Computer Number Systems

All digital computers, from supercomputers to your smartphone, are electronic devices and ultimately can do one thing: detect whether an electrical signal is on or off. That basic information, called a *bit* (**bi**nary digi**t**), has two values: a 1 (or *true*) when the signal is on, and a 0 (of *false*) when the signal is off.

Larger values can be stored by a group of bits. For example, there are 4 different values stored by 2 bits (00, 01, 10, and 11), 8 values for 3 bits (000, 001, 010, 011, 100, 101, 110, and 111), 16 values for 4 bits (0000, 0001, ..., 1111), and so on. However, large numbers, using 0s and 1s only, are quite unwieldy for humans. For example, a computer would need 19 bits to store the numbers up to 500,000! We use of different number systems to work with large binary strings that represent numbers within computers.

Contents

Different Number Systems

A *number system* is the way we name and represent numbers. The number system we use in our daily life is known as **decimal number system** or **base 10** because it is based on 10 different digits: 0, 1, 2, ..., 8, and 9. The base of number is written as subscript to a number. For example, the decimal number "twelve thousand three hundred and forty-five" is written as 12345_{10}. Without a base or explicit context, it's assumed that a number is in base 10.

In computer science, apart from the decimal system, three additional number systems are commonly used: **binary** (base-2), **octal** (base-8), and **hexadecimal** or just **hex** (base-16). Binary numbers are important because that is how numbers are stored in the computer. Octal and hexadecimal are used to represent binary numbers in a user-friendly way. Each octal symbol represents 3 binary bits and each hex digit represents 4 binary bits. For example, the decimal number 53201, stored as in the computer as the binary number 1000000111000101100, is represented by the octal number 2017054, and the hex number 81E2C. The table below compares the number systems:

Number System	Base	Digits Used	Examples
Binary	2	0,1	10110_2, 10110011_2
Octal	8	0,1,2,3,4,5,6,7	75021_8, 231_8, 60012_8
Decimal	10	0,1,2,3,4,5,6,7,8,9	97425_{10} or simply 97425
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F	$54A2DD0F_{16}$

In general, N bits have 2^N different values. The computers aboard the Apollo spacecraft had 8-bit words of memory. Each word of memory could store 256 different values, and the contents were displayed using 2 hex characters.

The following table shows the first 20 numbers in decimal, binary, octal, and hexadecimal:

Decimal	Binary	Octal	Hexadecimal
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

Converting to Decimal

As we learned in the first or second grade, the decimal value of a decimal number is simply sum of each digit multiplied by its place value:

$$12345 = 1 \times 10^4 + 2 \times 10^3 + 3 \times 10^2 + 4 \times 10^1 + 5 \times 10^0 = 10000 + 2000 + 300 + 40 + 5 = 12345$$

$$3079 = 3 \times 10^3 + 0 \times 10^2 + 7 \times 10^1 + 9 \times 10^0 = 3000 + 70 + 9 = 3079$$

Analogously, the decimal value of a number in an arbitrary base is the sum of each digit multiplied by its place value. Here are some examples of converting from one base into base 10:

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 8 + 4 + 0 + 1 = 13_{10}$$

$$175_8 = 1 \times 8^2 + 7 \times 8^1 + 5 \times 8^0 = 1 \times 64 + 7 \times 8 + 5 \times 1 = 64 + 56 + 5 = 125_{10}$$

$$A5E_{16} = 10 \times 16^2 + 5 \times 16^1 + 14 \times 16^0 = 10 \times 256 + 5 \times 16 + 14 \times 1 = 2560 + 80 + 14 = 2654_{10}$$

Converting from Decimal

The algorithm to convert a number from an arbitrary base requires finding how many times successive powers of the base go into the number, starting with the largest power of the base less than the starting number. For example, converting 3306 from base 10 to octal proceeds as follows:

1) The largest power of 8 that is less than or equal to 3306 is 512 (8^3). Divide 3306 by 512:

$$3306 = 6 * 8^3 + 234$$

2) The next power of 8 is 64 (8^2). Divide 234 by 64:

$$234 = 3 * 8^2 + 42$$

3) The next smaller power of 8 is 8 (8^1). Divide 42 by 8:

$$42 = 5 * 8^1 + 2$$

4) Finally, the next smaller power of 8 is 1 (8^0). Divide 2 by 1:

$$2 = 2 * 8^0$$

The answer is 6352_8.

Converting between Binary, Octal, and Hexadecimal

Converting from octal to binary is simple: replace each octal digit by its corresponding 3 binary bits. For example:

$$375_8 = \ \ 011 \ \ 111 \ \ 101_2 = 11111101_2$$

Converting from hex to binary is also simple: replace each hex digit by its corresponding 4 binary bits. For example:

$$FD_{16} = \ \ 1111 \ \ 1101_2 = 11111101_2$$

Converting from binary to either octal or hex is pretty simple as well: group the bits by 3s or 4s (starting at the right), and convert each group:

$$1000000111100010 1100 = 10\ 000\ 001\ 111\ 000\ 101\ 100 = 2017054_8$$

$$1000000111100010 1100 = 1000\ 0001\ 1110\ 0010\ 1100 = 81E2C_{16}$$

Converting between base 8 and 16 is easy by expressing the number in base 2 (easy to do!) and then converting that number from base 2 (another easy operation)! This is shown below in Sample Problem #1.

Using Hexadecimal Numbers to Represent Colors

Computers use hexadecimal numbers to represent various colors in computer graphics because all computer screens use combinations of red, green, and blue light or RGB to represent thousands of different colors. Two digits are used for each so the hexadecimal number "#FF0000" represents the color red, "#00FF00" represents green, and "#0000FF" represents blue. The color black is "#000000" and white is "#FFFFFF".

The hash tag or number sign is used to denote a hexadecimal number. $FF_{16} = F(15) \times 16 + F(15) \times 1 = 240 + 15 = 255_{10}$ so there are 0 to 255 or 256 different shades of each color or $256^3 = 16,777,216$ different colors.

The following web site has nearly every color name, along with its hex code and decimal values:

https://www.rapidtables.com/web/color/RGB_Color.html

For example “salmon” is “#FA8072” which represents the decimal numbers 250 (hex FA), 128 (hex 80), and 114 (hex 72).

Resources

Ryan's Tutorials (https://ryanstutorials.net/) covers this topic beautifully. Rather than trying to duplicate that work, we'll point you to the different sections:

1. Number Systems (https://ryanstutorials.net/binary-tutorial/) - An introduction to what numbers systems are all about, with emphasis on decimal, binary, octal, and hexadecimal. ACSL will typically identify the base of a number using a subscript. For example, 123_8 is an octal number, whereas 123_{16} is a hexadecimal number.

2. Binary Conversions (https://ryanstutorials.net/binary-tutorial/binary-conversions.php) - This section shows how to convert between binary, decimal, hexadecimal and octal numbers. In the *Activities* (https://ryanstutorials.net/binary-tutorial/binary-conversions.php#activities) section, you can practice converting numbers.

3. Binary Arithmetic (https://ryanstutorials.net/binary-tutorial/binary-arithmetic.php) - Describes how to perform various arithmetic operations (addition, subtraction, multiplication, and division) with binary numbers. ACSL problems will also cover basic arithmetic in other bases, such as adding and subtracting together 2 hexadecimal numbers. ACSL problems will not cover division in other bases.

4. Negative Numbers (https://ryanstutorials.net/binary-tutorial/binary-negative-numbers.php) - ACSL problems will not cover how negative numbers are represented in binary.

5. Binary Fractions and Floating Point (https://ryanstutorials.net/binary-tutorial/binary-floating-point.php) - The first part of this section is relevant to ACSL: fractions in other bases. ACSL will not cover floating point numbers in other basis. So, focus on the section Converting to a Binary Fraction (https://ryanstutorials.net/binary-tutorial/binary-floating-point.php#convertfraction), but keep in mind that ACSL problems may also cover octal and hexadecimal fractions.

The CoolConversion.com online calculator (https://coolconversion.com/math/binary-octal-hexa-decimal/) is another online app for practicing conversion from/to decimal, hexadecimal, octal and binary; this tool shows the steps that one goes through in the conversion.

The AskNumbers.com site has a nice description of the conversions between binary, octal, decimal and hexadecimal (https://www.asknumbers.com/hex-to-octal.aspx) numbers.

Format of ACSL Problems

The problems in this category will focus on converting between binary, octal, decimal, and hexadecimal, basic arithmetic of numbers in those bases, and, occasionally, fractions in those bases.

To be successful in this category, you must know the following facts cold:

1. The binary value of each octal digit 0, 1, ..., 7
2. The binary value of each hex digit 0, 1, ..., 9, A, B, C, D, E, F
3. The decimal value of each hex digit 0, 1, ..., F
4. Powers of 2, up to 4096
5. Powers of 8, up to 4096
6. Powers of 16, up to 65,536

Sample Problems

Sample Problem 1

Solve for x where $x_{16} = 3676_8$.

Solution: One method of solution is to convert 3676_8 into base 10, and then convert that number into base 16 to yield the value of x.

An easier solution, less prone to arithmetic mistakes, is to convert from octal (base 8) to hexadecimal (base 16) through the binary (base 2) representation of the number:

$$\begin{aligned} 3676_8 &= 011\ 110\ 111\ 110_2 && \text{convert each octal digit into base 2} \\ &= 0111\ 1011\ 1110_2 && \text{group by 4 bits, from right-to-left} \\ &= 7\ \mathrm{B}\ \mathrm{E}_{16} && \text{convert each group of 4 bits into a hex digit} \end{aligned}$$

Sample Problem 2

Solve for x in the following hexadecimal equation: $x = \mathrm{F5AD}_{16} - 69\mathrm{EB}_{16}$

Solution: One could convert the hex numbers into base 10, perform the subtraction, and then convert the answer back to base 16. However, working directly in base 16 isn't too hard. As in conventional decimal arithmetic, one works from right-to-left, from the least significant digits to the most.

> The rightmost digit becomes 2, because D-B=2.
> The next column is A-E. We need to *borrow* a one from the 5 column, and 1A-E=C
> In the next column, 4-9=B, again, borrowing a 1 from the next column.
> Finally, the leftmost column, E-6=8

Combining these results of each column, we get a final answer of $8BC2_{16}$.

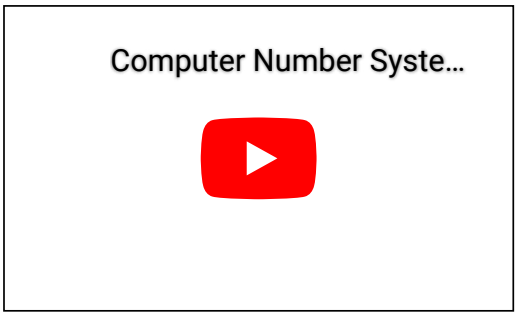
Sample Problem 3

How many numbers from 100 to 200 in base 10 consist of distinct ascending digits and also have distinct ascending hex digits when converted to base 16?

Solution: There are 13 numbers that have ascending digits in both bases from 100 to 200. They are (in base 10): 123 (7B), 124, 125, 126, 127 (7F), 137 (89), 138, 139 (8B), 156 (9C), 157, 158, 159 (9F), 189 (BD)

Video Resources

ACSL Videos

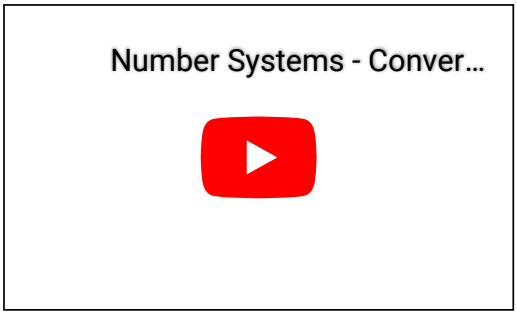


Computer Number Systems - ACSL Short Problems Topic (**Next Generation Academy**) (https://youtu.be/2v5hDcKCfbo)

A great tutorial covering this ACSL category, and shows how to solve many ACSL problems that have appeared in previous contests.

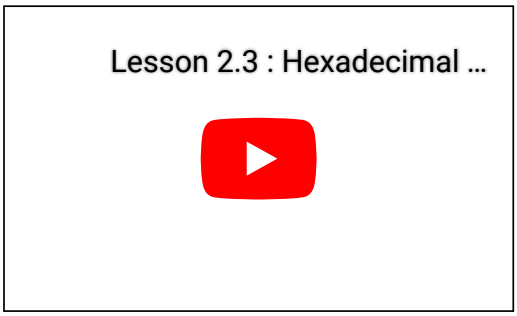
Other Videos

There are many YouTube videos about computer number systems. Here are a handful that cover the topic nicely Some of the videos contain ads; ACSL is not responsible for the ads and does not receive compensation in any form for those ads.



Number Systems - Converting Decimal, Binary and Hexadecimal (**Joe James**) (https://youtu.be/aW3qCcH6Dao)

An introduction to number systems, and how to convert between decimal, binary and hexadecimal numbers.



Lesson 2.3 : Hexadecimal Tutorial (**Carl Herold**) (https://youtu.be/m1JtWKuTLR0)

The video focuses on hexadecimal numbers: their relationship to binary numbers and how to convert to decimal numbers.

*Hexes and the Magic of Base 16 - Vaidehi Joshi - May 2017' (**DonutJS**)* (https://youtu.be/WR67syBDzew)

A fun introduction to hexadecimal numbers, focusing a bit on using hex numbers for specifying RGB colors.

Collins Lab: Binary & Hex (**Adafruit Industries**) (https://youtu.be/jvx-NrILgpE)

A professionally produced video that explains the number systems, how and why binary numbers are fundamental to computer science, and why hexadecimal is important to computer programmers.
