

Topic: 1.1.1 Number representation

Fundamentals of Data Representation:

Before we jump into the world of number systems, we'll need a point of reference; I recommend that you copy the following table that you can refer to throughout this chapter to check your answers.

Hexadecimal	Binary	Denary
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15





Topic: 1.1.1 Number representation

Denary/Decimal

Denary is the number system that you have most probably grown up with. It is also another way of saying base 10. This means that there are 10 different numbers that you can use for each digit, namely:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Notice that if we wish to say 'ten', we use two of the numbers from the above digits, 1 and 0.

Thousands	Hundreds	Tens	Units
10^3	10^2	10^1	10^0
1000	100	10	1
5	9	7	3

Using the above table we can see that each column has a different value assigned to it. And if we know the column values we can know the number, this will be very useful when we start looking at other base systems. Obviously, the number above is: five-thousands, nine-hundreds, seven-tens and three-units.

$$5 \times 1000 + 9 \times 100 + 7 \times 10 + 3 \times 1 = (5973)_{10}$$

Binary Number System:

Binary is a base-2 number system, this means that there are two numbers that you can write for each digit 0, 1.

With these two numbers we should be able to write (or make an approximation) of all the numbers that we could write in denary. Because of their digital nature, a computer's electronics can easily manipulate numbers stored in binary by treating 1 as "on" and 0 as "off."

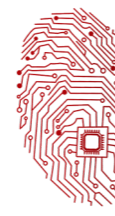
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	1	1	0	1	0	1	0

Using the above table we can see that each column has a value assigned to it that is the power of two (the base number!), and if we take those values and the corresponding digits we can work out the value of the number: $1 \times 64 + 1 \times 32 + 1 \times 8 + 1 \times 2 = 106$.

If you are asked to work out the value of a binary number, the best place to start is by labeling each column with its corresponding value and adding together all the columns that hold a 1. Let's take a look at another example: $(00011111)_2$

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





Topic: 1.1.1 Number representation

So now all we need to do is to add the columns containing 1s together:

$$1*16 + 1*8 + 1*4 + 1*2 + 1*1 = 31$$

Exercise: Binary

Zak Convert the following binary numbers into denary
>>(00001100)₂

Answer :

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

$$8+4 = (12)_{10}$$

Zak Convert the following binary numbers into denary
(01011001)₂

Answer :

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

$$64 + 16 + 8 + 1 = (89)_{10}$$

Zak Convert the following binary numbers into denary
>>(00000111)₂

Answer :

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

$$4 + 2 + 1 = (7)_{10}$$

Zak Convert the following binary numbers into denary
>>(01010101)₂

Answer :

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

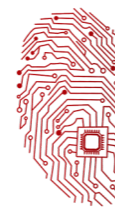
$$64 + 16 + 4 + 1 = (85)_{10}$$

Zak How do we tell if a binary number is odd?

Answer :

It's right most digit is a one





Topic: 1.1.1 Number representation

Is there a short cut to working out a binary number that is made of solid ones, such as: $(01111111)_2$?

Answer :

Yes, take the first 0's column value and minus one

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

$$= 128 - 1 = 127 = 64 + 32 + 16 + 8 + 4 + 2 + 1$$

Bit patterns in a computer:

The language that a computer understands is very simple, so simple that it only has 2 different numbers: 1 and 0. This is called **Binary**. Everything you see on a computer, images, sounds, games, text, videos, spreadsheets, websites etc. Whatever it is, it will be stored as a string of ones and zeroes.

What is a bit?

A bit is the smallest unit in digital representation of information. A bit has only two values, ON and OFF where ON is represented by 1 and OFF by 0. In terms of electrical signals a 1 (ON) is normally a 5 volt signal and a 0 (OFF) is a 0 volt signal.

Bit
1

What is a nibble?

A group of 4 bits are referred to as a nibble.

Nibble			
1	0	0	1

What is a byte?

In the world of computers and microcontrollers, 8 bits are considered to be a standard group. It is called a byte. Microcontrollers are normally byte oriented and data and instructions normally use bytes. A Byte can be broken down into two nibbles.

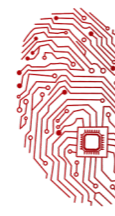
Byte							
1	0	0	1	0	1	1	1

What is a word?

Going bigger, a group of bytes is known as a word. A word normally consists of two and sometimes more bytes belonging together.

Word															
1	1	0	1	0	1	1	0	0	0	1	1	0	1	1	1





Topic: 1.1.1 Number representation

Hexadecimal

You may notice from the table that one hexadecimal digit can represent exactly 4 binary bits. Hexadecimal is useful to us as a shorthand way of writing binary, and makes it easier to work with long binary numbers.

Hexadecimal is a base-16 number system which means we will have 16 different numbers to represent our digits. The only problem being that we run out of numbers after 9, and knowing that 10 is counted as two digits we need to use letters instead:

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

We can do exactly the same thing as we did for denary and binary, and write out our table.

16^5	16^4	16^3	16^2	16^1	16^0
1 048 576	65536	4096	256	16	1
0	0	3	4	A	F

So now all we need to do is to add the columns containing values together, but remember that A = 10, B = 11, C = 12, D = 13, E = 14, F = 15.

$$3 \times 4096 + 4 \times 256 + (A)10 \times 16 + (F)15 \times 1 = (13487)_{16}$$

You might be wondering why we would want to use hexadecimal when we have binary and denary, and when computer store and calculate everything in binary. The answer is that it is entirely for human ease. Consider the following example:

```

*** STOP: 0x0000001E (0x80000003,0x80106fc0,0x8025ea21,0xfd6829e8)
Unhandled Kernel exception c0000047 from fa8418b4 (8025ea21,fd6829e8)

Dll Base Date Stamp - Name                Dll Base Date Stamp - Name
80100000 2be154c9 - ntoskrnl.exe          80400000 2bc153b0 - hal.dll
80258000 2bd49628 - nrcr710.sys           8025c000 2bd49688 - SCSIPTOR.SYS
80267000 2bd49683 - scsidisk.sys         802a6000 2bd496b9 - Fastfat.sys
fa800000 2bd49666 - Floppy.SYS           fa810000 2bd496db - Hpfs_Rec.SYS
fa820000 2bd49676 - Null.SYS             fa830000 2bd4965a - Beep.SYS
fa840000 2bdaab00 - i8042prt.SYS         fa850000 2bd5a020 - SERMOUSE.SYS
fa860000 2bd4966f - kbdclass.SYS        fa870000 2bd49671 - MOUCLASS.SYS
fa880000 2bd9c0be - Videoprt.SYS        fa890000 2bd49638 - NCC1701E.SYS
fa8a0000 2bd4a4ce - Vga.SYS              fa8b0000 2bd496d0 - Msfs.SYS
fa8c0000 2bd496c3 - Npfs.SYS             fa8e0000 2bd496c9 - Ntfs.SYS
fa940000 2bd496df - NDIS.SYS             fa930000 2bd49707 - wlan.sys
fa970000 2bd49712 - TDI.SYS              fa950000 2bd5a7fb - nbfs.sys
fa980000 2bd72406 - streams.sys          fa9b0000 2bd4975f - ubnb.sys
fa9c0000 2bd5bdf7 - usbser.sys           fa9d0000 2bd4971d - netbios.sys
fa9e0000 2bd49678 - Parallel.sys         fa9f0000 2bd4969f - serial.SYS
faa00000 2bd49739 - mup.sys              faa40000 2bd4971f - SMETRSUP.SYS
faa10000 2bd6f2a2 - srv.sys              faa50000 2bd4971a - afd.sys
faa60000 2bd6fd80 - rdr.sys              faaa0000 2bd49735 - browser.sys

Address dword dump Dll Base                - Name
801afc20 80106fc0 80106fc0 00000000 00000000 80149905 : fa840000 - i8042prt.SYS
801afc24 80149905 80149905 ff8e6b8c 80129c2c ff8e6b94 : 8025c000 - SCSIPTOR.SYS
801afc2c 80129c2c 80129c2c ff8e6b94 00000000 ff8e6b94 : 80100000 - ntoskrnl.exe
801afc34 801240f2 80124f02 ff8e6df4 ff8e6f60 ff8e6c58 : 80100000 - ntoskrnl.exe
801afc54 80124f16 80124f16 ff8e6f60 ff8e6c3c 8015ac7e : 80100000 - ntoskrnl.exe
801afc64 8015ac7e 8015ac7e ff8e6df4 ff8e6f60 ff8e6c58 : 80100000 - ntoskrnl.exe
801afc70 80129bda 80129bda 00000000 80088000 80106fc0 : 80100000 - ntoskrnl.exe

Kernel Debugger Using: COM2 (Port 0x2f8, Baud Rate 19200)
Restart and set the recovery options in the system control panel
or the /CRASHDEBUG system start option. If this message reappears,
contact your system administrator or technical support group.
    
```




Topic: 1.1.1 Number representation

Error messages are written using hex to make it easier for us to remember and record them

Representation	Base
EF FE 11	base-16 hexadecimal
15728145	base-10 denary
1110111111111111000010001	base-2 binary

All the numbers are the same and the easiest version to remember/understand for humans is the base-16. Hexadecimal is used in computers for representing numbers for human consumption, having uses for things such as memory addresses and error codes. NOTE: Hexadecimal is used as it is shorthand for binary and easier for people to remember. It **DOES NOT** take up less space in computer memory, only on paper or in your head! Computers still have to store everything as binary whatever it appears as on the screen.

Exercise: Hexadecimal

 Convert the following Hex numbers into decimal/denary:

>>A1

Answer :

16	1
A	1

$$16 * 10 + 1 * 1 = (161)_{10}$$

 Convert the following Hex numbers into decimal/denary:

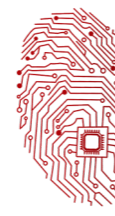
>>FF

Answer :

16	1
F	F

$$16 * 15 + 1 * 15 = (255)_{10}$$





Topic: 1.1.1 Number representation

Zak Convert the following Hex numbers into decimal/denary:

>>0D

Answer :

16	1
0	D

$$16 * 0 + 1 * 13 = (13)_{10}$$

Zak Convert the following Hex numbers into decimal/denary:

>> 37

Answer :

16	1
3	7

$$16 * 3 + 1 * 7 = (55)_{10}$$

Zak Why would we use the Hexadecimal system?

Answer :

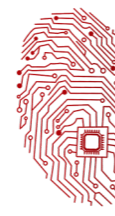
Hexadecimal is used for humans, it is easier to understand and write

Zak Name a use of the hexadecimal system?

Answer :

Hexadecimal is used for error message codes and memory addresses





Topic: 1.1.1 Number representation

Converting Between Bases

The sum that you saw previously to convert from hex to denary seemed a little cumbersome and in the exam you wouldn't want to make any errors, we therefore have to find an easier way to make the conversion.

Since 4 binary bits are represented by one hexadecimal digit, it is simple to convert between the two. You can group binary bits into groups of 4, starting from the right, and adding extra 0's to the left if required, and then convert each group to their hexadecimal equivalent. For example, the binary number 0110110011110101 can be written like this:

0110 1100 1111 0101

and then by using the table given at the beginning, you can convert each group of 4 bits into hexadecimal:

0110	1100	1111	0101
(2+4=6)	(4+8=12)	(1+2+4+8=15)	(1+4=5)
6	C	F	5

So the binary number 0110110011110101 is 6CF5 in hexadecimal. We can check this by converting both to denary. First we'll convert the binary number, since you already know how to do this:

32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
0	1	1	0	1	1	0	0	1	1	1	1	0	1	0	1

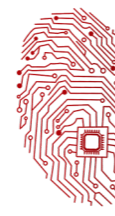
$$32768*0 + 16384*1 + 1*1 = 27893$$

By multiplying the columns and then adding the results, the answer is 27893.

Notice that the column headings are all 2 raised to a power, $1 = 2^0, 2 = 2^1, 4 = 2^2, 8 = 2^3$, and so on. To convert from hexadecimal to denary, we must use column headings that are powers with the base 16, like this:

$16^3 = 4096$	$16^2 = 256$	$16^1 = 16$	$16^0 = 1$
6	C	F	5





Topic: 1.1.1 Number representation

$$5 \times 1 = 5$$

$$15 \times 16 = 240 \quad (\text{You should memorize the values A-F})$$

$$12 \times 256 = 3072$$

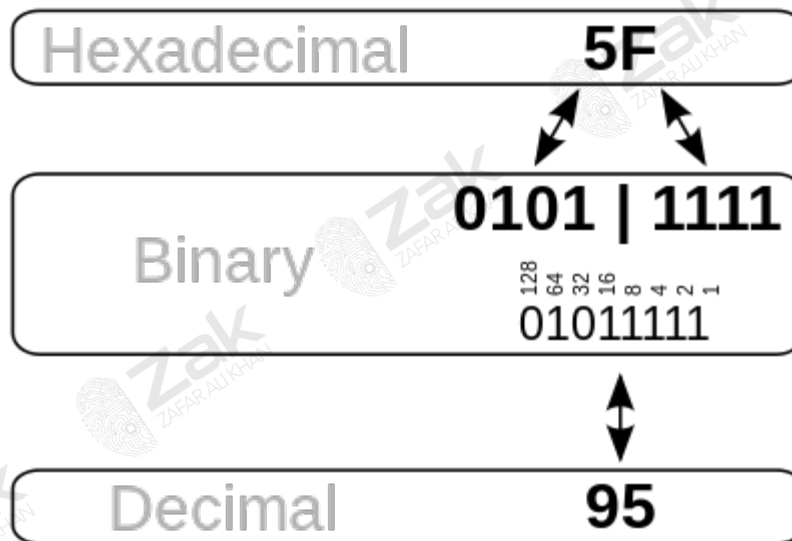
$$6 \times 4096 = 24576$$

Totaling them all up gives us 27893, showing that 0110110011110101 is equal to 6CF5.

To convert from denary to hexadecimal, it is recommended to just convert the number to binary first, and then use the simple method above to convert from binary to hexadecimal.

In summary, to convert from one number to another we can use the following rule:

Hexadecimal \leftrightarrow Binary \leftrightarrow Denary





Topic: 1.1.1 Number representation

Exercise: Hexadecimal and Base Conversion

Zak Convert the following Hexadecimal values into Denary:

>>(12)₁₆

Answer :

1	2	(Hex)
0001	0010	(Binary)

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

2+16 = 18 (decimal)

Zak Convert the following Hexadecimal values into Denary:

>> (A5)₁₆

Answer :

A	5	(Hex)
1010	0101	(Binary)

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

128+32+4+1 = 165 (decimal)

Zak Convert the following Hexadecimal values into Denary:

>> (7F)₁₆

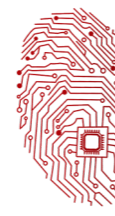
Answer :

7	F	(Hex)
0111	1111	(Binary)

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

64+32+8+4+2+1 = 127 (decimal)





Topic: 1.1.1 Number representation

Zak Convert the following Hexadecimal values into Denary:

>> $(10)_{16}$

Answer :

1	0	(Hex)
0001	0000	(Binary)

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

16(decimal)

Zak Convert the following Binary numbers into hex:

>> $(10101101)_2$

Answer :

1010	1101	(Binary)
A	D	(Hex)

Hence, $(10101101)_2 \gg (AD)_{16}$

Zak Convert the following Binary numbers into hex:

>> $(110111)_2$

Answer :

0011	0111	(Binary)
3	7	(Hex)

Hence, $(110111)_2 \gg (37)_{16}$

In this example, the question has given you a binary number with only six bits. You must complete the byte by adding zeroes to the left of the byte to make it complete

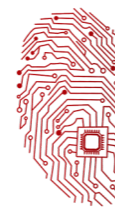
Zak Convert the following Binary numbers into hex:

>> $(10101111)_2$

Answer :

1010	1111	(Binary)
A	F	(Hex)





Topic: 1.1.1 Number representation

Convert the following Binary numbers into hex:

$\gg(111010100001)_2$

Answer :

1110	1010	0001	(Binary)
E	A	1	(Hex)

Hence, $(111010100001)_2 \gg (EA1)_{16}$

Convert the following decimal numbers into hex:

$\gg(87)_{10}$

Answer :

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

$0101\ 0111 = 64+16+4+2+1 = 87(\text{decimal})$

0101	0111	(Binary)
5	7	(Hex)

Hence, $(87)_{10} \gg (57)_{16}$

Convert the following decimal numbers into hex:

$\gg(12)_{10}$

Answer :

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

$00001100 = 8+4 = 12(\text{decimal})$

0000	1100	(Binary)
0	C	(Hex)

Hence, $(12)_{10} \gg (0C)_{16}$





Topic: 1.1.1 Number representation

Zak Convert the following decimal numbers into hex:

$>>(117)_{10}$

Answer :

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

$01110101 = 64 + 32 + 16 + 4 + 1 = 117(\text{decimal})$

0111	0101	(Binary)
7	5	(Hex)

Hence, $(117)_{10} >> (75)_{16}$

Zak Why might you use Hexadecimal?

Answer :

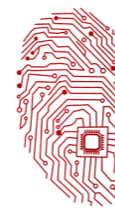
So that it makes things such as error messages and memory address easier for humans understand and remember.

Zak Give two uses of hexadecimal?

Answer :

- Zak** Error message codes
- Zak** Memory address locations





Topic: 1.1.1 Number representation

Positive & negative Integers

Nearly all computers work purely in binary. That means that they only use ones and zeros, and there's no - or + symbol that the computer can use. The computer must represent negative numbers in a different way.

We can represent a negative number in binary by making the most significant bit (MSB) a sign bit, which will tell us whether the number is positive or negative. The column headings for an 8 bit number will look like this:

-128	64	32	16	8	4	2	1
MSB							LSB
1	0	1	1	1	1	0	1

Here, the most significant bit is negative, and the other bits are positive. You start with -128, and add the other bits as normal. The example above is -67 in denary because: $(-128 + 32 + 16 + 8 + 4 + 1 = -67)$.

-1 in binary is: 11111111. $(-128 + 127)$

Note that you only use the most significant bit as a sign bit if the number is specified as signed. If the number is unsigned, then the msb is positive regardless of whether it is a one or not.

Two's Complement:

The MSB stays as a number, but is made negative. This means that the column headings are

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

+117 does not need to use the MSB, so it stays as 01110101.

-117 = -128 + 11

= -128 + (8 + 2 + 1)

fitting these in the columns give 10001011

Two's complement seems to make everything more complicated for little reason at the moment, but later it becomes essential for making the arithmetic easier.



Topic: 1.1.1 Number representation

The Standard ASCII Character Set

Bytes are frequently used to hold individual characters in a text document. In the ASCII character set, each binary value between 0 and 127 is given a specific character. Most computers extend the ASCII character set to use the full range of 256 characters available in a byte. The upper 128 characters handle special things like accented characters from common foreign languages.

You can see the 127 standard ASCII codes below. Computers store text documents, both on disk and in memory, using these codes. For example, if you use Notepad in Windows OS to create a text file containing the words, "Four score and seven years ago," Notepad would use 1 byte of memory per character (including 1 byte for each space character between the words --

ASCII character 32). When Notepad stores the sentence in a file on disk, the file will also contain 1 byte per character and per space.

Try this experiment: Open up a new file in Notepad and insert the sentence, "Four score and seven years ago" in it. Save the file to disk under the name getty.txt. Then use the explorer and look at the size of the file. You will find that the file has a size of 30 bytes on disk: 1 byte for each character. If you add another word to the end of the sentence and re-save it, the file size will jump to the appropriate number of bytes. Each character consumes a byte.

If you were to look at the file as a computer looks at it, you would find that each byte contains not a letter but a number -- the number is the ASCII code corresponding to the character (see below). So on disk, the numbers for the file look like this:

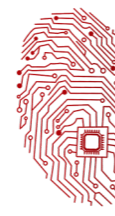
Four and Seven (example)

F	o	u	r	<spc>	a	n	d	<spc>	s	e	v	e	n
70	111	117	114	32	97	110	100	32	115	101	118	101	110

By looking in the ASCII table, you can see a one-to-one correspondence between each character and the ASCII code used. Note the use of 32 for a space -- 32 is the ASCII code for a space. We could expand these decimal numbers out to binary numbers (so 32 = 00100000) if we wanted to be technically correct - that is how the computer really deals with things.

The first 32 values (0 through 31) are codes for things like carriage return and line feed. The space character is the 33rd value, followed by punctuation, digits, uppercase characters and lowercase characters. To see all 127 values try Google "ASCII codes".





Topic: 1.1.1 Number representation

UNICODE:

Unicode is a computing industry standard for the consistent encoding, representation and handling of text expressed in most of the world's writing systems or languages. Developed in conjunction with the Universal Character Set standard and published in book form as The Unicode Standard, the latest version of Unicode consists of a repertoire of more than 107,000 characters covering 90 scripts for the correct display of text containing both right-to-left scripts, such as Arabic and Hebrew, and left-to-right scripts).

The Unicode Consortium, the nonprofit organization that coordinates Unicode's development, has the ambitious goal of eventually replacing existing character encoding schemes like ASCII with Unicode as many of the existing schemes are limited in size and scope and are incompatible with multilingual environments. Before 1996 UNICODE used 16 bits (2 bytes) however after 1996 size was not restricted to 16bits and enhanced further to cover every possible variation of multilingual environment.

Notes: All the characters that a system can recognise are called its character set.

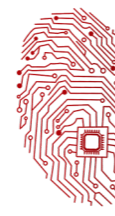
ASCII uses 8 bits so there are 256 different codes that can be used and hence 256 different characters. (This is not quite true, we will see why in chapter 1.5 with reference to parity checks.)

A problem arises when the computer retrieves a piece of data from its memory. Imagine that the data is 01000001. Is this the number 65, or is it A?

They are both stored in the same way, so how can it tell the difference?

The answer is that characters and numbers are stored in different parts of the memory, so it knows which one it is by knowing whereabouts it was stored.





Topic: 1.1.1 Number representation

Binary Coded Decimal (BCD):

Some numbers are not proper numbers because they don't behave like numbers. A barcode for chocolate looks like a number, and a barcode for sponge cake look like a number, but if the barcodes are added together the result is not the barcode for chocolate cake. The arithmetic does not give a sensible answer. Values like this that look like numbers but do not behave like them are often stored in binary coded decimal (BCD). Each digit is simply changed into a four bit binary number which are then placed after one another in order.

e.g. 398602 in BCD

Answer: 3 = 0011 9 = 1001
 8 = 1000 6 = 0110
 0 = 0000 2 = 0010

So 398602 = 001110011000011000000010 (in BCD)

Note: All the zeros are essential otherwise you can't read it back.

Application

The BIOS in many personal computers stores the date and time in BCD because the MC6818 real-time clock chip used in the original IBM PC AT motherboard provided the time encoded in BCD. This form is easily converted into ASCII for display.

The Atari 8-bit family of computers used BCD to implement floating-point algorithms. The MOS 6502 processor used has a BCD mode that affects the addition and subtraction instructions.

Early models of the PlayStation 3 store the date and time in BCD. This led to a worldwide outage of the console on 1 March 2010. The last two digits of the year stored as BCD were misinterpreted as 16 causing an error in the unit's date, rendering most functions inoperable. This has been referred to as the Year 2010 Problem.

