

Number Systems

Lecture 2 – Intro to Info. & Comm.
Technologies

Number Systems

Number System Name	Number System and Description
Decimal Number System	Base 10 Digits used: 0 to 9
Binary Number System	Base 2 Digits used : 0, 1
Octal Number System	Base 8. Digits used : 0 to 7
Hexadecimal Number System	Base 16. Digits used: 0 to 9, Letters used : A- F

Binary Number System

- Each digit of a binary number is called a bit. For example the number $(1010)_2$ has 4 bits.
- The more the number of bits, the more the total numbers that can be represented. For example if we use 8 bits, then total numbers that can be represented by 8 bits is $2^8=256$.

Binary Number System (contd)

- Suppose we have 8 bits. We know that we can store 256 numbers in these 8 bits.
- If we store only positive numbers, then the smallest number that we can store is 0, and the largest number that we can store is $2^8 - 1 = 255$.
- 0 in binary (using 8 bits) is $(00000000)_2$ and 255 in binary is $(11111111)_2$

How to compute the decimal value of a binary number

- Technique
 - Multiply each bit by 2^n , where n is the position of the digit. So, 2^n is the value of the bit.
 - The weight is the position of the bit, starting from 0 on the right
 - Add the results

How to compute the decimal value of a binary number

- **Example**
- Binary Number: 10101
- Calculating Decimal Equivalent:

Step	Binary Number	Decimal Number
Step 1	10101	$(1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$
Step 2	10101	$16 + 0 + 4 + 0 + 1$
Step 3	10101	21

How to compute the decimal value of a binary number (shortcut)

Suppose we want to find the decimal value of binary number 10101011.

128	64	32	16	8	4	2	1
1	0	1	0	1	0	1	1

Simply add the decimal values where the corresponding bit is 1.

Hexadecimal Number System

The **hexadecimal number system** has 16 digits.

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Examples of hexadecimal numbers:

1. 1F
2. 1AB
3. 100
4. 54

How to compute the decimal value of a hexadecimal number

- Technique
 - Multiply each digit by 16^n , where n is the position of the digit. So, 16^n is the value of the digit.
 - positions of the bit start from 0 on the right
 - Add the results

How to compute the decimal value of a hexadecimal number

Step	Hex Number	Decimal Number
Step 1	19FDE	$(1 \times 16^4) + (9 \times 16^3) + (F \times 16^2) + (D \times 16^1) + (E \times 16^0)$
Step 2	19FDE	$(1 \times 16^4) + (9 \times 16^3) + (15 \times 16^2) + (13 \times 16^1) + (14 \times 16^0)$
Step 3	19FDE	$65536 + 36864 + 3840 + 208 + 14$
Step 4	19FDE	106462

Decimal Number Systems

- One of the most common number systems.
- Has 10 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Decimal to Binary

- Technique
 - Divide by two, keep track of the remainder

$$125_{10} = ?_2$$

2		125	
		62	1
2		31	0
		15	1
2		7	1
		3	1
2		1	1

$$125_{10} = 1111101_2$$

Decimal to Hexadecimal

$$1234_{10} = ?_{16}$$

$$\begin{array}{r|l} 16 & 1234 \\ \hline 16 & 77 \\ \hline 16 & 4 \\ \hline & 0 \end{array}$$

$$\begin{array}{l} 2 \\ 13 = D \\ 4 \end{array}$$

$$1234_{10} = 4D2_{16}$$

Converting Hexadecimal to Binary and Vice Versa

Hexadecimal to Binary

- We know that the largest digit in hexadecimal is F, and this digit requires 4 bits for binary representation.
- So, find 4-digit binary value of each hexadecimal digit.
- The binary equivalent of the hexadecimal number is simply the concatenation of 4-digit binary values of all digits in order.

Conversion Chart

Decimal	Binary	Hex
00	0000	0
01	0001	1
02	0010	2
03	0011	3
04	0100	4
05	0101	5
06	0110	6
07	0111	7
08	1000	8
09	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Hexadecimal to Binary

Example

$$10AF_{16} = ?_2$$

1	0	A	F
↓	↓	↓	↓
0001	0000	1010	1111

$$10AF_{16} = 0001000010101111_2$$

Binary to Hexadecimal

- Make Quadruplets of bits from right to left.
- Find the hexadecimal value of each quadruplet.
- The final hexadecimal number is the concatenation of hex digits in order.

Binary to Hexadecimal

Example:

$$1010111011_2 = ?_{16}$$

10	1011	1011
↓	↓	↓
2	B	B

$$1010111011_2 = 2BB_{16}$$

Representing Signed Binary Numbers in Computer Memory

Signed Magnitude Representation

- Signed binary numbers can be represented in computer memory by using the most significant bit as a signed bit. This representation is also called sign & magnitude representation.
- If the signed bit is on, the number is negative, and if the signed bit is off, the number is positive.
- **What is the total number of values that can be stored using this signed representation?**

Signed Magnitude Representation

Examples:

- $(01101101)_2 = +(109)_{10}$
- $(11101101)_2 = -(109)_{10}$
- $(00101011)_2 = +(43)_{10}$
- $(10101011)_2 = -(43)_{10}$

Signed Magnitude Representation

- Range of values for positive number:
1 to 127
- Range of values for negative numbers:
-127 to -1.

Problem:

- Positive zero and negative zero.
- So, The range for positive numbers becomes +0 to +127, and for negative numbers the range becomes -127 to -0.

Converting a Binary Number Represented by Signed Magnitude into Decimal.

- Suppose there are N bits in the given binary number represented by signed magnitude form. The most significant bit will be used to represent the sign.
- Simply find the decimal equivalent of remaining $N-1$ bits, and put $-$ sign if the sign bit is 1 else put $+$.
- **Example:** (11010001)
Find the decimal of (1010001) which is 81. Since the sign bit is 1, so the number is -81.

1's Complement

- To represent a negative number in 1's complement, first find its positive binary counterpart.
- Then inverse the bits.
- **For example:** Represent -2 using 1 byte in 1's complement.
 - ❑ 2 = 00000010
 - ❑ -2 = 11111101

Note: Applying 1's complement again on the binary of a negative number (represented by 1's complement) will convert it into the positive number.

Range of 1's Complement Representation

- The range of signed numbers using one's complement is $-(2^{N-1} - 1)$ to $(2^{N-1} - 1)$ with ± 0 .
- **For example:** if we use 8 bits, then range is -127_{10} to $+127_{10}$ with zero being either 00000000 (+0) or 11111111 (-0).

Converting a Binary Number Represented by 1's Complement into Decimal.

- Suppose there are N bits in the given binary number represented by 1's complement.
- If the most significant bit is 0, then the number is positive, and simply find its decimal equivalent.
- If the most significant bit is 1, then the number is negative. So, first apply 1's complement (to convert the negative binary number into positive binary number), then find the decimal equivalent and put – sign.
- **Example:** (11010011)
 1. Finding 1's complement since msb is 1: 00101100.
 2. Now find the decimal equivalent: 44
 3. Put negative sign. (-44)

2's complement

- First find 1's complement.
- Now add 1 to the 1's complement.
- It does not have positive and negative zero problem.
- **For example:** Represent -2 using 1 byte in 2's complement.
 - ☐ 2 = 00000010
 - ☐ 1's complement = 11111101
 - ☐ 2's complement = 11111101 + 1 = 11111110
 - ☐ -2 = 11111110

Note:

Applying 2's complement again on the binary of a negative number (represented by 2's complement) will convert it into the binary of the same positive number. For example, apply 2's complement on the binary of -2 (calculated above) and see the result.

Range of 2's Complement Representation

- The range of signed numbers using two's complement is $-(2^{N-1})$ to $(2^{N-1} - 1)$. There is only one representation of 0 in this representation.
- **For example:** if we use 8 bits, then range is -128_{10} to $+127_{10}$ with zero being 00000000.

Converting a Binary Number Represented by 2's Complement into Decimal.

- Suppose there are N bits in the given binary number represented by 2's complement.
- If the most significant bit is 0, then number is positive and simply find its decimal equivalent.
- If the most significant bit is 1, then number is negative. So, first apply 2's complement (to convert it into positive binary number), then find the decimal equivalent of positive binary number and put – sign.
- **Example:** (11010011)
 1. Finding 2's complement since msb is 1: 00101101.
 2. Now find the decimal equivalent: 45
 3. Put negative sign. (-45)

Data Representation in Computers

What is Data

Data is simply
any **numbers**, **letters** or **symbols** that can be
entered into a computer system.

Data values **don't have any meaning** unless we
put them into **context**

Information = **Data** + **Context**

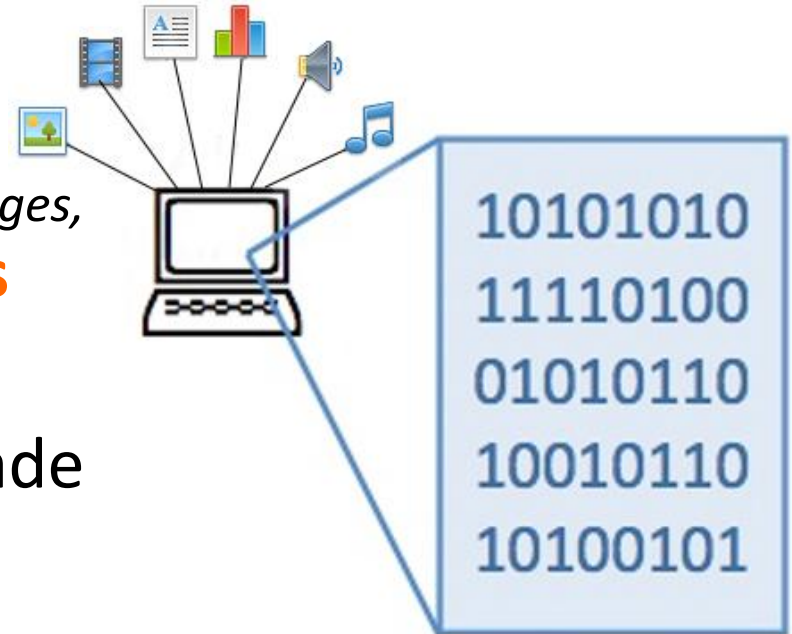
Computers Process Data

- Computers are used to process all types of information in a broad spectrum fields.
 - **Numeric data** consisting of Integers and real numbers are used in programs calculating payroll. We typically perform **arithmetic operations** on numeric data.
 - **Strings** of **alphabets** and numbers (**Alphanumeric Data**) are processed in customer record keeping systems.
 - **Multimedia** content including images, sound and text are frequently used in a large collection of application areas.
 - **Signals** representing various types of information like temperature, pressure, presence or absence of objects etc. are processed by computers in Robotics, IoT, monitoring and control applications.

How is Data Actually Stored in Computer

Everything that is stored and processed inside a computer *(all data, information, instructions, files, images, etc.)* is stored as **Binary Numbers**

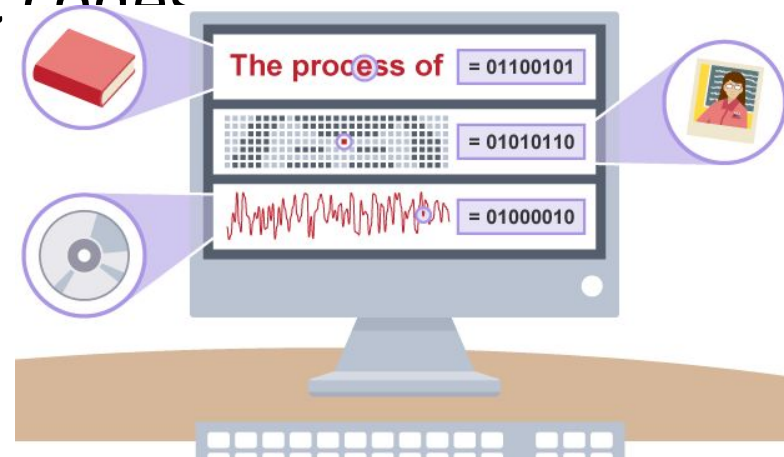
Digital computers have been made such that all data and instructions(program) for processing must be stored in computers memory before processing.



How to store *text* and *pictures* as numbers?

- The solution is to use **numeric codes**:
 - Different **letters** in a text document are given different numeric codes
 - Different **pixels** (colored dots) in an image are given different numeric codes
 - Different **sounds** in a music file are given different numeric codes

Everything is numbers!



Memory Measuring Units

(As viewed by computer scientists)

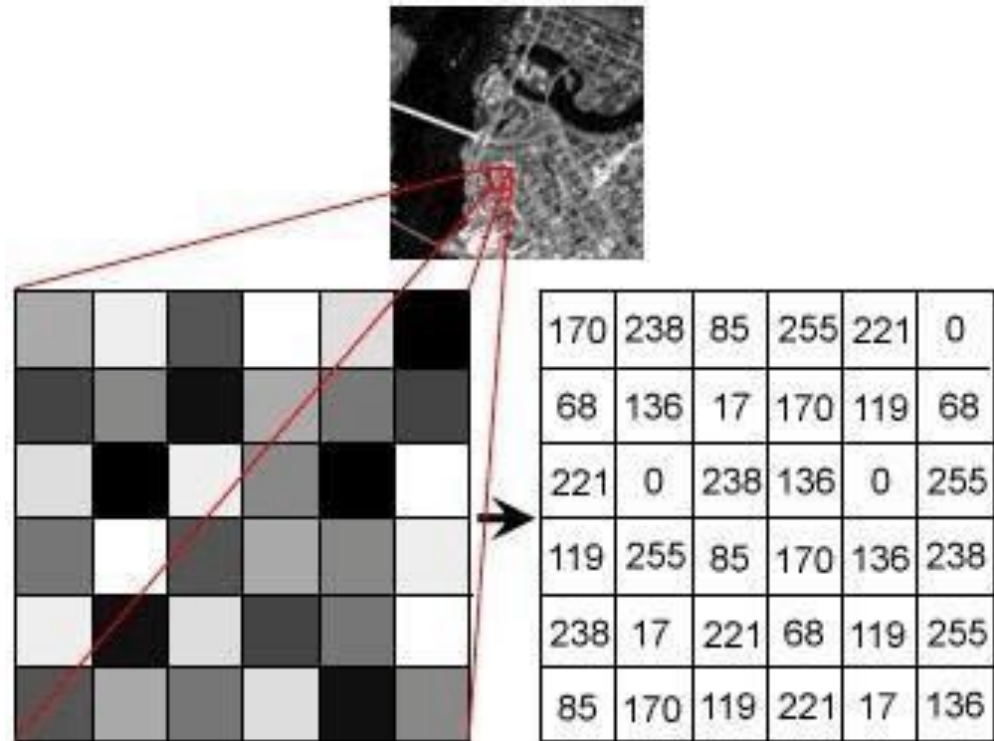
- Bits can be grouped together to make them easier to work with. A group of 8 bits is called a **byte**.

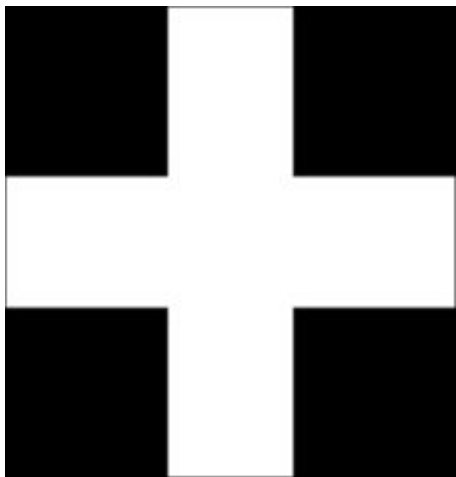
UNIT	ABBREVIATION	STORAGE
Bit	B	Binary Digit, Single 1 or 0
Nibble	-	4 bits
Byte/Octet	B	8 bits
Kilobyte	KB	1024 bytes
Megabyte	MB	1024 KB
Gigabyte	GB	1024 MB
Terabyte	TB	1024 GB
Petabyte	PB	1024 TB
Exabyte	EB	1024 PB
Zettabyte	ZB	1024 EB
Yottabyte	YB	1024 ZB

Storage units (www.byte-notes.com)

Most computers can process millions of bits every second. A hard drive's storage capacity is measured in gigabytes or terabytes. RAM is often measured in megabytes or gigabytes.

How Images are Stored in Memory

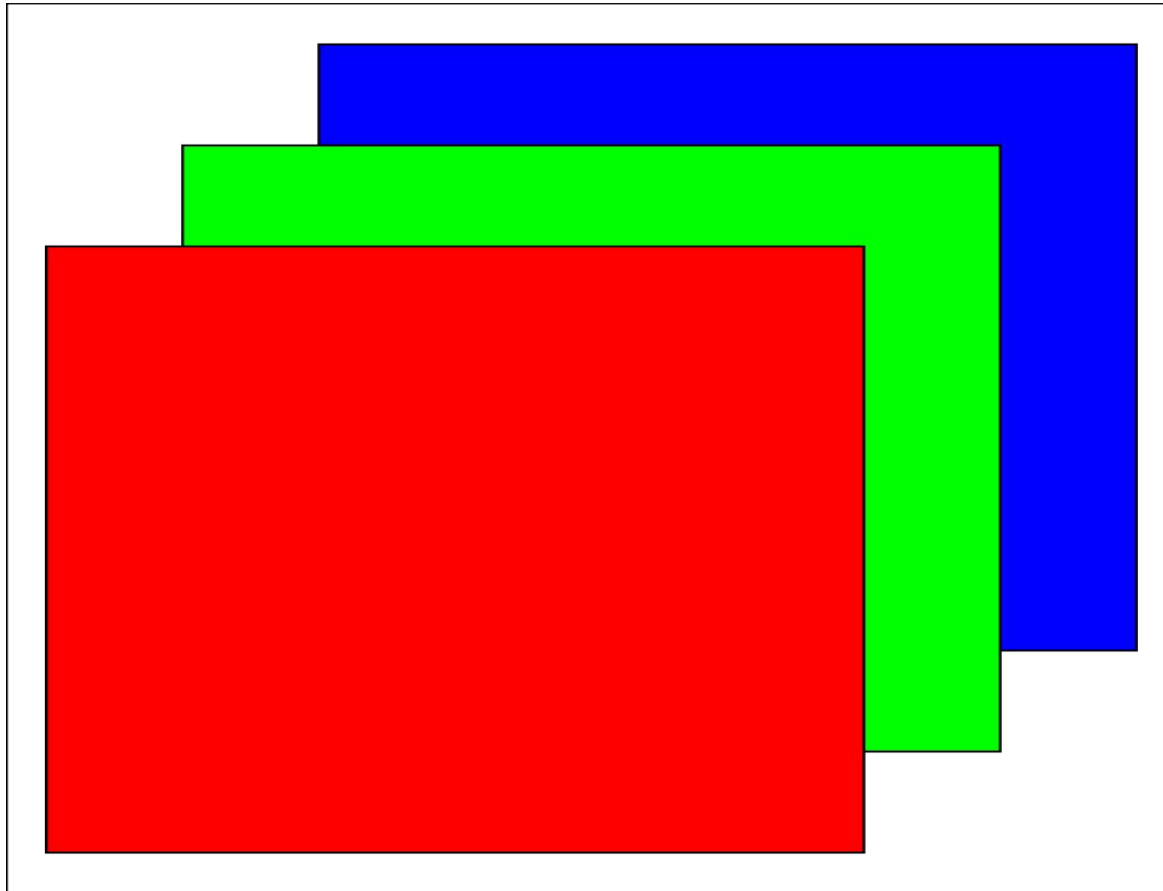




0	0	255	0	0
0	0	255	0	0
0	0	255	0	0
255	255	255	255	255
0	0	255	0	0
0	0	255	0	0
0	0	255	0	0

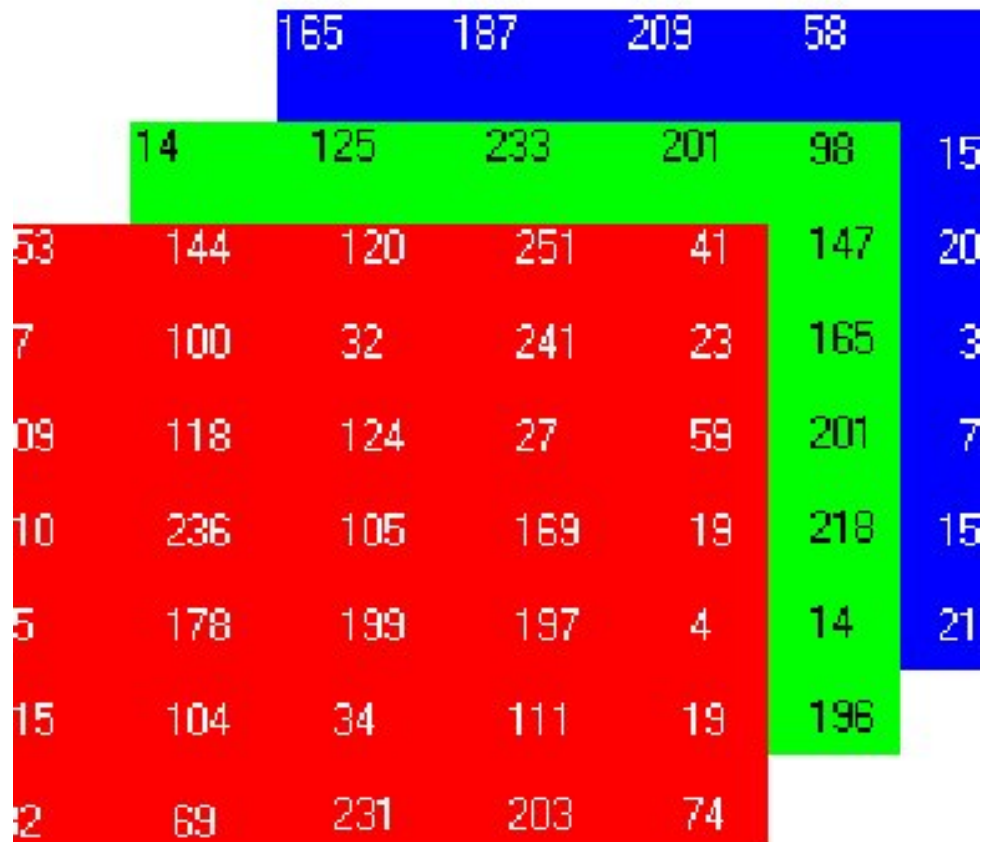
How To Represent Color Images in Memory

RGB



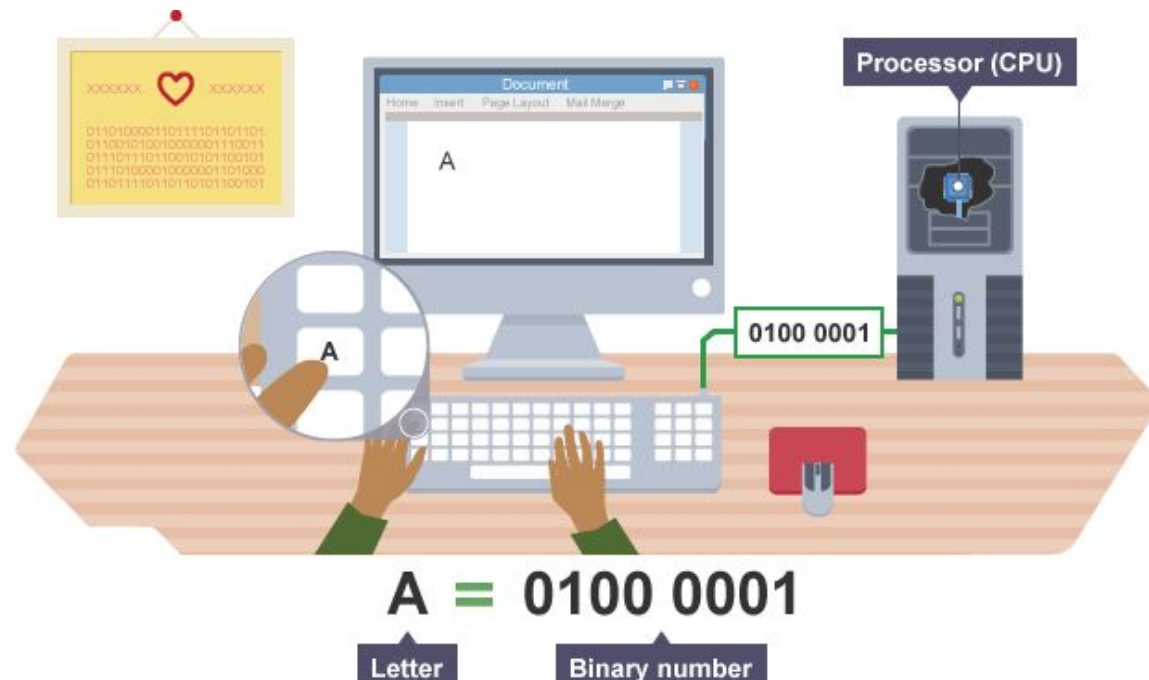
Three Separate Matrices for R, G and B, respectively.

How To Represent Color Images in Memory



How can we represent a character?

- IDEA.
 - Assign numeric codes to characters and represent each character in a Byte using it's numeric code.
 - Can we assign numeric codes of our choice to each character?. What might be a problem with this approach?



How can we represent a character?

- **IDEA**
- Create a Standard coding scheme so that information can be easily shared between devices from different vendors.
- Standard Codes
 - ASCII (American Standard Code for Information Interchange)
 - Unicode
 - Unicode Transformation Format(UTF) UTF-8, UTF-16
 - ANSI Character Set

ASCII Character Encoding

Letter
 Number
 Punctuation
 Symbol
 Other
 undefined

ASCII (1977/1986)

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
	0000 0	0001 1	0002 2	0003 3	0004 4	0005 5	0006 6	0007 7	0008 8	0009 9	000A 10	000B 11	000C 12	000D 13	000E 14	000F 15
1_	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
	0010 16	0011 17	0012 18	0013 19	0014 20	0015 21	0016 22	0017 23	0018 24	0019 25	001A 26	001B 27	001C 28	001D 29	001E 30	001F 31
2_	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
	0020 32	0021 33	0022 34	0023 35	0024 36	0025 37	0026 38	0027 39	0028 40	0029 41	002A 42	002B 43	002C 44	002D 45	002E 46	002F 47
3_	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
	0030 48	0031 49	0032 50	0033 51	0034 52	0035 53	0036 54	0037 55	0038 56	0039 57	003A 58	003B 59	003C 60	003D 61	003E 62	003F 63
4_	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	0040 64	0041 65	0042 66	0043 67	0044 68	0045 69	0046 70	0047 71	0048 72	0049 73	004A 74	004B 75	004C 76	004D 77	004E 78	004F 79
5_	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
	0050 80	0051 81	0052 82	0053 83	0054 84	0055 85	0056 86	0057 87	0058 88	0059 89	005A 90	005B 91	005C 92	005D 93	005E 94	005F 95
6_	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
	0060 96	0061 97	0062 98	0063 99	0064 100	0065 101	0066 102	0067 103	0068 104	0069 105	006A 106	006B 107	006C 108	006D 109	006E 110	006F 111
7_	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL
	0070 112	0071 113	0072 114	0073 115	0074 116	0075 117	0076 118	0077 119	0078 120	0079 121	007A 122	007B 123	007C 124	007D 125	007E 126	007F 127

ASCII to Binary 0s and 1s

Converting the text "hope" into binary

Characters:	h	o	p	e
ASCII Values:	104	111	112	101
Binary Values:	01101000	01101111	01110000	01100101
Bits:	8	8	8	8

Recommended

- <https://www.youtube.com/watch?v=1GSjbWt0c9M>
- <https://www.khanacademy.org/computing/computer-science/how-computers-work2/v/khan-academy-and-codeorg-introducing-how-computers-work>
- <https://www.youtube.com/watch?v=ptzGl9VaZmQ>

Activity