**CSCI 360 Number Systems**

**Decimal Number System (base 10)**

* Uses 10 digits: 0 – 9
* 125 = 1\*102 + 2\*101 + 5\*100

**Binary Number System (base 2)**

* Uses 2 digits: 0 and 1
* 110101 = 1\*25 + 1\*24 + 0\*23 + 1\*22 + 0\*21 + 1\*20

**Hexadecimal Number System (base 16)**

* Uses 16 symbols: 0 – 9, A, B, C, D, E, F
* 19F = 1\*162 + 9\*161 + 15\*160

**The Numbers 0 to 15 in Decimal, Binary & Hex**

|  |  |  |
| --- | --- | --- |
| **Decimal** | **Binary** | **Hexadecimal** |
| 0 | 0000 | 0 |
| 1 | 0001 | 1 |
| 2 | 0010 | 2 |
| 3 | 0011 | 3 |
| 4 | 0100 | 4 |
| 5 | 0101 | 5 |
| 6 | 0110 | 6 |
| 7 | 0111 | 7 |
| 8 | 1000 | 8 |
| 9 | 1001 | 9 |
| 10 | 1010 | A |
| 11 | 1011 | B |
| 12 | 1100 | C |
| 13 | 1101 | D |
| 14 | 1110 | E |
| 15 | 1111 | F |

**Converting Binary to Hexadecimal**

* Starting from the right, divide the 0’s and 1’s into groups of 4. Pad on the left with 0’s, if needed, to form a group of 4. Find the corresponding hex value from the table.
* 11011011100011 = 0011 0110 1110 0011 = 3 6 E 3 = 36E3

**Converting Hexadecimal to Binary**

* Convert each symbol to its corresponding binary value.
* 2AF = 2 A F = 0010 1010 1111 = 001010101111

**Converting Binary/Hexadecimal to Decimal**

* Multiply each symbol by the base value raised to a positional power and then add each product.
* 11011 = 1\*24 + 1\*23 + 0\*22 + 1\*21 + 1\*20 = 27
* 2AF = 2\*162 + 10\*161 + 15\*160 = 687

**Converting Decimal to Binary/Hexadecimal**

* Divide by the base value until the quotient is 0. If converting to hex, convert the remainders to hex.
* 415 (in decimal) = 19F (in hex)

0

16 | 1 R1 => 1

16 | 25 R9 => 9

16 |415 R15 => F

**Arithmetic**

**Binary Addition**

1 11

0 0 1 1 1

+ 0 + 1 + 0 + 1 + 1

0 1 1 10 11

111111

11010110

+ 1101101

101000011

**Binary Subtraction**

02

0 1 1 ~~10~~

- 0 - 0 - 1 - 1

0 1 0 1

1

0~~2~~202 02

1~~10010~~11~~10~~

- 11010001

1001011101

**Hexadecimal Addition**

A27CB4 39CDF106

+ 6E3095 + A6F278C

110AD49 443D1892

**Hexadecimal Subtraction**

A52CF3 3B0029

- 2B7169 - 1765A4

79BB8A 239A85

**Storage**

The main storage of a computer’s memory is made up of bits (aka binary digits).

1 bit => binary 0 or 1

1 byte => 8 bits => 2 hex digits

1 halfword => 2 bytes => 16 bits => 4 hex digits

1 fullword => 4 bytes => 32 bits => 8 hex digits => 2 halfwords

1 doubleword => 8 bytes => 64 bits => 16 hex digits => 2 full words

Largest positive hexadecimal value that can be stored: 7FFFFFFF

* If first digit is 0 – 7, positive hex number.
* If first digit is 8 – F, negative hex number.

Largest positive binary value: 01111111111111111111111111111111

* First digit is called the **sign bit**
* If 0, positive binary number
* If 1, negative.

Negative numbers are stored by taking the two’s complement of the absolute value of the number.

**To find binary 2’s complement:**

1. Switch all of the 0s to 1s and 1s to 0s (finding the 1’s complement)
2. Add 1

100111100 => 011000011

+ 1

011000100

**To find hexadecimal 2’s complement:**

1. Subtract the number from FFFFFFFF
2. Add 1

FFFFFFFF FFFFFFFF FFFFFFFF

- 002BCF06 - 00000001 - FFD430FA

FFD430F9 FFFFFFFE 002BCF05

+ 1 + 1 + 1

FFD430FA FFFFFFFF 002BCF06

**Arithmetic using the two’s complement:**

* **Addition** – same
* **Subtraction** – Find the two’s complement of subtrahend (# after the subtraction sign) and add it to the number

**Overflow:**

Occurs when a number becomes too large for its representation scheme.

To check for overflow:

1. Convert the 1st digit of each number to binary
2. Add the binary values together
3. If the last two carry bits are the same, no overflow.
4. If they are different, overflow.

00 <= NO overflow

729B6320 7 => 0111

+ 8A5C973C 8 => 1000

FCF7FA5C 1111

10 <= overflow

92B176C0 9 => 1001

+ 859237A4 8 => 1000

1843AE64 0001

01111 <= overflow

328AC105 328AC105 3 => 0011

- 807B96AF => 7F846951 => + 7F846951 7 => + 0111

B20F2A56 1011