

LEARNING BINARY AND HEXADECIMAL

A Beginner's Guide to Talking like a Computer

Revision 1.3

PREFACE

If you have been programming for either a few months or a few years, chances are you have come across the terms **binary** (bin for short) and **hexadecimal** (hex for short) at some point in your career. For the programmers that have begun their careers in the years of late a thorough understanding of these enigmatic topics are typically known only to well-seasoned programmers. In contrast, some of the older, more experienced programmers were shown this to be a fundamental concept when programming for computers just to get their job done. This shift in thought results in less programmers learning binary and hex nowadays.


So, you may ask, with all the abstraction of today's high level programming languages and libraries, why should one bother with learning all of this binary and hex mumbo jumbo anyway? Well, to truly understand the eclectic nature of a computer (data storage, memory, TCP/IP, debugging, cryptography, bit-planes, compression, etc.) and the "how's" and "why's", an understanding of binary and hex is required. The truth is, virtually everything dealing with computers can be traced back to bits, bytes, memory, and CPU register addressing. Once you lift the hood, so-to-speak, to any modern technology you'll find the basics staring you right in the face. As such, knowing the lingo a computer speaks will put you a step ahead should you ever encounter a problem to solve that falls outside the scope of normal day-to-day development.

INTRODUCTION

What exactly is binary and hexadecimal? It may sound like an easy question, but do you really know? Unfortunately, sometimes things are assumed known in the programming community, and nobody bothers asking. When I was first learning, someone had to tell me just as I am telling you now. So, if you are unsure, then binary and hex are simply different counting systems.

Yup, that's it! But wait a minute... What's a counting system? Well, it's the way we as humans count varying amounts. How many ears do you have? Two I presume, correct? You are using a counting system to be able to represent that amount.

All systems of counting and writing ever used contain what we call glyphs to represent values, whether we are counting with roman numerals or ancient hieroglyphics. A glyph is basically a fancy way to say picture or character, but with the purpose to represent something in writing. Even today, we still use glyphs to symbolize meaning just as we did back in ancient times. The difference being the numbers we're used to seeing have Arabic origins, and we have been accustomed to seeing them in numbers since birth practically so we don't have to think about it much.

For instance, to represent a value of ten the way you have always done, you will most likely use a combination of Arabic-based pictures of a vertical line and then a circle next to it resembling this: **10**, but in Egyptian hieroglyphics you would use picture of an arch:  Most of us in recent years are just

not used to seeing it done this way, but perhaps if we went back in time, the **10** would look just as odd to the general populous.

Fast forwarding to today, the counting system you've been using your entire life is called **decimal**. It has the prefix "dec", which literally means **ten**. In the decimal counting system, we have only ten glyphs to work with. To represent a value in decimal, we are thus limited to ten choices in total: **0, 1, 2, 3, 4, 5, 6, 7, 8, and 9**. Society simply agrees on an amount each of these represents, and it stops at nine since zero or none is also a valid value.

However, by using a combination of them in a working system, we can create any value imaginable. That is why we have counting systems in the first place, and it is much easier to remember working system as opposed to memorizing billions of different glyphs for every amount ever needed.

Did you know?

Why have you been counting in decimal your whole life? For my theory, look at how many fingers you have. While history does not tell us the exact story, we do know that people were counting before we had labels and terminology for it. It's not a far stretch to hypothesize a ten-based mentality due to this.

THE BASICS

First, we need to cover something you already learned in the first grade – how to count to ten. You might say, "no sweat, been doing it all my life" right? Well, therein lays the problem for most people when learning binary or hexadecimal. You have been counting so long that you do not need to even think about numbers anymore; you just know the numbers. It helps to remember what you were taught as a child and to almost unlearn what you understand about the way you use numbers to count. The following will introduce you to a long-lost concept:

Me: How much is a dozen?

You: Why it's twelve of course!

Me: Yes, but how much is twelve?

You: Twelve is twelve, are you crazy?

Me: Perhaps, but how much is twelve, as a first grade teacher would say to a pupil?

You: She'd say twelve!

Me: No, if she's teaching a child how to count for the first time, she would say, "twelve is one set of tens and two ones."

Did a light bulb just turn on? Is it all coming back to you? Good! Remembering the basic principles of counting that you learned a long time ago is paramount. Stay in that mindset while learning!

As I mentioned earlier, in decimal (also called base 10), we have only ten glyphs or characters to work with: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. Each one of them represents a different value. Also, it is noteworthy to mention that when using glyphs for numerical purposes, it is commonplace to refer to them as digits. Now, if we started counting at zero and stopped at nine for everything, then ten digits would suffice, but that's certainly not the case.

Do you recall the concept of placeholders? They serve as containers to represent greater values. Every placeholder in a number to the left of the decimal point is ten greater than the previous one. With the number 123 (one hundred twenty-three), you should notice three. So, to represent a value greater than nine, we would use one or more placeholders (think of them as seats) that say, "If that digit parks its butt in this spot, then it is worth more than its face value alone."

Take the dozen amount from the dialog above. Twelve is more than nine, so if we want to count past nine, we must move over to the next placeholder and put the digit 1 there to indicate ten greater. We then follow it with a zero so everyone knows it belongs in the second placeholder and not the first. Now we have the amount ten, and it's easy to count to eleven, twelve, and so on.

Let's illustrate this idea. Also, for clarity's sake in this example, we will separate the placeholders with a pipe character:

1 | 2 | 3

Instead of seeing 123 as one big chunk, think of it as one set of hundreds, two sets of tens, and three ones. Man, if your first grade teacher could see you now! Now, let's look at the illustration again, but with labels for each placeholder.

Hundreds	Tens	Ones
1	2	3

Note the placeholder values; they are always ten greater than the one before it. Guess what, that notion right there is the cornerstone of every counting system on the planet. Other systems use different meanings for their placeholders, but the concept is the exact same regardless. In decimal, every placeholder is a multiple of ten greater than the previous, and that is why it is called base 10 in the first place. However, placeholder values can be swapped in and out like a new pair of socks.

Placeholders always exist, even if we don't use them. In practice we tend to only focus on the ones being used to make a number. If we wanted to show available placeholders, we could and we would still have the same number.

Where did the other six digits come from? Well, it was decided to use letters from the alphabet to represent them. We could use any glyph in the world to represent new digits, but in practice it was agreed on to borrow from letters. Even though the extra glyphs come from letters, they still act as digits now since they are intended to represent a certain value of something we all agree on.

So now we have a total of sixteen digits to work with instead of ten: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Now remember, a new counting system means we have new placeholder values. In fact, the amount of digits we have to utilize directly corresponds to the multiplicative value of the placeholders. The amount of digits and placeholder values work hand-in-hand since they are both based around the same amount, and that's also why a new system is scientifically referred to as a new base. Decimal is base 10 and hex is base 16 for that reason.

To get the ball rolling, let's count from zero to ten in hex. Ready? This should look mostly familiar, with a twist at the end.

0 1 2 3 4 5 6 7 8 9 A

What's that you say? Why is A after 9? Well, that's the digit chosen to represent the value of one greater than nine. The reason we didn't use two placeholders to indicate the number ten was because we didn't need to. Think about it, in decimal, we have ten digits. In hex, we have sixteen. What happened in decimal is that we ran out of digits to use at nine. We had nothing left to indicate a higher value, so we were forced to take advantage of the available placeholders. With hex though, we still have more digits to use. Therefore, we do not need to use a new one.

Can you guess what twelve is in hex? If you said C, you're right! If you didn't, then take a moment to stop and review a bit. The digits simply mean an amount of something. Try not to get caught up in what it actually looks like. Its look is simply a picture that we all agree on.

Now, let's count to fifteen in hex.

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

As you may have guessed, F equals fifteen. But, what if we want to count to sixteen instead? Then, we would face the same issue we did back in decimal when we counted past nine. We would run out of digits to use – F is where it stops in hex. To solve the problem of no more digits, we place a 1 in the next placeholder over.

In decimal, each placeholder (seat) is ten greater than the previous. Well, in hex, it is sixteen! Starting to see a similarity here? Using the pipe character again to separate placeholders let's write the number sixteen in hex.

1 | 0

It may look like ten, but it's not since we are in a new system with a new placeholder value as its base. So, this should be read as, "one set of sixteen and zero ones."

There is always a direct correlation between the amount of digits in a counting system and the value of its placeholders. This makes up the base. In decimal, every placeholder is times ten because of the ten digits we can use. In hex, every placeholder is times sixteen because we have sixteen digits to use. That's the nature of the game. Placeholders come into action when digits run out; consequently, they will always be the number of digits in your counting system times greater than the previous placeholder used in the number.

To solidify this, let's take a look at a number in hex and decimal, with labeled placeholders. Notice how the same digits are used, but they are interpreted differently.

Deci mal	=	Tens	Ones
Hex	=	Si xteens	Ones
Di gi ts	=	2	0

Here, we have the number 20. But, this can represent two different values depending on if we are using decimal or hex. In decimal, it is the same as saying $(2 \times 10) + (0 \times 1)$. The parentheses can be thought of as representing placeholders. With every new placeholder, we have the base number (ten for decimal; sixteen for hex) times the position of the placeholder.

If I wanted to show the number zero for instance, using decimal with four placeholders, I could say: $(0 \times 1,000) + (0 \times 100) + (0 \times 10) + (0 \times 1)$. Or, if I'm bold enough to show the number zero with four placeholders in hex it would look like this: $(0 \times 4,096) + (0 \times 256) + (0 \times 16) + (0 \times 1)$.

Now, looking at the number 20 above when translated as hex we get: $(2 \times 16) + (0 \times 1)$. Look at the placeholders now. They are incrementing in steps of sixteen. If you did the math, should notice that 20 in hex is 32 in decimal. Even if the characters look the same, when you change the values of the placeholders, you change the value of the entire number.

Here's one of my favorite illustrations. Let's use a different twist and count to the number thirty-three in both decimal and hex side-by-side.

Decimal	Hex	Decimal	Hex	Decimal	Hex
1	1	12	C	23	17
2	2	13	D	24	18
3	3	14	E	25	19
4	4	15	F	26	1A
5	5	16	10	27	1B
6	6	17	11	28	1C
7	7	18	12	29	1D
8	8	19	13	30	1E
9	9	20	14	31	1F
10	A	21	15	32	20
11	B	22	16	33	21

I hope, by now, you see a pattern emerging. Look at the decimal number 26 in the chart. Can you tell me why the hex equivalent is 1A? If you can, give yourself a pat on the back. If it's not that clear at the moment, don't worry. Take a break and come back to reread the hexadecimal section of this document again. Just think about sets of sixteens ($\ast 16$) + ($\ast 1$) for the placeholders and using more digits for each seat. It will eventually make sense if you think it out.

Did you know?

To become better at understanding hex or any counting system for that matter, you must practice. Once you understand hex really well, binary will be a snap. So, here are two exercises to try to get you started.

- Convert the following numbers to hex. 45, 8, 52, 23, 71, and 100.
- Why does the decimal number 255 equal the hex number FF?

BINARY

The prefix **bi** means two, so as you may have guessed binary is also called base 2 and **bin** for short. That means, we have only two digits to work with: 0 and 1. And every placeholder is only two times greater than the previous one. We have no letters to mess with this time around!

This time let's dive straight in head first and count in decimal and binary first side-by-side only to explain it later. I'll be using the same concepts, just under a different counting system. The idea behind this is to see something strange at first, and then apply your newly learned principles to have it become familiar to you instantly and reinforce how systems work.

Deci mal	Bi n
1	1
2	10
3	11
4	100
5	101

Deci mal	Bi n
6	110
7	111
8	1000
9	1001
10	1010

Deci mal	Bi n
11	1011
12	1100
13	1101
14	1110
15	1111

Just like 9 is the magic digit in decimal and F is the magic digit in hex, 1 is the magic digit in binary. Because we only have two to choose from we have to use placeholders like they're going out of style. So, after we've counted to just one we no longer have any more digits to use. We have to resort to taking advantage of the placeholders. And as you would expect, every placeholder is two times greater than the previous one.

Let's compare a number in all three systems at once. For this example, we'll use the digits 10. Each counting system produces a unique value.

Deci mal	=	Tens	Ones
Hex	=	Si xteens	Ones
Bi n	=	Twos	Ones
Di gi ts	=	1	0

In decimal, everyone knows 10 equals ten. In hex 10 is equal to sixteen. Now, in binary, since we go by twos 10 equals two. It's the same as saying, "one set of twos and zero ones" because of the value our placeholders represent. In binary, since you only have two digits, every time you count two numbers higher, you have to use a placeholder. Place holder usage increases dramatically because there are so few digits to work with.

If you look back at the chart above that counts to fifteen in binary, take notice that in every two steps, the next available placeholder is filled with 1. And, don't forget the importance of zeros! Leading zeros are ignored, but not trailing zeros. You can't do anything with 0 except to reserve a placeholder. The only available digit in binary with any real value is 1. After it's used, it's time for a new placeholder to help out.

Once again, here's a chart that counts from zero to seven in binary, but this time, it shows several placeholders instead of just assuming them, and then we separate them for illustration.

Thir ty-Twos	Si xteens	Ei ghts	Four s	Two s	One s
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	0	1
0	0	0	1	1	0
0	0	0	1	1	1

Take a moment to think about the points that we have discussed. Now, by using the charts as your aide, can you tell me what **1100** in binary is in decimal? If you said twelve, you're right. If you said something else, remember the formula: $(1 \times 8) + (1 \times 4) + (0 \times 2) + (0 \times 1)$.

Did you know?

Now that we have answered the "what", we should think about the "why" we use binary. At the heart of it all, computers only communicate with numbers. In the digital age it comes right down to simple digits. Computers represent those numbers with electric states in the circuitry. Electricity is simple in nature, you either have a current or you do not. Yes and no or on and off, as it were. Binary is perfect to represent this as each digit can show a state of either being on or off. 0 is no electric current, and 1 means a current is present. To simplify the concept, a computer can realize it is dealing with the number three with two jolts being transmitted (11 in binary).

CONCLUSION

Congratulations, whether you realize it or not, you just learned the basics of hex and binary. If you take a step back for a moment, you may realize all the letters, zeros, and ones combinations now actually have some meaning to it. In time, they won't even appear random anymore.

To sharpen your newfound skills, you have to practice them – even with larger numbers. I kept them small for the sake of clarity and illustration, but becoming familiar with larger numbers is essential when working with computers in today's world.

If you are using Windows, open up the calculator program by using the Windows key on your keyboard. Simply type **⊞** +R, input **calc**, and then hit Enter. Set the program **View** to **Programmer**, and you'll be able to work with and convert between decimal, hex, and binary numbers.

Once you are a seasoned pro with hex and binary, you can also use the same exact placeholder and digit concept to count in any system in the world that you want, like **ternary** (base 3), **octal** (base 8), or **duodecimal** (base 12).