HEXADECIMAL NUMBER SYSTEM

We often have to deal with large positive binary numbers. For instance, consider that computers connect to the Internet using a Network Interface Card (NIC).

Every NIC in the world is assigned a unique 48-bit identifier as an Ethernet address. The intent is that no two NICs in the world will have the same address. A sample Ethernet address might be: 000000000100011101011110011111111001001000110110

As another example, computer engineers must oftentimes look at the contents of a specific item in computer memory. You might, for instance, have to look at a variable that is stored at address:
00000000000100101111111101111100

Fortunately, large binary numbers can be made much more compact—and hence easier to work with—if represented in base-16, the so-called hexadecimal number system. You may wonder: Binary numbers would also be more compact if represented in base-10—why not just convert them to decimal? The answer, as you will soon see, is that converting between binary and hexadecimal is exceedingly easy—much easier than converting between binary and decimal.

The Hexadecimal Number System

The base 16 hexadecimal has 16 digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F). Note that the single hexadecimal symbol A is equivalent to the decimal number 10, the single symbol B is equivalent to the decimal number 11, and so forth, with the symbol F being equivalent to the decimal number 15.

For example, consider the hexadecimal number 1A9B. Indicating the values associated with the positions of the symbols, this number is illustrated as:

1 A 9 B

16^3 16^2 16^1 16^0

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 use another type of numbering system also commonly used in computer and digital systems called Hexadecimal Numbers.

$$2\,8_{16} = 2\,8_H = 2\times16^1 + 8\times16^0 = 40$$

$$= 32 \quad + \quad 8 \quad = \quad 40$$

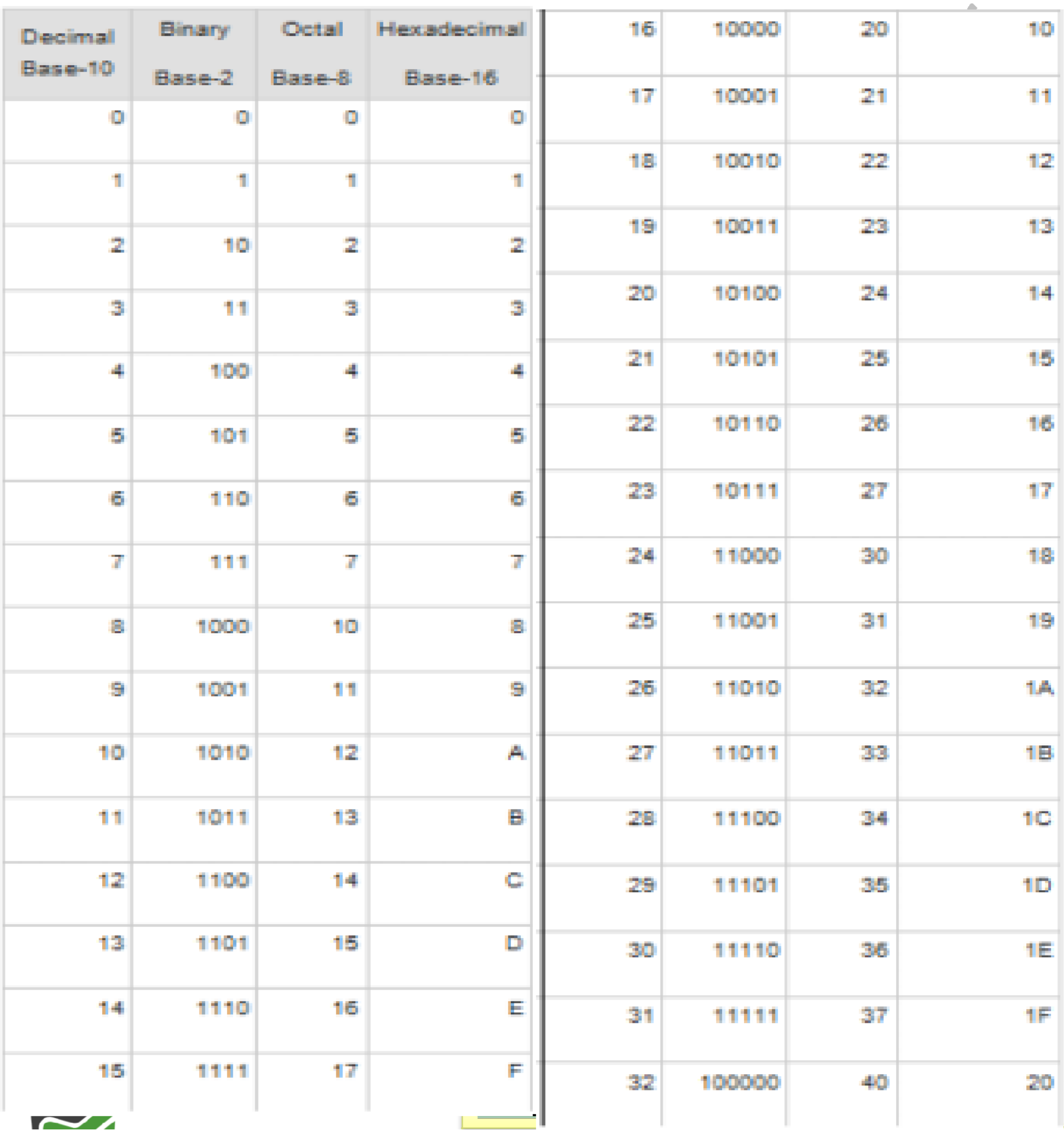
$$2\,F_{16} = 2\,F_H = 2\times16 + 15\times1 = 47$$

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. For example:

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



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

The binary number is 01010111

We will break number into 4 bits each as

0101 0111
Then we will start with the right side 4 bits
Starting from extreme right number

for 0101	**for 0111**
$0X2^3 + 1X2^2 + 0X2^1 + 1X2^0$	**$0X2^3 + 1X2^2 + 1X2^1 + 1X2^0$**
0X8 +1X4 +0X2 +1X1	**0X8 +1X4 +1X2 +1X1**
0+4+0+1=5	**0+4+2+1=7**
5	**7**

So Hexadecimal number is 57

Converting Hexadecimal Numbers to Binary Numbers

To convert a hexadecimal number to a binary number, we reverse the above procedure. We separate every digit of hexadecimal number and find its equivalent binary number and then we write it together.

Example 1.2.4

To convert the hexadecimal number 9F216 to binary, each hex digit is converted into binary form.

$$9\ F\ 2_{16} = (1001\ 1111\ 0010)_2$$

9 =1001 F=1111 2=0010

So Binary equivalent of Hexadecimal number is: 9F2= 100111110010 **Problems 1.2.6**
Convert hexadecimal 2BF9 to its binary equivalent.
Convert binary 110011100001 to its hexadecimal equivalent. (Below is working area)

Converting a Hexadecimal Number to a (Denry) Decimal Number

To convert a hexadecimal number to a decimal number, write the hexadecimal number as a sum of powers of 16. For example, considering the hexadecimal number 1A9B above, we convert this to decimal as:

1	A	9	B
16^3	16^2	16^1	16^0

$1A9B = 1(16^3) + A\ (16^2) + 9(16^1) + B\ (16^0)$

= 4096 + 10(256) + 9(16) + 11(1) = 6811

So $1A9B_{16} = 6811_{10}$

Converting a Decimal Number to a Hexadecimal Number

The easiest way to convert from decimal to hexadecimal is to use the same division algorithm that you used to convert from decimal to binary, but repeatedly dividing by 16 instead of by 2. As before, we keep track of the remainders, and the sequence of remainders forms the hexadecimal representation.

For example, to convert the decimal number **746** to hexadecimal, we proceed as follows:

`

```
                    Remainder
16 | 746
   | 46                  10 =   A
   | 2              14 =   E
   | 0                    2
```

We read the number as last is first and first is last.

2EA

So, the decimal number 746 = 2EA in 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

Using hexadecimal

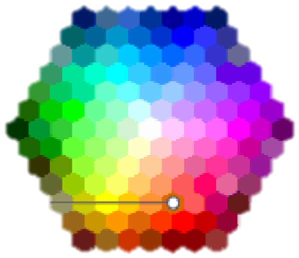
Hex codes are used in many areas of computing to simplify **binary codes**. It is important to note that computers do not use hexadecimal - it is used by humans to shorten binary to a more easily understandable form. Hexadecimal is translated into binary for computer use. Some examples of where hex is used include:

- colour references
- **assembly language** programs
- error messages

Colours

Hex can be used to represent **colours** on web pages and image-editing programs using the format **#RRGGBB** (RR = reds, GG = greens, BB = blues). The # symbol indicates that the number has been written in hex format.

This system uses two hex digits for each colour, eg #FF6600.



As one hex digit represents 4 **bits**, two hex digits together make 8 bits (1 **byte**). The values for each colour run between 00 and FF. In binary, 00 is 0000 0000 and FF is

1111 1111. That provides 256 possible values for each of the three colours.
That gives a total spectrum of 256 reds x 256 greens x 256 blues - which is over 16 million colours in total.

- **#FF0000** will be the purest red - red only, no green or blue.
- Black is **#000000** - no red, no green and no blue.
- White is **#FFFFFF.**

An orange colour can be represented by the code #FF6600. The hex code is much easier to read than the binary equivalent 1111 1111 0110 0110 0000 0000.
If you are making a web page with HTML or CSS you can use hex codes to choose the colours.

RGB colour model

Hex values have equivalents in the **RGB colour model**. The RGB model is very similar to the hex colour model, but instead of combining hex values you use a value between 0 and 255 for each colour. So an orange colour that is #FF6600 in hex would be 255, 102, 0 in RGB.

Errors

Hex is often used in error messages on your computer. The hex number refers to the memory location of the error. This helps programmers to find and then fix problems.

ASCII and Unicode

Two standard character sets are ASCII and Unicode.

ASCII

The **ASCII** character set is a 7-bit set of codes that allows **128 different characters**. That is enough for every upper-case letter, lower-case letter, digit and punctuation mark on most keyboards. ASCII is only used for the English language.

This table shows some examples of letters represented using the ASCII character set:

Character	Denary value	Binary value	Hex
N	78	1001110	4E

Character	Denary value	Binary value	Hex
O	79	1001111	4F
P	80	1010000	50
Q	81	1010001	51
R	82	1010010	52

Extended ASCII

Extended ASCII code is an 8-bit character set that represents **256 different characters**, making it possible to use characters such as é or ©. Extended ASCII is useful for European languages.

Hexadecimal in HTML Colour Codes

One example of how hexadecimal is used in computers would be HTML colour codes. In HTML colours are defined by how much red, green and blue there is on a scale of 0 to 255. The range 0 to 255 was chosen because that is the range of numbers that can be fitted into a single byte and a single byte can be represented as two hexadecimal numerals. HTML colour codes start with a hash symbol followed by 3 pairs of hexadecimal numbers. The first two numerals show how much red, the second pair show how much green and the final pair show how much blue. For example the HTML colour code #00FF00 would have 00 (0) units of red, FF (255) units of green and 00 (0) units of blue. The diagram below shows several examples of HTML colour codes. More can be found on the website html-color-codes.info.

#0000FF Blue	#00FF00 Green	#FF0000 Red	#FF00FF Pink	#FFFF00 Yellow	#000000 Black	#FFFFFF White

HTML colour codes are given in hexadecimal values rather than decimal values as this uses fewer characters which therefore makes the file smaller, which in turn would allow the page to load faster over a slow network connection. de.

Soon there were two very typical lines of HTML code being used to create background colors:

<body bgcolor="#FFFFFF"> for an all-white background And

<body bgcolor="#CCCCCC"> for a light-gray background

HTML Font color TAGs

<font color="#FF0000">RED</font> (RED)
<font color="#00FF00">GREEN</font>(GREEN) LIME
<font color="#0000FF">BLUE</font> (BLUE) <font color="#FFFF00">YELLOW</font>(YELLOW)
<font color="#FF00FF">VIOLET</font>(VIOLET) FUCHSIA
<font color="#00FFFF">CYAN</font> (CYAN) AQUA

USES OF HEXADECIMAL IN MAC Addresses:

What Is a MAC Address?

The MAC address is a unique value associated with a network adapter. MAC addresses are also known as **hardware** addresses or **physical** addresses. They uniquely identify an adapter on a LAN.
MAC addresses are 12-digit hexadecimal numbers (48 bits in length). By convention, MAC addresses are usually written in one of the following two formats:

MM:MM:MM:SS:SS:SS

The first half of a MAC address contains the ID number (MM:MM:MM) of **manufacturer**. These IDs are regulated by an Internet standards body. The second half of a MAC address represents (SS:SS;SS) **serial number** assigned to the adapter by the manufacturer. In the example,

00:A0:C9:14:C8:29

The prefix: **00A0C9** indicates the **manufacturer is Intel Corporation.**

MAC Addresses

All network adapters and network devices have a Media Access Control (MAC) address. This is also known as the 'physical address' and is a unique address determined during the manufacture of each device. This address is given as a set of 6 pairs of hexadecimal numbers. An example of a MAC address would be: A0-1D-48-FE-5E-F5.You can determine the physical address of the network adapters in a computer running the Windows operating system by typing the following command in to a command prompt:

```
ipconfig /all
```

Uses of Hexadecimal

Specification link

- *Show understanding of the reasons for choosing hexadecimal to represent numbers*
- *Represent numbers stored in registers and main memory as hexadecimal*
- *Identify current uses of hexadecimal numbers in computing* - 2016 CIE Syllabus p10

Hexadecimal is used as an intermediate step between binary and denary because it is easier for a computer to convert between binary and hexadecimal than between binary and denary whilst at the same time being easier for a Human to process than a binary number would be. It is also used because a single hexadecimal numeral can store 1 nibble and we can store a whole byte using only 2 hexadecimal numerals. Depending on the application this can have benefits in both storage space and processing time.

Why MAC Addresses?

Recall that TCP/IP and other mainstream networking architectures generally adopt the OSI model. In this model, network functionality is subdivided into layers. MAC addresses function at the data link layer (layer 2 in the OSI model). They allow computers to uniquely identify themselves on a network at this relatively low level.

MAC vs. IP Addressing

Whereas MAC addressing works at the data link layer, IP addressing functions at the network layer (layer 3). It's a slight oversimplification, but one can think of IP addressing as supporting the software implementation and MAC addresses as supporting the hardware implementation of the network stack. The MAC address generally remains fixed and follows the network device, but the IP address changes as the network device moves from one network to another.
IP networks maintain a mapping between the IP address of a device and its MAC address. This mapping is known as the **ARP cache** or **ARP table**. ARP, the Address Resolution Protocol, supports the logic for obtaining this mapping and keeping the cache up to date.
DHCP also usually relies on MAC addresses to manage the unique assignment of IP addresses to devices.

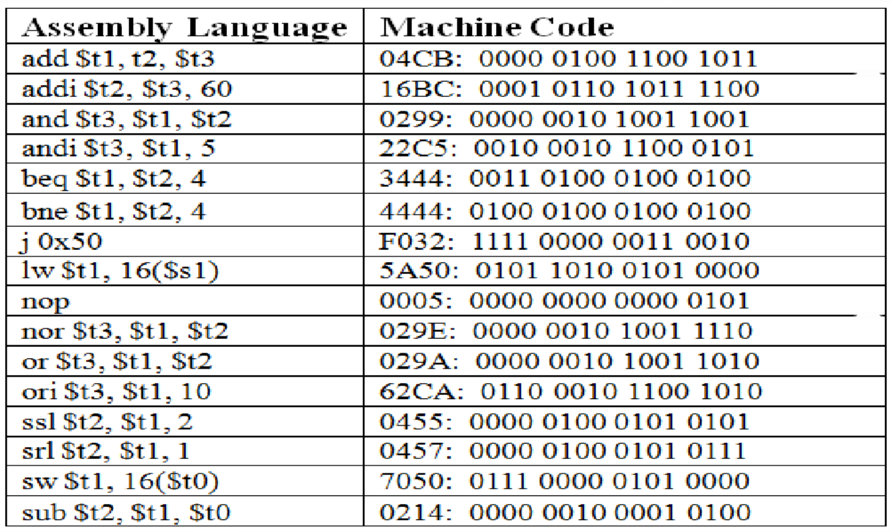
Assembly code and machine code

Computer memory can be referred to directly using machine code or assembly code. This can have many advantages to program developers or when carrying out troubleshooting.

Using hexadecimal makes it much easier, faster and less error prone to write code compared to binary. Using true machine code (which uses binary) is very cumbersome and it takes a long time to key in the values. It is also very easy to mistype the digits in a „sea of 1s and 0s". Here is a simple example:

STO FFA4 (assembly code)
A5E4 FFA4 (machine code using hexadecimal values)
1010 0101 1110 0100 1111 1111 1010 0100 (machine code using binary)

Machine code and assembly code are examples of low-level languages and are used by software developers when producing, for example, computer games, they have many advantages at the development stage of software writing (especially when trying to locate errors in the code).



Assembly Language	Machine Code
add $t1, t2, $t3	04CB: 0000 0100 1100 1011
addi $t2, $t3, 60	16BC: 0001 0110 1011 1100
and $t3, $t1, $t2	0299: 0000 0010 1001 1001
andi $t3, $t1, 5	22C5: 0010 0010 1100 0101
beq $t1, $t2, 4	3444: 0011 0100 0100 0100
bne $t1, $t2, 4	4444: 0100 0100 0100 0100
j 0x50	F032: 1111 0000 0011 0010
lw $t1, 16($s1)	5A50: 0101 1010 0101 0000
nop	0005: 0000 0000 0000 0101
nor $t3, $t1, $t2	029E: 0000 0010 1001 1110
or $t3, $t1, $t2	029A: 0000 0010 1001 1010
ori $t3, $t1, 10	62CA: 0110 0010 1100 1010
ssl $t2, $t1, 2	0455: 0000 0100 0101 0101
srl $t2, $t1, 1	0457: 0000 0100 0101 0111
sw $t1, 16($t0)	7050: 0111 0000 0101 0000
sub $t2, $t1, $t0	0214: 0000 0010 0001 0100