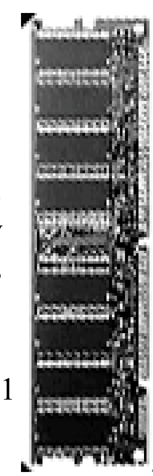
Chapter 1

Computer Hardware and Vocabulary

2005 Version by Steve Bergen and Lynne Schalman (www.summercore.com)
The *Original* Teaching Company, PO Box E, Lexington MA 02420 • 781-953-9698

16 pages in this chapter ... 16 chapters in this book ...
16 NBA championships for the Boston Celtics
Coincidence? We don't think so!

CPU: The **CPU** is the Central Processing Unit. It resides on one or more chips inside the computer. Each chip is a grid of silicon wires encased in a plastic container about the size of a fingernail. Although there are other chips involved and this is an oversimplification, it works well to think of the CPU as the CENTRAL component of your system. It functions the way your brain functions, coordinating the activities of the other devices. Although computers with vacuum tubes go back to 1946, the first microprocessor— i.e. computer on a chip was invented by Ted Hoff from Intel Corp in 1971.[1]



Another important chip — the **RAM** chip — is always blank when the electricity is off. When you start up a computer and begin to work, the information you type as well as the program that is in process all fit inside of RAM. When people refer to a 128 meg or 256 meg computer, they are referring to the size of the RAM capacity. RAM stands for Random Access Memory. A typical Mac or Windows computer now comes with 128-256 meg of RAM, i.e. the equivalent of 128-256 million pages of notebook paper. Older Macs and PCs in the early 1990s came with 1 meg of RAM. In the mid 1980s, typical computers had about 1/8 the of a meg of

[1] source: Newsweek magazine, winter 97-98, p29

RAM (128K) and very primitive graphics. The first microcomputer in 1975 (Altair) had under 1K of RAM which is about 1/1,000th of a meg. Gordon Moore, former CEO of INTEL and mentor of Andrew Grove (TIME magazine Man of the year 12/97) gets credit for what is now called “Moore's Law” — chip power doubling roughly every 2 years and costs falling by 50%.[2] Those of us who bought computers in the 1980s can clearly attest to the truth of Moore’s law!

ROM chips are crucial to the computer, but not very interesting to us as consumers. A ROM chip typically contains “frozen” information (fairly technical, esoteric stuff) that the manufacturer wants accessible at all times. BASIC, however, was built into ROM on the Apple II so that if you turned on that CPU without a disk drive, you could actually start programming. ROM stands for Read Only Memory. ROM chips are found not only in computers; many appliances and even toys contain ROM chips. In fact, the singing bass, Big Mouth Billie, contains a ROM chip.

In order for the CPU and other chips to talk to other devices, an **Interface** board is used to connect the electronics of one machine to another. The board contains a printed circuit with a few special purpose chips. The innovation of the MacPlus back in 1987 was that it had a SCSI Interface built-in, allowing for easy connection to hard disk drives. The technology rivalry between the East Coast (Rt 128 beltway) and West Coast (Silicon Valley[3] area) was symbolized by the pronunciation of the word SCSI. East Coasters wanted to pronounce it as "sexy" while West Coasters wanted “scuzzy.” As the Lakers proved more often in the 1980s, the West won and “scuzzy” became the accepted pronunciation. By the way **SCSI** stands for Small Computer System Interface and is increasingly another obsolete word, since USB is taking over (see page 15) as the faster, preferred way for data to travel!

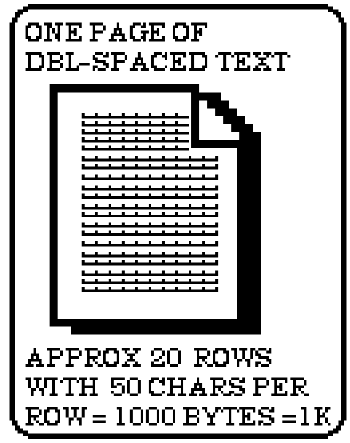
One special interface board — the **Video Board** — connects the computer to the screen; it frequently has multi-colored wires emerging from it, typically RGB for red, green and blue. Although most of us mixed the primary paint colors (red, yellow and blue) to make other paint colors, it turns out that the primary colors for electron beams are red, green and blue! Another important board — the **Motherboard** — is the main circuit board that all the RAM Chips, ROM Chips and other Interface Boards plug into.

Both the computer chip and the disk drive store information that is coded electronically. The basic unit of information is the **Byte** which is equivalent to one character or symbol of text. For example, the word “Hello” is a 5 byte word. Because spaces are just as significant as letters, “New York” is one 8 Byte word, which is the way most New Yorkers say it anyway. In fact, it is very useful in teaching or learning word processing to realize that Space Bar and Return each constitutes 1 Byte; you might say that the computer alphabet has 29 letters: A-Z, Space Bar, Tab and Return. Use the “show invisibles” command in your word processor to see all of these 29 letters!

> Of all the people I play against, the only one I truly fear or worry about is Larry Bird. Magic Johnson

[2] source: Time magazine, Dec 97 , p66
[3] Silicon Valley is south of San Francisco — towns such as Cupertino, Palo Alto, Menlo Park

1000 BYTES = 1 **K**: How many BYTES on a piece of paper? Well, a good approach is to consider double-spaced text with wide margins. For example, you might have 50 characters across the page, with a total of 20 lines down. Therefore, we have 20 X 50 bytes or 1000 bytes on the page. Computer people refer to 1000 bytes as 1 **Kilobyte** and then abbreviate it as 1 **K**.



Just as 1000 bytes equals 1 kilobyte (1K), 1 million bytes (1000K) is 1 **Megabyte** (usually pronounced **MEG**) and 1 billion bytes (1000 megabytes) is 1 **Gigabyte**. Hard disks range from 40 meg to 80 gigabytes. CDs hold up to 650,000K, which we call 650 meg or over 1/2 a Gigabyte. In Spring of 1998, Bill Gates launched the TerraServer project — a database of satellite pictures (www.terraserver.com) of the world. The web server contains over 1 terabyte ("of the terra") stored in a database of 173.6 million rows. One **Terabyte** contains 1,000 gigabytes (about 1500 CDs) or about 1 million floppy disks!

This chart summarizes basic computer literacy for various disks

• double density floppy disk	800K	800,000 bytes	.8 meg	
• high density floppy disk	1,400K	1,400,000 bytes	1.4 meg	
• typical hard disk drive	20,000,000K	20,000,000,000 bytes	20,000 meg	20 gig
• CD ROM disk	650,000K	650,000,000	650 meg	over 1/2 gig

How is each byte coded? Although this topic is not particularly useful, many people find it interesting. Via magnetism, it is possible to magnetize a spot on the plastic floppy disk or a piece of wire to be either ON or OFF. The inside of a chip looks like a huge grid of horizontal and vertical wires.

Consider a 1 inch chip that has 100 horizontal wires and 100 vertical wires. It looks like a gigantic tic-tac-toe board with 10,000 different junction spots or intersection points. Imagine that each one can be TURNED ON or OFF with magnetism, behaving like a light bulb. Let us call each junction of wires a **Bit** and let us think of grouping the thousands of **Bits** into groups of 8.

Using groups of 8 **Bits** at a time, we have a picture that looks like this:

```
 |    |    |    |    |    |    |    |
-·-  -·-  -·-  -·-  -·-  -·-  -·-  -·-
 |    |    |    |    |    |    |    |
```

It turns out that we can design a code of ONs and OFFs so that each group of 8 **Bits** holds one letter or symbol of the English alphabet. How? Well, first let's figure out how many different patterns of ON and OFF we would find with 8 **Bits** lined up. If we had just a 2 bit computer, we might have 4 different patterns for the 2 spots:

```
ON-ON      · ·
ON-OFF     · o
OFF-ON     o ·
OFF-OFF    o o
```

If we had just a 3 bit computer, we might have 8 different patterns for the 3 spots:

```
ON-ON-ON      · · ·      OFF-OFF-ON     o o ·
ON-ON-OFF     · · o      OFF-OFF-OFF    o o o
ON-OFF-ON     · o ·      OFF-ON-ON      o · ·
ON-OFF-OFF    · o o      OFF-ON-OFF     o · o
```

Using the symbols 1 for ON and 0 for off simplifies the explanation:

```
A 2-bit computer has 4 patterns:  00  01  10  11
A 3-bit computer has 8 patterns: 000 001 010 011 100 101 110 111
A 4-bit computer has 16 patterns:
          0000 0001 0010 0011 0100 0101 0110 0111
          0001 1001 1010 1011 1100 1101 1110 1111
```

This system of 1's and O's is really the **Base Two** or **Binary** number system in math; the word **Bit** is an acronym of the phrase **<u>BI</u>nary digi<u>T</u>.**

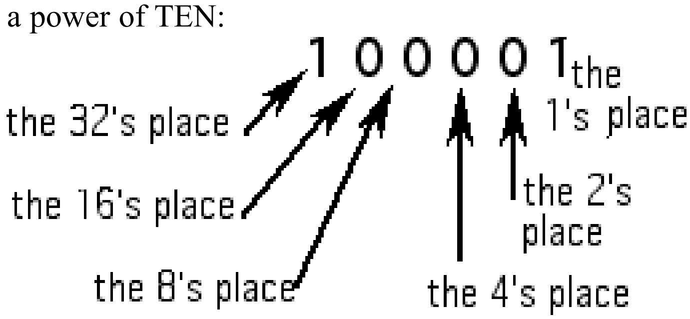
Here are the numbers from 1 to 14 written in **Base Two**:

```
0001 = 1     1000 = 8
0010 = 2     1001 = 9
0011 = 3     1010 = 10
0100 = 4     1011 = 11
0101 = 5     1100 = 12
0110 = 6     1101 = 13
0111 = 7     1110 = 14
```

Here is Larry Bird's uniform number (33) written as a BASE TWO number:

100001

Each position stands for a power of TWO instead of a power of TEN:



Now let's figure out how many patterns in an 8 bit computer? Since a 2 bit computer has 4 patterns and a 4 bit computer has 8 patterns, can you see that the number of patterns doubles as we add a bit. Can you figure out a good rationale for why it is so?

Take the 8 patterns for a 3 bit computer. Put a zero in front of each pattern or put a one in front of each pattern — you get the 16 patterns that for a 4 bit computer. The end result is that an 8 bit computer has 256 patterns. Each one is equal to one letter of the alphabet, a symbol on the keyboard or perhaps even a graphics character. Some of the 256 patterns may even be left unused. This provides another way to understand a Byte. A byte is made up of 8 bits and is a coded way of representing one letter of the alphabet.

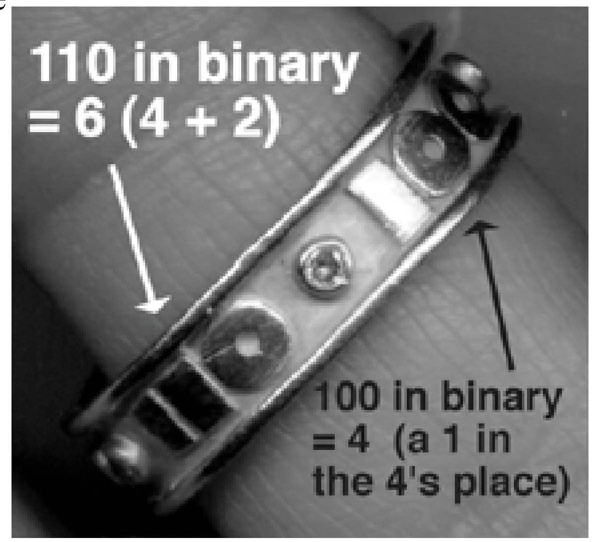
Example #1: let us change 89 from decimal to binary
The typical strategy is to write down 8 places, labeling them from 1 on the right to 128 on the left:

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

We now distribute the 89 (as if we playing the role of the banker in monopoly and have to pay $89 but can use only $128 bills, $64 bills, $32 bills, etc. and just ONE of each) We first put a 1 in the 64's place which leaves 89-64 or 25 left to distribute. We next put a 1 in the 16's place, leaving 9 dollars and so on. **We get an answer of 01011001.**

Example #2: change 01000101 from binary to decimal
We begin with our 8 positions

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



and now translate our 01000101 number into 1 in the 64 place PLUS 1 in the 4 place PLUS 1 in the 1's place, i.e. with our "monopoly money analogy" $64 + $4 + $1. **We get an answer of 69.**

Example #3: interpret the months and date on this wedding ring on the right. The 110 represents 6 (June) since there are 1's in the 2's place and the 4's place. The 110 represents 4 since there is a 1 in the 4's place and 0's in the 1's place and the 2's place. We get an answer of June 4th.

Example #4: Consider the 3 byte word **Yes** (which is what most Boston Celtics fans said when Rick Pitino quit his job as the coach.) This word is made of 3 bytes or 24 bits or 6 nybbles (yes, the word **Nybble** is legitimate — though esoteric — and means 4 bits or half of a byte). And so Rick can teach his children that 8 bits make a byte, 4 bits make a Nybble and 2 nybbles make a byte.

When you type the word **YES** on your computer, each letter is translated into its 8 bit code. Each **ASCII** code is 1 byte made up of 8 bits:

```
 Y Code 89 = 01011001 OFF ON OFF  ON  ON  OFF OFF ON     o·o··oo·
 E Code 69 = 01000101 OFF ON OFF OFF OFF   ON OFF ON     o·ooo·o·
 S Code 83 = 01010011 OFF ON OFF  ON OFF  OFF  ON ON     o·o·oo··
```

The computer is therefore storing 3 bytes of information to remember the word YES. Since each byte is made of 8 bits, the computer is storing the 3 bytes as a series of 24 bits. If we are looking at a disk, we would be referring to 24 spots of magnetism on one of the circular "tracks" of the record. Inside a chip, we would be referring to 24 intersections of wires on the grid.

As mentioned above, the most common coding system for relating binary numbers to English symbols is called the **ASCII** system. It means that A is code 65, B is code 66 and so on ... through Z which is code 90. **ASCII** stands for American Standard Coding Information Interchange (pronounced "ass-key") and it proves that the computer industry is not completely idiosyncratic and eccentric! Now that you know that A is code 65, look back at the word YES and see if the three ASCII Codes (89, 69, 83) make sense.

Another computer system for coding numbers is called **Hexadecimal** or base 16. In this system we use one symbol for each number from 1 to 15; we use A for 10, B for 11, C for 12, D for 13, E for 14 and F for 15. In the Hexadecimal system, after the one's place we have the 16's place, so the number A3 means 3 in the ONEs place and A (or 10) in the 16's PLACE, i.e. A3 = 10x16 + 3 = 163. Let's compare the 3 number systems:

decimal	**binary**	**hexadecimal**
called base 10	called base 2	called base 16
10 symbols used: 0 to 9	2 symbols used: 0 & 1	16 symbols used 0-9 and A,B,C,D,E,F
100,10 & 1's place	4,2 & 1's place	256,16 & 1's place

Below are the ASCII codes for the common 26 uppercase letters of the alphabet, A to Z, together with their Binary and Hexadecimal equivalents. On the old ImageWriter printer, if you turned on the machine incorrectly, you got a series of 2-character hexadecimal codes. If the printout was

4C 41 52 52 59 20 42 49 52 44 20 47 4F 20 44 41 4C 4C 41 53 0D

you can decipher using this chart what was really typed (20=SPACE BAR and 0D=RETURN).

• Thanks to Theresa Overall from Lamplighter School (Dallas, TX) for fixing this coded message •

Letter	ASCII Code	Binary Equivalent	Hexadecimal Equivalent	Letter	ASCII Code	Binary Equivalent	Hexadecimal Equivalent
A	65	01000001	**41**	**N**	78	01001110	**4E**
B	66	01000010	**42**	**O**	79	01001111	**4F**
C	67	01000011	**43**	**P**	80	01010000	**50**
D	68	01000100	**44**	**Q**	81	01010001	**51**
E	69	01000101	**45**	**R**	82	01010010	**52**
F	70	01000110	**46**	**S**	83	01010011	**53**
G	71	01000111	**47**	**T**	84	01010100	**54**
H	72	01001000	**48**	**U**	85	01010101	**55**
I	73	01001001	**49**	**V**	86	01010110	**56**
J	74	01001010	**4A**	**W**	87	01010111	**57**
K	75	01001011	**4B**	**X**	88	01011000	**58**
L	76	01001100	**4C**	**Y**	89	01011001	**59**
M	77	01001101	**4D**	**Z**	90	01011010	**5A**

Example #5: let us make sense of the use of hexadecimal numbers in the HTML command <BODY BGCOLOR = "#33B42D" TEXT= "#ABCDEF" > a command that is fancier than but similar to <BODY BGCOLOR="NAVY" TEXT="FUCHSIA">

Solution: the programmer is using hexadecimal numbers (base 16) to represent the mixture of RGB on the web page — 33 drops of red plus B4 drops of green plus 2D drops of blue. The 3 numbers 33, B4 and 2D are hexadecimal numbers which means that the positions are the ONES place and the SIXTEENS place. 33 therefore is 3x16 + 3 or 49. B4 therefore is 11x16 + 4 or 180 and 2D would be 2x16 + 14 or 46. Remember that in the hexadecimal alphabet A stands for 10, B stands for 11 and so on. Can you figure out amounts RGB intended for the text color of ABCDEF?

Exercises: Change 133 and 233 from decimal to binary and hexadecimal. Change BIRD into a series of ASCII numbers, a series of binary codes and a series of hexadecimal codes.