





# Binary to Decimal Conversion

# EXample - Convert (1101)<sub>2</sub> 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)<sub>16</sub> to Decimal

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

BASE0040

University of Idaho



Decimal to Binary Conversion

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

Quotient Remainder

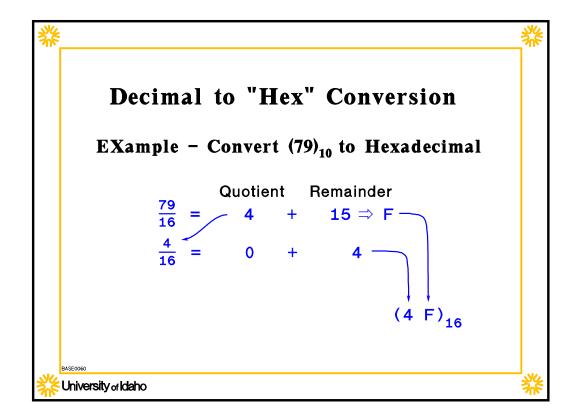
$$\frac{25}{2} = 12 + 1$$

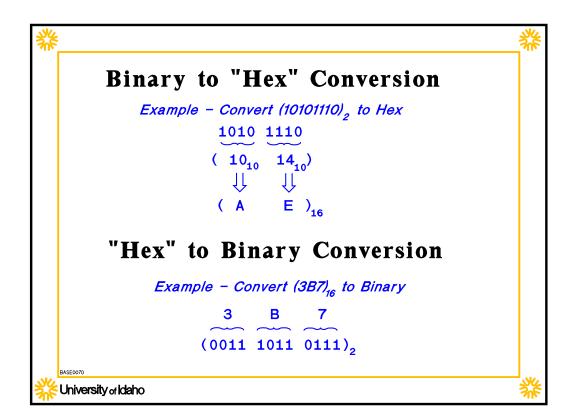
$$\frac{12}{2} = 6 + 0$$

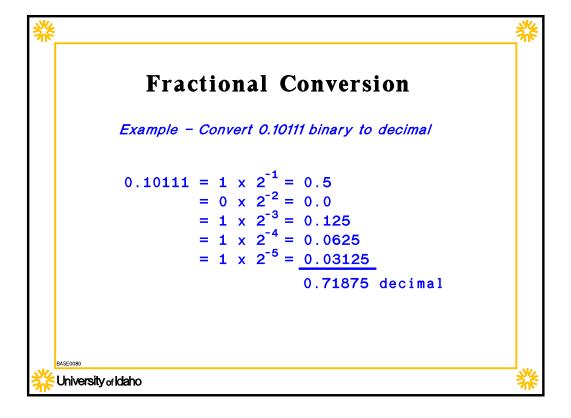
$$\frac{6}{2} = 3 + 0$$

$$\frac{3}{2} = 1 + 1$$

$$\frac{1}{2} = 0 + 1$$
Whiversity of Idaho











## Fractional Conversion

Example - Convert 0.4E1 Hex to decimal

$$0.4E1 = 4 \times 16^{-1} = 0.25$$

$$E \times 16^{-2} = 14 \times 16^{-2} = 0.0546875$$

$$1 \times 16^{-3} \approx 0.3049316 \text{ decimal}$$

BASE0090

University of Idaho



# Fractional Conversion

Example - Convert 0.642 Decimal to Binary

$$0.642 \times 2 = 1.284$$
 1  
 $0.284 \times 2 = 0.568$  0

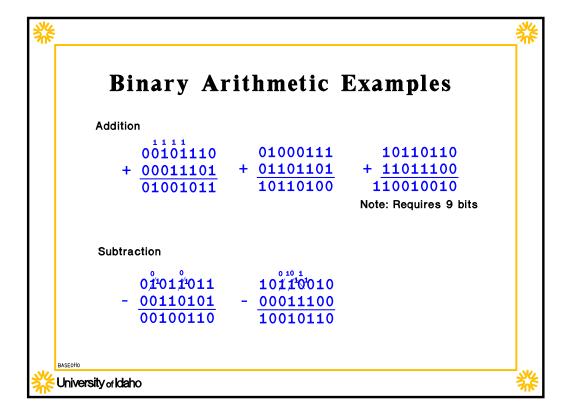
$$0.568 \times 2 = 1.136$$

$$0.136 \times 2 = 0.272$$

$$0.272 \times 2 = 0.544$$

University of Idaho





# 





# 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  $\implies$  0 0 1 1 1 0 0 1

-57  $\Rightarrow$  1 0 1 1 1 0 0 1

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

BASE0120

University of Idaho





# 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!

ASE0130









#### 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<sub>10</sub> as an 8-bit 1's complement number -5<sub>10</sub> = -(00000101)  $\implies$  11111010

Example 2 - Express -13<sub>10</sub> as an 8-bit 1's complement number -13<sub>10</sub> = -(00001101)  $\Rightarrow$  11110010

Example 3 - Express +14<sub>10</sub> as an 8-bit 1's complement number +14<sub>10</sub> = +(00001110)  $\Rightarrow$  00001110

Example 4 - Express  $-0_{10}$  as an 8-bit 1's complement number  $-0_{10} = -(00000000)$   $\Rightarrow$  11111111

BASE0140







#### One's Complement Values

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

Example 1 - 00101101

00101101  $\implies$  positive number, or +45

Example 2 - 11101101

11101101  $\Rightarrow$  negative number, so -(00010010), or -18

Example 3 - 11111110

11111110  $\implies$  negative number, so -(00000001), or -1







#### 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 as an 8-bit 1's complement number

$$-5_{10} = -(00000101) = 111111010 +1$$

$$\frac{+1}{11111011}$$

Example 2 - Express -13<sub>10</sub> as an 8-bit 1's complement number

$$-13_{10} = -(00001101) = 11110010$$

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

$$-0_{10} = -(00000000) = 111111111 \\ +1 \\ \hline 000000000$$

ASE0200









## Two's Complement Values

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

Example 1 - 00101101

00101101  $\implies$  positive number, or +45

Example 2 - 11101101

11101101  $\implies$  negative number, so -(00010011), or -19

Example 3 - 11111111

11111111  $\Rightarrow$  negative number, so -(00000001), or -1











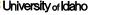
To "find the twos complement" of a number means to find the negative of a number:

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









# 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

$$+ \begin{array}{c} 0010 & 1101 \Longrightarrow & ( +45) \\ 0001 & 1011 \Longrightarrow + & ( +27) \\ \hline 0100 & 1000 \Longrightarrow & ( +72) \end{array}$$







# Two's Complement Examples

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

Note: throw away the carry out

BASE0250







#### 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.









# 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  $\pm 209$  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







#### 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)









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



