number: 25+128=153.
- Step 3: **Convert 153 to binary:** 153=10011001.
- The excess-128 binary representation of 25 is 10011001Excess-128.

IEEE 754 Floating-Point Rep

The base (radix) is assumed to be 2.

Two formats (We will focus on the single-precision format):

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

sign	exponent	mantissa
1	8	23

- Mantissa is normalised with an implicit leading bit 1

=> Can **NOT** represent 0

Double-precision (64 bits): 1-bit sign, 11-bit exponent with bias 1023 (excess-1023), and 52-bit mantissa

Appendix

1. [T] sign extension is value-preserving.

Let X be the *n*-bit signed integer and Y be the *m*-bit signed integer which is the sign-extended version of X.

If the MSB of X is zero, this is straightforward, since padding more 0's to the left adds nothing to the final value. If the MSB of X is one, then it is trickier to prove. In the original *n-bit* representation, the MSB has a weight of -2^{n-1} giving us

$$X = -2^{n-1} + b_{n-2} \cdot 2^{n-2} + b_{n-3} \cdot 2^{n-3} + \cdots + b_0.$$

Let $Z = b_{n-2} \cdot 2^{n-2} + b_{n-3} \cdot 2^{n-3} + \cdots + b_0$, then $X = -2^{n-1} + Z$.

In the new *m*-bit representation Y where *m* > *n*, the MSB of Y has a weight of -2^{m-1}, and since we copy the MSB (i.e. the leftmost bit) of X a total of *m* – *n* times, we get

$$Y = -2^{m-1} + 2^{m-2} + 2^{m-3} + \cdots + 2^n + 2^{n-1} + Z.$$

For $Y = X$, it suffices to show that $-2^{m-1} + 2^{m-2} + 2^{m-3} + \cdots + 2^n + 2^{n-1} = -2^{n-1}$.

For $Y = X$, it suffices to show that $-2^{m-1} + 2^{m-2} + 2^{m-3} + \cdots + 2^n + 2^{n-1} = -2^{n-1}$.

Recall that the sum of a Geometric Progression with N terms, initial value a and ratio r is given by: $\frac{a(r^N-1)}{r-1}$. We will use this formula to calculate $2^{m-2} + 2^{m-3} + \cdots + 2 + 2^{n-1}$, which has $N = (m-2) - (n-1) + 1 = m - n; a = 2^{n-1}$ and $r = 2$.

$$\begin{aligned}
&-2^{m-1} + (2^{m-2} + 2^{m-3} + \cdots + 2^n + 2^{n-1}) \\
&= -2^{m-1} + \frac{a(r^N - 1)}{r - 1} \\
&= -2^{m-1} + 2^{n-1}(2^{m-n} - 1) \\
&= -2^{m-1} + 2^{m-1} - 2^{n-1} \\
&= -2^{n-1}
\end{aligned}$$

Therefore, $Y = X$.

2. **Nice Values**

ASCII values:

Char	Dec	Hex	Bin
'0'	48	0x30	0b00110000
'A'	65	0x41	0b01000001
'a'	97	0x61	0b01100001

- **Nice numbers:**

$2^{15} - 1 = 32\,767$

$2^{16} - 1 = 65\,535$

$2^{31} - 1 = 2\,147\,483\,647$

$2^{32} - 1 = 4\,294\,967\,295$

3. **Power of 2 Table**

Exp	Val	Exp	Val	Exp	Val	Exp	Val
2^0	1	2^8	256	2^{16}	65,536	2^{24}	16,777,216
2^1	2	2^9	512	2^{17}	131,072	2^{25}	33,554,432
2^2	4	2^{10}	1,024	2^{18}	262,144	2^{26}	67,108,864
2^3	8	2^{11}	2,048	2^{19}	524,288	2^{27}	134,217,728
2^4	16	2^{12}	4,096	2^{20}	1,048,576	2^{28}	268,435,456
2^5	32	2^{13}	8,192	2^{21}	2,097,152	2^{29}	536,870,912
2^6	64	2^{14}	16,384	2^{22}	4,194,304	2^{30}	1,073,741,824
2^7	128	2^{15}	32,768	2^{23}	8,388,608	2^{31}	2,147,483,648

Exp	Val	Exp	Val
2^{-1}	0.5	2^{-9}	0.001953125
2^{-2}	0.25	2^{-10}	0.0009765625
2^{-3}	0.125	2^{-11}	0.00048828125
2^{-4}	0.0625	2^{-12}	0.000244140625
2^{-5}	0.03125	2^{-13}	0.0001220703125
2^{-6}	0.015625	2^{-14}	0.00006103515625
2^{-7}	0.0078125	2^{-15}	0.000030517578125
2^{-8}	0.00390625	2^{-16}	0.0000152587890625

Radix Complement

Number of Digits	n
Radix	b
(b-1)s complement	-x = b^n-x-1
(b)s complement	-x = b^n-x

Example: Consider the number $(43)_{10}$. We can express this in 5-trit[4] number as $(01121)_3$. The negations are:

- ***(b-1)s Complement***: 21101

  -43 = 3^5-43-1 = 199

  $(199)_{10} = (21101)_3$

- ***(b)s Complement***: 21102

  -43 = 3^5-43 = 200

  $(200)_{10} = (21102)_3$