

What is Binary? What is Hex?

Mark Terjeson
Russell, Tacoma, WA

ABSTRACT

If anyone is interested in what a megabyte is or how it got to be that way, what number bases are, what binary or hex means, or how to convert them, this will get you started!

INTRODUCTION

Once upon a time, there was the idea of a computer. Big ideas of Little tiny on/off switches. Early on there were four. Then somebody got excited and said, "Let's double what we got". Then there were eight Little tiny on/off switches (much like rabbits, except rabbits are exponential) (see another type of math class for the explanation on that).

Since we have electrical on/off switches, that gives us the possibility of representing two states. On or Off (of course, this is old news, I know!). But we humans could think of it as True/False, Yes/No, Good/Bad (oops, how'd that get in there!), One/Zero, etc. So numerically they decided on 1 and 0.

So remember back in 5th or 6th grade when we learned about Number Base. You know, that Base 10 stuff.

BASE 10

Base-10, if we look at each digit position in the number 123, that's one-hundred-twenty-three. We have a 3 in the ones position, a 2 in the tens position, and a 1 in the hundreds position. So visually it could be represented like this: (you remember all this stuff of course)

10,000	1,000	100	10	1
-----+	-----+	-----+	-----+	-----
		1	2	3

Yes, this looks all too familiar. Well in the scheme of things (this number base stuff) each position can have only one digit, letter, or symbol. Example: in the ones position we can have:

1	

0	meaning zero or nothing
1	meaning one
2	meaning two
3	
4	
5	
6	
7	
8	
9	meaning nine

Okay now, we said base 10, so we now have used up ten symbolizations (the 0-9). So now what do we do? (I know you're saying, yeah, yeah, yeah, we know this. But we gotta due thys stuph sew thah necst pahrt maques cents.) So we'll represent a ten by placing a 1 in the tens position.

10	1	
-----+	-----	
1	0	meaning one ten count and no ones

Yup, the highest representation in each column is one-less than the number base you are working with. Another interesting fact is that you can determine what each column heading is by multiplying the preceding lesser column times the number base value. i.e. to get the tens column multiple 1 times 10 and you get 10, thus the header for the second column. To get the third column multiply the second column 10, times the number base 10 and you get 100 for the header of the third column. Yep, $100 * 10 = 1000$, $1000 * 10 = 10000$, and so on, and so on, and so on, <Energizer Bunny® enters here>.

NUMBER BASES

Yeah, yeah, yeah, we know all this stuff, why bore us? Well, you can't believe how many people I've come across that know this stuff and then turn around and tell me they do not have a clue how to do other number bases!! They say, what is this octal stuff, hex mash, binary goop?

Okay since you asked . . .

Standard number prefixes are commonly used to assist in the description of the, or ability to remember very large or very small, units of measure. Milligram/kilogram, kilometer/centimeter. So it is with our most familiar number system "Decimal", otherwise known as Base-10. From the Greek words we have Deca which means 10, Octal 8, Hexa 6, and Penta 5, etc. These are combined at times such as HexaDecimal which means six and ten to become sixteen. Hexadecimal is also known as a number system with a base of 16, or Base-16, as is binary also known as Base-2, etc.

HOW CAN THIS BE HELPFUL TO ME?

At face value you may think number bases will be of no interest to you, but they actually come into play in many aspects of our lives, especially in the computer world. For the casual user you will encounter times that you need to convert hexadecimal to decimal, or figure out how to get the decimal value of some binary number. There are countless times that it would be handy to be able to read hexadecimal numbers as easily as we read decimal numbers. Even in SAS®, to conserve space and to easily address memory positions, we see hex numbers such as when we get an invalid data error message in our log. The line affected is displayed showing position, the character-letter representations, and the actual byte value of each character so that you are assisted in debugging your error. This byte value in your log uses hex representation because it only requires two letters and conserves display space and is uniform in size which is also advantages for display purposes. Another example is when an application gives you a big integer value that contains multiple stored elements and you need to figure out how to reverse engineer the individual pieces out of a single large decimal number. You actually already know what number base systems are and you use them every day and may not know it. Continue reading for a reminder of what number base systems are and how to read and write them.

BASE 5

Let's use the same rules as above for Base-10 and let's do Base-5. Okay, we gotta always start out with a ones column if nothing else, and we can only have a count of symbol representations for as many as our number base states, thus for Base-5 we can only have five symbols, letters, or digits:

```
1
-----
0      meaning zero or nothing
1      meaning one
2      meaning two
3      meaning three
4      ... and we've used up all our positions.
```

Also remember that our biggest representation is one-less than our number base. One of our other rules was to get the column heading for the next larger column we multiply the previous column (1) by the number base. $1 * 5 = 5$ So our next column heading is 5. The next column heading would be the last column(5), times the number base 5 and you get $5 * 5 = 25$.

So we see the headings starting to look like:

625	125	25	5	1	
-----+	-----+	-----+	-----+	-----	
				0	meaning zero or nothing
				1	meaning one
				2	meaning two
				3	meaning three
				4	meaning four

so 4 plus one more means that we can't go bigger in the ones column,
so add one, carry the 1, and we get:

625	125	25	5	1	
-----+	-----+	-----+	-----+	-----	
			0	0	meaning zero or nothing
			0	1	meaning one
			0	2	meaning two
			0	3	meaning three
			0	4	meaning four
			1	0	meaning one-5 and no-1s equals 5
			1	1	meaning one-5 and one-1 equals 6
			1	2	
			1	3	
			1	4	
			2	0	can't go bigger so Carry-the-1
			2	1	
			2	2	meaning two-5s and two-1s equals twelve
			2	3	meaning 2*5 and 3*1 equals 13
			2	4	meaning 2*5 and 4*1 equals 14
			3	0	meaning 3*5 and 0*1 equals 15
			3	1	meaning 3*5 and 1*1 equals 16

Yeah, ... you're gettin' it.

"14302" base5 would equal:

you guessed it:

625	125	25	5	1
-----+	-----+	-----+	-----+	-----
1	4	3	0	2

or

$$\begin{array}{r}
 1 * 625 = 625 \\
 4 * 125 = 500 \\
 3 * 25 = 75 \\
 0 * 5 = 0 \\
 2 * 1 = 2 \\
 \hline
 \text{equals } 1202 \text{ in base10}
 \end{array}$$

BASE 8

Base8 (octal) you say, same rules, 1 times base8 = 8, $8 * \text{base8} = 64$, $64 * \text{base8} = 512$, $512 * 8 = 4096$, ...

Base8:

4096	512	64	8	1	
-----	-----	-----	-----	-----	
		0	0	0	= 0
		0	0	1	= 1
		0	0	2	= 2
		0	0	3	= 3
		0	0	4	= 4
		0	0	5	= 5
		0	0	6	= 6
		0	0	7	= 7
		0	1	0	= 8
		0	1	1	= 9
		0	1	2	= 10
		0	1	3	= 11

BASE 16

Base16 (hexadecimal, or commonly known as hex), same rules, 1 times base16 = 16, $16 * \text{base16} = 256$, $256 * \text{base16} = 4096$, $4096 * 16 = 65536$, ... hmmm, nice chunks and increments of 8 again. Recognize many of the numbers?

Base16:

65536	256	16	1	
-----	-----	-----	-----	
		0	0	= 0
		0	0	= 1
		0	0	= 2
		0	0	= 3
		0	0	= 4
		0	0	= 5
		0	0	= 6
		0	0	= 7
		0	0	= 8
		0	0	= 9

now remember our rule is that in the current base16 we can have 16 single character symbols in each column and we just used up all ten of our digit symbols, so we will start with the letters of the alphabet to take over, so:

0	0	A	= 10
0	0	B	= 11
0	0	C	= 12
0	0	D	= 13
0	0	E	= 14
0	0	F	= 15
0	1	0	= 16 plus one was Carry-the-1
0	1	1	= 17
0	1	2	= 18
0	1	3	= 19
0	1	4	= 20 ($1*16 + 4*1$)

256	128	64	32	16	8	4	2	1
---	+	---	+	---	+	---	+	---
1	0	0	0	0	0	0	0	0

ANY OF THESE NUMBERS SOUND FAMILIAR YET?

Example for fun: (all of these represent the same decimal value of one-thousand-two hundred-two)

Decimal	Base10: 1202	$(1*1000 + 2*100 + 0*10 + 2*1) = 1202$
Hexadecimal	Base16: 4B2	$(4*256 + 11*16 + 2*1) = 1202$
Octal	Base8 : 2262	$(2*512 + 2*64 + 6*8 + 2*1) = 1202$
	Base5 : 14302	
	Base3 : 1052112	
Binary	Base2 : 10010110010	

See, you can write or convert any number base you want. Didn't know you already knew how, did ya?

Remember when those computer developers doubled their 4 switches and got 8? Well, those computer developers decided to double again and they got 16! Using the same rules above, the largest representation of 16 bits of binary base2 counting is:

32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	= 0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	= 1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	= 2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	= 3
																.
																.
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	= 65532
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	= 65533
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	= 65534
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	= 65535

any of these numbers sound familiar yet?

Maximum number of rows in an Excel2000 or an Excel97 spreadsheet - 65535. (Whoa, where have we just seen that!) I may not get my version numbers just exactly right, but Excel 95 or 4.0 had 32767 rows. Maximum string length in SAS v8 is 1 thru 32767 (the 32768th reserving zero). SAS number variables are 8. Many products reserve the subscript maximums of 255, meaning 256 values from 0-255. Excel 4.0 or 2.0 had 16384 rows. Lotus and other spreadsheets in the 1980's had 8192 and 4096 rows. In the 1970's early spreadsheets had 1024 rows. Other SAS maximum values: char 255, unsigned char 255, short 32767, signed short 32767, unsigned short 65535, etc.

any of these numbers sound familiar yet? (look at the previous table above) they're everywhere!

COMPUTER LINGO

What is a Bit, Nibble, Byte, Word, DoubleWord, Kilobyte, Megabyte, or Gigabyte?

A single on/off or 0/1 is called a "bit" in computer lingo. A chunk of 8bits is called a "byte". Two bytes for a chunk of 16bits is called a "word". Four bytes is a double-word. A half-byte is a "nibble".

(Many of you folks know all of this stuff but there are many who are new to this computer world that haven't got all these pieces yet, so bear with this, and everyone can just add to their repertoire the little pieces they didn't quite have yet.)

[illegible]

Then extending the column headings out to the first bit of byte three would be 65536 or in binary would display as 00000001 00000000 00000000. So what would 11111111 00000000 00000000 represent?

Let's practice identifying our boundaries and column headings for a moment.

Note that we have other shortcut representations to make things easier. If our binary column headings of

256	128	64	32	16	8	4	2	1
---	+	---	+	---	+	---	+	---
1	0	0	0	0	0	0	0	0

can be also rewritten as 2 raised to the power of x, where x is the number of bits used, or the number bit positions as you shift left. e.g. the binary 00000000 is bit positions 76543210.

```

256 = 2^8
128 = 2^7
64  = 2^6
32  = 2^5
16  = 2^4
8   = 2^3
4   = 2^2
2   = 2^1
1   = 2^0

```

If that is the case, then the digit 1 in 00000001 00000000 00000000 represents the least significant bit in the most significant of the three bytes, or bit17. The column heading (our decimal unit of measure) can either be figured out by starting at 1 and multiplying them out one column at a time, such as 1 2 4 8 16 32 ... until we get to the 17th column, or we can use math to do it faster. The column label for bit17 is 2^16 or 65536. Remember that zero is a valid integer, so two bytes represents 0(00000000 00000000) through 65535(11111111 11111111). The next column for 1 00000000 00000000 would be 65536 or 2^16. So what would 11111111 00000000 00000000 represent?

```

2^23 or 8388608 is 1 * 8388608
2^22 or 4194304 is 1 * 4194304
2^21 or 2097152 is 1 * 2097152
2^20 or 1048576 is 1 * 1048576
2^19 or 524288  is 1 * 524288
2^18 or 262144  is 1 * 262144
2^17 or 131072  is 1 * 131072
2^16 or 65536   is 1 * 65536

```

which adds up to 16711680

Binary 11111111 00000000 00000000 equals 16711680 decimal.

Using hex makes the binary values and boundaries easy to work with. If 0000binary equals 00hex then 1111binary equals FFhex. Thus, 11111111 00000000 00000000 is FF 00 00 in hex representation.

For our ID and flag values out of the decimal value 193273528322 the hex representation of decimal 193273528322 is 000000000000002D0000000000000002, for visualizing the WORDs we can use spaces and display as 00000000 0000002D 00000000 00000002. We can either use our scientific calculator or we can verify by hand as the bigint being two INTs concatenated together.

The upper INT is: (the 2^32 representing the 2D being shifted up (or left) by 4 bytes.)
00000000 0000002D hex (45decimal) * 2^32 equals 193273528320

The lower INT is: (the rightmost or lower 4 bytes.)

```

00000000 00000002 hex ( 2decimal)

```

which adds up to our 193273528322

Our task was to reverse engineer the decimal number to get the two separate pieces, the ID value and the flag value. You can either use one or more function calls to do the arithmetic for you if available or you can do the simple arithmetic yourself. Raise to the power capability for your platform can be either a special symbol or a function. For the data dataset: (Note: in SAS 2^{32} is written as 2^{**32})

```
data _null_;
    orig_val = 193273528322;
    ID = int(orig_val/(2**32));
    flag = orig_val - (int(orig_val/(2**32))*(2**32));
    put _all_;
run;
```

For your SAS SQL pass-through code to a SQL Server the syntax would be:

```
Select
    convert(bigint,orig_val/power(2,32)) as ID,
    (orig_val -(convert(bigint, orig_val /power(2,32))* power(2,32))) as flag
```

For performance you can save the call to the POWER() function by hard coding the constant:

```
Select
    convert(bigint,orig_val/4294967296) as ID,
    (orig_val -(convert(bigint,orig_val/4294967296)*4294967296)) as flag
```

So knowing how to count the units of measure in any number base system and knowing how to chop out chunks by merely dividing by the smallest unit of measure in the desired chunk works no matter what number base you need to work with. You've been doing it all your life in decimal, and in your head at that, so using the same rules as base 10, base 2 and base 16 can come to life for you and your programs.

CONCLUSIONS

See, you knew this stuff all the time. Yes ya did! You practice 1, 10, 100, 1000 everyday. Since you know all the rules, just change the unit of measure and you will be more fluent than you thought.

Thus, Mary had a Little Lamb and her fleece was white as, Little Red Riding Hood's, candlestick, where three men in the tub, said they'd blow your house in, and may your bytes live happily ever after!

ACKNOWLEDGMENTS

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.
Copyright 2005 Energizer Holdings, Inc.
Other brand and product names are registered trademarks or trademarks of their respective companies.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Mark Terjeson
Email mterjeson@russell.com or mtnw@comcast.net