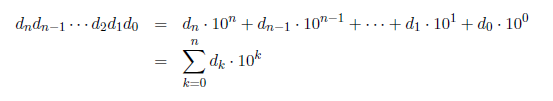
# CIS7 Chapter 1 Notes

## Number Representations

Computer stores data in 1s and 0s, known as **binary** or **base-2 number system.**

Non-negative or positive integers include 0, 1, 2, 3….

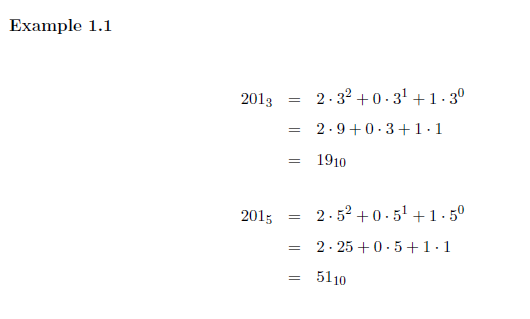
**Base-10** or **decimal numbers** (our counting numbers) represents (0, 1, 2, 3, 4, 5, 6, 7, 8, and 9).



For example, 60325 = 6 \* 104 + 0 \* 103 + 3 \* 102 + 2 \* 101 + 5 \* 100 = 60000+ 0000 +300 + 20+ 5.

**Theorem 1**: Let b be an integer greater than 1. Then if n is a positive integer, n can be expressed uniquely in the form where k is a nonnegative integer, a0, a1… , ak are nonnegative integers less than b and ak ≠ 0.

Integer theorem



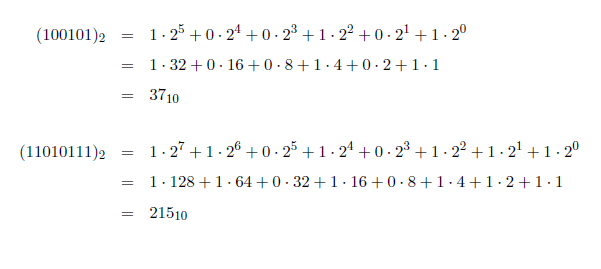
*The above example illustrates the conversion for base-3 and base-5 numbers to base-10 or decimal numbers.*

We say b is the base of expansion of n and we write n = (akak-1 \_ \_ \_ a1a0)b. For “short" numbers, typically those with three digits or less, we often eliminate the parentheses.

### Binary Representation

In the binary representation, the base is 2 and the integers, ak; ak-1…a1; a0 must be nonnegative and less than 2.

Example:



*You can use Addition/Subtraction method to convert binary to decimal numbers or vice-versa. Another method you would be division.*

### Decimal-Binary Conversion: Subtraction Method

*Example:* ***Convert decimal value 14 to a binary value****.*

***14 – 8 = 6***

***6 – 4 = 2***

***2 – 2 = 0***

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | *MSB* |  |  | *LSB* |
| Binary |  |  |  |  |  | ***1*** | ***1*** | ***1*** | ***0*** |
| Decimal | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| Decimal | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 |

### Binary-Decimal Conversion: Addition Method

***Example: 0110110***

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Binary |  |  | *0* | *1* | *1* | *0* | *1* | *1* | *0* |
| Decimal | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

***Add the decimal value of 1s’***

***32+16+4+2 = 54 🡨 decimal value of 01100110***

### Decimal- Binary Conversion: Division Method

1. Dividing each decimal number by “2” plus a remainder.
2. If the decimal number being divided is even then the result will be whole and the remainder will be equal to “0”. If the decimal number is odd then the result will not divide completely and the remainder will be a “1”.
3. The binary result is obtained by placing all the remainders in order with the least significant bit (LSB) being at the top and the most significant bit (MSB) being at the bottom.

Example: ***Convert decimal value 152 to a binary value.***

***152/2 = 76 remainder 0***

***76/2 = 38 remainder 0***

***38/2 = 19 remainder 0***

***19/2 = 9 remainder 1***

***9/2 = 4 remainder 1***

***4/2 = 2 remainder 0***

***2/2 = 1 remainder 0***

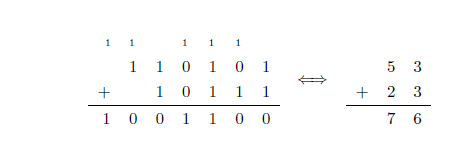
***1/2 = 0 remainder 1***

***Write binary value from bottom up for left to right: 10011000***

## Binary Arithmetic

### Binary Addition

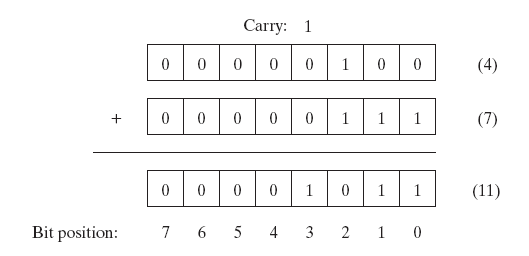
Like decimal integers addition, the same algorithm applies to adding binary integers but when you add the bits in a column, you either get a sum less than two and you write the single bit, 0 or 1, in that place.



**Binary Addition rules:**

|  |  |
| --- | --- |
| 0 + 0 = 0 | 0 + 1 = 1 |
| 1+0 = 1 | 1 + 1 = 10  The extra digit generates a carry to the next highest possible bit position. |

Example:



1. Add the following binary:

1 0 1 0 1 1 0 1

+0 1 0 1 0 1 1 1

1. Add the following binary:

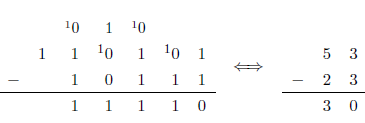
0 0 1 1 0 1 0 1

+1 0 1 0 1 1 1 1

### Binary Subtraction

**Binary Subtraction Rules**

|  |  |
| --- | --- |
| 0 – 0 = 0 | 1 – 0 = 1 |
| 1 – 1 = 0 | **0 – 1 = 1 with borrow 1** |

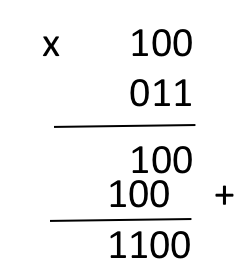


### Binary Multiplication

**Binary Multiplication Rules**

|  |  |
| --- | --- |
| 0 x 0 = 0 | 0 x 1 = 0 |
| 1 x 0 = 0 | **1 x 1 = 1** |

**Example:**



## Bytes

A **byte** is an 8-bit binary number with leading zeros allowed.

There are 256 different bytes and they represent the integers from 0 (00000000) to 255 (11111111).

A **word** is a basic unit of storage whose size depends on the particular computer. Words are commonly composed of four or eight bytes**, 32 or 64 bits** respectively.

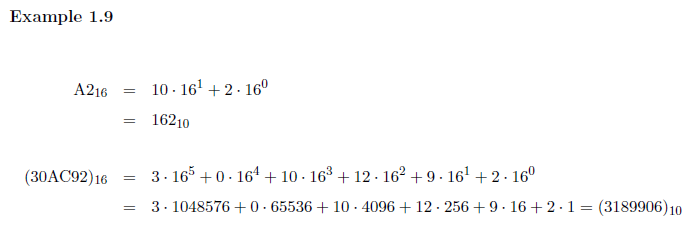
**ASCII** uses the ***lower 7-bits of a byte***, **0 to 127**, to represent letters and special characters;

**ISO Latin-1 and Mac-Roman** (now obsolete) use values ***above 127 for accented letters and additional special characters***.

**Unicode** is an international standard intended to encode all characters in all languages as well as mathematical and other specialized characters. UTF-32, also called UCS-4, uses four bytes to encode Unicode characters.

## Hexadecimal Representation

**Hexadecimal** use the ordinary decimal digits and the letters A, B, C, D, E, and F (or a, b, c, d, e, and f) to represent 10, 11, 12, 13, 14, and 15 respectively.



***Each hex-digit*** corresponds ***to four bits which is half a byte*** or a **nibble**.

A **byte** can be represented ***by two hex-digits*** instead of 8 bits, and a 32-bit word can be written with only 8 hex-digits.

### Hexadecimal Conversion to Decimal

1. Divide the number by 16.
2. Get the integer quotient for the next iteration.
3. Get the remainder for the hex digit.
4. Repeat the steps until the quotient is equal to 0.

**Example #1**

Convert 756210 to hex:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Division by 16 | Quotient | Remainder (decimal) | Remainder (hex) | Digit # |
| 7562/16 | 472 | 10 | A | 0 |
| 472/16 | 29 | 8 | 8 | 1 |
| 29/16 | 1 | 13 | D | 2 |
| 1/16 | 0 | 1 | 1 | 3 |

So 756210 = 1D8A16

### Decimal – Hexadecimal Conversion

***Multiply each digit of the hex number with its corresponding power of 16 and sum:***

decimal = *dn-1×16n-1 + ... + d3×163 + d2×162 + d1×161+d0×160*

**Example #1**

3B in base 16 is equal to each digit multiplied with its corresponding 16n:

3B16 = 3×161+11×160 = 48+11 = 5910

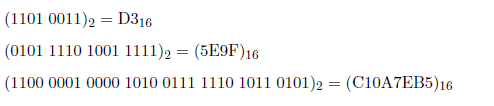
**Example #2**

E7A9 in base 16 is equal to each digit multiplied with its corresponding 16n:

E7A916 = 14×163+7×162+(10×161)+(9×160) = 57344+1792+160+9 = 5930510

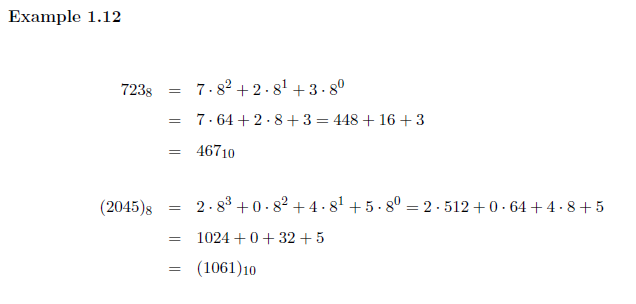
### Hexadecimal-Binary Conversion

To convert a binary integer to hex, each four-bit cluster corresponds to a single hex-digit.

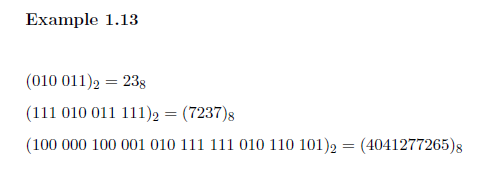


## Octal System

Octal system used to be widespread back when many computers used 6- bit bytes, as a 6-bit byte can be conveniently written as a two-digit octal number. Since nowadays a byte is almost always 8-bits long the octal system lost most of its appeal to the hexadecimal system.



To convert a binary integer to octal, each three-bit cluster corresponds to a single octal digit. If the number of bits in the binary integer is not a multiple of three, add zeros to the left, e.g., 11011 = 011011. There are spaces between to separate the three-bit clusters.



## Representing Negative Numbers

A single bit is used to represent the sign of the number (+ or -) and the remaining bits to represent the magnitude of the number (how positive or negative it is).

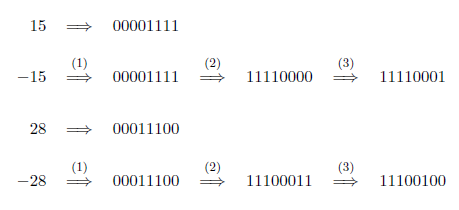
The most common representation of positive and negative integers is two's complement. In two's complement, positive integers are represented in standard binary, as in signed magnitude. However, the representation of a negative number is determined as follows:

1. compute a binary representation of the magnitude of the number,
2. Flip all the bits
3. Add 1

Example:

Example of negative decimal conversion to binary. 

The **most-significant bit** of 1 again signifies a negative number, but the remaining bits do not encode the magnitude in the usual way.



To convert a negative two’s complement number back to decimal, follow these steps:

1. flip all the bits,
2. add 1
3. Interpret the result as a binary representation of the magnitude and add a negative sign.

Example:

