Topic: 1.1.2 Hexadecimal

At the very launch of computer expansion it was realized that people had many complexities in managing binary numbers. For this reason, a new number system using 16 different symbols was developed. The one main disadvantage of binary numbers is that the binary string equivalent of a large decimal base-10 number can be quite long. When working with large digital systems, such as computers, it is common to find binary numbers consisting of 8, 16 and even 32 digits which makes it difficult to both read and write without producing errors especially when working with lots of 16 or 32-bit binary numbers.

One common way of overcoming this problem is to arrange the binary numbers into groups or sets of four bits (4-bits). These groups of 4-bits uses another type of numbering system also commonly used in computer and digital systems called Hexadecimal Numbers.

REPRESENTING INTERGERS AS HEXADECIMAL NUMBERS:

The base 16 notational system for representing real numbers. The digits used to represent numbers using hexadecimal notation are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.

"H" denotes hex prefix.

Examples:

28_{16} = 28H = $2\times16^1+8\times16^0$ = 40

$2F_{16}$ = 2FH = $2\times16^1+15\times16^0$ = 47

$BC12_{16}$ = BC12H = $11\times16^3+12\times16^2+1\times16^1+2\times16^0$ = 48146

Topic: 1.1.2 Hexadecimal

Numeral systems conversion table:

Decimal Base-10	Binary Base-2	Octal Base-8	Hexadecimal Base-16
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17
24	11000	30	18
25	11001	31	19
26	11010	32	1A
27	11011	33	1B
28	11100	34	1C
29	11101	35	1D
30	11110	36	1E
31	11111	37	1F
32	100000	40	20

Topic: 1.1.2 Hexadecimal

WHY USE HEXADECIMAL NUMBER SYSTEMS?

The main reason why we use hexadecimal numbers is because it is much easier to express binary number representations in hex than it is in any other base number system. Computers do not actually work in hex (don't laugh, beginning students do ask that question). Let's look at an example, using a byte. Bytes are typically 8 bits, and can store the values 0 – 255 (0000 0000 – 1111 1111 in binary). For people, expressing numbers in binary is not convenient. I am not going to turn around to my co-worker and tell him that my phone number is 101 101 101 001 010 001 010 for obvious reasons. Imagine having to try and work with that on a daily basis. So a more convenient expression is needed for the humans.

Since a byte is 8 bits, it makes sense to divide that up into two groups, the top 4 bits and the low 4 bits. Since 4 bits gives you the possible range from 0 – 15, a base 16 system is easier to work with, especially if you are only familiar with alphanumeric characters. It's easier to express a binary value to another person as “A” then it is to express it as “1010”. This way I can simply use 2 hex values to represent a byte and have it work cleanly. This way even if my math is poor, I only need to memorize the multiplication tables up to 15. So if I have a hex value of “CE”, I can easily determine that $(12*16^1) + (14*16^0) = 206$ in decimal, and can easily write it out in binary as 1100 1110. Trying to convert from binary would require me to know what each place holder represents, and add all the values together (128 + 64 + 8 + 4 + 2 = 206). It's much easier to work with binary through hex than any other base system.

Topic: 1.1.2 Hexadecimal

CONVERSION TECHNIQUES:

HEXADECIMAL TO BINARY AND BINARY TO HEXADECIMAL NUMBER SYSTEM:

Let's assume we have a binary number of: 01010111

Hexadecimal =??

8	4	2	1	8	4	2	1
0	**1**	**0**	**1**	**0**	**1**	**1**	**1**
0x8	1x4	0x2	1x1	0x8	1x4	1x2	1x1
0	4	0	1	0	4	2	1
0+4+0+1=5				0+4+2+1=7			
5				7			

Hence the hexadecimal equivalent is **57**

From Denary to Hexadecimal:

117 (in denary) is 7 lots of 16 (112) plus an extra 5. Fitting this in the columns gives

256	16	1
0	7	5

Notice that 7 in binary is 0111 and that 5 is 0101, put them together and we get 01110101 which is the binary value of 117 again. So binary and hexadecimal are all related in some way.

FROM HEXADECIMAL TO BINARY AND DENARY:

Hexadecimal number BD stands for 11 lots of 16 and 13 units

= 176 + 13

= 189 (in denary)

Note: B = 11, which in binary = 1011

D = 13, which in binary = 1101

Put them together to get 10111101 = the binary value of 189.

Topic: 1.1.2 Hexadecimal

USE OF HEXADECIMAL NUMBER IN COMPUTER REGISTERS AND MAIN MEMORY:

Computers are comprised of chips, registers, transistors, resistors, processors, traces, and all kinds of things. To get the binary bits from one place to the next, software programmers convert binary to hex and move hex values around. In reality, the computer is still shoving 1's and 0's along the traces to the chips.

There are two important aspects to the beauty of using Hexadecimal with computers: **First**, it can represent 16-bit words in only four Hex digits, or 8-bit bytes in just two; thus, by using a numeration with more *symbols*, it is both easier to work with (saving paper and screen space) *and* makes it possible to understand some of the vast streams of data inside a computer merely by looking at the Hex output. This is why programs such as DEBUG, use only Hexadecimal to display the actual Binary bytes of a Memory Dump rather than a huge number of ones and zeros!

The **second** aspect is closely related: Whenever it is necessary to convert the Hex representation back into the actual Binary bits, the process is simple enough to be done in your own mind. For example, ***FAD7* hex** is 1111 1010 1101 0111 (F=1111, A=1010, D=1101, 7=0111) in Binary. The reason one might wish to do this is in order to work with "logical" (**AND**, **OR** or **XOR**) or "bit-oriented" instructions (Bit tests, etc.) which *may* make it easier (at times) for a programmer to comprehend.

For example, if you wanted to ***logically* AND** the hex number FAD7 with D37E, you might have a difficult time without first changing these numbers into Binary. If you jot them out in Binary on scratch paper, the task will be much easier:

FAD7(hex)	**1 1 1 1**	**1 0 1 0**	**1 1 0 1**	**0 1 1 1**
D37E(hex)	**1 1 0 1**	**0 0 1 1**	**0 1 1 1**	**1 1 1 0**
ANDing gives	**1 1 0 1**	**0 0 1 0**	**0 1 0 1**	**0 1 1 0**
Answer (in **hex**)	**D**	**2**	**5**	**6**

Topic: 1.1.2 Hexadecimal

USES OF HEXADECIMAL IN COMPUTING:

HTML COLORS:

Along with the advances in Microcomputers, "the Internet" has experienced many changes as well. A code still used by "Web browsers" today had been invented to transfer information from servers to terminals in a way that made the Internet a much more effective tool for research. That code is called "Hyper-Text Markup Language" (or **HTML**) and it soon included a method which could theoretically reproduce background and text with a total of 16,777,216 different colors!

The hardware available today has already advanced way beyond that limit. The main reason was a push to display pictures "in living color;" now a common reality. And one of the first things a new computer user should always do is make sure their display can be set to what's called "24-bit" or "True Color" (for those 16-million plus possible colors). As a matter of fact, almost every video card today has the capacity to reproduce what's called "32-bit" color. But all those extra bits are ***not*** used for increasing the number of colors! Why? Well, since the human eye is only capable of distinguishing something like **7 million** or so different colors that would be a real waste of technology! But you'll have to look for another page about video cards if you want to know more; our task here is to simply explain the use of color with HTML code.

Hexadecimal color values are supported in all major browsers. A hexadecimal color is specified with: #RRGGBB, where the RR (red), GG (green) and BB (blue) hexadecimal integers specify the components of the color. All values must be between 00 and FF.

For example, the #0000FF value is rendered as blue, because the blue component is set to its highest value (FF) and the others are set to the lowest value (00).Computers are comprised of chips, registers, transistors, resistors, processors, traces, and all kinds of things. To get the binary bits from one place to the next, software programmers convert binary to hex and move hex values around. In reality, the computer is still shoving 1's and 0's along the traces to the chips.

ASSEMBLY LANGUAGE:

In 8 bit PC register, the largest number is 1111 1111 which has 8 bits. At the same time the 2 digit hexadecimal number for 1111 1111 is 'FF'.

Do you see how skillfully it is used? Don't forget that computers process 8-digit binary numbers. For the convenient use of programmer in Assembly language, it is easy to manipulate hexadecimal numbers rather than binary numbers.

In Assembly Language, you would move a HEX value into a register like so:

Mov AX,03

The command is MOV or move. The item to move is the value of 3. Where? Into the AX register.

Topic: 1.1.2 Hexadecimal

MAC ADDRESS:

A typical MAC address looks something like this: `af-14-b3-c2-14-45`
You may be wondering why we've got letters and numbers in this address. MAC addresses are expressed in hexadecimal, which gives us the ability to express more values with the same number of bits. Theoretically, every single NIC in the world should have a totally unique MAC address, and the only way to do this is to express MAC addresses in hexadecimal.