Number Systems

A number of the form:

$(IJKL)_B$

can be evaluated in the following way:

$$L \times B^0 + K \times B^1 + J \times B^2 + I \times B^3$$

BASE0010

Base Conversion

Decimal	*Binary*	*Octal*	*Hexadecimal*
0	00000	00	0x00
1	00001	01	0x01
2	00010	02	0x02
3	00011	03	0x03
4	00100	04	0x04
5	00101	05	0x05
6	00110	06	0x06
7	00111	07	0x07
8	01000	10	0x08
9	01001	11	0x09
10	01010	12	0x0A
11	01011	13	0x0B
12	01100	14	0x0C
13	01101	15	0x0D
14	01110	16	0x0E
15	01111	17	0x0F
16	10000	20	0x10

BASE0020

Binary to Decimal Conversion

EXample – Convert $(1101)_2$ to Decimal

$$1101 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$
$$= 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1$$
$$= 8 + 4 + 0 + 1$$
$$= (13)_{10}$$

BASE0030

University of Idaho

Hexadecimal to Decimal Conversion

EXample – Convert $(E42)_{16}$ to Decimal

$$E42 = E \times 16^2 + 4 \times 16^1 + 2 \times 16^0$$
$$= 14 \times 256 + 4 \times 16 + 2 \times 1$$
$$= 3584 + 64 + 2$$
$$= 3650$$

BASE0040

University of Idaho

Decimal to Binary Conversion

EXample – Convert $(25)_{10}$ to Binary

	Quotient		Remainder
$\frac{25}{2}$ =	12	+	1
$\frac{12}{2}$ =	6	+	0
$\frac{6}{2}$ =	3	+	0
$\frac{3}{2}$ =	1	+	1
$\frac{1}{2}$ =	0	+	1

1 1 0 0 1

BASE0050

University of Idaho

Decimal to "Hex" Conversion

EXample – Convert $(79)_{10}$ to Hexadecimal

	Quotient		Remainder
$\frac{79}{16}$ =	4	+	15 ⇒ F
$\frac{4}{16}$ =	0	+	4

$(4\ F)_{16}$

BASE0060

University of Idaho

Binary to "Hex" Conversion

Example – Convert $(10101110)_2$ to Hex

$$\underbrace{1010}\ \underbrace{1110}$$
$$(\ 10_{10}\quad 14_{10})$$
$$\Downarrow\quad\Downarrow$$
$$(\ A\quad E\ )_{16}$$

"Hex" to Binary Conversion

Example – Convert $(3B7)_{16}$ to Binary

$$\overbrace{3}\quad \overbrace{B}\quad \overbrace{7}$$
$$(0011\ 1011\ 0111)_2$$

BASE0070

University of Idaho

Fractional Conversion

Example – Convert 0.10111 binary to decimal

$$\begin{aligned}0.10111 &= 1 \times 2^{-1} = 0.5\\ &= 0 \times 2^{-2} = 0.0\\ &= 1 \times 2^{-3} = 0.125\\ &= 1 \times 2^{-4} = 0.0625\\ &= 1 \times 2^{-5} = \underline{0.03125}\\ & \qquad\qquad\quad 0.71875 \text{ decimal}\end{aligned}$$

BASE0080

University of Idaho

Fractional Conversion

Example – Convert 0.4E1 Hex to decimal

$$0.4E1 = 4 \times 16^{-1} = 0.25$$
$$E \times 16^{-2} = 14 \times 16^{-2} = 0.0546875$$
$$1 \times 16^{-3} \approx 0.0002441$$
$$\approx 0.3049316 \text{ decimal}$$

Fractional Conversion

Example – Convert 0.642 Decimal to Binary

0.642 x 2 =	1.284	1
0.284 x 2 =	0.568	0
0.568 x 2 =	1.136	1
0.136 x 2 =	0.272	0
0.272 x 2 =	0.544	0
0.544 x 2 =	1.088	1
		⋮

Binary Arithmetic Examples

Addition

```
     1 1 1 1
   00101110         01000111          10110110
 + 00011101       + 01101101        + 11011100
   --------         --------          ---------
   01001011         10110100          110010010
                                      Note: Requires 9 bits
```

Subtraction

```
    0    0             0 10 1
   01̸¹01 1̸¹011        101̸1̸¹0̸¹010
 - 00110101         - 00011100
   --------           --------
   00100110           10010110
```


Hex Arithmetic Examples

Addition

```
       1            1
      3B4          1AF
   +  6A5       +  345
      ---          ---
      A59          4F4
```

Subtraction

```
         A          4  1C
      3  B̸ ¹4       5̸  D̸ ¹1
   -  1  9  5    -  1  D  8
      -------       -------
      2  1  F       2  F  9
```

Sign–Magnitude Notation

We represent negative numbers by using a "sign–plus–magnitude" notation. Binary values can also be represented this way, by appending a 0 for positive, 1 for negative.

+57 ⇒ | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |

-57 ⇒ | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |

We use sign–magnitude as our "normal" system
But sign–magnitude is clumsy for the computer!

Rules for Sign–Magnitude Arithmetic

1. *If both numbers are positive, then add them, and prepend a 0 (for positive).*

2. *If both numbers are negative, then add the absolute values and then prepend a 1 (for negative)*

3. *If one number is positive and the other negative, then determine which has the larger absolute value. Subtract the larger absolute value from the smaller one. Then if the larger number was positive, prepend a 0, otherwise prepend a 1 (for negative).*

WHEW!

University of Idaho

One's Complement Representation

A negative binary number can be represented by taking the positive representation of the number and changing all the 0's to 1's and all the 1's to 0's.

Example 1 – Express -5_{10} as an 8-bit 1's complement number

-5_{10} = -(00000101) ⇒ 11111010

Example 2 – Express -13_{10} as an 8-bit 1's complement number

-13_{10} = -(00001101) ⇒ 11110010

Example 3 – Express $+14_{10}$ as an 8-bit 1's complement number

$+14_{10}$ = +(00001110) ⇒ 00001110

Example 4 – Express -0_{10} as an 8-bit 1's complement number

-0_{10} = -(00000000) ⇒ 11111111 !

BASE0140

University of Idaho

One's Complement Values

What values do the following bit patterns represent in the one's complement system?

Example 1 – 00101101

00101101 ⇒ positive number, or +45

Example 2 – 11101101

11101101 ⇒ negative number, so –(00010010), or –18

Example 3 – 11111110

11111110 ⇒ negative number, so –(00000001), or –1

BASE0150

University of Idaho

Finding the One's Complement

To "find the one's complement" of a number means to find the negative of a number:

Example 1 – Find the one's complement of 00101101

−(00101101) = 11010010

Example 2 – Find the one's complement of 11101101

−(11101101) = 00010010

BASE0160

Two's Complement Representation

A negative binary number can be represented by taking the one's complement of the number and then adding 1. This is called the two's complement representation.

Example 1 – Express -5_{10} as an 8-bit 1's complement number

$$
\begin{array}{rcrcr}
-5_{10} & = & -(00000101) & = & 11111010 \\
 & & & & +1 \\
\hline
 & & & & 11111011
\end{array}
$$

Example 2 – Express -13_{10} as an 8-bit 1's complement number

$$
\begin{array}{rcrcr}
-13_{10} & = & -(00001101) & = & 11110010 \\
 & & & & +1 \\
\hline
 & & & & 11110011
\end{array}
$$

⇒

Example 3 – Express -0_{10} as an 8-bit 1's complement number

$$
\begin{array}{rcrcr}
-0_{10} & = & -(00000000) & = & 11111111 \\
 & & & & +1 \\
\hline
 & & & & 00000000
\end{array}
$$

BASE0200

Two's Complement Values

What values do the following bit patterns represent in the two's complement system?

Example 1 – 00101101

00101101 ⇒ positive number, or +45

Example 2 – 11101101

11101101 ⇒ negative number, so –(00010011), or –19

Example 3 – 11111111

11111111 ⇒ negative number, so –(00000001), or –1

BASE0210

University of Idaho

Finding the Two's Complement

To "find the twos complement" of a number means to find the negative of a number:

Example 1 – Find the twos complement of 00101101

–(00101101) = 11010010 + 1 = 11010011

Example 2 – Find the two's complement of 11101101

–(11101101) = 00010010 + 1 = 00010011

BASE0220

University of Idaho

Two's Complement Arithmetic

To add:

1. *Add the two numbers, including sign bits. Ignore any carry out of the sign bit.*

To subtract:

1. *Take the 2's complement of the second number and then add.*

BASE0230

Two's Complement Examples

Perform the following 2's complement arithmetic, and show the decimal equivalents of each problem

```
    0010 1101 ⇒     ( +45)
  + 0001 1011 ⇒ +   ( +27)
    ---------       ------
    0100 1000 ⇒     ( +72)

    0110 1101 ⇒                      (+109)
  + 1111 0010 ⇒ +(-00001110)   =   (  -14)
    ---------                      -------
    0101 1111 ⇒                      ( +95)
```

BASE0240

Two's Complement Examples

Perform the following 2's complement arithmetic, and show the decimal equivalents of each problem

```
    0010 1001 ⇒  ( +41)      0010 1001
  - 0001 0111 ⇒ -( +23)  ⇒  +1110 1001
                             0001 0010     (+18)

    1110 1010 ⇒  ( -22)      1110 1010
  - 1101 1000 ⇒ -( -40)  ⇒  +0010 1000
                           1  0001 0010     (+18)
```

Note: throw away the carry out

BASE0250

University of Idaho

Overflow

In a real computer, a finite number of bits are used in calculations.

Whenever an attempt is made to represent or compute a number that is too large to be represented in the number of bits available, overflow occurs.

This is analogous to trying to represent a number larger than 1000 in only three decimal digits.

Example - Represent the decimal value 343 as an unsigned 8-bit value

ANS - The unsigned representation requires 9 bits, 101010111, so cannot be represented as an 8 bit number.

BASE0260

University of Idaho

Overflow in 1's and 2's Complement

Since the MSB is reserved for the sign, one less bit is available for magnitudes

EXample – Represent the value +209 decimal as an 8-bit 2's complement number.

Ans – The binary representation for +209 is 1101 0001. However, this pattern represents a negative number, since the MSB or sign bit is a 1. This value cannot be represented as an 8 bit 2's complement number.

BASE0270

University of Idaho

Maximum/Minimum Values

For unsigned integers:

$0 - 2^n - 1$, where n is number of bits

For 8 bits: 0 – 255
For 16 bits: 0 – 65535 (0 – 64K)
For 32 bits: 0 – 4294967295 (0 – 4G)

For 2's complement (ints):

$-2^{n-1} - 2^{n-1} - 1$, where n is number of bits:

For 8 bits: -128 – +127
For 16 bits: -32768 – +32767 (±32k)
For 32 bits: -2147483648 – +2147483547 (±2G)

BASE0280

University of Idaho

Detecting Overflow

For unsigned integers:

Overflow occurs when the value doesn't fit with in the number of bits allotted the number.

For 1's/2's complement (two ways):

1. Determine the answer. Overflow if it is larger/smaller than limits.

2. Check carries into and out of sign bit – if neither or both occur, then OK. If only one occurs, overflow