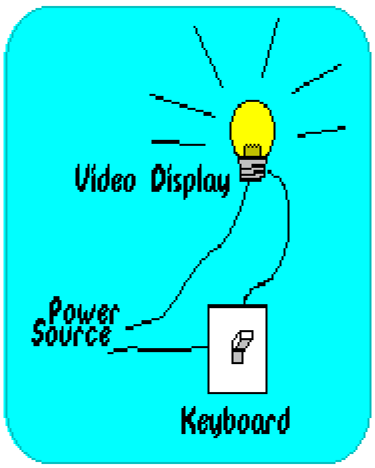
The World's Simplest Computer
Table of Contents

Return to SunShine Table of Contents

Last Page
Next Page

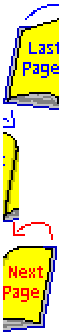
The World's Simplest Computer



The world's simplest digital electronic computer can handle one piece of information.

It has a keyboard with one switch, and it determines if that switch is on or off. If the switch is on, it sends electrical energy to the video display to shine a light. If the switch is off, then it cuts off this energy leaving darkness.

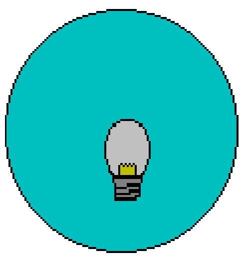
Other digital computers work the same way. They just combine switches in sophisticated ways.

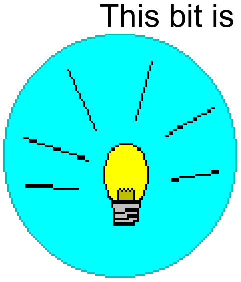


A Bit Is Like a Bulb

A bit is one piece of information and is the smallest unit of data which a computer can hold. Millions of bits can fit on a single computer chip. The actual physical qualities of a bit inside a computer can vary, but a bit can be, for example, a surge of electrical energy or a magnetic field.

Imagine that a lit light bulb represents a bit which is on. An unlit light bulb represents a bit which is off. Imagine that the computer can detect whether the bit is on or off.

This bit is off:  This bit is on:



Last Page
Next Page

Bits Can Be Many Things...

/ / / /

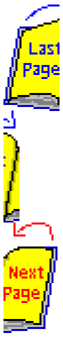
Off/On No/Yes False/True 0/1

/ / / /

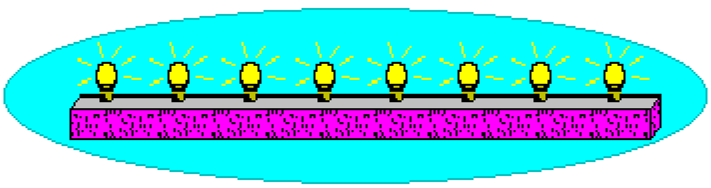
Male/Female Pepper/Salt Night/Day None/One

Someone who does not know about bits/Someone who is learning about bits

...It Is Up to the Programmer!



Eight Bits Make a Byte

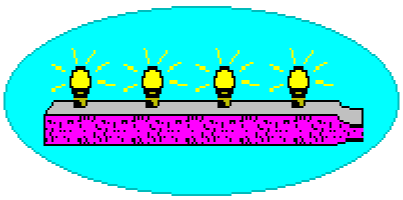


Other sized bytes are possible, but this is the generally accepted size for IBM compatible personal computers. It is easy to remember the difference between bits and bytes. The word *byte* is longer than the word *bit* and bytes are longer than bits.

Byte is pronounced *bite.* It is spelled with the character *y* instead of the character *i* to help distinguish the word *byte* from the word *bit*.

A lot of word plays are done on *byte*. Prepare a groan for the next page.

Half of a Byte Is a Nibble

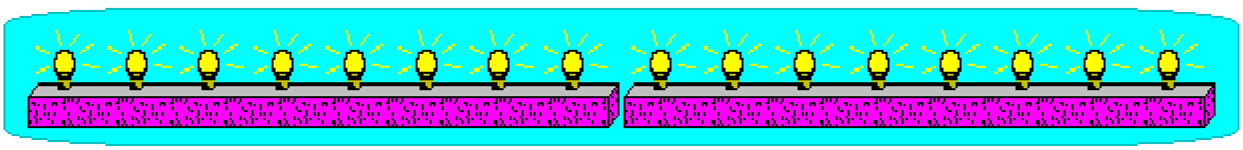


Really!

Nibble is sometimes spelled *nybble* to be consistent with the spelling of *byte*.

Programmers like nibbles because they make it easier to translate from the binary numerical system to the hexidecimal numerical system.   You may not want to know the details about why this is true, but it is fun to know that half a byte is a nibble!

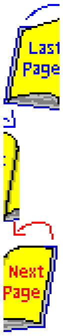
A Word Is One or More Bytes



The size of a word is set by the manufacturer of a computer and represents how many bits a computer can process at the same time.

The current trend towards *32-bit computing* means that 32 bits, or four bytes, is the word size.

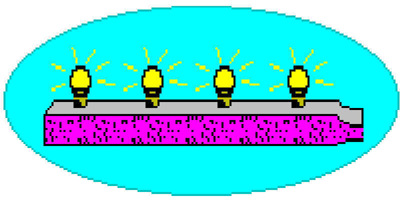
The above illustration shows a 16-bit, or two-byte, word.



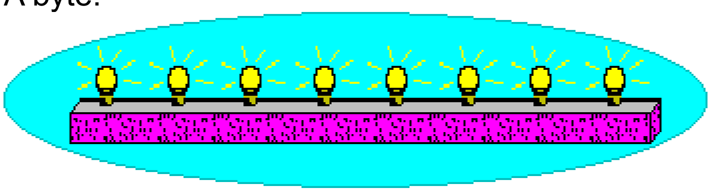
Review

A bit: 

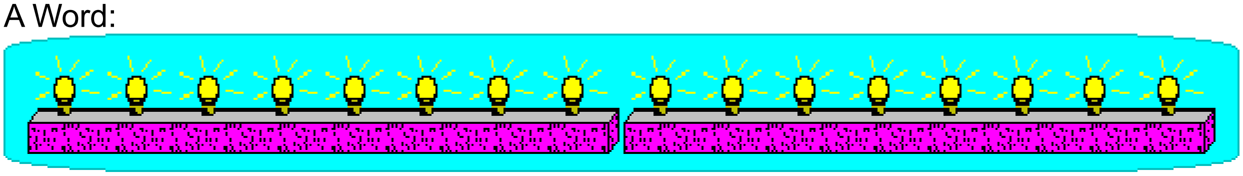
A nibble:



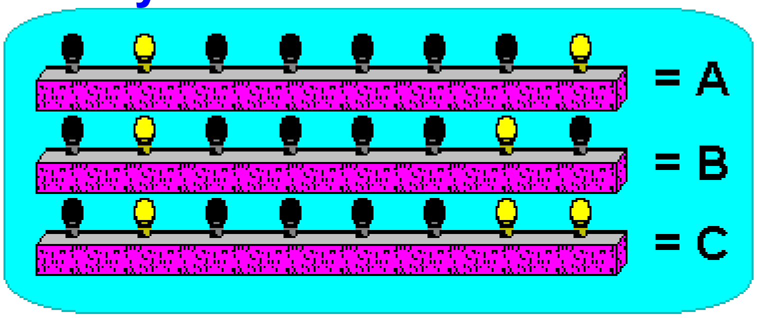
A byte:



A Word:



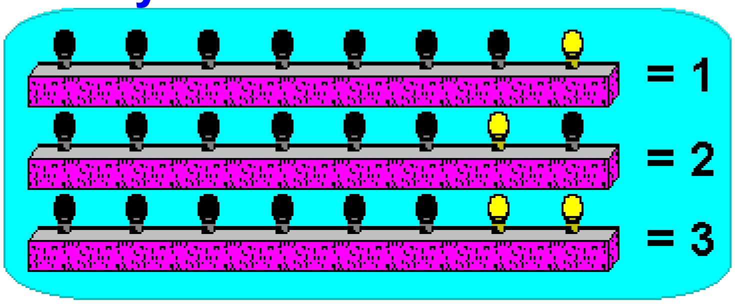
Bytes Are Like Characters



Characters make up real words and bytes make up computer words. However, there is another way that bytes are like characters. A computer stores characters in bytes. The combination of *on* bits in a byte signifies to the computer which character it represents.

Note: There is a current movement, called *Unicode,* towards having two bytes per character instead of one byte.

Bytes Are Also Numbers

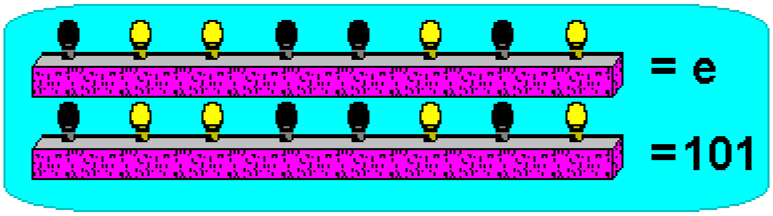


Bytes can represent numbers as well as characters.

When a byte is a number, the combination of lit bits signifies to the computer what number it represents.

How does a computer know if a bit is a number or a character? The programmer tells it through programming code.

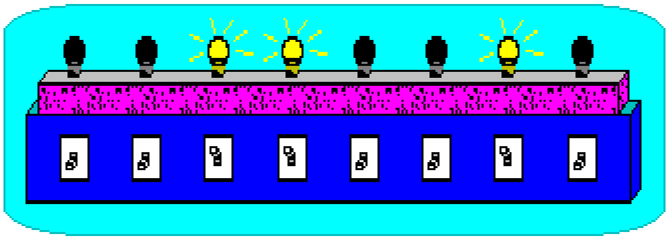
One Byte, Two Meanings



The programmer instructs a computer to interpret particular bytes as numbers or as characters. These computer code instructions are also represented as bytes. So bytes can represent numbers, characters, or computer code. They can also represent other things.

When a computer starts, it looks at a particular byte and interprets it as computer code for what to do next. Programmers take over from there to instruct the computer on how to interpret other bytes.

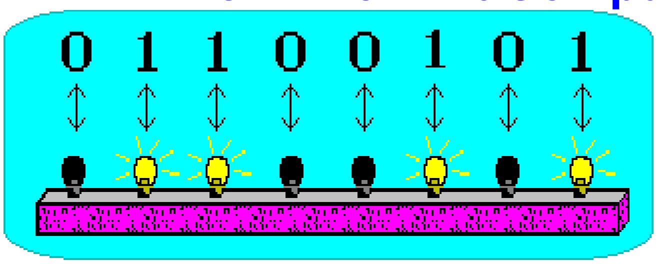
A Primitive Computer



This is an imaginary and oversimplified concept of a primitive computer. However, early personal computers were similar to this. For example, they had switches to set individual bits. After the computing was finished, the user had to read and interpret lights which also represented individual bits.

These early computers did not have keyboards, video displays, printers, or disk drives.

A Five-Million-Bit Computer



Most personal computers have at least 640k of Random Access Memory (RAM). Since a single *k* is 1,024, this means that these computers have (640 times 1,024 equals) 655,360 bytes of RAM. Since each byte has eight bits, this means that these computers have (8 times 655,360 equals) 5,242,880 bits in RAM.

While the light bulbs have been convenient, so far, in representing bits, they would become clumsy as this discussion of bits and bytes progresses. Therefore, a new method is going to be used: That of using *0*'s and *1*'s. A *0* represents a bit *(light)* which is off, and a *1* represents a bit *(light)* which is on.

The byte in the illustration can now be stated simply as being *01100101* without the use of any graphical pictures of lights.

256 Variations of Byte

The bits in a byte can be arranged 256 ways. Scroll down the illustration and you may be able to detect and abstract pattern of how the bit counting progresses. (If you can't, don't fret. Just recognize that their **are** 256 possibilities.)

00000000 = 0
00000001 = 1
00000010 = 2
00000011 = 3
00000100 = 4
00000101 = 5
00000110 = 6
00000111 = 7
00001000 = 8
00001001 = 9
00001010 = 10
00001011 = 11
00001100 = 12
00001101 = 13
00001110 = 14
00001111 = 15
00010000 = 16
00010001 = 17
00010010 = 18
00010011 = 19
00010100 = 20
00010101 = 21
00010110 = 22
00010111 = 23
00011000 = 24
00011001 = 25
00011010 = 26
00011011 = 27
00011100 = 28
00011101 = 29
00011110 = 30
00011111 = 31
00100000 = 32
00100001 = 33
00100010 = 34
00100011 = 35
00100100 = 36

00100101 = 37
00100110 = 38
00100111 = 39
00101000 = 40
00101001 = 41
00101010 = 42
00101011 = 43
00101100 = 44
00101101 = 45
00101110 = 46
00101111 = 47
00110000 = 48
00110001 = 49
00110010 = 50
00110011 = 51
00110100 = 52
00110101 = 53
00110110 = 54
00110111 = 55
00111000 = 56
00111001 = 57
00111010 = 58
00111011 = 59
00111100 = 60
00111101 = 61
00111110 = 62
00111111 = 63
01000000 = 64
01000001 = 65
01000010 = 66
01000011 = 67
01000100 = 68
01000101 = 69
01000110 = 70
01000111 = 71
01001000 = 72
01001001 = 73
01001010 = 74
01001011 = 75
01001100 = 76
01001101 = 77
01001110 = 78
01001111 = 79
01010000 = 80
01010001 = 81
01010010 = 82

01010011 = 83
01010100 = 84
01010101 = 85
01010110 = 86
01010111 = 87
01011000 = 88
01011001 = 89
01011010 = 90
01011011 = 91
01011100 = 92
01011101 = 93
01011110 = 94
01011111 = 95
01100000 = 96
01100001 = 97
01100010 = 98
01100011 = 99
01100100 = 100
01100101 = 101
01100110 = 102
01100111 = 103
01101000 = 104
01101001 = 105
01101010 = 106
01101011 = 107
01101100 = 108
01101101 = 109
01101110 = 100
01101111 = 111
01110000 = 112
01110001 = 113
01110010 = 114
01110011 = 115
01110100 = 116
01110101 = 117
01110110 = 118
01110111 = 119
01111000 = 120
01111001 = 121
01111010 = 122
01111011 = 123
01111100 = 124
01111101 = 125
01111110 = 126
01111111 = 127
10000000 = 128

10000001 = 129
10000010 = 130
10000011 = 131
10000100 = 132
10000101 = 133
10000110 = 134
10000111 = 135
10001000 = 136
10001001 = 137
10001010 = 138
10001011 = 139
10001100 = 140
10001101 = 141
10001110 = 142
10001111 = 143
10010000 = 144
10010001 = 145
10010010 = 146
10010011 = 147
10010100 = 148
10010101 = 149
10010110 = 150
10010111 = 151
10011000 = 152
10011001 = 153
10011010 = 154
10011011 = 155
10011100 = 156
10011101 = 157
10011110 = 158
10011111 = 159
10100000 = 160
10100001 = 161
10100010 = 162
10100011 = 163
10100100 = 164
10100101 = 165
10100110 = 166
10100111 = 167
10101000 = 168
10101001 = 169
10101010 = 170
10101011 = 171
10101100 = 172
10101101 = 173
10101110 = 174

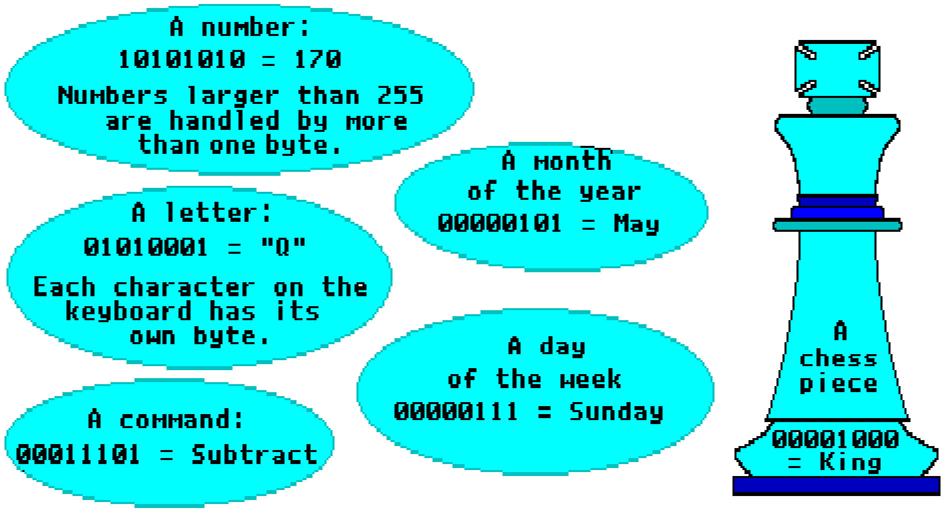
10101111 = 175
10110000 = 176
10110001 = 177
10110010 = 178
10110011 = 179
10110100 = 180
10110101 = 181
10110110 = 182
10110111 = 183
10111000 = 184
10111001 = 185
10111010 = 186
10111011 = 187
10111100 = 188
10111101 = 189
10111110 = 190
10111111 = 191
11000000 = 192
11000001 = 193
11000010 = 194
11000011 = 195
11000100 = 196
11000101 = 197
11000110 = 198
11000111 = 199
11001000 = 200
11001001 = 201
11001010 = 202
11001011 = 203
11001100 = 204
11001101 = 205
11001110 = 206
11001111 = 207
11010000 = 208
11010001 = 209
11010010 = 210
11010011 = 211
11010100 = 212
11010101 = 213
11010110 = 214
11010111 = 215
11011000 = 216
11011001 = 217
11011010 = 218
11011011 = 219
11011100 = 220

11011101 = 221
11011110 = 222
11011111 = 223
11100000 = 224
11100001 = 225
11100010 = 226
11100011 = 227
11100100 = 228
11100101 = 229
11100110 = 230
11100111 = 231
11101000 = 232
11101001 = 233
11101010 = 234
11101011 = 235
11101100 = 236
11101101 = 237
11101110 = 238
11101111 = 239
11110000 = 240
11110001 = 241
11110010 = 242
11110011 = 243
11110100 = 244
11110101 = 245
11110110 = 246
11110111 = 247
11111000 = 248
11111001 = 249
11111010 = 250
11111011 = 251
11111100 = 252
11111101 = 253
11111110 = 254
11111111 = 255

That's only 255!?

No it isn't. When you count the first byte, which starts at 0, then it's 256 variations.

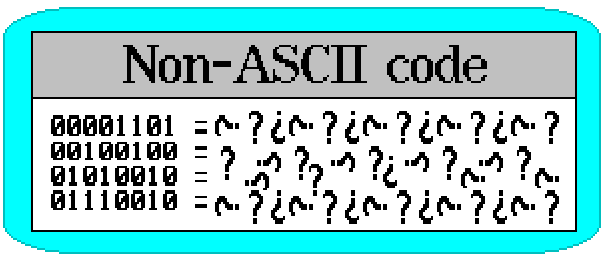
A Byte Can Be Many Things



How can anybody tell when a byte stands for what?

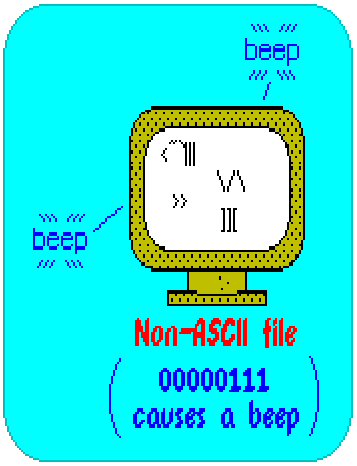
ASCII ("As-Key") to the Rescue!

ASCII stands for the *American Standard Code for Information Interchange.* It is used so that bytes can be utilized in a consistent manner.



With non-ASCII code, only the programmer knows for sure what the bytes stand for.

Is it ASCII?



One can tell if a file uses the ASCII format by loading it into an ASCII text editor (such as Windows Notepad). If the file can be read, it is ASCII. ASCII files are sometimes called *text files.* If the file is not ASCII, it produces nonsense, and sometimes beeps.

The bytes listed below stand for the indicated characters and cause the message in the illustration to be shown on the video display.

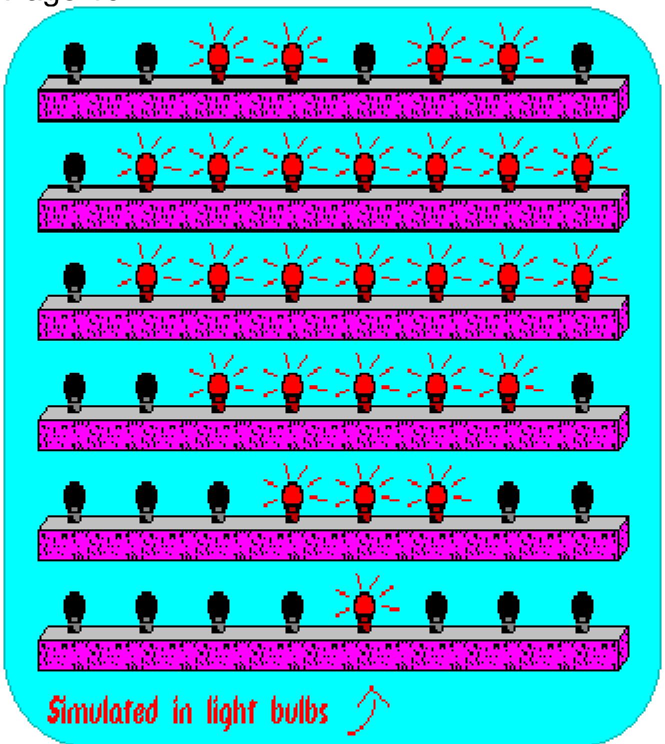
01000100 = "D"
01100001 = "a"
01100100 = "d"
00101100 = ","
00001101 = Carriage Return
00001010 = Line Feed
00100000 = (Space)
01010011 = "S"
01100101 = "e"
01101110 = "n"
01100100 = "d"
00001101 = Carriage Return
00001010 = Line Feed
00100000 = (Space)
01101101 = "m"
01101111 = "o"
01101110 = "n"
01100101 = "e"
01111001 = "y"
00011010 = End Of File

Pixels, Light Bulbs, and Bits

A pixel is a single dot on a video screen. *Pixel* stands for *picture element.* A video screen contains many thousands of pixels. If a pixel is *on* for a monochrome monitor, it displays a dot on the screen. Otherwise, it does not. Pixels have similarities to light bulbs and bits.

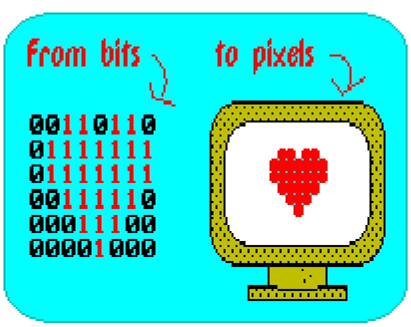
This ***off*** pixel, ○,
is similar to this light bulb,
,
which is similar to this bit, 0.

This ***on*** pixel,
●,
is similar to this light bulb,
,
which is similar to this bit, 1.

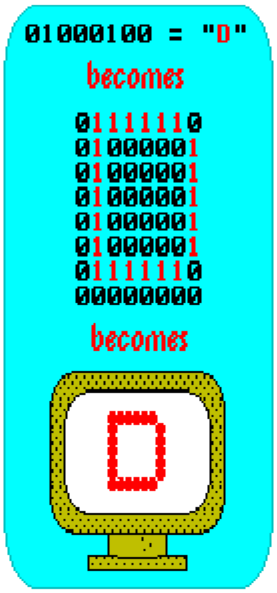


Pixels Copy Bits

For monochrome graphical displays, the pixels copy the status of bits located in a certain part of the computer's memory.



From Characters to Pixels



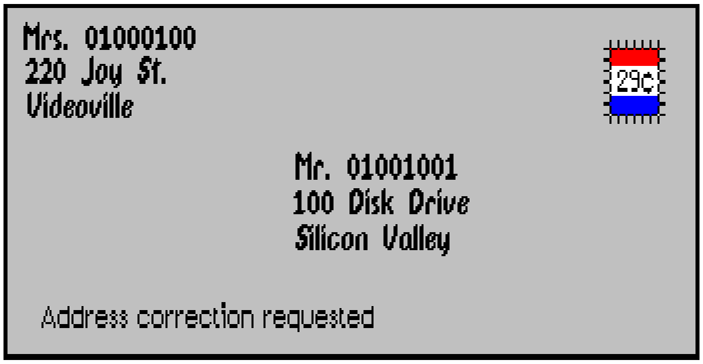
Characters are first changed from their ASCII codes to their graphical bit structures.

The manufacturer of the computer often places these bit structures in the memory of the computer when it is made.

Then, the status of each bit is copied to the corresponding pixel on the video display.

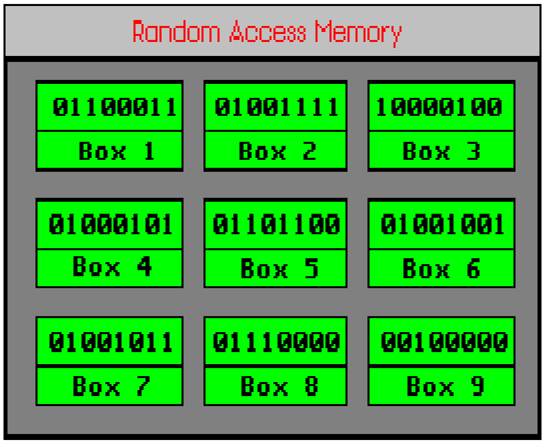
Color displays are done by using more bits, which contol the different colors for each pixel.

Bytes Have Addresses



Every byte in the computer has its own address where it can be located immediately.

The Byte Post Office



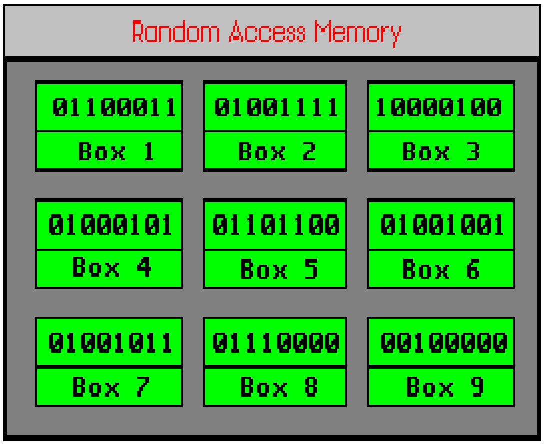
The best way to visualize byte addresses is to think of them as being post office boxes.

This way, they can be referred to by their box numbers.

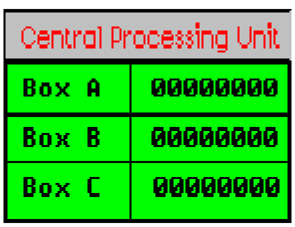
For example, the byte in Box 6 is 01001001.

Any random byte can be directly accessed this way. That is why this method is called *random* access memory (RAM).

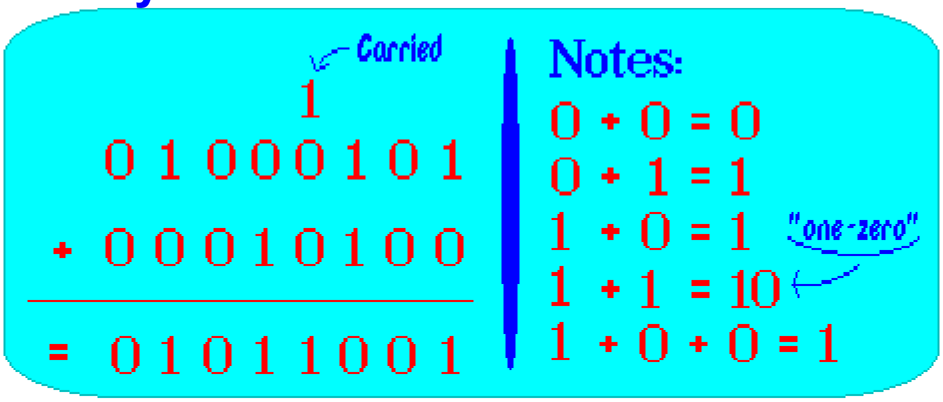
The CPU Is the Handling Area



CPU stands for *Central Processing Unit.* It is where the computer actually does things with the bytes (besides just storing them). The CPU also has addresses for its bytes.

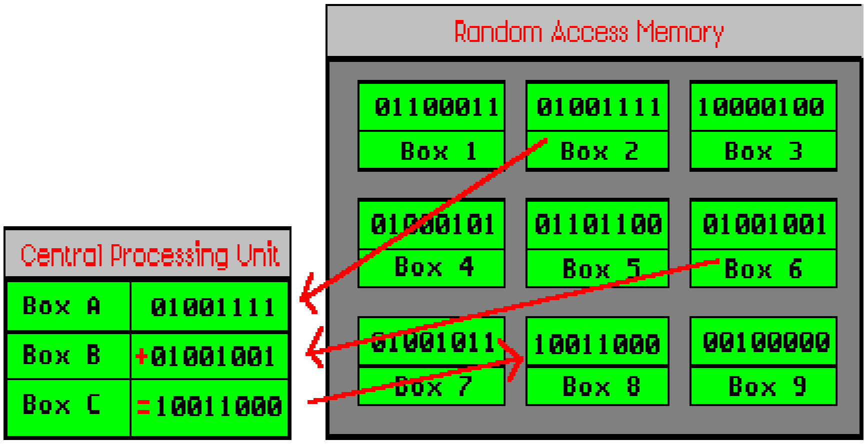


Byte Addition



Before proceeding, it is desirable to know something about how to add two bytes. It is the same as normal arithmetic, except that the highest possible digit is 1. It is not necessary to understand exactly how to add bytes. But one should know that a method does exist.

Adding Bytes from Memory



Any two bytes can be added from anywhere in RAM and the result placed anywhere in RAM.

Each number is placed in the CPU where the addition takes place.

The answer is placed back in memory. The programmer decides where the bytes come from and where the answer goes to.