Base Systems

Jacqueline A. Jones

People use the decimal number system to perform arithmetic operations. Computers, on the other hand, use the binary system, which contains only two digits: 0 and 1. We need a way to convert numbers from one system to another.

Base Systems and Positional Notation

All the number systems we use operate on some basic principles. Each has a base: we will look at decimal (base 10), binary (base 2), and hexadecimal (base 16). The base denotes the number of digits and the powers to be used: base 10 has 10 digits, numbered from 0-9, and uses powers of 10; base 2 has 2 digits, numbered from 0-1, and uses powers of 2; and base 16 has 16 digits, numbered from 0-F (where the letters A-F serve as the digits following 9), and uses powers of 16.

Each of these number systems is a positional notation system, which means that the position of a digit in the number determines its value: for example, the value of 9 in the decimal number 973 is not the same as the value of 9 in 379 or 793.
(As an example of a non-positional notation system, consider the tally system used in small elections, where we write 4 straight bars, then cross them with a fifth bar to form a group of 5. 𝍸𝍸 || is 12, but so is || 𝍸 𝍸 and so is 𝍸 || 𝍸.)

We'll use the decimal number system to show how the positional notation system works. Suppose we have the number 973. What does this number mean? Each digit represents that digit times the power of 10 represented by its position. Keep in mind that the decimal point is assumed to be on the right of the number if it is not explicitly shown. Starting at the decimal point and moving to the left, the columns represent increasing powers of 10, as shown:

10^2 10^1 10^0 .

9 7 3

The base is 10, and the exponent (the superscript) tells what power of 10. The exponents increase by 1 with each move to the next column to the left.

Recall that any number to the zeroth power is 1, and any number to the first power is that number itself. So the column labeled 10_0 represents how many 1's are in the number, the column labeled 10_1 represents how many 10's are in the number, and the column labeled 10_2 represents how many 100's are in the number:

100's column	10's column	1's column
9	7	3

Thus this number has 3 ones, 7 tens, and 9 hundreds:

$3 * 10_0 = 3 * 1 = 3$
$7 * 10_1 = 7 * 10 = 70$
$9 * 10_2 = 9 * 100 = 900$

Summing the three values yields 973. We evaluated each digit, based on its position, and then summed the results. In essence, we have done a decimal to decimal conversion. Later, we'll use this same method to convert from other bases to decimal.

Note that the exponents also decrease by 1 with each column move to the right, so the first position to the right of the decimal place tells how many 10^{-1}'s are in the number, and the second position to the right of the decimal place tells how many 10^{-2}'s are in the number, etc. Thus .25 means $2 * 10^{-1}$ or 2 times 1/10 or 2/10 (or 20/100), while .05 means $5 * 10^{-2}$ or 5 * 1/100 or 5/100), and the sum of those is 20/100 + 5/100 or 25/100ths. (For simplicity, we will ignore decimal places in our discussion below.)

Counting

Counting in all number systems works the same way: start at 0 and increment by 1 till you reach the maximum digit in the number system. Then put down a 0 and carry a 1 to the next column, adding it (if necessary) to the value already there. As an example, in decimal we count from 0 to 9, which is the maximum digit; then we put down a 0 and carry the 1, yielding 10. We do the same process in the rightmost column until we reach 19; then we put down a 0, carry the 1 and add it to the 1 already there to produce 20. When we reach the maximum number in every column, we put down 0's and carry the 1 to the next column: for example, we move from 99 to 100, or from 999 to 1000.

Binary Number System

The binary number system (base 2) has two digits, numbered from 0 to 1. The computer uses the binary number system to represent numbers internally (it uses the decimal system for input/output of numeric values). The smallest unit of storage in the computer is the bit, which stands for binary digit. A bit can hold one of two values, 0 or 1; these are the only symbols that are used to represent numbers in the binary system. All other values are represented by combinations of bits. The 0 and 1 values are often interpreted as false and true, or as an electrical impulse being off and on. A sample binary number looks like this: 101110. Note: this is also a number in the decimal system – one hundred and one thousand, one hundred and ten. We will always try to make clear which system is being discussed.

Counting in Binary

Counting in binary is the same as in decimal, but we hit the maximum digit in a column more often (in fact, it happens every other number--do you see why?):

Decimal	**Binary**
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000

Reading Numbers in Binary

Students who are first learning about the binary system often make the mistake of reading binary numbers like 10 as "ten" and 11 as "eleven" and 100 as "one hundred." This is misleading. "Ten" and "eleven" and "one hundred" are names of numbers in the decimal number system, and they do not apply to binary. We should refer to binary 10 as one-zero, to 11 as one-one, and to 100 as one-zero-zero.

Positions in the Binary Number System

As we noted before, the positions in the binary number system are powers of 2, rather than powers of 10: Here is an example using a number with 6 bits:

2^5 2^4 2^3 2^2 2^1 2^0
1 0 1 1 1 1

That means that we can label the columns as follows:

32's column	16's column	8's column	4's column	2's column	1's column
1	0	1	1	1	1

We will use these positions to evaluate binary numbers for conversion to decimal.

Conversion from Binary to Decimal

Often we want to know what a binary number represents in the decimal number system. To find out, we can perform a binary to decimal conversion. One simple method is this:

1. write out the binary number
2. write the powers of two above the digits, from right to left
3. add up the numbers with 1's under them.

Let's use this method to convert 101111 from binary to decimal. Here is the number with the powers of two written above the digits:

```
32 16 8  4  2  1
 1   0   1  1  1
```

This shows that the number 101111 has one 32, no 16's, one 8, one 4, one 2 and one 1. Thus it evaluates to 32+8+4+2+1 = 47 in decimal.

As another example, let's convert 1001110 from binary to decimal. Here is the number written with the powers of two written above the digits:

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

Adding the numbers with 1's under them gives us 64+8+4+2 = 78, which is the value of the number in decimal.

Conversion from Decimal to Binary

Reversing this method gives us the ability to convert a decimal number to binary. Here is an algorithm to convert from decimal to binary:

1. write out all of the powers of 2 that are less than the number
2. starting at the largest, determine if that power of two is contained in the number (that is, determine whether the power of 2 can be subtracted from the number)
3a. if the power of 2 is contained in the number, write a 1 under the power of 2 and subtract its value from the number
3b. if the power of 2 is not contained in the number, write a 0 under the power of two
4. repeat for each power of two until the number has been reduced to 0.
5. write 0's under any powers of 2 which have nothing underneath.

As an example, let's convert the decimal number 41 to binary. Start by writing out the powers of two that are less than 41. (Starting at 1 and working left is easiest for this.)

```
41          32 16 8 4 2 1
```

The next power of 2 is 64, and that is larger than 41, so we stop here.

Now start at the left. Is there a 32 contained in 41? Yes, so write 1 under 32, and subtract 32 from 41:

```
  41          32  16  8  4  2  1
- 32           1
------
   9
```

Continue: is there a 16 contained in 9? No, so write 0 under 16:

```
  41          32 16  8  4  2  1
- 32           1   0
-------
   9
```

Is there an 8 contained in 9? Yes, so write 1 under 8 and subtract 8 from 9:

```
  9           32 16  8  4  2  1
- 8            1   0  1
-------
  1
```

Is there a 4 contained in 1? No, so write 0 under 4:

```
  9           32 16  8  4  2  1
- 8            1   0  1  0
-------
  1
```

Is there a 2 contained in 1? No, so write 0 under 2:

```
  9             32 16  8  4  2  1
- 8               1  0  1  0  0
-------
  1
```

Is there a 1 contained in 1? Yes, so write 1 under 1 and subtract 1 from 1; this yields 0, so we are done.

```
  9           32 16  8  4  2  1
- 1            1    0  1  0  0  1
-------
  0
```

The number 41 in decimal is 101001 in binary. We can check it by summing the numbers with 1's under them: 32+8+1 = 41.

Let's convert the decimal number 78 to binary. Start by writing out the powers of two that are less than 78.

```
  78            64 32 16  8  4  2  1
```

The next power of 2 is 128, and that is larger than 78, so we stop at 64.

Now start at the left. Is there a 64 contained in 78? Yes (the first answer must be yes--do you see why?), so write 1 under 64 and subtract 64 from 78:

```
  78            64 32 16  8  4  2  1
- 64            1
-------
  14
```

Continue: is there a 32 contained in 14? No, so write 0 under 32:

```
  78            64 32 16  8  4  2  1
- 64              1  0
-------
  14
```

Continue: is there a 16 contained in 14? No, so write 0 under 16:

```
  78            64 32 16  8  4  2  1
- 64              1   0   0
-------
  14
```

Is there an 8 contained in 14? Yes, so write 1 under 8 and subtract 8 from 14:

```
  14           64 32 16  8  4  2  1
  - 8            1  0    0  1
-------
```

```
   6
```

Is there a 4 contained in 6? Yes, so write 1 under 8 and subtract 4 from 6:

```
  6              64 32 16  8  4  2  1
- 4                1   0   0  1  1
-------
  2
```

Is there a 2 contained in 2? Yes, so write 1 under 2 and subtract 2 from 2:

```
  2              64 32 16  8  4  2  1
- 2                1   0   0  1  1  1
-------
  0
```

This yields 0, so we are done. However, there is a power of two with no number under it. Finish by writing a 0 under each of the remaining powers of 2:

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

The number 78 in decimal is 1001110 in binary. We can check it by summing the numbers with 1's under them: 64+8+4+2 = 78.

Another Method of Decimal to Binary Conversion

Although the previous method is quite simple, there is another algorithm that some people prefer. This algorithm follows:

1. divide the number by 2 and write down the remainder, which must be 0 or 1.
2. divide the quotient by 2 and write down the remainder to the left of the previous remainder.
3. repeat this process until the result of the division is 0.

As an example, let's convert 19 from decimal to binary:

```
2 | 19
     9 rem 1                    1
2 | 9
    4 rem 1                    11
2 | 4
    2 rem 0                   011
2 | 2
   1 rem 0                   0011
2 | 1
    0 rem 1                 10011
```

19 in decimal is 10011 in binary. To check the result, add up the positions: 16+2+1 = 19.

Binary Arithmetic

Binary arithmetic is quite simple. There are only four addition facts to learn (versus 100 for decimal arithmetic--up to 9+9).

0 + 0 = 0
1 + 0 = 1
0 + 1 = 1
1 + 1 = 10

From a computer's point of view, this simplicity means that the circuitry for addition is incredibly simplified in base 2 vs base 10.

Binary Addition

Here are some examples of binary addition. In decimal, this first one is 5 plus 3:

```
101
+11
------
```

First use rule 1+1 = 10. Put down the 0 and carry the 1.

```
 1
101
+11
------
  0
```

The second column is done in two steps: 1+0 = 1 and 1+1 is 10. Again put down the 0 and carry the 1.

```
 11
 101
+ 11
------
  00
```

1+1 is 10. Again put down the 0 and carry the 1. This time, the carry doesn't have to be added to anything, so we just bring it down to the result, writing down 10:

```
 11
 101
+ 11
------
1000
```

The answer is 1000, which in decimal is 8. Our answer checks, since 5 + 3 = 8.

The next example is about as hard as addition in binary gets (in decimal, this adds 7 + 3):

```
 111
+ 11
-------
```

We apply the rule 1+1 = 10, writing down the 0 and carrying the 1.

```
  1
 111
+ 11
------
    0
```

Now we have 1+1+1, but this is done two at a time, so it is just 1+ 1 = 10 and then 10 + 1 = 11 (write down the 1 and carry the 1):

```
 1 1
 111
+ 11
------
   10
```

We again have 1+1 = 10 (write down the 0 and carry the 1, but since there are no more digits, we bring down the last 1):

```
 1 1
 111
+ 11
------
1010
```

In decimal, we added 7 + 3 and got 10.

Hexadecimal Number System

Suppose we write down this number and say that you have 5 seconds to memorize it:

10101001010001111011110

Are you done? Have you memorized it? Not likely. You've probably barely even read the digits.Computers love long strings of ones and zeroes, but humans don't. It is hard for us to read binary numbers and recollect them. This is why computer scientists use another number system, which we are about to introduce. As we will see later on, this number system allows easy representation and recollection of binary numbers.

The number system we will use is base 16 or hexadecimal (also known as hex). The fact that 16 is 2^4 permits easy conversion from base 2 to base 16, as we will see below. The hexadecimal number system has 16 digits, numbered from 0 to F, where the letters A through F represent the numbers after 9. The letters A through F are used because there must be a single digit to represent the values that decimal represents by 10, 11, 12, 13, 14, and 15, which are 2-digit numbers. Any six symbols could have been chosen, but the letters have the advantage of already having an order.

Decimal	Hexadecimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	10

Reading Numbers in Hexadecimal

Just as with binary, it is a common mistake to read hexadecimal numbers with the names from decimal. Hex 30 is not "thirty"; it is read as three-zero. Similarly, hex 15 is not "fifteen," but one-five.

Positions in the Hexadecimal Number System

The positions in the binary number system are powers of 16, rather than powers of 10 (as in decimal) or powers of 2 (as in binary).

16^3 16^2 16^1 16^0

This means that we can label the columns as follows:

```
 4096's    256's     16's     1's
column   column   column   column
```

We will use these positions to evaluate hexadecimal numbers for conversion to decimal.

Hexadecimal to Decimal Conversion

Here is an algorithm for converting from hexadecimal to decimal:

1. write out the hexadecimal number
2. above the digits, write the powers of 16, starting with 1 at the right and increasing as you move to the left
3. multiply each digit by the power of 16 above it
4. sum the results.

(Note that each digit A through F should be converted to decimal before multiplication.)

Here's an example. Let's convert A5 from hexadecimal to decimal. First we write out the number and write the powers of 16 above the digits. Note that it is easier to write 16 and 1 than 16_1 and 16_0, so that's what we'll do from here on:

```
16 1
 A 5
```

Multiply each digit of the number by the power of 16 above it. Remember to convert each digit A through F to its decimal equivalent before multiplication. The digit A is 10, and 10 times 16 is 160. Of course, 5 times 1 is 5. Sum the results:

```
A * 16 = 10 * 16 = 160
 5 *  1          =   5
                   -----
                   165
```

Notice that the final result requires three digits in decimal but only two in hex. That's because each digit in hex represents one of 16 different values, while in decimal each digit represents only 10 values.

Here's another example. Let's convert 1CE from hexadecimal to decimal. First we write out the number and write the powers of 16 above the digits:

```
256 16  1
  1   C  E
```

Multiply each digit of the number by the power of 16 above it. Remember to convert each digit A through F to its decimal equivalent before multiplication, so that C becomes 12 and E becomes 14. Sum the results:

```
1 * 256          = 256
C * 16 = 12 * 16 = 192
E * 1 = 14 * 1   =  14
                  ------
                   462
```

Hexadecimal Arithmetic

Hexadecimal arithmetic has many more facts that decimal. For each operation (addition, subtraction or multiplication), there are 16*16 facts, but we won't show those here. Some hexadecimal arithmetic may look quite funny, as you add "letters"; for example

A + 1 = B
C + 2 = D
F - 1 = E
F - 5 = A

Some hexadecimal arithmetic is just like base 10; that can be misleading, because the arithmetic gets tricky when it produces results that violate our decimal experience; for example,

2 + 5 = 7

That's no problem, but what about 9+1?

9 + 1 = A
2 + 8 = A
9 + 2 = B

These additions violate our expectations and our years of training, and they are the ones that will confuse you on a test.

Another point of confusion arises when we move back from letters to numbers. For example, what is F + 1? F + 1 = 10 (in decimal, this is 15 + 1 = 16). Here are some other examples:

E + 2 = 10
E + 3 = 11
F + 2 = 11

Revising the Binary Number Chart

Let's put together all three number systems into one chart. In addition, we'll modify the Binary column a bit, so that each binary number is represented by exactly 4 digits. That means we'll add some leading zeroes to each of the lower numbers (adding leading zeroes never changes the value of a number).

Decimal	**Binary**	**Hexadecimal**
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
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
	…	
32	100000	20

Producing the Chart of Equivalents

It is not necessary to memorize the whole chart. When you need it, you can reproduce it by counting. Still, if you need to reproduce it quickly--as on a test--there is a simple pattern to the binary numbers. The rightmost column starts with 0 and alternates 0, 1, 0, 1– it alternates by 1's.The second column from the right starts with 0 and alternates 0, 0, 1, 1–that is, it alternates by

2's. The third column starts with 0 and alternates 0,0,0,0, 1,1,1,1–that is, it alternates by 4's. Finally, the leftmost column starts at 0 and alternates by 8's: 8 zeroes followed by 8 ones. If you can remember that, you can easily reproduce the binary column and the entire chart.

Binary to Hexadecimal Conversion

Now that we've seen all three number systems, there are two more conversions to learn--binary to hex and hex to binary--but they will turn out to be quite easy with the revised chart. Here is the algorithm for converting from binary to hexadecimal:

1. write down the binary number
2. divide it into groups of 4 digits, starting at the RIGHT side
3. look up each group of 4 digits in the chart above, and write the corresponding hex digit below the group of digits.

Remember that long scary number we suggested you memorize? Let's convert it to hex:

```
10101001010001111011110
```

Divided into groups of 4, it becomes the following:

```
101 0100 1010 0011 1101 1110
```

Note that the reason we start dividing into groups on the right is so that if any group has fewer than 4 digits, that group will be the one on the left, where we can pad it with leading zeroes without changing the value of the number.

Now, let's look each group of digits up in the chart:

1110 is E, so we write E under 1110:

```
101 0100 1010 0011 1101 1110
                          E
```

1101 is D and 0011 is 3:

```
101 0100 1010 0011 1101 1110
                3    D    E
```

1010 is A and 0100 is 4:

```
101 0100 1010 0011 1101 1110
      4    A   3    D     E
```

Finally, 101 is 5 (it's the same as 0101):

```
101 0100 1010 0011 1101 1110
 5    4    A    3    D    E
```

The number in hex is 54A3DE. Can you memorize that in 10 seconds? You are a lot more likely to be able to memorize 6 hex symbols rather than 23 bits. That's why computer scientists use hexadecimal, as shorthand for long strings of binary numbers.

Hexadecimal to Binary Conversion

A hexadecimal to binary conversion is just the reverse of the binary to hex conversion. Here's the method:

1. write down the hexadecimal number (spread out the digits a bit)
2. look up each hex digit in the chart
3. under the hex digit, write the equivalent 4 binary digits (you can leave out leading zeroes on the leftmost group of bits, but ONLY on that one!)

As an example, let's convert C8 to binary:

```
C    8
```

In the chart, 8 is 1000:

```
C    8
    1000
```

Again in the chart, C is 1100:

```
C    8
1100 1000
```

Thus, C8 in hex is 11001000 in binary.

Let's do one last example, 34:

```
3    4
```

4 is 0100:

```
3    4
     0100
```

3 is 0011:

```
3    4
0011 0100
```

34 in hex is 110100 in binary. Note that since the leftmost group of bits has leading zeroes, we omit them (but we cannot omit leading zeroes on any other group but the leftmost, or we change the value).

Enrichment

In case you want more…

Binary Subtraction

Here is an example of subtraction (in decimal, this is 5 - 3):

```
101
- 11
-----
```

1-1 = 0, so we have this:

```
101
- 11
-----
   0
```

0-1 can't be done, so we borrow 1 from the next column. 10-1 =1, so our result is this:

```
101
- 11
------
  10
```

The answer is 10 or 2 in decimal.

Binary Multiplication

Finally, here is an example of multiplication:

```
 101
* 11
-------
```

Starting at the top right and bottom right, 1*1 = 1, 0 * 1 = 0, and 1 * 1 = 1, so we have this:

```
 101
* 11
-------
  101
```

Again starting at the top right, but using the bottom left, we have the same pattern, but shifted one to the left:

```
 101
* 11
-------
  101
101
-------
```

Adding the two columns is a series of 1+0 = 1 computations, giving this result:

```
 101
* 11
-------
 101
101
```

```
-------
1111
```

The answer is 1111, or 15 in decimal, which checks because we multiplied 5 times 3.

Decimal to Hexadecimal Conversion

Going from decimal to hexadecimal requires a different method (reversing the hex to decimal method is possible, but it is not as easy as reversing the binary to decimal method). The algorithm is as follows; note that this is the same as the second method for converting from decimal to binary. It works for converting from decimal to any other base if you substitute that base for 16 in the algorithm:

1. divide the number by 16 and write down the remainder (if the remainder is between 10 and 15, convert it to hex before writing it down)
2. divide the quotient by 16 and write down the remainder to the left of the previous remainder.
3. repeat this process until the result of the division is 0.

As an example, let's convert 74 from decimal to hex.

```
16 | 74
  16 | 4   rem 10          A
        0   rem 4         4A
```

First we divided 16 into 74; it goes in 4 times with a remainder of 10, which is A in hexadecimal. Therefore A is the rightmost digit of the answer.

Next we divided 16 into the previous quotient, which was 4. Since 16 doesn't go into 4, the quotient is 0 and the remainder is 4. We write the 4 to the left of the A, and the final result is 4A. So decimal 74 is equivalent to hexadecimal 4A. This is correct since 4A is 4*16+10, which equals 74 in decimal.

Actually, this method can be used to convert from decimal to any other base. Dividing by 16 gives remainders in base 16: numbers from 0 to F. Dividing by 2 gives remainders in base 2: numbers from 0 to 1.

Tricks in Hexadecimal Addition

The hardest additions may be those like D + C. One simple solution is to convert each digit to decimal, add in decimal, and convert the result back to hex (but don't forget to convert it back to hex!)

To add D + C:

convert D to 13
convert C to 12
add 13 + 12 to get 25
convert decimal 25 back to hex 19

Hexadecimal Subtraction

Subtraction that crosses between numbers and letters can be confusing. For example,

10 - 1 = F
10 - 2 = E
11 - 2 = D

Seeing 10-1, your automatic reaction will be 9, but try to focus on the number system you are using.

Summary on the Use of Number Systems

You might wonder how these number systems are used. The computer can use all three, and people have set up computers to use the most appropriate system for various purposes. For example, generally the computer uses decimal* for input and output, since I/O is where the computer interacts with people, and people work best with decimal. The computer uses binary for internal representation of everything, since that is most efficient system for processing and storage. The computer uses hexadecimal when printing out or displaying long strings of binary numbers for people to read. For example, if a program crashes, the computer might generate a "core dump," which is a display of the contents of memory at the time of the crash. This dump is printed in hex to make it easier for the programmer to read.

*Actually, I/O is done in character form, but that is a subject for another time and place.