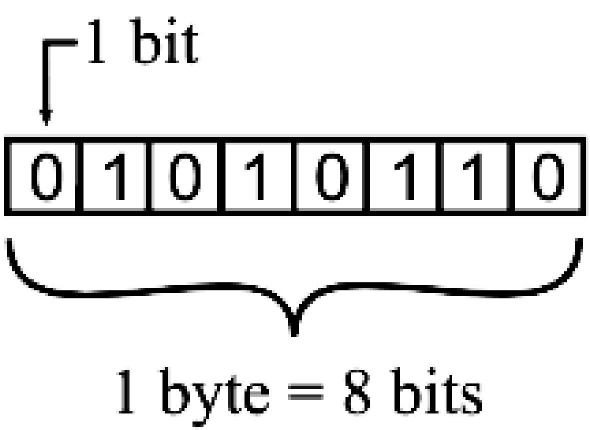
HNDIT1032 Computer and Network Systems

Week 2-Data Representation in Computers

Digital Data

- Most computers are Digital
- Understands only two discrete values
  - 0 (Off)
  - 1 (On)
- Each on or off value is called a **bit** (binary digit)



How Computers Represent Data?

- A computer is an **electronic device**
- Electronic devices process data by manipulating electricity

Presence of electricity (1)

Absence of electricity (0)

1011

Data Units use in a Computer

- A ***bit*** is the most basic unit of information in a computer.
  - It is a state of “on” or “off” in a digital circuit.
  - Sometimes they represent **high** or **low** voltage

- A ***byte*** is a group of eight bits.. It is the smallest possible *addressable* unit of computer storage.

Data Units use in a Computer...

- A ***word*** is a contiguous group of bytes.
  - Words can be any number of bits or bytes.
  - Word sizes of 16, 32, or 64 bits are most common.

- A group of four bits is called a ***nibble***.
  - Bytes, therefore, consist of two nibbles: a "high-order nibble," and a "low-order" nibble

Data Units use in a Computer…

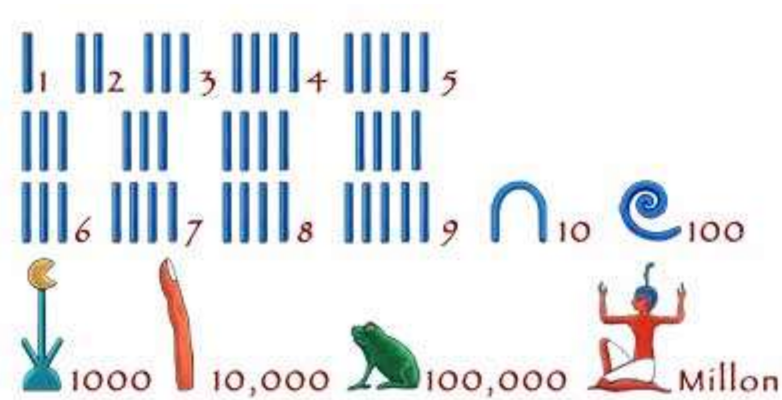
- **Bit**: It is the smallest unit of information used in a computer system. It can either have the value 0 or 1. Derived from the words *binary digit*.
- **Nibble**: It is a combination of 4 bits.
- **Byte**: It is a combination of 8 bits.
- **Word**: It is a combination of 16 bits.
- **Double word**: It is a combination of 32 bits.
- **Kilobyte (KB)**: It is used to represent the 1024 bytes of information.
- **Megabyte (MB)**: It is used to represent the 1024 KBs of information.
- **Gigabyte (GB)**: It is used to represent the 1024 MBs of information.

Types of Data Representations

- Character Representation
  - A, a, ?, @
- Number Representation
  - 1, 235, -10, 0

Number systems

- Positional Number System / Weighted Number System
- Non positional number system / non Weighted Number System



The Non-weighted/ Non Positional Numbers

- The non-weighted numbers are **not positional weighted** .
- That are not assigned with any weight to each digit position.
- position *independent*
- Ex-
  - Roman number system
    - ❖ Roman numerals symbols with different values: I (1), V (5), X (10), C (50), M (100)
    - ❖ Examples: I, II, III, IV, VI, VI, VII, VIII, IX
  - Egyptian number system

Weighted Numbers/ Positional Number

- The weighted numbers are those that obey the **position weighting principle**
- which states that the **position of each number represent a specific weight**.
- Numeric values are represented by a *sequence* of digit symbols. Each digit position has a value called a weight associated with it
- Ex:
  - decimal numbers
  - Binary numbers
  - Octal numbers
  - Hexadecimal numbers

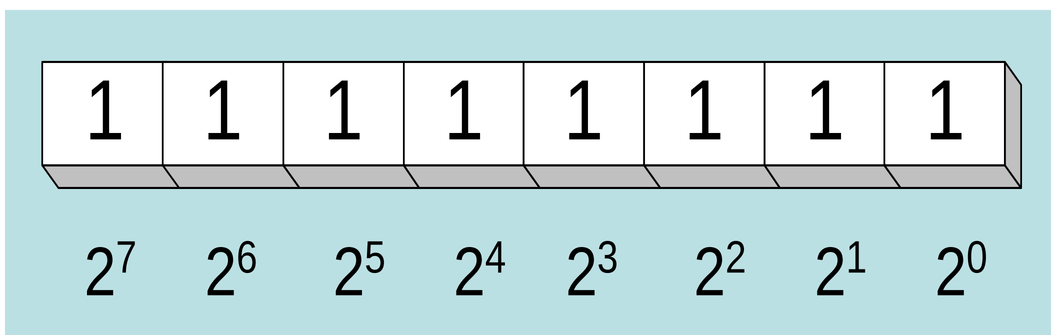
Number systems

- In computers, all numbers, letters, pictures, sounds are represented as numbers.
- There're different number systems

NUMBER SYSTEM	BASE VALUE	SYMBOLIC CHARACTER SET
Binary	2	0,1
Octal	8	0, 1, 2, 3, 4, 5, 6, 7
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, and 9
Hexadecimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Binary Number System

- Each digit (bit) is either 1 or 0
- Each bit represents a power of 2
- Every binary number is a sum of powers of 2



Octal Number System

- Contains eight digits (0, 1, 2, 3, 4, 5, 6, 7)
- The base is 8
- Each digit in an octal number represents a specific power of its base (8).
- The three binary digits can be represented with a single octal digit.

Decimal Number System

- Contains digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
- The base is 10.
- Each digit in decimal number represents a specific power of the base (10) of the number system.
- Widely used in our day to day life.

Hexadecimal Number System

- Contains 16 digits (0 to 9 and A to F)
- The base is 16.
- The A to F alphabets represent 10 to 15 decimal numbers.
- Each digit in a hexadecimal number represents a specific power of base (16) of the number system.
- Also known as alphanumeric number system

Converting Decimal to Binary

- $156_{10} = 010011100_2$

$$
\begin{array}{rl}
2\,\underline{)156} & \\
2\,\underline{)78} & 0 \\
2\,\underline{)39} & 0 \\
2\,\underline{)19} & 1 \\
2\,\underline{)9} & 1 \\
2\,\underline{)4} & 1 \\
2\,\underline{)2} & 0 \\
1 & 0
\end{array}
$$

Converting Decimal to Octal

- $156_{10} = 234_8$

$$
\begin{array}{l l}
8\underline{)156} & \\
8\underline{)19} & 4 \\
\quad\ 2 & 3
\end{array}
$$

Converting Decimal to Hexadecimal

- $156_{10} = 9C_{16}$

$$16\overline{)156}$$

9 C

Converting Binary to Decimal

$11001 = 1\times 2^4 + 1\times 2^3 + 0\times 2^2 + 0\times 2^1 + 1\times 2^0$

$= 16+8+0+0+1$

$=25$

Converting Octal to Decimal

$$325_8 = 3 \times 8^2 + 2 \times 8^1 + 5 \times 8^0$$

$$= 3 \times 64 + 2 \times 8 + 5 \times 1$$

$$= 192 + 16 + 5$$

$$= 213_{10}$$

Converting Hexadecimal to Decimal

$$(2056)_{16} = 2 \times 16^3 + 0 \times 16^2 + 5 \times 16^1 + 6 \times 16^0$$

$$= 2 \times 4096 + 0 + 80 + 6$$

$$= 8192 + 0 + 80 + 6$$

$$= (8278)_{10}$$

Representing Octal number using Binary

Octal	Binary
0	000
1	001
2	010
3	011

Octal	Binary
4	100
5	101
6	110
7	111

Example

- **$(53)_8$ in Binary**
  - Binary equivalent of **5** is **$(101)_2$**.
  - Binary equivalent of **3** is **$(011)_2$**.

$(53)_8$

$(101)(011)$

$(101011)_2$

Representing Hexadecimal Number Using Binary

Hex	Binary
0	0000
1	0001
2	0010
3	0011

Hex	Binary
4	0100
5	0101
6	0110
7	0111

Hex	Binary
8	1000
9	1001
A	1010
B	1011

Hex	Binary
C	1100
D	1101
E	1110
F	1111

Example

- **$(f3)_{16}$ in Binary**
  - Binary equivalent of **f** is **$(1111)_2$**.
  - Binary equivalent of **3** is **$(0011)_2$**.

$(f3)_{16}$

$(1111)(0011)$

$(11110011)_2$

Converting Octal to Hexadecimal

Ex : Convert 752_8 to hex

First convert the octal to binary:

$111\ 101\ 010_2$

re-group by 4 bits (add leading zeros)

$0001\ 1110\ 1010$

Then convert the binary to hex:

1 E A

So $752_8 = 1EA_{16}$

Converting Hexadecimal to Octal

Ex : Convert $E8A_{16}$ to octal

First convert the hex to binary:

$1110\ 1000\ 1010_2$

$111\quad 010\quad 001\quad 010$ and re-group by 3 bits (starting on the right)

Then convert the binary to octal:

$7\quad 2\quad 1\quad 2$

So $E8A_{16} = 7212_8$

Character Representation

- Any text-based data is stored by the computer in the form of bits(a series of 1s and 0s)
- The combinations of 0s and 1s used to represent data are defined by patterns called **coding schemes**
  - BCD
  - ASCII
  - Extended ASCII
  - EBCDIC
  - Unicode

BCD (Binary Coded Decimal)

- BCD uses 6 bits and can represent $2^6 = 64$ characters

Digit	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
Zones	
1111	Unsigned
1100	Positive
1101	Negative

FIGURE 2.5 **Binary-Coded Decimal**

Example

- 7093_{10} = ? (in BCD)

7	0	9	3
0111	0000	1001	0011

ASCII (American Standard Code for Information Exchange)

- Uses 7 bits and can represent 2^7 =128 characters
- Starts from *(ANSI) AMERICAN NATIONAL STANDARD INSTITUTE*
- Assigns standard numeric values to letters, numerals, punctuation marks, and other characters used in computers
- Every character is a unique ASCII code.
- The ASCII code for an uppercase A is 1000001.

ASCII Character Set

Low Order Bits		High Order Bits							
		0000 **0**	0001 **1**	0010 **2**	0011 **3**	0100 **4**	0101 **5**	0110 **6**	0111 **7**
0000	**0**	NUL	DLE	Space	0	@	P	`	p
0001	**1**	SOH	DC1	!	1	A	Q	a	q
0010	**2**	STX	DC2	"	2	B	R	b	r
0011	**3**	ETX	DC3	#	3	C	S	c	s
0100	**4**	EOT	DC4	$	4	D	T	d	t
0101	**5**	ENQ	NAK	%	5	E	U	e	u
0110	**6**	ACK	SYN	&	6	F	V	f	v
0111	**7**	BEL	ETB	'	7	G	W	g	w
1000	**8**	BS	CAN	(	8	H	X	h	x
1001	**9**	HT	EM	)	9	I	Y	i	y
1010	**A**	LF	SUB	*	:	J	Z	j	z
1011	**B**	VT	ESC	+	;	K	[	k	{
1100	**C**	FF	FS	,	<	L	\	l	|
1101	**D**	CR	GS	-	=	M	]	m	}
1110	**E**	SO	RS	.	>	N	^	n	~
1111	**F**	SI	US	/	?	O	_	o	DEL

“Hello, world” Example

		Binary		Hexadecimal		Decimal
H	=	01001000	=	48	=	72
e	=	01100101	=	65	=	101
l	=	01101100	=	6C	=	108
l	=	01101100	=	6C	=	108
o	=	01101111	=	6F	=	111
,	=	00101100	=	2C	=	44
	=	00100000	=	20	=	32
w	=	01110111	=	77	=	119
o	=	01100111	=	67	=	103
r	=	01110010	=	72	=	114
l	=	01101100	=	6C	=	108
d	=	01100100	=	64	=	100

Extended ASCII

- Uses 8 bits and can represent 2^8 =256 characters
- Extended version of ASCII
- Uses 8 bits for each character
- Introduced by IBM in 1981 for use in its first PC
- Extended ASCII represents the uppercase letter A as 01000001.
- Does not include enough code combinations to support all written languages.

EBCDIC (Extended Binary Coded Decimal Interchange Code)

- Extended BCD Interchange Code (pronounced *ebb'-se-dick*)
- 8-bit code
- Developed by IBM
- Rarely used today
- IBM mainframes only

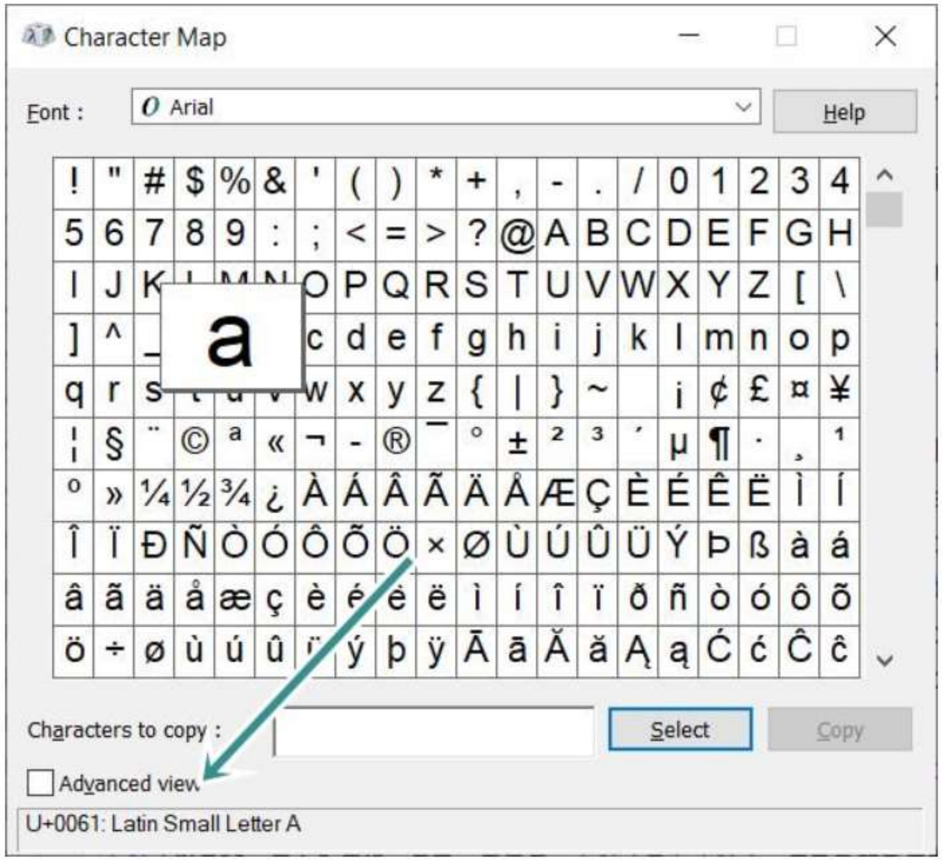
Unicode

- Unicode is a Universal Encoding System (UES)
- Uses sixteen bits and provides codes or 65,000 characters.
- Can support all the written languages
- Most common character-encoding system on the World Wide Web
- Unicode assigns code to every character
- The code is an integer value.

Example

- You can refer character map to see all the code for characters.
- For example the code point of **a** (Latin small letter) is **0061** or **U+0061**.

Example


Character Map
Font :
Arial
Help
a
Characters to copy :
Select
Copy
Advanced view
U+0061: Latin Small Letter A

Next Week Discussion

- Building blocks of Digital Systems (Logic Gates)