School of Engineering and Computer Science

Mathematics Applied to Digital Engineering

(NUMBER SYSTEM)

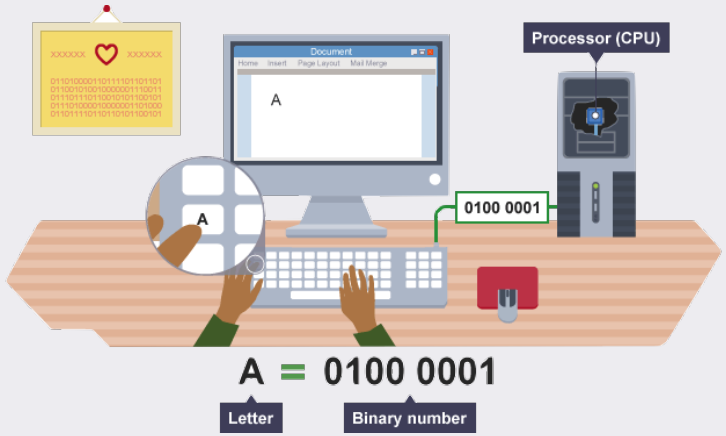
Kamel ATTAR
kamel.attar@epita.fr

Week **#2** ✦ Friday 3/NOV/2023 ✦
Week **#3** ✦ Friday 10/NOV/2023 ✦
Week **#4** ✦ Friday 17/NOV/2023 ✦

Definition (Number System)

- A number system is defined as the representation of numbers by using digits or other symbols in a consistent manner and are understood by computers.
- Each number is represented by a string of digits where the position of each digit is associated with a weight:

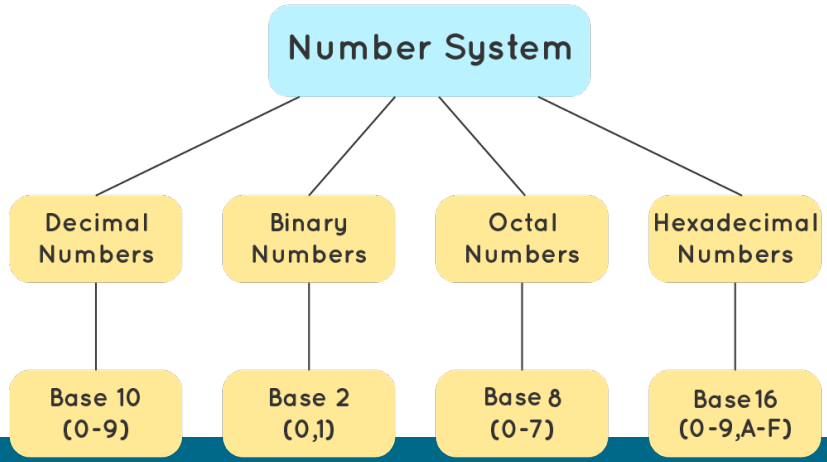
$$d_k d_{k-1} \cdots d_1 d_0$$



where d_k is referred to as the **most significant digit** (MSD) and d_0 as the **least significant digit** (LSD). Each digit position d_i has an associated weight b^i where b is called the base .

There are different types of number systems in which the four main types are as follows.

1. **Binary number system**
2. **Octal number system**
3. **Decimal number system**
4. **Hexadecimal (hex) number system**



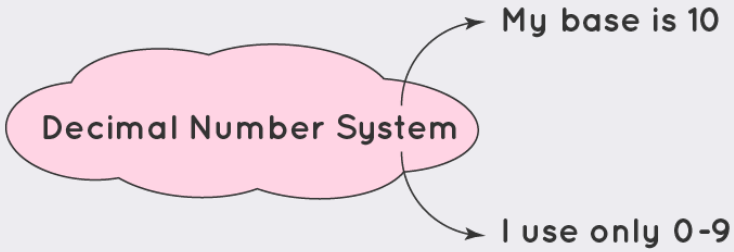
Number is expressed with digits, while a numeral is a word describing a number.

★ Number system ★

Decimal Number System

Definition (Decimal Number System)

- **Name** decem (Latin) $\Longrightarrow$ ten
- **Characteristics**
  - It has only ten (**10**) digits from **0** to **9**. Every number (value) represents with **0**, **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8** and **9** in this number system.
  - It is positional number system **2945** $\neq$ **2495**.
- The base of decimal number system is **10**, because it has only **10** digits.

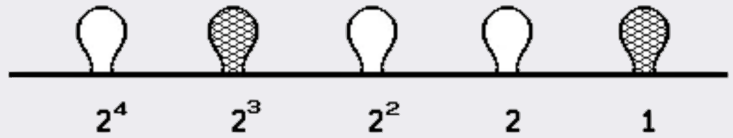


$$d_{n-1}d_{n-2}\cdots d_1d_0 = d_{n-1}10^{n-1} + d_{n-2}10^{n-2}\cdots d_1 10^1 + d_0 10^0.$$

- (Most) people use the decimal number system.
- If any number is represented without a base, it means that its base is **10**.

Definition (Binary Number System)

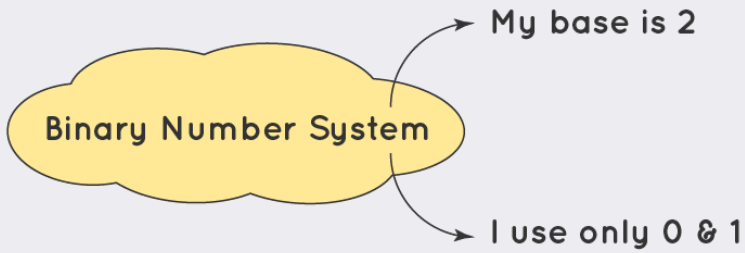
- **Name** binarius (Latin) $\Longrightarrow$ two



- **Characteristics**
  - It has only two digits that are **0** and **1**. Every number (value) represents with **0** and **1** in this number system.
  - It is positional number system in which the allowable digits **0** and **1** that are called “bits”. $(1010)_2 \neq (1100)_2$.
- A Binary number The base of binary number system is **2**, because it has only two digits.

$$n = a_k 2^k + a_{k-1} 2^{k-1} + \cdots + a_1 2^1 + a_0 .$$

Here the coefficients $a_0, a_1, \cdots, a_k$ can take the value of **0** or **1**.



- Computers store all data as binary digits, but we may need to convert this to a number system we are familiar with.

When you look at a byte, the rightmost bit is the "one's place". The next bit is the "two's place". The next the "four's place", The next the "eight's place" and so on.

1024	512	256	128	64	32	16	8	4	2	1
2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Example

$(11011)_2 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 16 + 8 + 2 + 1 = 27$.

✎ **Exercise 1.** What is the decimal expansion of the integer that has $(\mathbf{1\ 0101\ 1111})_{\mathbf{2}}$ as its binary expansion?

✎ **Exercise 2.** A particular binary number has **3** digits.

a. What are the largest and smallest possible binary numbers?

b. Convert these numbers to base **10**.

✎ **Exercise 3.** When a particular base **10** number is converted it gives a **4**-digit binary number. What could the original base **10** number be? (*Write the numbers as a list. e.g.* **21**, **22**, **23**, *etc*)

✎ **Exercise 4.** A **4**-digit binary number has **2** zeros and **2** ones.

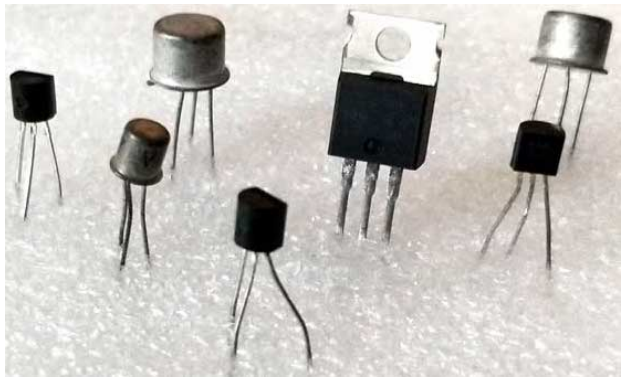
a. List all the possible binary numbers with these digits.

b. Convert these numbers to base **10**.

✎ **Exercise 5.** A binary number has **8** digits and is to be converted to base **10**. What is the *smallest* and *largest* possible base **10** answer?

★ Why Binary Numbers are Used? ★

- The first and foremost reason is that electronic components, as a natural coincidence, operate in a binary mode. A switch is either open/off (called **0** state) or closed/on (called **1** state); a transistor is either not conducting (**0** state) or is conducting (**1** state).

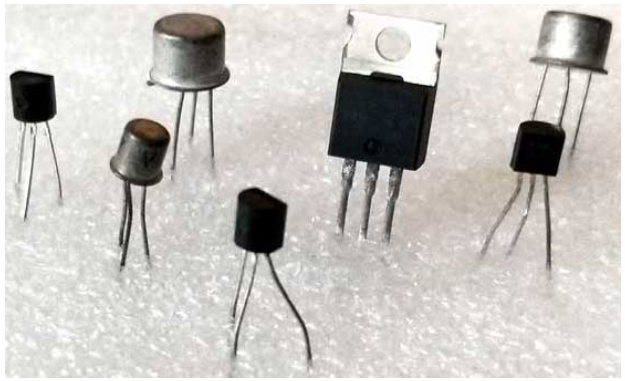


This two-state nature of the electronic components can be easily expresses with the help of binary numbers.

- The second reason is that computer circuits primarily use a binary system, which consists of only two digits: **0** and **1** instead of **10** digits of the decimal system. This fundamental difference simplifies the design of the machine, reduces the cost and improves the reliability.

★ Why Binary Numbers are Used? ★

- The first and foremost reason is that electronic components, as a natural coincidence, operate in a binary mode. A switch is either open/off (called **0** state) or closed/on (called **1** state); a transistor is either not conducting (**0** state) or is conducting (**1** state).



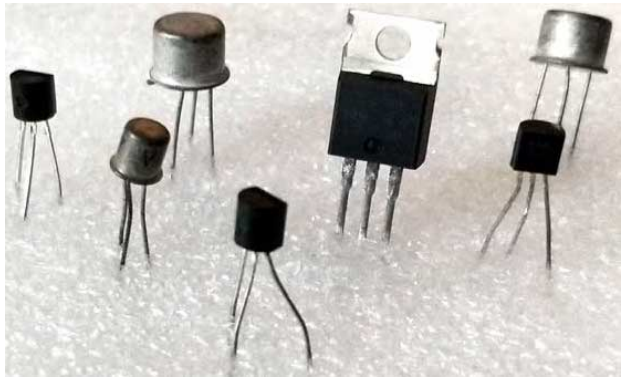
This two-state nature of the electronic components can be easily expresses with the help of binary numbers.

- The second reason is that computer circuits primarily use a binary system, which consists of only two digits: **0** and **1** instead of **10** digits of the decimal system. This fundamental difference simplifies the design of the machine, reduces the cost and improves the reliability.

Another reason why Binary Numbers are used ?

★ Why Binary Numbers are Used? ★

- The first and foremost reason is that electronic components, as a natural coincidence, operate in a binary mode. A switch is either open/off (called **0** state) or closed/on (called **1** state); a transistor is either not conducting (**0** state) or is conducting (**1** state).



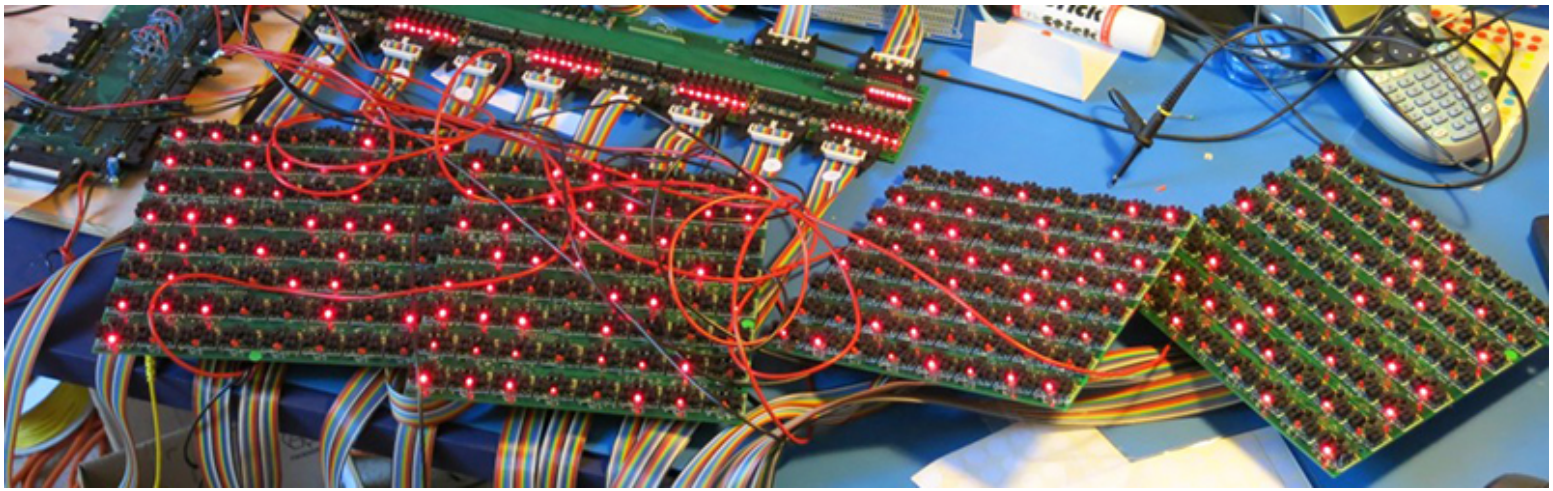
This two-state nature of the electronic components can be easily expresses with the help of binary numbers.

- The second reason is that computer circuits primarily use a binary system, which consists of only two digits: **0** and **1** instead of **10** digits of the decimal system. This fundamental difference simplifies the design of the machine, reduces the cost and improves the reliability.
- Lastly, binary number system is used because all the operations that can be done in the decimal system can also be done with a binary number of radix **2**.

★ Bit is a code ★

Wires conduct electricity. When there is a potential difference, it causes electric difference and flow of electric current.

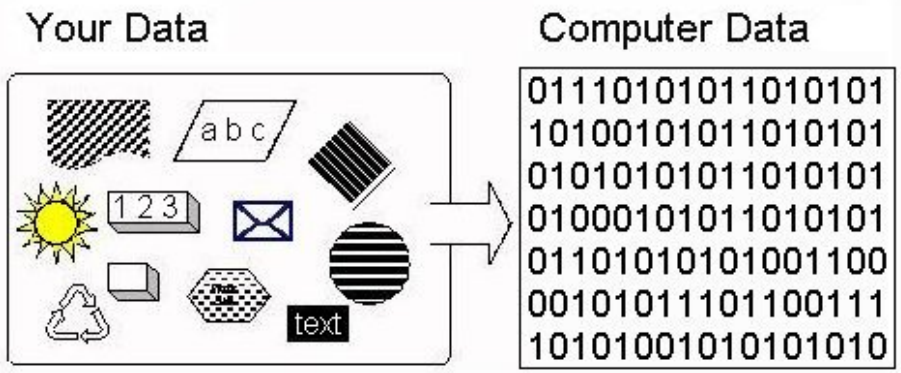
- High or low amount of electric current can represent a two types of signals...
  ...so one wire can represent a logical zero or a logical one.
- Eight wires can represent **256** unique combinations of **1**s and **0**s.
- In general, k wires can represent 2^k different combinations of **1**s and **0**s
- A combination of bit patterns is a *code* that represents a *data value*.



✎ **Exercise 6.** The base **10** number **999** is to be converted to binary. How many more digits does the binary number have than the number in base **10**?

★ Representing data in the computer ★

- We have all seen computers do many things with all kinds of sounds, pictures, graphics, numbers, and text. It seems that we can build a replica of parts of our world inside the computer.
- All of the wonderful multi-media that we see on modern computers is all constructed from simple ON/OFF switches.
- The trick is to take all of the real-world sound, picture, number etc data that we want in the computer and convert it into the kind of data that can be represented in switches, as shown in the figure.



- The computer has switches to represent data and switches have only two states: ON and OFF. Binary has two digits to do the counting: **0** and **1** - a natural fit to the two states of a switch (**0** = OFF, **1** = ON).

★ Representing data in bytes ★

- bit: Short for binary digit, the smallest unit of information on a machine.
  A single bit can hold only one of two values: **0** or **1**.
  So we can't store much in **1** bit, we group bits together into larger units to obtain meaningful information.
- nibble: Half a byte – four bits. Nibbles are important in hexadecimal and BCD representations. The term is sometimes spelled nybble.
- byte: Abbreviation for binary term, a unit of storage capable of holding a single character. On almost all modern computers, a byte is equal to **8** bits.
  1 byte = **8** bits = — — — — — — — —

Number of bits	Number of patterns	**2** raised to the power	Number of bytes	Unit	
1	**2**	**1**		Bit	**0/1**
2	**4**	**2**			
3	**8**	**3**			Octal unit
4	**16**	**4**		Nibble	Hexadecimal unit
5	**32**	**5**			
6	**64**	**6**			
7	**128**	**7**			
8	**256**	**8**	**1**	Byte	One character e.g. 'A' or 'x' or '$'

The computer uses a single **byte** and **binary number** to represent a single character.

For example, the letter 'A' is represented by **01000001** and 'C' is represented by **01000011**.

Character	Bit pattern	Byte number	Character	Bit pattern	Byte number
A	**01000001**	**65**	.	**00101110**	**46**
B	**01000010**	**66**	:	**00111010**	**58**
C	**01000011**	**67**	>	**00111110**	**62**
a	**01100001**	**97**	1	**00110001**	**49**
b	**01100010**	**98**	2	**00110010**	**50**

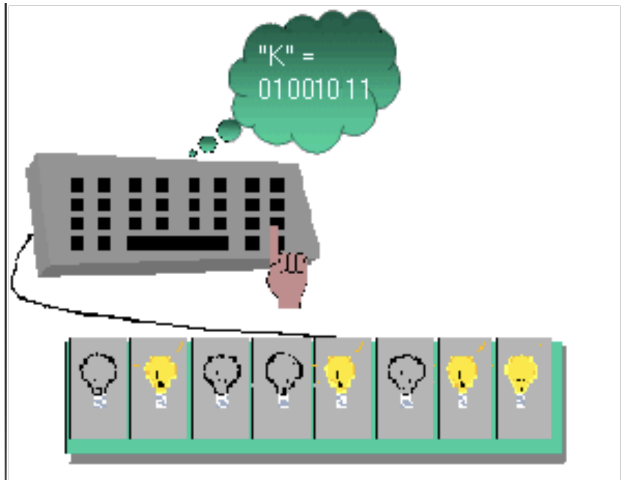
Thus, when you type a 'K' on the keyboard, circuitry on the keyboard and in the computer converts the 'K' to the byte **01001011** and stores the letter in the computer's memory.

✎ Exercise 7.

a. What is the minimum number of bits that are required to uniquely represent the characters of English alphabet? (Consider upper case characters alone)

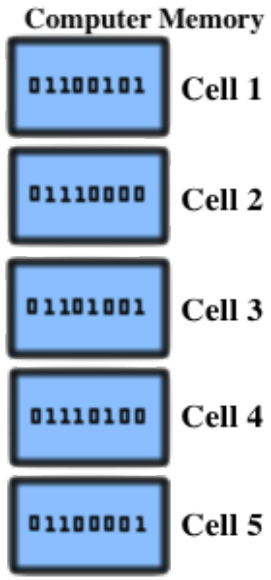
b. How many more characters can be uniquely represented without requiring additional bits?

c. What are the **8**-bit patterns used to represent each of the characters in the string "***CS/ECE* 252**" and "***Badgers* 20**"? (Only represent the characters between the quotation marks.) ▸ Go to the Character Table



★ Unit of information storage ★

- **1** word = **4** bytes = **4** × **8** = **32** = 2^5 bits (usually). If the person typed the word "epita", it would be represented by the following **5** bytes in the computer's memory (think of it as five rows of eight switches in memory being **ON** or **OFF**):
- But we can't store much in **1** — **4** bytes either, so we refer to larger storage capacities using abbreviations as shown below (notice how each unit is **2** power of **10**) .
- Large amounts of memory are indicated in terms of kilobytes, megabytes, and gigabytes. A disk that can hold **1.44** megabytes, for example, is capable of storing approximately **1.4** million characters, or about **3, 000** pages of information.



Numbers of bytes	2 raised to the power	Unit	
1		Byte	One character
1024	**10**	KiloByte (Kb)	Small text
1, 048, 576	**20**	MegaByte (Mb)	A book
1, 073, 741, 824	**30**	GigaByte (Gb)	An large encyclopedia
1, 099, 511, 627, 776	**40**	TeraByte	

The total memory size represents the total amount of bits that can be stored in memory.

memory size = number of addresses(Cells) x size of a word

Here, number of possible addresses = 2^b, where b is the number of bits needed to address a word.

1. A memory stores **8**-bit words (**1** byte) and has 2^{16} addresses (Cells). What is the total memory size in kilobytes (KB)?
2. A memory stores **16**-bit words (**2** bytes) and requires **8** bits to address them (2^8 addresses (Cells)). What is the total memory size in bytes?
3. A memory has a total size of **32** MB and can store **32**-bit words. How many bits do we need to represent the addresses in this memory?

Solution

The total memory size represents the total amount of bits that can be stored in memory.

memory size = number of addresses(Cells) x size of a word

Here, number of possible addresses = 2^b, where b is the number of bits needed to address a word.

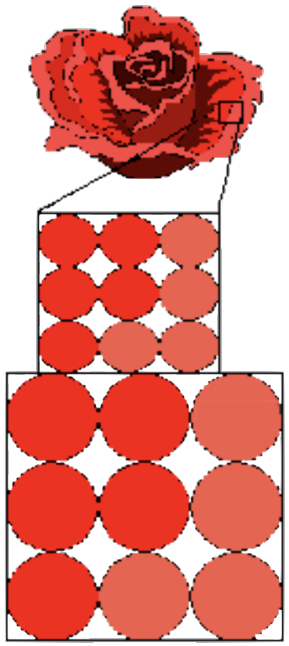
1. A memory stores **8**-bit words (**1** byte) and has 2^{16} addresses (Cells). What is the total memory size in kilobytes (KB)?
2. A memory stores **16**-bit words (**2** bytes) and requires **8** bits to address them (2^8 addresses (Cells)). What is the total memory size in bytes?
3. A memory has a total size of **32** MB and can store **32**-bit words. How many bits do we need to represent the addresses in this memory?

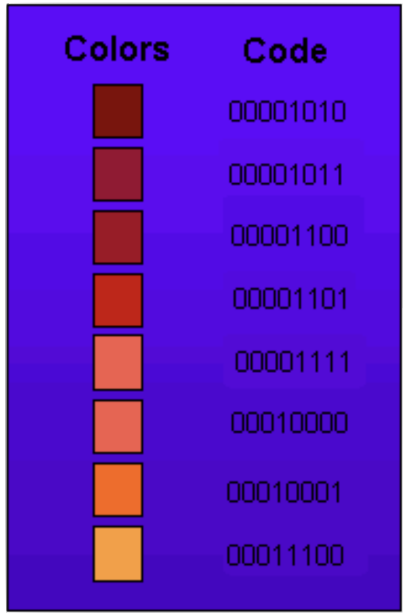
Solution

1. Memory size = $1 \times 2^{16} = 2^6 \times 2^{10} = 64KB$
2. Memory size = $2 \times 2^8 = 2^9 = 512B$
3. size of words = $32MB \div 4B = (32 \times 2^{20}) \div 4 = 2^{25} \div 2 = 2^{23}$

★ Bitmap ★

- Computer graphic data like pictures, frames of a movie, drawings, or frames of an animation are represented by a grid of pixels. A 'pixel' is short for 'Picture Element'.
- Bitmaps are ways of storing images on a computer. Bitmap images are used on cameras, smartphones, etc.
- A bitmap is laid out in a grid format with each box on the grid containing 1 pixel each. (resolution)
- Bitmap images can become poor quality if they are zoomed in too far. This is because each pixel is stretched and becomes a lot bigger than the original.
- Each pixel can only be one single colour at a time, however when thousands of pixels are used together they can create very detailed images.
- Each pixel can determine what colour to display as it is represented by a binary value that corresponds to a colour.
- A pixel in one byte can be one of **256**(=8bits=1byte) shades of grey (usually with **0** being white and **255** being black).
- The greater the number of pixels within a specific area, the higher the image quality.

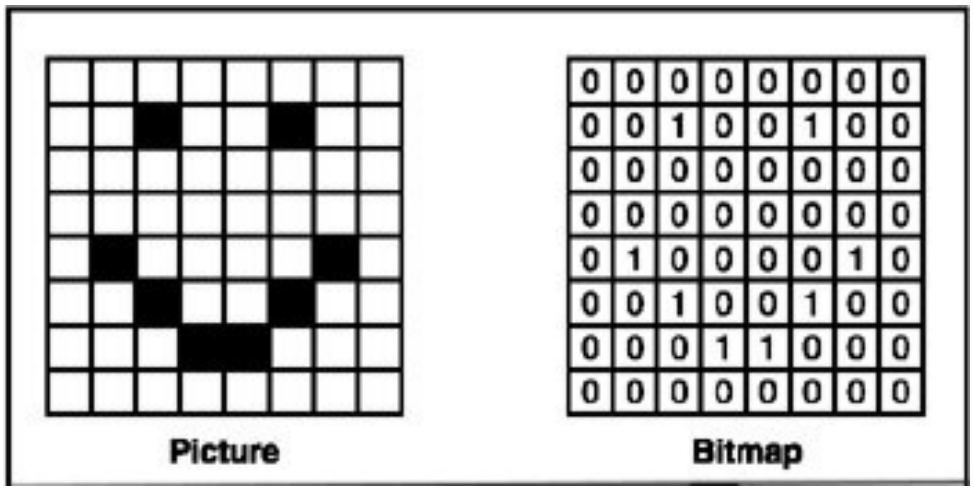




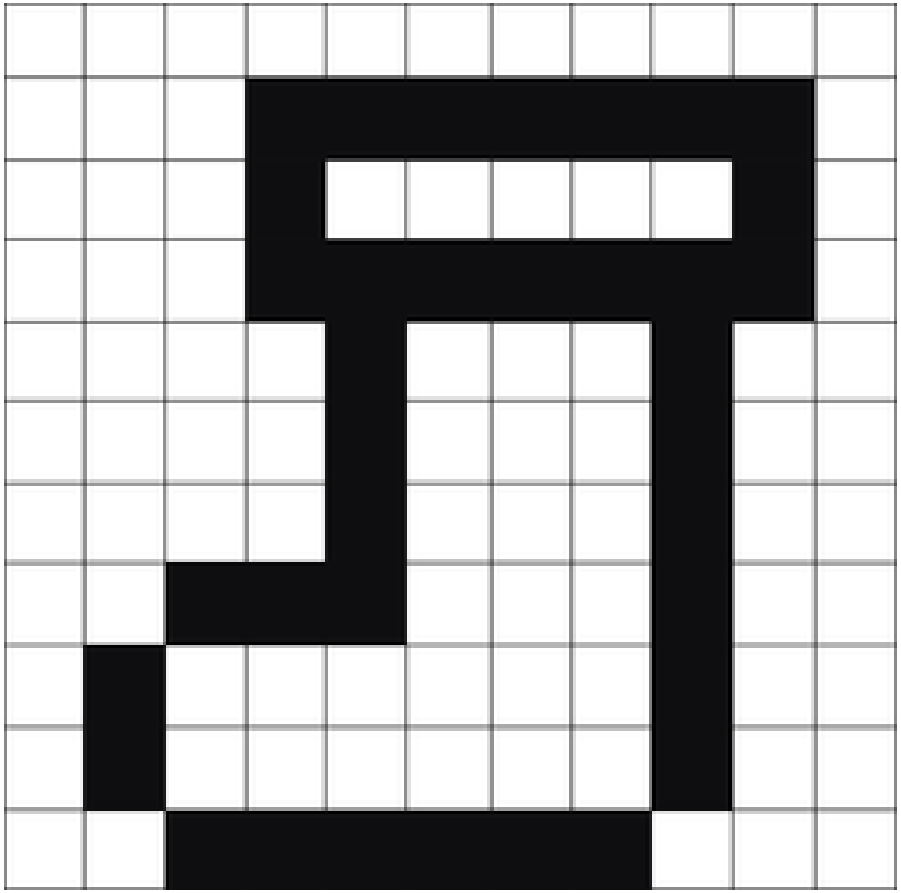
Bitmap

How is the image formed?

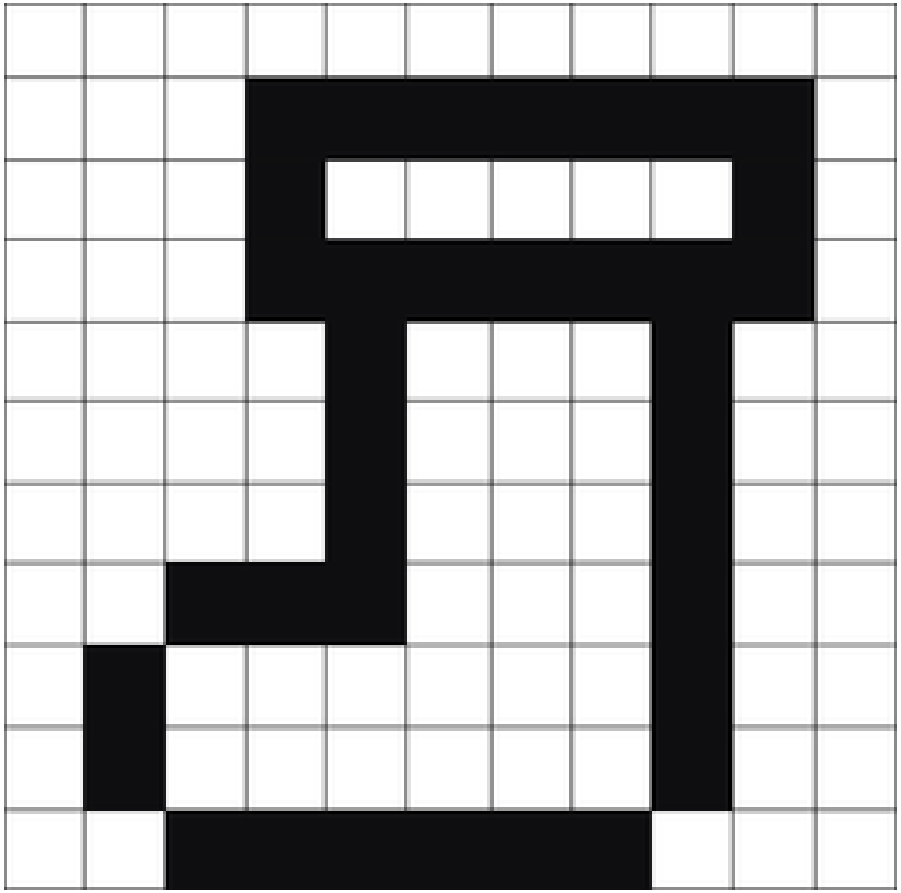
- In simple graphics (those without many colors), a byte can represent a single pixel.
- In a graphic representation called greyscale each pixel is a shade of grey from black at one extreme to white at the other.
- Allowing only one bit per pixel you can create two colours, black and white. These colours will be represented by the binary equivalent. So for the colour white, the binary representation would be '0'. And for Black the binary representation would be '1'. Here is an example:



Write down the binary code for that image



Write down the binary code for that image

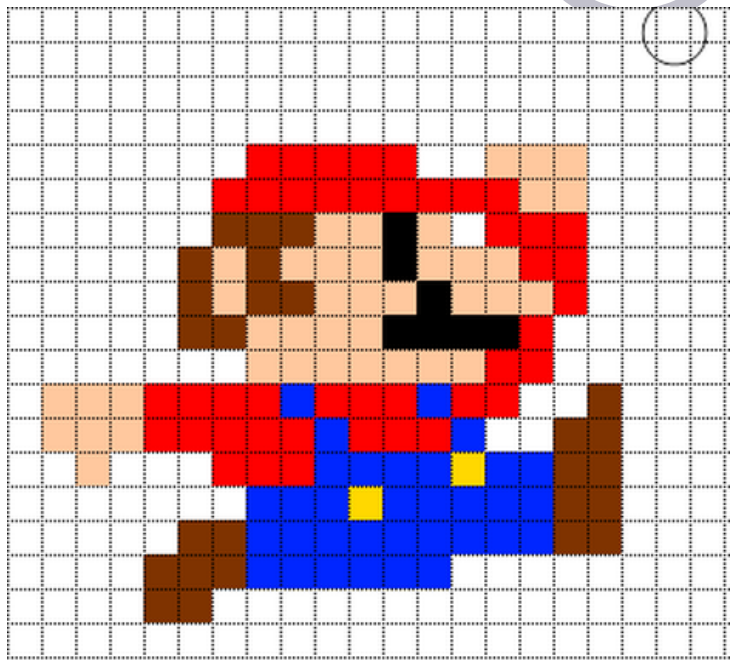


0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	1	1	0
0	0	0	1	0	0	0	0	0	1	0
0	0	0	1	1	1	1	1	1	1	0
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	1	0	0	0	1	0	0
0	0	1	1	1	0	0	0	1	0	0
0	1	0	0	0	0	0	0	1	0	0
0	1	0	0	0	0	0	0	1	0	0
0	0	1	1	1	1	1	1	0	0	0

- If you wanted a coloured picture, you would have to allow for more bits per pixel, thus increasing the file size.
- With **2** bits per pixel a maximum of **4** colours can be created.

00	11	10	01
White	Blue	Green	Red

- Modern video games and colorful graphics use several bytes for each pixel (Nintendo **64** uses eight bytes = **64** bits for each pixel to get a huge array of possible colors).



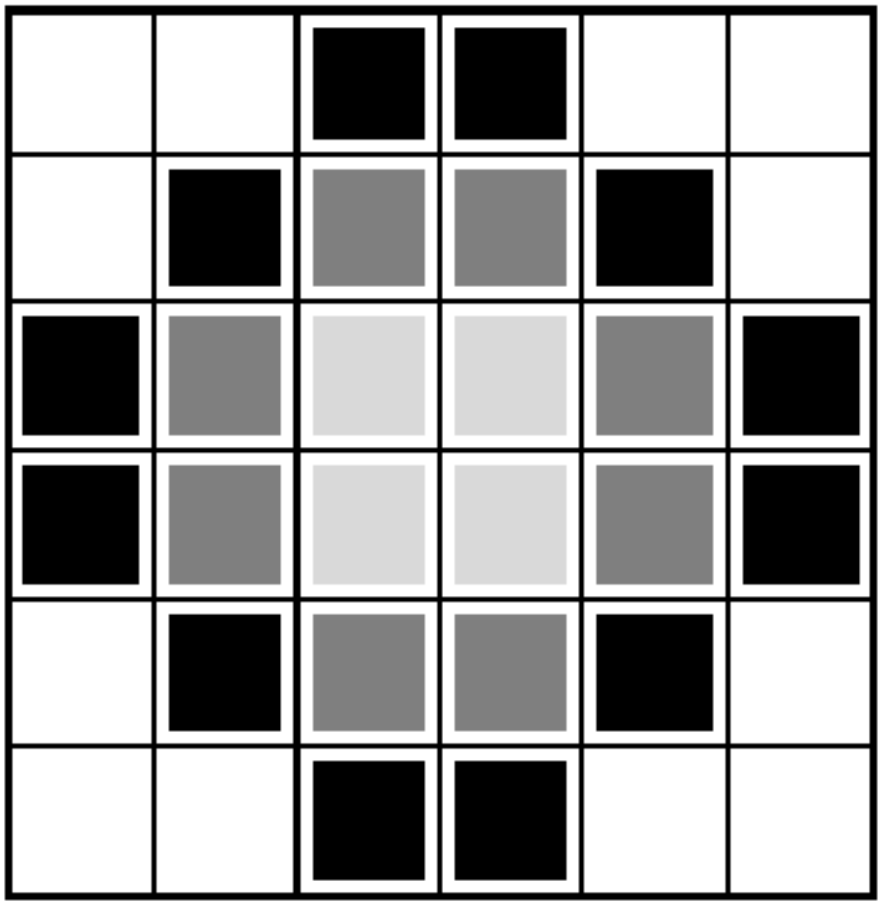
Example

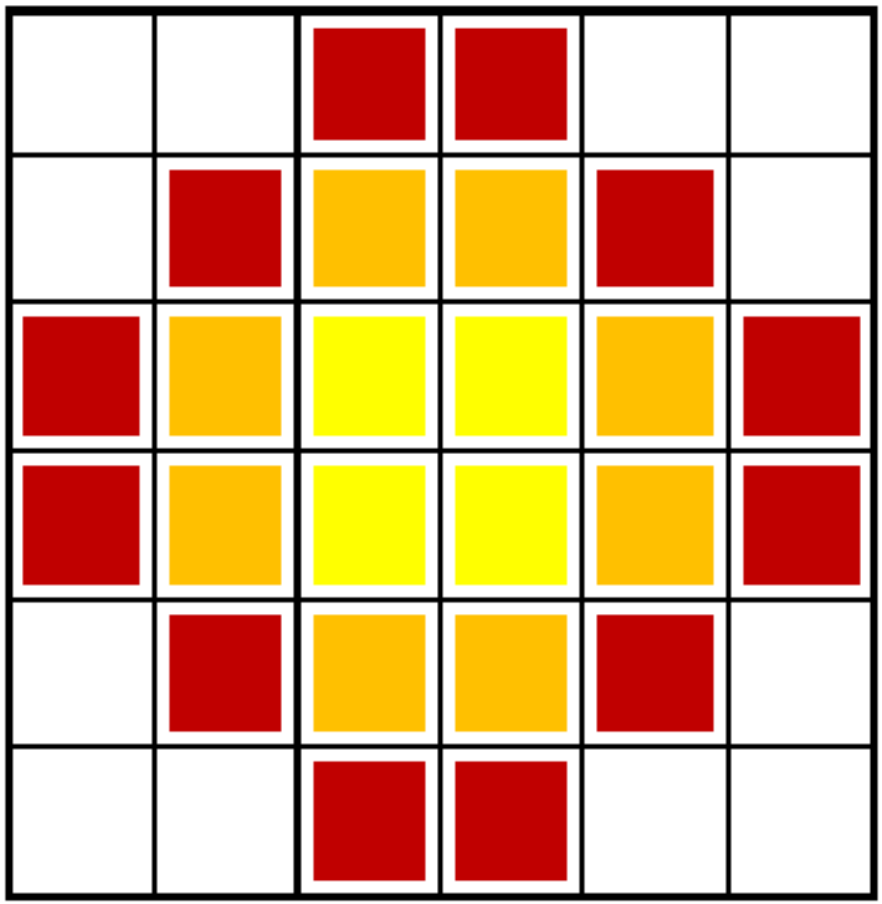
Using the Mario image, we can see that there are **7** different colours. To represent **7** different colours we will need to use **3** bits per pixel as a combination of **3** bits can offer up to **8** different states:

111	110	101	011	010	001	100	000
Black	White	Blue	Yellow	Red	Brown	Creme	

Question: How many colors can you encode with a **5**-bit word?

✎ Exercise 8. Write down the binary code for that image





Bitmap

Number of colours - Number of bits

Each time you increase the number of colours in an image you will need to increase the number of bits being used to represent each pixel. Below is a table illustrating how increasing the bits increases the number of available colour options.

Number of Colours	Bits Required
$2^1 = 2$	1
$2^2 = 4$	2
$2^3 = 8$	3
$2^4 = 16$	4
$2^5 = 32$	25

Effect on file size

Increasing the number of pixels

It is important to note that if you increase the resolution of an image it will increase its quality, however it will also increase the file size as there will be more pixels each represented by bits (data).

Increasing the number of colours

It is also important to note that increasing the number of colors will improve the quality of the image, however it will also increase the file size as each pixel will now be represented by more bits (data)

★ Picture and graphic data ★

Calculating file size

It is possible to calculate the size of the image file.

Step 1 Have an image to use, we will use this duck image

Step 2 Determine how many pixels the image has. This image has **16** $\times$ **18 = 288** Pixels

Step 3 Determine how many bits are being used to represent each pixel. This image has **4** colours so it is using **2** bits.

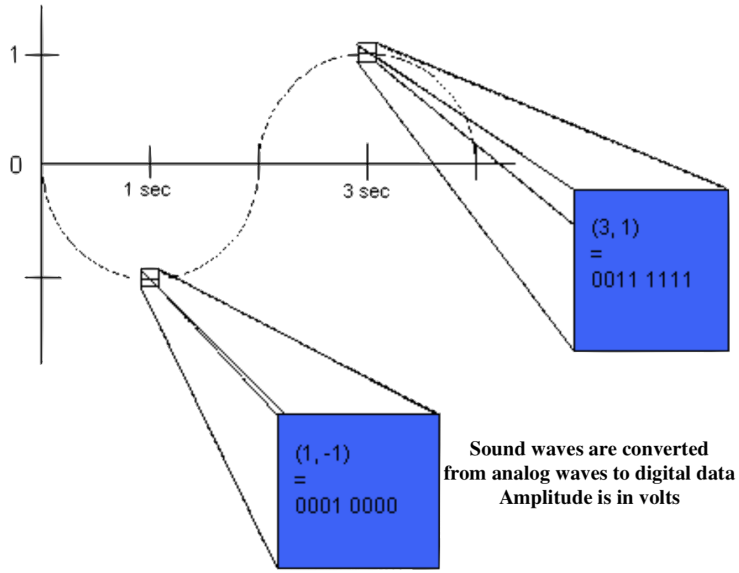
Step 4 Multiply the number of bits by the number of pixels.

288(Pixels) $\times$ **2(Bits) = 576Bits**

576 $\div$ **8 = 72 Bytes**

★ Sound data as bytes ★

- To convert an analog wave into digital, converters use a process called sampling. They sample the height of the sound wave at regular intervals of time, often small fractions of a second.
- If one byte is used to hold a single sample of an analog wave, then the wave can be one of **256** different heights (**0** being the lowest height and **255** being the highest).
- These heights represent the decibel level of the sound. Thus a spoken word might occupy several hundred bytes - each being a sample of the sound wave of the voice at a small fraction of a second.
- If these **100** bytes were sent to a computer's speaker, the spoken word would be reproduced.



Sound waves are converted from analog waves to digital data Amplitude is in volts

★ Number system ★

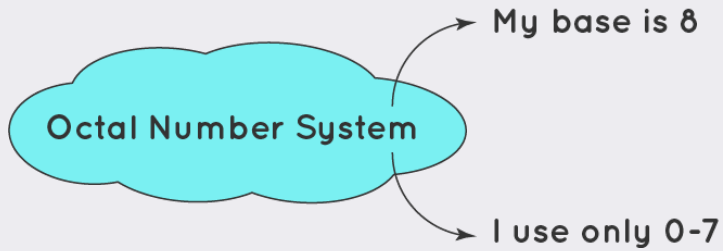
Octal Number System

Definition (Octal Number System)

- **Name** octo (Latin) $\Longrightarrow$ eight
- **Characteristics**
  - It has only only eight (**8**) digits from **0** to **7**. Every number (value) represents with **0**, **1**, **2**, **3**, **4**, **5**, **6** and **7** in this number system.
  - It is positional number system $(1743)_8 \neq (7314)_8$.
- The **base** of octal number system is **8**, because it has only **8** digits.

$$n = a_k 8^k + a_{k-1} 8^{k-1} + \cdots + a_1 8^1 + a_0 \, .$$

$a_0, a_1, \cdots, a_k \in \{0, 1, 2, 3, \cdots, 7\}$.



- Computer programmers often use the octal number system
- Octal numbers can be converted to binary numbers, binary numbers to octal numbers.

Example

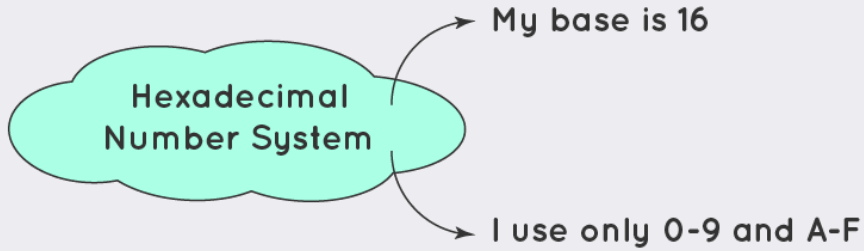
$(175)_8 = 1 \times 8^2 + 7 \times 8^1 + 5 \times 8^0 = 64 + 56 + 5 = (125)_{10}$.

✎ **Exercise 9.** What is the decimal expansion of the following numbers with octal expansion ?

a) $(7016)_8$ b) $(111)_8$

Definition (Hexadecimal Number System)

- **Name**
  - hexa (Latin) $\Longrightarrow$ six
  - decem (Latin) $\Longrightarrow$ ten
- **Characteristics**
  - It has has sixteen (**16**) alphanumeric values from **0** to **9** and ***A*** to ***F***. Every number (value) represents with **0**, **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, ***A***, ***B***, ***C***, ***D***, ***E*** and ***F***.
  - It is positional number system i $(A13D)_{16} \neq (3DA1)_{16}$.
- The **base** of hexadecimal number system is **16**, because it has **16** alphanumeric values.



$$n = a_k 16^k + a_{k-1} 16^{k-1} + \cdots + a_1 16^1 + a_0 .$$

$a_0, a_1, \cdots, a_k \in \{0, 1, 2, 3, \cdots, 9, A, B, \cdots, F\}$.

- Here ***A*** is **10**, ***B*** is **11**, ***C*** is **12**, ***D*** is **14**, ***E*** is **15** and ***F*** is **16**.
- Computer programmers often use the hexadecimal number system

★ Number system ★

Hexadecimal Number System

Example

$(2AE0B)_{16} = 2 \times 16^4 + 10 \times 16^3 + 14 \times 16^2 + 0 \times 16^1 + 11 \times 16^0 = (175627)_{10}$.

✎ **Exercise 10.** What is the decimal expansion of the following numbers with hexadecimal expansion?

a) $(1E5)_{16}$ b) $(1A2F)_{16}$

Why do we use a hexadecimal number system in computers?

Why do we use a hexadecimal number system in computers?

- The hexadecimal numbering system is often used by programmers to simplify the binary numbering system.
- Computers use binary numbering systems while humans use hexadecimal numbering systems to shorten binary and make it easier to understand.
- Writing out a binary number ends up with a long string of digits because there's only **2** digits, and hexadecimal are shorter numbers to write out for a human.
- It's easier to convert between binary and hexadecimal (for both humans and computers) than between binary and decimal, because every **4** bits corresponds exactly to **1** hex digit. In general it's easy to convert between a base number and that number to any integer power (e.g. **2** and **16**).
- Because the **4** bits that constitute a hex digit fits exactly into an **8**-bit byte (twice, as it happens).

Example

Do you want to write out **1101 0101 1101 0011 0010 0011 0011 0000** like that? ***D5D32330*** much easier to spell, and far less likely to evoke an error in writing or copying.

Hexadecimal are used in the following:

- **To define locations in memory**. Hexadecimals can characterise every byte as two hexadecimal digits only compared to eight digits when using binary.
- **To define colours on web pages**. Each primary colour – red, green and blue is characterised by two hexadecimal digits. The format being used is RRGGBB. RR stands for red, GG stands for green and BB stands for blue.
- **To represent Media Access Control (MAC) addresses**. MAC addresses consist of 12-digit hexadecimal numbers. The format being used is either MM:MM:MM:SS:SS:SS or MMMM-MMSS-SSSS. The first 6 digits of the MAC address represent the ID of the adapter manufacturer while the last 6 digits represent the serial number of the adapter.
- **To display error messages**. Hexadecimals are used to define the memory location of the error. This is useful for programmers in finding and fixing errors.

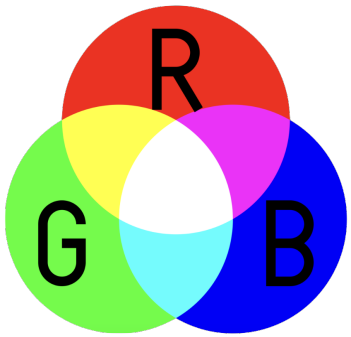
★ Hexadecimal Number System ★

Color Codes

- A very common use of hexadecimal numbers is for use of color codes. There are many different ways to define color. The two main ways are to use a hexadecimal value or an rgb value.

$$\text{rgb}(\mathbf{255}, \mathbf{255}, \mathbf{255}) \iff \#\mathit{FFFFFF}$$

- There are the three base colors red, green, and blue. In order to mix these colors in values of **0** – **255**. Hexadecimal applies the same way. ***FF*** is the hexadecimal representation of the number **255**. Hex color codes are just hexadecimal representations of an rgb color code.
- Below is a simple chart showing how the colors mix. Obviously there are many more than these **7** colors. Take note how the secondary colors are Cyan, Magenta and Yellow: the colors your printer uses.



★ Hexadecimal Number System ★

Color Codes

Since there are **256** values for each of the three colors (including **0**), we can make **256** × **256** × **256** different colors. That's **16, 777, 216** different colors! Obviously there are so many colors to make and we don't have place to display each of the over **16** million colors. For that reason we did the basic ROY G BIV rainbow below.

RGB	Hexadecimal	Name	Color
0,0,0	#000000	Black	
255,0,0	#FF0000	Red	
255,165,0	#FFA500	Orange	
255,255,0	#FFFF00	Yellow	
0,128,0	#008000	Green	
0,0,255	#0000FF	Blue	
75,0,130	#4B0082	Indigo	
238,130,238	#EE82EE	Violet	
255,255,255	#FFFFFF	White	

Uses of Octal and Hexa-decimal Numbers

- We will see that octal and hexa-decimal numbers can be easily converted to binary numbers and vice-versa. For this reason, octal and hexa-decimal numbers are useful in representing binary numbers in a compact form.
- Computer memory is often specified with numbers represented in groups of **8** bits which makes hexa-decimal system particularly useful at this stage.
- Both octal and hexa-decimal number system are used for input-output operations and for storing data and information in memory.
- The major disadvantage of Octal and Hexadecimal numbers system is that computer does not understand octal and hexadecimal numbers system directly (the computer uses only binary systems), so we need octal and hexadecimal to binary converter.

★ Representations of integers ★

Theorem

Let b *be a positive integer greater than* 1 *(*$b \in \mathbb{N}$ *and* $b > 1$*). Then if* n *is a positive integer, it can be expressed uniquely in the form*

$$n = a_k b^k + a_{k-1} b^{k-1} + \cdots + a_1 b^1 + a_0,$$

where k *is a nonnegative integer,* $a_0, a_1, \cdots, a_k$ *are nonnegative integers less than* b*, and* $a_k \neq 0$*.*

The representation of n is known as the base b expansion of n and is denoted by

$$(a_k a_{k-1} \cdots a_1 a_0)_b$$

Example

$$\begin{aligned}
267 &= 5 \times 7^2 + 3 \times 7^1 + 1 \times 7^0 && = (531)_7 \\
32,569 &= 4 \times 20^3 + 1 \times 20^2 + 8 \times 20^1 + 9 \times 20^0 && = (4,1,8,9)_{20} \\
32,569 &= 9 \times 60^2 + 2 \times 60^1 + 49 \times 60^0 && = (9,2,49)_{60} \\
10705679 &= 10 \times 16^5 + 3 \times 16^4 + 5 \times 16^3 + 11 \times 16^2 + 0 \times 16 + 15 && = (A35B0F)_{16}
\end{aligned}$$

★ Constructing Base b Expansions ★

Division Method

To construct the base b expansion of an integer n:

1st Divide n by b to obtain a quotient and remainder, that is

$$n = bq_0 + a_0 \qquad (0 \leq a_0 \leq b)$$

- The remainder, a_0 , is the rightmost digit in the base b expansion of n.

2nd Next, divide q_0 by b to obtain

$$q_0 = bq_1 + a_1 \qquad (0 \leq a_1 \leq b)$$

- The remainder, a_1, is the second digit from the right in the base b expansion of n.

nth Continue this process, successively dividing the quotients by b, obtaining additional base b digits as the remainders.

End This process terminates when we obtain a quotient equal to zero.

- It produces the base b digits of n from the right to the left.

The binary equivalent is found by successively dividing the integer part of the decimal number repeatedly by **2** (÷**2**), noting the remainders in reverse order from the least significant bit (LSB) to the most significant bit (MSB), until the value becomes "**0**" producing the binary equivalent.

Example

The octal representation of **1830** is $(3446)_8$ in fact:
Successively dividing **1830** by **8** gives:

$$\begin{aligned} 1830 &= 8 \times 228 + \underbrace{6}_{LSD} \\ 228 &= 8 \times 28 + 4 \\ 28 &= 8 \times 3 + 4 \\ 3 &= 8 \times 0 + \underbrace{3}_{MSD} \end{aligned}$$

The remainders are the digits from right to left yielding $(3446)_8$

✎ Exercise 11.

a. Find the octal expansion of $(12345)_{10}$.
b. Find the base **12** representation of $(19151)_{10}$.
c. Find the binary expansion of $(1830)_{10}$.
d. Find the hexadecimal expansion of $(177130)_{10}$.

★ Constructing Base b Expansions ★

Expansion Method

To find the binary representation of a small number, the following method is often easier than the above method
In expansion method the conversion of decimal numbers to their binary equivalents are shown with the help of the examples.

Convert the decimal numbers 77 to their binary equivalents?

Powers of 2

1024	512	256	128	64	32	16	8	4	2	1
2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

★ Constructing Base b Expansions ★

Expansion Method

1024	512	256	128	64	32	16	8	4	2	1
2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Convert the decimal numbers 77 to their binary equivalents?

- The fiven number **77** is less than **128** but greater than **64**. We therefore put **1** in the slot corresponding to **64** in the first row.

64	32	16	8	4	2	1
1						

Next, we subtract **64** from **77** and get **13** as remainder. **77 − 64 = 13**

- The remainder **13** is less than **16** and greater than **8**. So we put **1** in the slot corresponding to **8**

64	32	16	8	4	2	1
1			1			

we subtract **8** from **13**, this gives **13 − 8 = 5**.

★ Constructing Base b Expansions ★

Expansion Method

- The remainder **5** is less than **8** and greater than **4**. Hence we put **1** in the slot corresponding to **4** and subtracting **4** from **5** we get **1**.

64	32	16	8	4	2	1
1			1	1		

- Now, **1** is present in the right hand most slot of the first row. We, therefore, put **1** in the corresponding slot and fill all other slots with zeros.

64	32	16	8	4	2	1
1	0	0	1	1	0	1

Thus $\mathbf{77 = (1001101)_2}$.

★ Constructing Base b Expansions ★

Expansion Method

✎ **Exercise 12.** Convert $(675)_{10}$ to its Binary equivalent by using expansion method.

✎ **Exercise 13.** Complete the following table

Octal	Binary	Hexadecimal	Decimal
	100000000		
		50	
			10
			20
			40
			100

★ All possible numbers of n digits and base b without leading zeros ★

- **Binary number** $\underbrace{2\,2\,2\,\cdots\,2}_{n \text{ digits}} = \underbrace{2 \times 2 \times 2 \times \cdots \times 2}_{n \text{ times}} = 2^n$.
- **Octal number** $\underbrace{8\,8\,8\,\cdots\,8}_{n \text{ digits}} = \underbrace{8 \times 8 \times 8 \times \cdots \times 8}_{n \text{ times}} = 8^n$.
- **Decimal number** $\underbrace{10\,10\,10\,\cdots\,10}_{n \text{ digits}} = \underbrace{10 \times 10 \times 10 \times \cdots \times 10}_{n \text{ times}} = 10^n$.
- **Hexadecimal number** $\underbrace{16\,16\,16\,\cdots\,16}_{n \text{ digits}} = \underbrace{16 \times 16 \times 16 \times \cdots \times 16}_{n \text{ times}} = 16^n$.

Conclusion: Number of possible codes with n digits in base b is b^n.

✎ **Exercise 14.**

a. What is the largest number that could be written with **2** digits in base **10**?
b. How many different codes can be written with **2** digits in base **16**?
c. How many different codes can be written with **16** bits?
d. How many bits we need to write $(1234567)_8$ in binary number.

★ CONVERSION RULES of NUMBER SYSTEM ★

Number system	Base(Radix)	Used digits	Example
Binary	2	0,1	$(11110000)_2$
Octal	8	0,1,2,3,4,5,6,7	$(360)_8$
Decimal	10	0,1,2,3,4,5,6,7,8,9	$(240)_{10}$
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9, A,B,C,D,E,F	$(F0)_{16}$

A number can be converted from one number system to another number system using number system formulas:

Binary $\longrightarrow$ Others ; Octal $\longrightarrow$ Others ; Hexa $\longrightarrow$ Others ; Decimal $\longrightarrow$ Others

★ Binary to Others ★

Binary to Decimal

To convert a number from the binary to decimal system, we use the following steps.

1. Multiply each digit of the given number, starting from the rightmost digit, with the exponents of base $b = 2$.
2. The exponents should start with **0** and increase by **1** every time we move from right to left.
3. Simplify each of the above products and add them.

Example

	1	1	0	1	1	0	0	1
Positions	7	6	5	4	3	2	1	0
values of each digit	1×2^7 128	1×2^6 64	0×2^5 0	1×2^4 16	1×2^3 8	0×2^2 0	0×2^1 0	1×2^0 1
$1 + 8 + 16 + 64 + 128 = 217$								

Thus $(11011001)_2 = (217)_{10}$.

★ Binary to Others ★

Binary to Octal

.

Observation: Since a string of **3** bits can have **8** different permutations $8^1 = 2^3$, it follows that every **1** octal digit corresponds to **3** binary digits.

1. Group binary digits into sets of three, starting with the least significant (rightmost) digits.

■■■■■■■■■ = $\underbrace{■■■}\ \underbrace{■■■}\ \underbrace{■■■}$

2. Pad the most significant digits with zeros if necessary to complete a group of three.

■■■■■■■■■■ = $\underbrace{00■}\ \underbrace{■■■}\ \underbrace{■■■}\ \underbrace{■■■}$

3. For each group of **3** binary digits, use this table to lookup the corresponding octal digit,

■	*Binary*	000	001	010	011	100	101	110	111
●	*Octal*	0	1	2	3	4	5	6	7

■■■■■■■■■■ = $\underbrace{00■}_{●}\ \underbrace{■■■}_{●}\ \underbrace{■■■}_{●}\ \underbrace{■■■}_{●}$

Binary	000	001	010	011	100	101	110	111
Octal	0	1	2	3	4	5	6	7

Example

What is the octal expansion of the following binary number :

a. $(11010110)_2$ b. $(1110101110)_2$ c. $(001100101101110)_2$

Solution

a.

Binary	011	010	110	
Octal	3	2	6	326

$\Longrightarrow (11010110)_2 = (326)_8$.

b. $1110101110 = 001\ 110\ 101\ 110 = (1656)_8$.

c.

Binary	001	100	101	101	110
Octal	1	4	5	5	6

$\Longrightarrow (001100101101110)_2 = (14556)_8$.

★ Binary to Others ★

Binary to Hexadecimal

Observation: Since a string of **4** bits has **16** different permutations $\mathbf{16^1 = 2^4}$. Every **1** hexadecimal digit corresponds to **4** binary digits.

❶ Group binary digits into sets of **four**, starting with the least significant (rightmost) digits.

■■■■■■■■■■■ = ■■■ ■■■■ ■■■■

❷ Pad the most significant digits with zeros if necessary to complete a group of four.

■■■■■■■■■■■ = 0■■■ ■■■■ ■■■■

❸ For each group of **4** binary digits, use this table to lookup the corresponding hexadecimal digit.

■	*Binary*	0000	0001	0010	0011	0100	0101	0110	0111
▲	*Hexa*	0	1	2	3	4	5	6	7
■	*Binary*	1000	1001	1010	1011	1100	1101	1110	1111
▲	*Hexa*	8	9	*A*	*B*	*C*	*D*	*E*	*F*

■■■■■■■■■■■ = 0■■■ ■■■■ ■■■■

▲ ▲ ▲

Binary	0000	0001	0010	0011	0100	0101	0110	0111
Hexa	0	1	2	3	4	5	6	7
Binary	1000	1001	1010	1011	1100	1101	1110	1111
Hexa	8	9	*A*	*B*	*C*	*D*	*E*	*F*

Example

What is the octal expansion of the following number:

a. $(11010110)_2$ b. $(1111101101)_2$

Solution

a.

Binary	1101	0110	
Hexa	*D*	6	*D*6

$\Longrightarrow (11010110)_2 = (D6)_{16}$.

b. $1111101101 = 0011\ 1110\ 1101 = (3ED)_{16}$.

★ Octal to Others ★

Octal to Decimal

To convert a number from the octal to decimal system, we use the following steps.

1. Multiply each digit of the given number, starting from the rightmost digit, with the exponents of base $b = 8$.
2. The exponents should start with **0** and increase by **1** every time we move from right to left.
3. Simplify each of the above products and add them.

Example

	1	2	1
Positions	2	1	0
values of each digit	1×8^2 64	2×8^1 16	1×8^0 1
$121 = 64 + 16 + 1 = 81$			

Thus $(121)_8 = (81)_{10}$.

★ Octal to Others ★

Octal to Binary

Converting from octal to binary is as easy as converting from binary to octal. Simply look up each octal digit to obtain the equivalent group of three binary digits.

Octal	0	1	2	3	4	5	6	7
Binary	000	001	010	011	100	101	110	111

Example

What is the binary expansion of the number with octal expansion $(345)_8$?

Solution

★ Octal to Others ★

Octal to Binary

Converting from octal to binary is as easy as converting from binary to octal. Simply look up each octal digit to obtain the equivalent group of three binary digits.

Octal	0	1	2	3	4	5	6	7
Binary	000	001	010	011	100	101	110	111

Example

What is the binary expansion of the number with octal expansion $(345)_8$?

Solution

Octal	3	4	5	
Binary	011	100	101	011100101

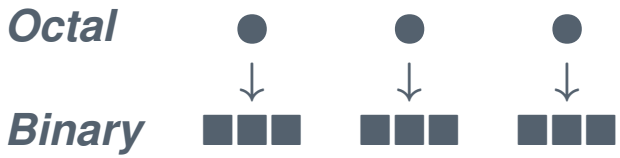
Thus $(345)_8 = (11100101)_2$.

★ Octal to Others ★

Octal to Hexadecimal

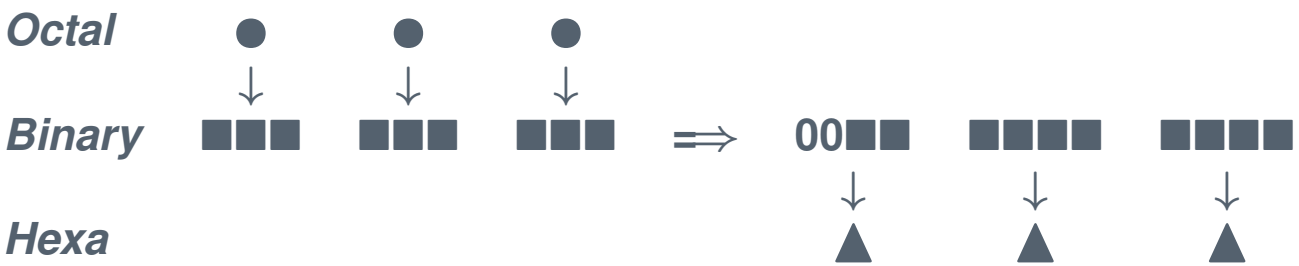
An easy way to convert from octal to hexadecimal:

1. First convert the octal number into binary .



2. Drop any leading zeros or pad with leading zeros to get groups of four binary digits:

3. Then, look up the groups in a table to convert to hexadecimal digits.



Example

a. What is the hexadecimal expansion of the number with octal expansion $(273)_8$?

b. Convert the octal number $(456)_8$ to a hexadecimal number.

Solution

Example

a. What is the hexadecimal expansion of the number with octal expansion $(273)_8$?

b. Convert the octal number $(456)_8$ to a hexadecimal number.

Solution

a.

Octal* − − > *Binary

Octal	2	7	3	
Binary	010	111	011	010111011

$\Longrightarrow$

Binary* − − > *Hexa

Binary	1011	1011	
Hexa	*B*	*B*	*BB*

Thus $(273)_8 = (010111011)_2 = (BB)_{16}$.

b.

Octal* − − > *Binary

Octal	4	5	6	
Binary	100	101	110	100101110

$\Longrightarrow$

Binary* − − > *Hexa

Binary	0001	0010	1110
Hexa	1	2	*D*

Thus $(456)_8 = (100101110)_2 = (12D)_{16}$.

★ Hexadecimal to Other ★

Hexadecimal to Decimal

To convert a number from hexadecimal to decimal system, we use the following steps.

1. Obtain the decimal equivalent of hexadecimal from the conversion table.

Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

2. Multiply each digit of the given number, starting from the rightmost digit, with the exponents of base $b = 16$.
2. The exponents should start with **0** and increase by **1** every time we move from right to left.
3. Simplify each of the above products and add them.

Example

	A	5
Positions	1	0
values of each digit	10×16^1	5×16^0
	160	5

$\Longrightarrow \quad (A5)_{16} = (165)_{10}$

★ Hexadecimal to Other ★

Hexadecimal to Binary

Converting from hexadecimal to binary is as easy as converting from binary to hexadecimal. Simply look up each hexadecimal digit to obtain the equivalent group of four binary digits.

Hexa	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Binary	**0000**	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**
Hexa	**8**	**9**	***A***	***B***	***C***	***D***	***E***	***F***
Binary	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**

Example

Hexadecimal	***A***	**2**	***D***	***E***	
Binary	**1010**	**0010**	**1101**	**1110**	**1010001011011110**

Thus $(A2DE)_{16} = (1010001011011110)_2$.

★ Hexadecimal to Other ★

Hexadecimal to Octal

An easy way to convert from hexadecimal to octal:

1. First convert the hexadecimal number into binary . For example,

Hexa	*A*	2	*B*	
Binary	1010	0010	1011	101000101011

2. Drop any leading zeros or pad with leading zeros to get groups of three binary digits (bits):

$$(A2B)_{16} = (101000101011)_2$$

3. Then, look up the groups in a table to convert to octal digits.

Binary	000	001	010	011	100	101	110	111
Octal	0	1	2	3	4	5	6	7

Binary	101	000	101	011
Octal	5	0	5	3

Thus $(A2B)_{16} = (101000101011)_2 = (5053)_8$.

★ Others to Decimal ★

Horner's Method

Converting binary, octal or hexadecimal to decimal can be done with repeated division.

1. Start the decimal result at **0**.
2. Remove the most significant octal digit (leftmost) and add it to the result.
3. If all octal digits have been removed, you're done. Stop.
4. Otherwise, multiply the result by **8**.
5. Go to step **2**.

$$\ll \text{(Leftmost Digit} \times \text{Decimal Result)} \times \boldsymbol{b} \gg$$

Octal Digits	Leftmost Digit	Operation	Decimal Result
			0
345	**3**	$(3+0)\times 8 = 24$	**24**
45	**4**	$(4+24)\times 8 = 224$	**224**
5	**5**	$5 + 224 = 229$	**229**

$$(345)_8 = \Big(\big((3+0)\times 8 + 4\big)\times 8\Big) + 5 = (229)_{10}\,.$$

★ Exercises ★

✎ **Exercise 15.** Find the decimal expansion of the followings by the method of your choice

a) $(C0CA)_{16}$ b) $(1011011)_2$ c) $(236)_8$ d) $(FFF)_{16}$.

✎ **Exercise 16.**

a. Find the octal and hexadecimal expansions of $(11111010111100)_2$.
b. Find the binary expansion of $(A8D)_{16}$.
c. What is the Octal expansion of the hexadecimal expansion of $(2AE08)_{16}$.

✎ **Exercise 17.** Using the Horner's method, find the decimal expansions of the followings:

a) $(203)_8$ b) $(101010)_2$ c) $(20A)_{16}$.

✎ **Exercise 18.** Complete the following table

Octal	Binary	Hex
12		
5655		
244371		

★ Bit ★

- a "bit" is atomic: the smallest unit of storage
- A bit stores just a **0** or **1**
- "In the computer it's all **0**'s and **1**'s" ... bits
- Anything with two separate states can store **1** bit
- In a chip: electric charge = **0**/**1**
- In a hard drive: spots of North/South magnetism = **0**/**1**
- A bit is too small to be much use
- Group **8** bits together to make **1** byte

Go Back

Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII
0	00000000	000	00	NUL	32	00100000	040	20	SP	64	01000000	100	40	@	96	01100000	140	60	`
1	00000001	001	01	SOH	33	00100001	041	21	!	65	01000001	101	41	A	97	01100001	141	61	a
2	00000010	002	02	STX	34	00100010	042	22	"	66	01000010	102	42	B	98	01100010	142	62	b
3	00000011	003	03	ETX	35	00100011	043	23	#	67	01000011	103	43	C	99	01100011	143	63	c
4	00000100	004	04	EOT	36	00100100	044	24	$	68	01000100	104	44	D	100	01100100	144	64	d
5	00000101	005	05	ENQ	37	00100101	045	25	%	69	01000101	105	45	E	101	01100101	145	65	e
6	00000110	006	06	ACK	38	00100110	046	26	&	70	01000110	106	46	F	102	01100110	146	66	f
7	00000111	007	07	BEL	39	00100111	047	27	'	71	01000111	107	47	G	103	01100111	147	67	g
8	00001000	010	08	BS	40	00101000	050	28	(	72	01001000	110	48	H	104	01101000	150	68	h
9	00001001	011	09	HT	41	00101001	051	29	)	73	01001001	111	49	I	105	01101001	151	69	i
10	00001010	012	0A	LF	42	00101010	052	2A	*	74	01001010	112	4A	J	106	01101010	152	6A	j
11	00001011	013	0B	VT	43	00101011	053	2B	+	75	01001011	113	4B	K	107	01101011	153	6B	k
12	00001100	014	0C	FF	44	00101100	054	2C	,	76	01001100	114	4C	L	108	01101100	154	6C	l
13	00001101	015	0D	CR	45	00101101	055	2D	-	77	01001101	115	4D	M	109	01101101	155	6D	m
14	00001110	016	0E	SO	46	00101110	056	2E	.	78	01001110	116	4E	N	110	01101110	156	6E	n
15	00001111	017	0F	SI	47	00101111	057	2F	/	79	01001111	117	4F	O	111	01101111	157	6F	o
16	00010000	020	10	DLE	48	00110000	060	30	0	80	01010000	120	50	P	112	01110000	160	70	p
17	00010001	021	11	DC1	49	00110001	061	31	1	81	01010001	121	51	Q	113	01110001	161	71	q
18	00010010	022	12	DC2	50	00110010	062	32	2	82	01010010	122	52	R	114	01110010	162	72	r
19	00010011	023	13	DC3	51	00110011	063	33	3	83	01010011	123	53	S	115	01110011	163	73	s
20	00010100	024	14	DC4	52	00110100	064	34	4	84	01010100	124	54	T	116	01110100	164	74	t
21	00010101	025	15	NAK	53	00110101	065	35	5	85	01010101	125	55	U	117	01110101	165	75	u
22	00010110	026	16	SYN	54	00110110	066	36	6	86	01010110	126	56	V	118	01110110	166	76	v
23	00010111	027	17	ETB	55	00110111	067	37	7	87	01010111	127	57	W	119	01110111	167	77	w
24	00011000	030	18	CAN	56	00111000	070	38	8	88	01011000	130	58	X	120	01111000	170	78	x
25	00011001	031	19	EM	57	00111001	071	39	9	89	01011001	131	59	Y	121	01111001	171	79	y
26	00011010	032	1A	SUB	58	00111010	072	3A	:	90	01011010	132	5A	Z	122	01111010	172	7A	z
27	00011011	033	1B	ESC	59	00111011	073	3B	;	91	01011011	133	5B	[	123	01111011	173	7B	{
28	00011100	034	1C	FS	60	00111100	074	3C	<	92	01011100	134	5C	\	124	01111100	174	7C	|
29	00011101	035	1D	GS	61	00111101	075	3D	=	93	01011101	135	5D	]	125	01111101	175	7D	}
30	00011110	036	1E	RS	62	00111110	076	3E	>	94	01011110	136	5E	^	126	01111110	176	7E	~
31	00011111	037	1F	US	63	00111111	077	3F	?	95	01011111	137	5F	_	127	01111111	177	7F	DEL

★ ASCII - Character Table ★

	0_	1_	2_	3_	4_	5_	6_	7_
_0	NUL	DLE	SP	0	@	P	'	p
_1	SOH	DC1	!	1	A	Q	a	q
_2	STX	DC2	"	2	B	R	b	r
_3	ETX	DC3	#	3	C	S	c	s
_4	EOT	DC4	$	4	D	T	d	t
_5	ENQ	NAK	%	5	E	U	e	u
_6	ACK	SYN	&	6	F	V	f	v
_7	BEL	ETB	'	7	G	W	g	w
_8	BS	CAN	(	8	H	X	h	x
_9	HT	EM	)	9	I	Y	i	y
_A	LF	SUB	*	:	J	Z	j	z
_B	VT	ESC	+	;	K	[	K	}
_C	FF	FS	,	<	L	\	l	|
_D	CR	GS	-	=	M	]	m	}
_E	SO	RS	.	>	N	^	n	~
_F	SI	US	/	?	O	_	o	DEL

- **ASCII** stands for *American Standard Code for Information Interchange*.
- Used for transferring codes of characters between **CPU** and **I/O** devices.
- Supports **8**-bit representation of character encodings.

Go Back

★ Comparison of Hexadecimal, Octal, and Binary ★

TABLE 1 Hexadecimal, Octal, and Binary Representation of the Integers 0 through 15.

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

- Each octal digit corresponds to a block of **3** binary digits.
- Each hexadecimal digit corresponds to a block of **4** binary digits.
- So, conversion between binary, octal, and hexadecimal is easy.