Dublin City University
School of Electronic Engineering

EE102: Software Engineering I

Section 5 – Storing Information

5.1 Motivation:

- **Various information types**
  - Various types of numbers
  - Text
  - Images, Audios, Videos

- **Storage and processing by computers**
  - Ease of representation within computer memory/storage
  - Simplicity of data processing
  - Minimum effort of conversion between real world format and internal computer representation and vice-versa

5.2 Solution:

- **Binary representation**
  - Any data is converted into and from binary when inputted and outputted respectively
  - Internally all processing is performed in binary format

5.3 Binary number system:

- **Binary to decimal conversion**

Rank: 4 3 2 1 0

No: **1 0 0 1 1** $_2$

$$10011_2 = 1 * 2^0 + 1 * 2^1 + 0 * 2^2 + 0 * 2^3 + 1 * 2^4 =$$

$$= 1 * 1 + 1 * 2 + 0 * 4 + 0 * 8 + 1 * 16 =$$

$$= 19_{10}$$

- **Decimal to binary conversion**

Decimal No.	Base Divisor	Quotient	Binary Remnant
19	2	9	1
9	2	4	1
4	2	2	0
2	2	1	0
1	2	0	1

$$19_{10} = 10011_2$$

- **Notes about number systems**
  - **Decimal system (base 10)** is based on powers of 10
  - It has 10 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

$$43_{10} = 4 * 10^1 + 3 * 10^0 = 40 + 3$$

$$957_{10} = 9 * 10^2 + 5 * 10^1 + 7 * 10^0 = 900 + 50 + 7$$

  - **Binary system (base 2)** is based on powers of 2
  - It has 2 digits: 0 and 1

$$110_2 = 1 * 2^2 + 1 * 2^1 + 0 * 2^0 = 4 + 2 + 0 = 6_{10}$$

$$101_2 = 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 4 + 0 + 1 = 5_{10}$$

  - **Octal system (base 8)** is based on powers of 8
  - It has 8 digits: 0, 1, 2, 3, 4, 5, 6, 7

$$123_8 = 1 * 8^2 + 2 * 8^1 + 3 * 8^0 = 64 + 16 + 3 = 83_{10}$$

$$47_8 = 4 * 8^1 + 7 * 8^0 = 32 + 7 = 39_{10}$$

  - **Hexadecimal system (base 16)** is based on powers of 16
  - It has 16 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

$$B3_{16} = B * 16^1 + 3 * 16^0 = 11 * 16 + 3 * 1 = 179_{10}$$

$$47_{16} = 4 * 16^1 + 7 * 16^0 = 4 * 16 + 7 * 1 = 71_{10}$$

- **Correspondence Table**

Decimal	Binary	Octal	Hexadecimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

- **Hexadecimal to binary conversion**
  - Each hexadecimal digit corresponds to a four digit binary number
  - Replace each hexadecimal digit with corresponding four digit binary number

Hexadecimal	**A**	**3**	**E**	**9**	**3**	$_{16}$
Binary	**1010**	**0011**	**1110**	**1001**	**0011**	$_{2}$

- **Binary to hexadecimal conversion**
  - Each group of four binary digits (a binary four digit number) corresponds to one hexadecimal digit
  - Group binary digits in clusters of four, starting from right
  - Add leading 0-s in front of the binary number if necessary
  - Replace each four digit binary group with the corresponding hexadecimal digit

Binary	**0010**	**0101**	**1011**	**1000**	**0001**	$_{2}$
Hexadecimal	**2**	**5**	**B**	**8**	**1**	$_{16}$

- **Homework**
  - Octal to binary conversion
  - Binary to octal conversion
  - Practice conversions as much as you can

5.4 Definitions:

- **Bit**
  - Single binary digit (1 or 0)
- **Byte**
  - A set of 8 adjacent bits
- **Least Significant Bit**
  - Left most bit in a byte
- **Most Significant Bit**
  - Right most bit in a byte

5.5 Binary operations:

- **Arithmetic**
  - Based on the following tables

X+Y	Y=0	Y=1
X=0	0	1
X=1	1	10

X-Y	Y=0	Y=1
X=0	0	-1
X=1	1	0

X*Y	Y=0	Y=1
X=0	0	0
X=1	0	1

X/Y	Y=0	Y=1
X=0	-	0
X=1	-	1

- **Logic**
  - Based on the following tables

X or Y	Y=0	Y=1
X=0	0	1
X=1	1	1

X and Y	Y=0	Y=1
X=0	0	0
X=1	0	1

X xor Y	Y=0	Y=1
X=0	0	1
X=1	1	0

	not X
X=0	1
X=1	0

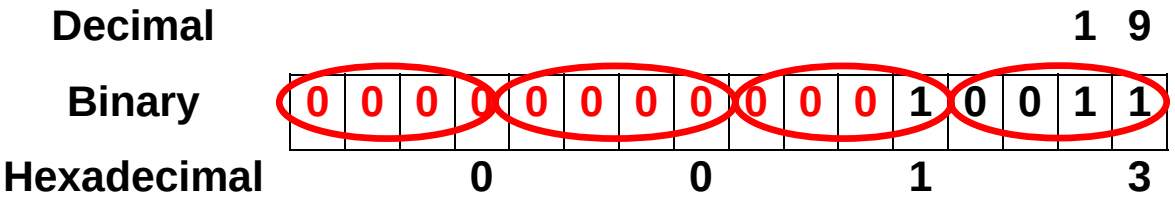
E.g.

X	1	0	0	0	1	1	0	1	0	1	0	0	0	1	1	0
Y				1	1	1	1	0	1	1	0	0	1	1	0	1
X + Y	1	0	1	0	1	1	0	0	0	0	0	1	0	0	1	1
Hex				A				C				1				3

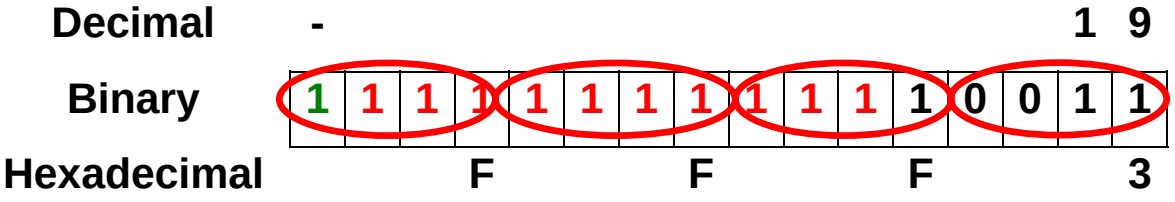
X	1	0	0	0	1	1	0	1	0	1	0	0	0	1	1	0
Y				1	1	1	1	0	1	1	0	0	1	1	0	1
X or Y	1	0	0	1	1	1	1	1	1	1	0	0	1	1	1	1
Hex				9				F				C				F

5.6 Representing Various Numbers:

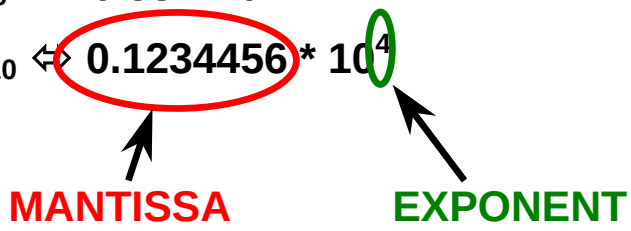
- **Positive Integers**
  - Binary representation
  - In 16-bit or 32-bit long locations of memory
  - Leading 0-s are added in front
  - Hexadecimal is used for easiness of notation



- **Sign Integers**
  - Binary representation + a sign bit in the most significant location
  - In 16-bit or 32-bit long locations of memory
  - Leading 0-s (for positive numbers) or 1-s (for negative numbers) are added in front
  - Hexadecimal is used for easiness of notation



- **Real Numbers**
  - Binary representation (in mantissa + exponent format) + a sign bit in the most significant location
  - In 32-bit or 64-bit long locations of memory
  - Leading 0-s (for positive numbers) or 1-s (for negative numbers) are added in front
  - Hexadecimal is used for easiness of notation
  - E.g. **$5{,}000{,}000_{10} \Leftrightarrow 0.5 * 10^{7}$**
  - E.g. **$-0.00053_{10} \Leftrightarrow -0.53 * 10^{-3}$**
  - E.g. **$1234.456_{10} \Leftrightarrow 0.1234456 * 10^{4}$**

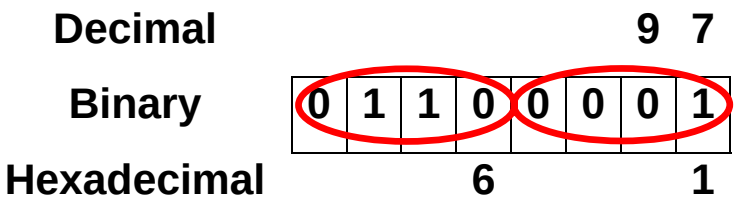


5.7 Representing Text:

- **Characters**
  - General term used for letters, digits and punctuation
  - Each character is assigned a unique numerical code
  - How many codes are needed?
  - Digits (10): 0, 1, 2, 3,…, 9
  - Lower case letters (26): a, b, c,…, z
  - Upper case letters (26): A, B, C,…, Z
  - Punctuation (16): . , ; : ? ! “ ‘ ` { } [ ] ( )
  - Special characters (18): # $ € ¥ £ % ^ & * + - = \ / < > | ~
  - Other characters: § © ® ¶ ± ¢ ¼
  - Characters for other languages: ß â ç ê ţ Њ љ Ω ə

- **ASCII code**
  - 7 bits allocated to store each code
  - 2^7 = 128 codes for 128 possible characters
  - Covers digits, lower and upper case letters, punctuation, special and other characters
  - Does not cover characters for other languages (accents, umlauts, fadas)

- **Latin-1**
  - Specified by the International standards Organization (ISO)
  - A subset of ISO-8859 that includes several sets of characters for writing in Cyrillic, Arabic, Hebrew, etc.
  - Extends ASCII, including additional characters used in some West European languages such as Irish, French and German
  - One byte (8 bits) allocated to store each code
  - 2^8 = 256 codes for 256 possible characters
  - Text encoded in Latin-1 can be transmitted through e-mail and be printed on any computer system, being accepted as basis for every text file formats
  - E.g. **'a' <=> 97_{10} = 1100001_2 = 61_{16}**



- **Words (Strings)**
  - A word is represented as a sequence of 0 or more characters in form of a string or an array of characters
  - E.g. **“ee102” = [‘e’, ‘e’, ‘1’, ‘0’, ‘2’] = [101, 101, 49, 48, 50]**

Binary	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**1**	**Hexa**	**6** **3**
Binary	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**1**	**Hexa**	**6** **3**
Binary	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**1**	**Hexa**	**3** **1**
Binary	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**Hexa**	**3** **0**
Binary	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**Hexa**	**3** **2**