CertificationKits

# How & Why We Subnet
# Lab Workbook

**CertificationKits.com How & Why We Subnet Workbook**

Copyright 2013 CertificationKits LLC

All rights reserved. No part of this book maybe be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without the written permission from the publisher, except for the inclusion of brief quotations in a review.

Printed in the United States of America

Third Printing March 2013

Library of Congress Cataloging-in-Publication Number:

ISBN: 978-1-60458-796-8

**Warning and Disclaimer**
This book is designed to provide self-study labs to help you prepare for your Cisco CCNA certification exam. Every effort has been made to make this book as complete and accurate as possible, but no warranty or fitness is implied.

The information is provided on an "as is" basis. The author, CertificationKits LLC, and publisher shall have neither liability nor responsibility to any persons or entity with respect to any loss or damages arising from the information contained in this book for from the use of the discs or programs that may accompany it.

**Trademark Acknowledgements**
All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. The publisher, author or CertificationKits LLC cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

# Contents

# How to Use This Workbook

There are many publications out there that will just focus on how to do subnetting to pass your Cisco exam. But they really do not explain with real world scenarios of why we subnet. There are also many other publications out there that explain subnetting in a 15 page chapter. I am sure you know from experience that a simple 15 page chapter does not cover subnetting in depth. It is just the same old information regurgitated by various authors. So we feel you are going to be pleased with what we have put together for you. Instead of just telling you how to convert to binary, decimal, find the number of subnets and valid hosts, we are going to tie all these concepts together with real world examples and lots and lots of practice.

By the time you are done reading this will you have had enough practice to confidently walk into your Cisco certification exam and ace every subnetting question. Additionally you will also really understand in depth why we subnet and how the data travels between nodes at the packet level. So this is not a quick read. Take your time and go through all the review questions and make sure you understand them fully.

So without further ado, we are going to jump right in and start off detailing base concepts of IP addresses, IP address classes and default subnet masks. As we progress in the chapter, we will start to discuss VLSM (variable length subnet masks) so you can see how to alter subnet masks to meet the requirements of a specific number of hosts, subnets and a combination of hosts and subnets. Some of these items can be explained using more than one method. In those cases, we may show you a method that is pure mathematical such as *x* to the 2nd power or an alternate method using a chart as a visual aid so you can "see" it. We will do this all the while providing you the real world examples and practice questions you need to really understand subnetting!

# Chapter 1

## What Are IP Addresses & Why Do We Need Them?

## The Internet

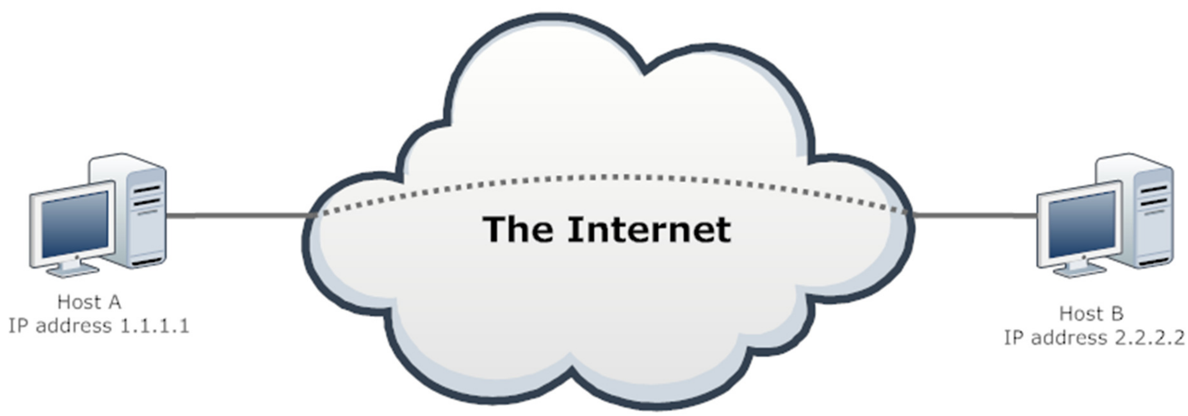
Originally, the goal of the Internet was to enable government branches, facilities and universities to connect to each other and share information and research. It has come a long way since then as nearly every modern home in the world has Internet connectivity.

Consider the fact that all computers connected to the Internet need to send and receive data. How does that work? In the current implementation, the TCP/IP suite of protocols provides end-to-end connectivity between hosts on the Internet. However, that presents us with another problem. How do we uniquely identify each and every host connected to the Internet to insure that data is sent and received by the correct host?

## IP Addresses

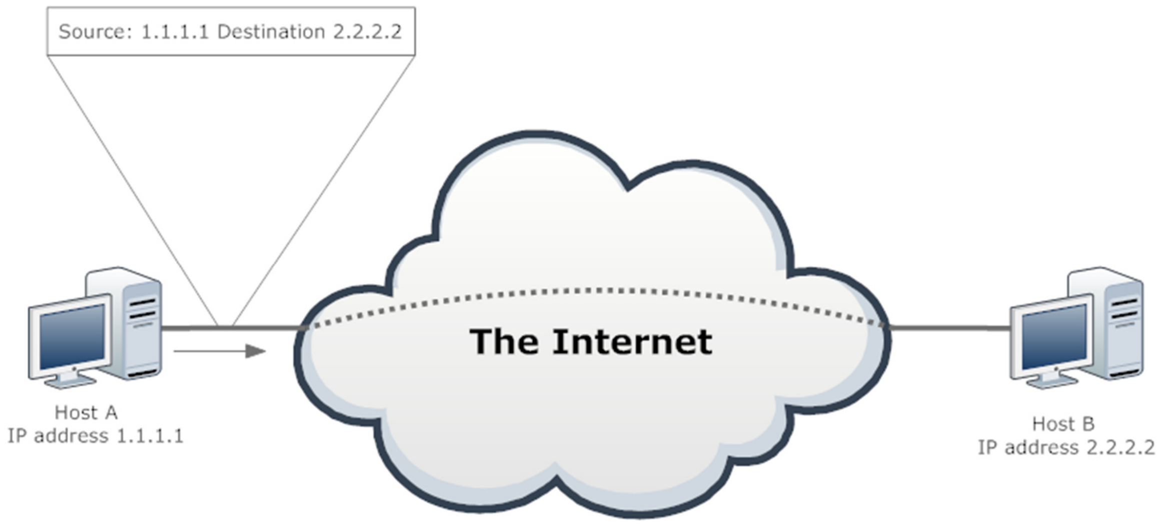
To correctly identify hosts on the Internet, IP addresses were devised. Each host is assigned an address that uniquely identifies the host on the Internet for communication.

**Figure 1.1: Two hosts connected to the Internet**




Consider Figure 1.1 above, Hosts A and B are both connected to the Internet. Assume that Host A wants to send Host B data. Since each host on the Internet has its own IP address to uniquely identify itself, Host A can specify the destination IP address it wants the data sent to as an end point. Another component of the information that will be sent in the data packet is the source IP address the packet was sent from. Figure 1.2, demonstrates this concept.

**Figure 1.2: Source and destination addressing**




Now that we have established what IP addresses are and what their purpose is, how do we keep the IP addresses organized so that no two hosts get assigned the same IP address? The Internet Assigned Numbers Authority (IANA) was created to allocate and keep track of the IP addresses assigned. They in turn delegate allocations of IP addresses to regional Internet registries (RIRs). The RIRs then divide their allocated address pools into smaller blocks and delegate them to Internet service providers and other organizations in their respective regions.

## Format of IP Addresses

IP addresses are 32 bit values. These 32 bits (32 bits equals 4 bytes) are broken up into four octets which are 8 bits each (8 bits equals 1 byte). The IP address is represented in the form of a.b.c.d where a, b, c and d are numbers in the range 0 – 255. While they are represented in regular decimal digits commonly referred to as the dotted decimal notation for human convenience, they are actually binary or base 2 and are seen by the IP devices as such.

This brings up the question of what is binary? You have probably heard of binary and bits before, which are represented using 1s and 0s. As mentioned above, binary is base 2, meaning it is comprised of 2 digits (1, 0), in comparison to decimal, which is base 10 and consists of 10 digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). This means for every place or character in a decimal number there is a possibility of 10 digits. However, in binary you only have two possibilities; the value of a single digit is either 0 or 1. We will see some examples of this shortly.

To summarize, an IP address is represented in decimal for human convenience and actual network operations are handled in binary. Thus, we will need to be able to proficiently convert from binary to decimal and from decimal to binary to really understand what is happening at the IP level.

First we need to familiarize ourselves with the basic operation of decimal and binary, which is discussed in the next section.

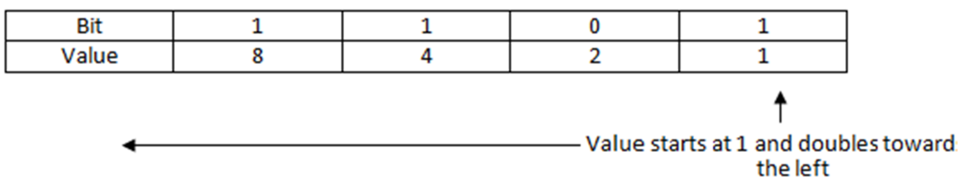
# Understanding Binary

## Converting from Binary to Decimal

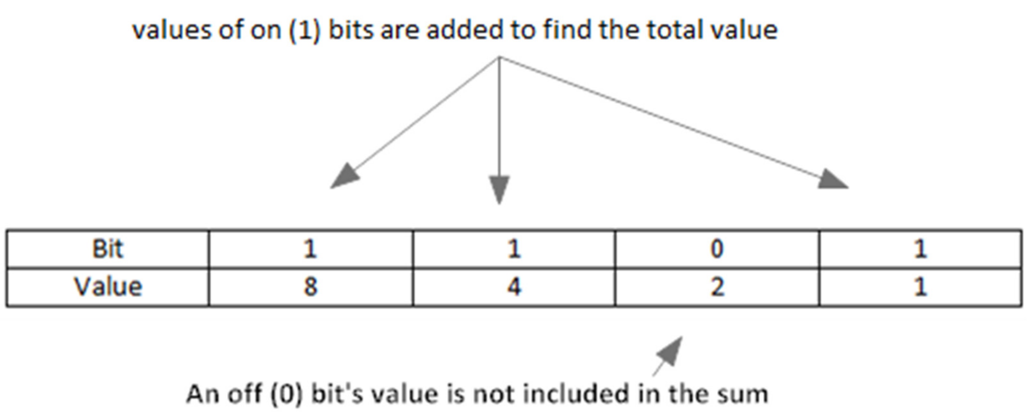
So how do we represent base 2 or binary numbers? It is quite simple. We will use the chart below in Figure 1.3 to help us explain it. The bits start from the right at a value of 1, and the value doubles as we move towards the left. Each bit has a corresponding value depending on where it resides in our chart. If the bit has a value of 1, it is considered on and that value will be included in our total. If the bit has a value of 0, it is considered off and it is not included in our total.

For example, consider the binary number 1101. Let us start from the rightmost bit, which is a 1. Since it is the rightmost bit, it represents a value of 1 and signifies that it is (on). The next bit towards the left is a 0, which has a value double to that of the first bit, 2. Notice that it is a 0 (off) however, meaning that it does not factor in the total value. The third bit is a 1 (on) and again represents double the value of the previous bit, 4. The fourth and final bit is double the third bit; it has a value of 8.

**Figure 1.3: Binary Chart**




Once we have figured out which bits represent what values, we add the values represented of the on bits (1s) to find out the decimal representation of the binary bits.



Adding the values of the on bits is the next step.

| Bit | 1 | 1 | 0 | 1 |
|---|---|---|---|---|
| Value | 8 | 4 | 2 | 1 |

8 + 4 + 1 = 13

Therefore, binary string 1101 is equal to 13 in decimal. As you have seen in the demonstration above, it is a simple process. Let us consider another example to familiarize ourselves with the process.

Convert the binary string 01001101 to decimal.

First we build a blank chart similar to the one we used above with the values starting at one on the right and doubling towards the left.

**Figure 1.4: Blank Binary Chart**

| Bit | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

Then we will fill in the binary numbers in the chart as shown in Figure 1.5. Again, values of on bits (1) are added to find the total value and values of off (0) bits are ignored. Then we will add up the value of all the on bits to reveal the decimal value of our binary number as shown in Figure 1.5.

**Figure 1.5: Binary Chart**

| Bit | **0** | **1** | **0** | **0** | **1** | **1** | **0** | **1** |
|---|---|---|---|---|---|---|---|---|
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

0 + 64 + 0 + 0 + 8 + 4 + 0 + 1 = 77

Thus the binary string 01001101 has a value of 77. In this case, note that the string started with a 0, 0s in string beginnings are called "leading zeros." Leading zeros have no significance whatsoever on the actual value and could be omitted; for example, 000010011 is the same as 10011. However, they usually listed due to a syntax requirement, such as in IP addresses. IP addresses are composed of 32 bits divided into 4 octets (an octet is an 8 bit string). Hence, if there are leading zeros, they should be included to have a complete 32 bit address. For example; assume we have a 32 bit IP address with the string 00101100 00001101 11010100 00101101. We can omit the leading zeros and represent it as 101100 1101 11010100 101101, but that may not be recognized properly.

## Converting from Decimal to Binary

The process of converting decimal to binary is a little different from converting binary to decimal. Instead of having the bits already in the columns and summing up their values to get the decimal number; we start off with a decimal number and determine which bits are 1s and which are 0s in our binary column chart. We will do this by placing a 1 (or as some may say, turning on the bit as IP devices see bits represented by 1s as on and bits represented by 0s as off) in the highest binary column that can be turned on while subtracting the column value from the decimal number we are converting without creating a negative remainder. We will repeat this process until we have a decimal remainder of zero. I know that may sound a little confusing, but I am sure a few examples will clarify it for you and in no time you will find the process quite simple.

For clarification, consider this example.

Convert 52 to binary. We will use the same table.

| Bit | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

Since we need to find out what the binary string is, we will fill in the bits fields following the procedure demonstrated previously.

Can we subtract 128 from 52 without resulting in a number less than zero (negative number)? No. That would not be possible, because we will get -76. So put a 0 bit in the 128 value and try the next value.

| Bit | **0** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

Can we subtract 64 from 52 without resulting in a number less than zero? No. That would not be possible, because we will get -12. So put a 0 bit in the 64 value and try the next value.

| Bit | **0** | **0** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

The next value is 32; can we subtract 32 from 52 without resulting in a number less than 0? Yes, we would have 20 left, so we put a 1 bit in the 32 value and find the next value that we can subtract from 20 without resulting in a number less than zero.

| Bit | **0** | **0** | **1** | | | | | |
|---|---|---|---|---|---|---|---|---|
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

We now have 32 of 52, we need to allocate the remaining 20. Can we subtract 16 from 20? Yes we can, and we would have a remainder of 4. Put a 1 in the 16 value bit field and find the next value we can subtract 4 from without resulting in a number less than 0.

| Bit | **0** | **0** | **1** | **1** | | | | |
|---|---|---|---|---|---|---|---|---|
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

We still have 4 left. Can we subtract 8 from 4 without resulting in a number less than zero? No we cannot, that would result in a -4. So we put a 0 in the 8 value bit field and move on to the next one.

| Bit | **0** | **0** | **1** | **1** | **0** | | | |
|---|---|---|---|---|---|---|---|---|
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

Can we subtract 4 from 4 without resulting in a number less than 0? Yes we can! We would be left with a 0, which is exactly what we want the result to be! Put a 1 in the 4 field.

| Bit | **0** | **0** | **1** | **1** | **0** | **1** | | |
|---|---|---|---|---|---|---|---|---|
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

Now that we have a 0; can we subtract 2 from 0 without having a number less than zero? How about 1 from 0? We cannot do either, as both would result in a number less than 0. So we put 0s in their respective value fields.

| Bit | **0** | **0** | **1** | **1** | **0** | **1** | **0** | **0** |
|---|---|---|---|---|---|---|---|---|
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

Therefore, 52 is 00110100 in binary. Following the leading zeros rule, we can drop them and have an answer of 110100. Note that both are considered correct, one is just a little shorter and some may say easier to read.

As previously mentioned, IP addresses are divided into 4 sets of 8 bits known as octets (Note that 8 bits or an octet is also equal to a byte, 1 byte = 8 bits) which can range from 0-255. The tables of bit values used above demonstrate this; they make it clear why an octet is limited to a maximum total value of 255. If we turn on all the bits (set them to 1) and sum up all the values; 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 the sum will be equal to 255.

****Exam Tip –** Currently when you take the exam you are provided 90 minutes to complete it. Trust me, you will need every one of those precious minutes. So how can you afford yourself a few extra minutes? Well the exam does not start until you click the Start Exam button. Thus draw out all of your charts on the laminated boards provided to you by the testing center before you click the Start Exam button. This just might give you the extra 5 minutes you need to finish all the questions in the allotted time. Always ensure you double check your chart as if it is wrong you may get all your subnetting questions wrong!

## IP Address Conversion - Decimal to Binary

To convert IP addresses to binary we will now take the decimal numbers and convert to binary. For example, let us convert the IP address 12.89.176.155 to binary.

To see how simple it is, we will tackle it octet by octet. Since each octet has a maximum of 8 bits, we should have fields for 8 bits and their corresponding values.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

Let us consider the first octet, 12. Can we subtract 128 from 12? No? How about 64 from 12? Or 32 from 12? Or even 16 from 12? We can't subtract any of those from 12 without getting a number less than zero, so we put 0s in the respective value's bit field and keep trying to find the biggest value we can subtract 12 from, without having a result less than zero.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit | **0** | **0** | **0** | **0** | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

Can we subtract 8 from 12? Of course! That leaves us with 4. So we put a 1 in the 8 value bit field. Now can we subtract 4 from 4 without having a result less than zero? Yes! Once we have 0 left, we can just enter 0s in the remaining fields towards the right as we can't subtract anything from zero without resulting in a number less than zero.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit | **0** | **0** | **0** | **0** | **1** | **1** | **0** | **0** |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

Great! So 12 in binary is 00001100, with 4 leading zeros; which can be eliminated based on the leading zeros rule but might cause confusion because an octet is expected to have 8 bits.

The next octet is 89. We follow the same steps to convert it to binary.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

Again we find the largest value we can subtract from 89 without resulting in a number less than zero. Can we subtract 128 from 89? We cannot, so a 0 goes into that bit field. How about 64? Yes we can, and that leaves us with 25. Put a 1 in the 64 value bit field.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit | **0** | **1** | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

We are left with 25; can we subtract 32 from 25? No, so a 0 goes into that field. How about 16 from 25? Yes, so a 1 goes into that field leaving us with 9.

| Bit | **0** | **1** | **0** | **1** | | | | |
|---|---|---|---|---|---|---|---|---|
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

Now we have to find the largest value towards the right that we can subtract from 9 without resulting in a number less than 0. Can we subtract 8 from 9? Yes we can, which leaves us with 1. So a 1 goes into that value's bit field.

| Bit | **0** | **1** | **0** | **1** | **1** | | | |
|---|---|---|---|---|---|---|---|---|
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

Now what number can we subtract from 1 without having an answer less than 0? Well 4 and 2 both don't meet the requirements, so zeros go into those values' respective bit fields. We are left with 1 itself! Can we subtract 1 from 1 without resulting in a result less than 0? Of course!

| Bit | **0** | **1** | **0** | **1** | **1** | **0** | **0** | **1** |
|---|---|---|---|---|---|---|---|---|
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

So 89 is 01011001 in binary. Great, let us convert the two remaining octets, 176 and 155.

You should be getting the hang of it by now; the steps in converting decimal to binary will be summarized for the last two octets.

Can 128 be subtracted from 176? Yes? Good! That leaves us with 48; a 1 bit goes into the 128 value bit field. Can we subtract 64 from 48? No we cannot, so a 0 goes into the 64 value bit field. How about 32 from 48? Yes we can. That leaves us with 16, so a 1 goes into the 32 value bit field. Next, can we subtract 16 from 16? Yes! That leaves us with a remainder of 0. So we enter 1 in the 16 value bit field and 0s in the subsequent bit fields.

| Bit | **1** | **0** | **1** | **1** | **0** | **0** | **0** | **0** |
|---|---|---|---|---|---|---|---|---|
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

The Last octet is 155, same steps as all of the previous examples. Can we subtract 128 from 155? Yes we can, it leaves us with 27. So a 1 goes into the 128 value bit field. How about 64 from 27? Or 32 from 27? We cannot subtract either of those from 27 without resulting in a number less than 0, so 0s go into their value bit fields. How about 16? Yes, we can subtract 16 from 27. That leaves us with 11, so a 1 goes into the 16 value bit field. Can we subtract 8 from 11? Yes that leaves us with a remainder of 3, so a 1 goes there as well. Is it possible to subtract 4 from 3? No it is not, so a 0 goes in that value bit field. 2 from 3? Yes! That leaves us with 1, so a 1 goes in the 2 value bit field. Finally, we subtract 1 from 1 yielding a remainder of 0 and also putting a 1 in the 1 value bit field.

| Bit | **1** | **0** | **0** | **1** | **1** | **0** | **1** | **1** |
|---|---|---|---|---|---|---|---|---|
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

As you have seen, we basically put a 1 in the field if we can subtract the value from the number while meeting the condition of not having a result less than 0. If it doesn't meet the condition, we simply put a 0 in the value bit field and try the next value.

****Exam Tip –** When you are performing your decimal to binary conversions, your remainder at the end of the process should be 0. If you have any other remainder, you made a mistake and should redo the question.

## IP Address Conversion - Binary to Decimal

Our example will be to convert binary strings 01011000. 10110010. 00101101. 10101001 to decimal. All we have to do is fill in the bits in the respective octet fields. Again, to avoid getting overwhelmed, we will tackle this octet by octet.

| Bits | Eighth (leftmost)bit | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First(rightmost) bit |
|---|---|---|---|---|---|---|---|---|
| 1st octet | **0** | **1** | **0** | **1** | **1** | **0** | **0** | **0** |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

All we have to do is add all of the values with 1 bits set. Which are 64, 16 and 8, 64 + 16+ 8 = 88; giving this octet a total value of 88. Giving us 88.b.c.d

| Bits | Eighth (leftmost)bit | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First(rightmost) bit |
|---|---|---|---|---|---|---|---|---|
| 2nd octet | **1** | **0** | **1** | **1** | **0** | **0** | **1** | **0** |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

Get the values of bits with 1s in the bit field and add them. 128 + 32 + 16 + 2 = 178, so it is 88.178.c.d

| Bits | Eighth (leftmost)bit | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First(rightmost) bit |
|---|---|---|---|---|---|---|---|---|
| 3rd octet | **0** | **0** | **1** | **0** | **1** | **1** | **0** | **1** |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

Repeat the same steps, 32 + 8 + 4 + 1 = 45. Making it 88.178.45.d

| Bits | Eighth (leftmost)bit | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First(rightmost) bit |
|---|---|---|---|---|---|---|---|---|
| 4th octet | **1** | **0** | **1** | **0** | **1** | **0** | **0** | **1** |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

Last octet! 128 + 32 + 8 = 168, yielding a final answer of 88.178.45.168.

Basically it works the same way, just more to do.

> ****Practice Tip –** you can download our CCNA Subnet Calculator for some additional practice at http://www.CertificationKits.com/cisco-ccna-subnet-calculator/ and also check your answers with it.

**Exam Tip –** If during your conversion process you resulting answer is a 0 or 255, it does not mean your answer is necessarily incorrect. Remember that they are both valid numbers. A few examples would be 192.168.0.1 or 192.168.1.255.

## Exercise 1.1: Binary to Decimal Practice

Convert the following digits from binary to decimal.

1) 11100111.00110010.00101101.11111011

| Bits | Eighth (leftmost) | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First bit | **Total value** |
|---|---|---|---|---|---|---|---|---|---|
| 1st octet | | | | | | | | | |
| 2nd octet | | | | | | | | | |
| 3rd octet | | | | | | | | | |
| 4th octet | | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |

The decimal IP address is _______________________________________________.

2) 00001110.01101110.00011001.00101010

| Bits | Eighth (leftmost) | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First bit | **Total value** |
|---|---|---|---|---|---|---|---|---|---|
| 1st octet | | | | | | | | | |
| 2nd octet | | | | | | | | | |
| 3rd octet | | | | | | | | | |
| 4th octet | | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |

The decimal IP address is _______________________________________________.

3) 11011101.00000001.11011101.01110110

| Bits | Eighth (leftmost) | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First bit | **Total value** |
|---|---|---|---|---|---|---|---|---|---|
| 1st octet | | | | | | | | | |
| 2nd octet | | | | | | | | | |
| 3rd octet | | | | | | | | | |
| 4th octet | | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |

The decimal IP address is _______________________________________________

4) 01010000.11111101.11110100.00010110

| Bits | Eighth (leftmost) | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First bit | **Total value** |
|---|---|---|---|---|---|---|---|---|---|
| 1st octet | | | | | | | | | |
| 2nd octet | | | | | | | | | |
| 3rd octet | | | | | | | | | |
| 4th octet | | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |

The decimal IP address is _______________________________________________.

5) 00000011.11111001.00110011.00111101

| Bits | Eighth (leftmost) | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First bit | **Total value** |
|---|---|---|---|---|---|---|---|---|---|
| 1st octet | | | | | | | | | |
| 2nd octet | | | | | | | | | |
| 3rd octet | | | | | | | | | |
| 4th octet | | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |

The decimal IP address is _______________________________________________.

6) 11000110.01011110.01111111.11111110

| Bits | Eighth (leftmost) | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First bit | **Total value** |
|---|---|---|---|---|---|---|---|---|---|
| 1st octet | | | | | | | | | |
| 2nd octet | | | | | | | | | |
| 3rd octet | | | | | | | | | |
| 4th octet | | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |

The decimal IP address is ________________________________________________.

7) 11111000.00000000.00000010.01100101

| Bits | Eighth (leftmost) | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First bit | **Total value** |
|---|---|---|---|---|---|---|---|---|---|
| 1st octet | | | | | | | | | |
| 2nd octet | | | | | | | | | |
| 3rd octet | | | | | | | | | |
| 4th octet | | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |

The decimal IP address is ________________________________________________.

8) 01000101.11111111.01011100.01111110

| Bits | Eighth (leftmost) | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First bit | **Total value** |
|---|---|---|---|---|---|---|---|---|---|
| 1st octet | | | | | | | | | |
| 2nd octet | | | | | | | | | |
| 3rd octet | | | | | | | | | |
| 4th octet | | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |

The decimal IP address is ________________________________________________.

9) 00000100.00001000.00000100.11111010

| Bits | Eighth (leftmost) | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First bit | **Total value** |
|---|---|---|---|---|---|---|---|---|---|
| 1st octet | | | | | | | | | |
| 2nd octet | | | | | | | | | |
| 3rd octet | | | | | | | | | |
| 4th octet | | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |

The decimal IP address is _______________________________________________.

10) 11011111.01110011.01000001.00101011

| Bits | Eighth (leftmost) | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First bit | **Total value** |
|---|---|---|---|---|---|---|---|---|---|
| 1st octet | | | | | | | | | |
| 2nd octet | | | | | | | | | |
| 3rd octet | | | | | | | | | |
| 4th octet | | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |

The decimal IP address is _______________________________________________.

## Exercise 1.2 Decimal to Binary Practice

Convert the following decimal numbers to binary.

1) 231.50.45.251

| Bits | Eighth (leftmost) | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First bit | **Total value** |
|---|---|---|---|---|---|---|---|---|---|
| 1st octet | | | | | | | | | |
| 2nd octet | | | | | | | | | |
| 3rd octet | | | | | | | | | |
| 4th octet | | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |

The binary representation is __________________________________________________.

2) 14.110.25.42

| Bits | Eighth (leftmost) | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First bit | **Total value** |
|---|---|---|---|---|---|---|---|---|---|
| 1st octet | | | | | | | | | |
| 2nd octet | | | | | | | | | |
| 3rd octet | | | | | | | | | |
| 4th octet | | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |

The binary representation is __________________________________________________.

3) 221.1.221.118

| Bits | Eighth (leftmost) | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First bit | **Total value** |
|---|---|---|---|---|---|---|---|---|---|
| 1st octet | | | | | | | | | |
| 2nd octet | | | | | | | | | |
| 3rd octet | | | | | | | | | |
| 4th octet | | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |

The binary representation is ______________________________________________

4) 80.253.244.22

| Bits | Eighth (leftmost) | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First bit | **Total value** |
|---|---|---|---|---|---|---|---|---|---|
| 1st octet | | | | | | | | | |
| 2nd octet | | | | | | | | | |
| 3rd octet | | | | | | | | | |
| 4th octet | | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |

The binary representation is ______________________________________________.

5) 3.249.51.61

| Bits | Eighth (leftmost) | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First bit | **Total value** |
|---|---|---|---|---|---|---|---|---|---|
| 1st octet | | | | | | | | | |
| 2nd octet | | | | | | | | | |
| 3rd octet | | | | | | | | | |
| 4th octet | | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |

The binary representation is ______________________________________________.

6) 198.94.127.254

| Bits | Eighth (leftmost) | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First bit | **Total value** |
|---|---|---|---|---|---|---|---|---|---|
| 1st octet | | | | | | | | | |
| 2nd octet | | | | | | | | | |
| 3rd octet | | | | | | | | | |
| 4th octet | | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |

The binary representation is ______________________________________________.

7) 248.0.2.101

| Bits | Eighth (leftmost) | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First bit | **Total value** |
|---|---|---|---|---|---|---|---|---|---|
| 1st octet | | | | | | | | | |
| 2nd octet | | | | | | | | | |
| 3rd octet | | | | | | | | | |
| 4th octet | | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |

The binary representation is ______________________________________________.

8) 69.255.92.126

| Bits | Eighth (leftmost) | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First bit | **Total value** |
|---|---|---|---|---|---|---|---|---|---|
| 1st octet | | | | | | | | | |
| 2nd octet | | | | | | | | | |
| 3rd octet | | | | | | | | | |
| 4th octet | | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |

The binary representation is ______________________________________________.

9) 4.8.4.250

| Bits | Eighth (leftmost) | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First bit | **Total value** |
|---|---|---|---|---|---|---|---|---|---|
| 1st octet | | | | | | | | | |
| 2nd octet | | | | | | | | | |
| 3rd octet | | | | | | | | | |
| 4th octet | | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |

The binary representation is ________________________________________________.

10) 223.115.65.43

| Bits | Eighth (leftmost) | Seventh bit | Sixth bit | Fifth bit | Fourth bit | Third bit | Second bit | First bit | **Total value** |
|---|---|---|---|---|---|---|---|---|---|
| 1st octet | | | | | | | | | |
| 2nd octet | | | | | | | | | |
| 3rd octet | | | | | | | | | |
| 4th octet | | | | | | | | | |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |

The binary representation is ________________________________________________.