The Binary System

A pretty damn clear guide to a quite confusing concept by Christine R. Wright with some help from Samuel A. Rebelsky.

Table of Contents

Basic Concepts Behind the Binary System

To understand binary numbers, begin by recalling elementary school math. When we first learned about numbers, we were taught that, in the decimal system, things are organized into columns:

```
H | T | O
1 | 9 | 3
```

such that "H" is the hundreds column, "T" is the tens column, and "O" is the ones column. So the number "193" is 1-hundreds plus 9-tens plus 3-ones.

Years later, we learned that the ones column meant 10^0, the tens column meant 10^1, the hundreds column 10^2 and so on, such that

```
10^2|10^1|10^0
  1 |  9 |  3
```

the number 193 is really {(1*10^2)+(9*10^1)+(3*10^0)}.

As you know, the decimal system uses the digits 0-9 to represent numbers. If we wanted to put a larger number in column 10^n (e.g., 10), we would have to multiply 10*10^n, which would give 10^(n+1), and be carried a column to the left. For example, putting ten in the 10^0 column is impossible, so we put a 1 in the 10^1 column, and a 0 in the 10^0 column, thus using two columns. Twelve would be 12*10^0, or 10^0(10+2), or 10^1+2*10^0, which also uses an additional column to the left (12).

The binary system works under the exact same principles as the decimal system, only it operates in base 2 rather than base 10. In other words, instead of columns being

```
10^2|10^1|10^0
```

they are

```
2^2|2^1|2^0
```

Instead of using the digits 0-9, we only use 0-1 (again, if we used anything larger it would be like multiplying 2*2^n and getting 2^n+1, which would not fit in the 2^n column. Therefore, it would shift you one column to the left. For example, "3" in binary cannot be put into one column. The first column we fill is the right-most column, which is 2^0, or 1. Since 3>1, we need to use an extra column to the left, and indicate it as "11" in binary (1*2^1) + (1*2^0).

Examples: What would the binary number 1011 be in decimal notation?

Click here to see the answer

Try converting these numbers from binary to decimal:

- 10
- 111
- 10101
- 11110

Remember:

```
2^4| 2^3| 2^2| 2^1| 2^0
   |    |    |  1 |  0
   |    | 1  |  1 |  1
1  | 0  | 1  |  0 |  1
1  | 1  | 1  |  1 |  0
```

Click here to see the answer

Return to Table of Contents

Binary Addition

Consider the addition of decimal numbers:

```
 23
+48
___
```

We begin by adding 3+8=11. Since 11 is greater than 10, a one is put into the 10's column (carried), and a 1 is recorded in the one's column of the sum. Next, add {(2+4) +1} (the one is from the carry)=7, which is put in the 10's column of the sum. Thus, the answer is 71.

Binary addition works on the same principle, but the numerals are different. Begin with one-bit binary addition:

```
 0    0    1
+0   +1   +0
___  ___  ___
 0    1    1
```

1+1 carries us into the next column. In decimal form, 1+1=2. In binary, any digit higher than 1 puts us a column to the left (as would 10 in decimal notation). The decimal number "2" is written in binary notation as "10" (1*2^1)+(0*2^0). Record the 0 in the ones column, and carry the 1 to the twos column to get an answer of "10." In our vertical notation,

```
 1
+1
___
10
```

The process is the same for multiple-bit binary numbers:

```
  1010
 +1111
 ______
```

- Step one:
  Column 2^0: 0+1=1.
  Record the 1.
  Temporary Result: 1; Carry: 0
- Step two:
  Column 2^1: 1+1=10.
  Record the 0, carry the 1.
  Temporary Result: 01; Carry: 1
- Step three:
  Column 2^2: 1+0=1 Add 1 from carry: 1+1=10.
  Record the 0, carry the 1.
  Temporary Result: 001; Carry: 1
- Step four:
  Column 2^3: 1+1=10. Add 1 from carry: 10+1=11.
  Record the 11.
  Final result: 11001

Alternately:

```
  11   (carry)
  1010
 +1111
______
 11001
```

Always remember

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

Try a few examples of binary addition:

```
  111      101      111
 +110     +111     +111
_____    _____    _____
```

Click here to see the answer

Return to Table of Contents

Binary Multiplication

Multiplication in the binary system works the same way as in the decimal system:

- 1*1=1
- 1*0=0
- 0*1=0

```
 101
* 11
____
 101
1010
____
1111
```

Note that multiplying by two is extremely easy. To multiply by two, just add a 0 on the end.

Return to Table of Contents

Binary Division

Follow the same rules as in decimal division. For the sake of simplicity, throw away the remainder.

For Example: 111011/11

```
    10011 r 10
   ______
11)111011
  -11
   ______
    101
    -11
   ______
     101
      11
   ______
      10
```

Return to Table of Contents

Decimal to Binary

Converting from decimal to binary notation is slightly more difficult conceptually, but can easily be done once you know how through the use of algorithms. Begin by thinking of a few examples. We can easily see that the number 3= 2+1. and that this is equivalent to (1*2^1)+(1*2^0). This translates into putting a "1" in the 2^1 column and a "1" in the 2^0 column, to get "11". Almost as intuitive is the number 5: it

is obviously 4+1, which is the same as saying [(2*2) +1], or 2^2+1. This can also be written as [(1*2^2)+(1*2^0)]. Looking at this in columns,

```
2^2 | 2^1 | 2^0
 1     0     1
```

or 101.

What we're doing here is finding the largest power of two within the number (2^2=4 is the largest power of 2 in 5), subtracting that from the number (5-4=1), and finding the largest power of 2 in the remainder (2^0=1 is the largest power of 2 in 1). Then we just put this into columns. This process continues until we have a remainder of 0. Let's take a look at how it works. We know that:

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

and so on. To convert the decimal number 75 to binary, we would find the largest power of 2 less than 75, which is 64. Thus, we would put a 1 in the 2^6 column, and subtract 64 from 75, giving us 11. The largest power of 2 in 11 is 8, or 2^3. Put 1 in the 2^3 column, and 0 in 2^4 and 2^5. Subtract 8 from 11 to get 3. Put 1 in the 2^1 column, 0 in 2^2, and subtract 2 from 3. We're left with 1, which goes in 2^0, and we subtract one to get zero. Thus, our number is 1001011.

Making this algorithm a bit more formal gives us:

1. Let D=number we wish to convert from decimal to binary
2. Repeat until D=0
    - a. Find the largest power of two in D. Let this equal P.
    - b. Put a 1 in binary column P.
    - c. Subtract P from D.
3. Put zeros in all columns which don't have ones.

This algorithm is a bit awkward. Particularly step 3, "filling in the zeros." Therefore, we should rewrite it such that we ascertain the value of each column individually, putting in 0's and 1's as we go:

1. Let D= the number we wish to convert from decimal to binary
2. Find P, such that 2^P is the largest power of two smaller than D.
3. Repeat until P<0
    - If 2^P<=D then
        - put 1 into column P
        - subtract 2^P from D
    - Else
        - put 0 into column P
    - End if
    - Subtract 1 from P

Now that we have an algorithm, we can use it to convert numbers from decimal to binary relatively painlessly. Let's try the number D=55.

- Our first step is to find P. We know that 2^4=16, 2^5=32, and 2^6=64. Therefore, P=5.
- 2^5<=55, so we put a 1 in the 2^5 column: `1-----`.
- Subtracting 55-32 leaves us with 23. Subtracting 1 from P gives us 4.
- Following step 3 again, 2^4<=23, so we put a 1 in the 2^4 column: `11----`.
- Next, subtract 16 from 23, to get 7. Subtract 1 from P gives us 3.
- 2^3>7, so we put a 0 in the 2^3 column: `110---`
- Next, subtract 1 from P, which gives us 2.
- 2^2<=7, so we put a 1 in the 2^2 column: `1101--`
- Subtract 4 from 7 to get 3. Subtract 1 from P to get 1.
- 2^1<=3, so we put a 1 in the 2^1 column: `11011-`
- Subtract 2 from 3 to get 1. Subtract 1 from P to get 0.
- 2^0<=1, so we put a 1 in the 2^0 column: `110111`
- Subtract 1 from 1 to get 0. Subtract 1 from P to get -1.

- P is now less than zero, so we stop.

Another algorithm for converting decimal to binary

However, this is not the only approach possible. We can start at the right, rather than the left.

All binary numbers are in the form

```
a[n]*2^n + a[n-1]*2^(n-1)+...+a[1]*2^1 + a[0]*2^0
```

where each a[i] is either a 1 or a 0 (the only possible digits for the binary system). The only way a number can be odd is if it has a 1 in the 2^0 column, because all powers of two greater than 0 are even numbers (2, 4, 8, 16...). This gives us the rightmost digit as a starting point.

Now we need to do the remaining digits. One idea is to "shift" them. It is also easy to see that multiplying and dividing by 2 shifts everything by one column: two in binary is 10, or (1*2^1). Dividing (1*2^1) by 2 gives us (1*2^0), or just a 1 in binary. Similarly, multiplying by 2 shifts in the other direction: (1*2^1)*2=(1*2^2) or 10 in binary. Therefore

```
{a[n]*2^n + a[n-1]*2^(n-1) + ... + a[1]*2^1 + a[0]*2^0}/2
```

is equal to

```
a[n]*2^(n-1) + a[n-1]*2^(n-2) + ... + a[1]2^0
```

Let's look at how this can help us convert from decimal to binary. Take the number 163. We know that since it is odd, there must be a 1 in the 2^0 column (a[0]=1). We also know that it equals 162+1. If we put the 1 in the 2^0 column, we have 162 left, and have to decide how to translate the remaining digits.

Two's column: Dividing 162 by 2 gives 81. The number 81 in binary would also have a 1 in the 2^0 column. Since we divided the number by two, we "took out" one power of two. Similarly, the statement a[n-1]*2^(n-1) + a[n-2]*2^(n-2) + ... + a[1]*2^0 has a power of two removed. Our "new" 2^0 column now contains a1. We learned earlier that there is a 1 in the 2^0 column if the number is odd. Since 81 is odd, a[1]=1. Practically, we can simply keep a "running total", which now stands at 11 (a[1]=1 and a[0]=1). Also note that a1 is essentially "remultiplied" by two just by putting it in front of a[0], so it is automatically fit into the correct column.

Four's column: Now we can subtract 1 from 81 to see what remainder we still must place (80). Dividing 80 by 2 gives 40. Therefore, there must be a 0 in the 4's column, (because what we are actually placing is a 2^0 column, and the number is not odd).

Eight's column: We can divide by two again to get 20. This is even, so we put a 0 in the 8's column. Our running total now stands at a[3]=0, a[2]=0, a[1]=1, and a[0]=1.

We can continue in this manner until there is no remainder to place.

```
  Let's formalize this algorithm:
1.  Let D= the number we wish to convert from decimal to binary.
2.  Repeat until D=0:
    a) If D is odd, put "1" in the leftmost open column, and subtract 1 from D.
    b) If D is even, put "0" in the leftmost open column.
    c) Divide D by 2.
    End Repeat
For the number 163, this works as follows:
1.  Let D=163
2.  b) D is odd, put a 1 in the 2^0 column.
  Subtract 1 from D to get 162.
    c) Divide D=162 by 2.
Temporary Result: 01     New D=81
D does not equal 0, so we repeat step 2.

2.  b) D is odd, put a 1 in the 2^1 column.
  Subtract 1 from D to get 80.
     c) Divide D=80 by 2.
Temporary Result: 11     New D=40
D does not equal 0, so we repeat step 2.

2.  b) D is even, put a 0 in the 2^2 column.
    c) Divide D by 2.
Temporary Result:011    New D=20

2.  b) D is even, put a 0 in the 2^3 column.
    c) Divide D by 2.
Temporary Result: 0011     New D=10

2.  b) D is even, put a 0 in the 2^4 column.
    c) Divide D by 2.
Temporary Result:  00011      New D=5

2.  a) D is odd, put a 1 in the 2^5 column.
```

```
       Subtract 1 from D to get 4.
    c) Divide D by 2.
Temporary Result:  100011   New D=2

2.  b) D is even, put a 0 in the 2^6 column.
    c) Divide D by 2.
Temporary Result:  0100011   New D=1

2.  a) D is odd, put a 1 in the 27 column.
       Subtract 1 from D to get D=0.
    c) Divide D by 2.
Temporary Result:  10100011   New D=0

D=0, so we are done, and the decimal number 163 is equivalent to the binary number 10100011.
```

Since we already knew how to convert from binary to decimal, we can easily verify our result. 10100011=(1*2^0)+(1*2^1)+(1*2^5)+(1*2^7)=1+2+32+128= 163.

Return to Table of Contents

Negation in the Binary System

These techniques work well for non-negative integers, but how do we indicate negative numbers in the binary system?

Before we investigate negative numbers, we note that the computer uses a fixed number of "bits" or binary digits. An 8-bit number is 8 digits long. For this section, we will work with 8 bits.

Signed Magnitude:

The simplest way to indicate negation is signed magnitude. In signed magnitude, the left-most bit is not actually part of the number, but is just the equivalent of a +/- sign. "0" indicates that the number is positive, "1" indicates negative. In 8 bits, 00001100 would be 12 (break this down into (1*2^3) + (1*2^2)). To indicate -12, we would simply put a "1" rather than a "0" as the first bit: 10001100.

One's Complement:

In one's complement, positive numbers are represented as usual in regular binary. However, negative numbers are represented differently. To negate a number, replace all zeros with ones, and ones with zeros - flip the bits. Thus, 12 would be 00001100, and -12 would be 11110011. As in signed magnitude, the leftmost bit indicates the sign (1 is negative, 0 is positive). To compute the value of a negative number, flip the bits and translate as before.

Two's Complement:

Begin with the number in one's complement. Add 1 if the number is negative. Twelve would be represented as 00001100, and -12 as 11110100. To verify this, let's subtract 1 from 11110100, to get 11110011. If we flip the bits, we get 00001100, or 12 in decimal.

In this notation, "m" indicates the total number of bits. For us (working with 8 bits), it would be excess 2^7. To represent a number (positive or negative) in excess 2^7, begin by taking the number in regular binary representation. Then add 2^7 (=128) to that number. For example, 7 would be 128 + 7=135, or 2^7+2^2+2^1+2^0, and, in binary,10000111. We would represent -7 as 128-7=121, and, in binary, 01111001.

Note:

- Unless you know which representation has been used, you cannot figure out the value of a number.
- A number in excess 2^(m-1) is the same as that number in two's complement with the leftmost bit flipped.

To see the advantages and disadvantages of each method, let's try working with them.

Using the regular algorithm for binary adition, add (5+12), (-5+12), (-12+-5), and (12+-12) in each system. Then convert back to decimal numbers.

Click here to see the answers Return to Table of Contents

Answers

What would the binary number 1011 be in decimal notation?

```
  1011=(1*2^3)+(0*2^2)+(1*2^1)+(1*2^0)
      = (1*8) + (0*4) + (1*2) + (1*1)
  = 11 (in decimal notation)
```

Go back to the question

Try converting these numbers from binary to decimal:

```
10=(1*2^1) + (0*2^0) = 2+0 = 2
111 = (1*2^2) + (1*2^1) + (1*2^0) = 4+2+1=7
10101= (1*2^4) + (0*2^3) + (1*2^2) + (0*2^1) + (1*2^0)=16+0+4+0+1=21
11110= (1*2^4) + (1*2^3) + (1*2^2) + (1*2^1) + (0*2^0)=16+8+4+2+0=30
```

Go back to the question

Try a few examples of binary addition:

```
                      1               1
          111            111            111
         +110                 +110           +110
        ______            ______         _____
               1                   01          1101

           1                      11            1
  101            101    101
        +111                 +111            +111
        _____                ____            _____
              0                    00          1100

              1                   1                       1
     111    111           111
           +111                 +111                 +111
           _____                _____                _____
         0                      10                 1110
```

Click here to return to the question

Using the regular algorithm for binary adition, add (5+12), (-5+12), (-12+-5), and (12+-12) in each system. Then convert back to decimal numbers.

```
Signed Magnitude:

    5+12         -5+12         -12+-5          12+-12

 00000101       10000101       10001100       00001100
 00001100       00001100       10000101       10001100
__________      ________       ________       _________
 00010001       10010001       00010000       10011000

    17            -17            16             -24

One' Complement:

 00000101       11111010       11110011       00001100
 00001100       00001100       11111010       11110011
_________       ________       ________       ________
 00010001       00000110       11101101       11111111

    17             6             -18            0

Two's Complement:

 00000101       11111011       11110100       00001100
 00001100       00001100       11111011       11110100
 ________       ________       ________       ________
 00010001       00000111       11101111       00000000

     17            7             -17             0

Signed Magnitude:

 10000101       01111011       01110100       00001100
 10001100       10001100       01111011       01110100
 ________       ________       ________       ________
 00010001       00000111       11101111       01111100

    109           119            111            124
```

Click here to return to the question