

CSI 2350 Notes for Aug 21-23.

Read pages 199-208 (Chapter 5 section 6, Binary and Hexadecimal numbers)

We are so used to representing numbers in decimal, that it seems natural to do so. In fact, the number system we use is completely artificial. We have arbitrarily made up nine digits, plus the zero, and use these symbols to write our numbers. We place the symbols in order, 0,1,2,3,4,5,6,7,8,9, and begin writing numbers by writing down the symbols in order. When we run out of symbols we add a 1 to the left, place a zero after it, and proceed to list the symbols in order again. This procedure gives us a way to write any number we want by proceeding in an orderly fashion.

This process will work with any number of symbols. The number of symbols is called the base or the radix of the number system. Our usual number system is a base-10 or a radix-10 number system because there are ten symbols. The binary system uses only two symbols 1 and 0. The trinary system uses three symbols, 0, 1, and 2. The octal system uses 8 symbols 0, 1, 2, 3, 4, 5, 6, and 7. The hexadecimal system uses 16 symbols 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E, and F. (Sometimes lower case letters are used.) Regardless of which system is used, we proceed in the same manner as in decimal, and write down numbers in an orderly way. Whichever system we use, we can write any number we choose by applying the same set of rules to our symbols. The following table shows a list of numbers in several different bases. Each line shows the same number in each different base.

Decimal	Binary	Trinary	Octal	Hexadecimal
1	1	1	1	1
2	10	2	2	2
3	11	10	3	3
4	100	11	4	4
5	101	12	5	5
6	110	20	6	6
7	111	21	7	7
8	1000	22	10	8
9	1001	100	11	9
10	1010	101	12	A
11	1011	102	13	B
12	1100	110	14	C
13	1101	111	15	D
14	1110	112	16	E
15	1111	120	17	F
16	10000	121	20	10
17	10001	122	21	11
18	10010	200	22	12
19	10011	201	23	13
20	10100	202	24	14

21	10101	210	25	15
22	10110	211	26	16
23	10111	212	27	17
24	11000	220	30	18
25	11001	221	31	19
26	11010	222	32	1A
27	11011	1000	33	1B
28	11100	1001	34	1C
29	11101	1002	35	1D
30	11110	1010	36	1E
31	11111	1011	37	1F
32	100000	1012	40	20

Although decimal might be natural for humans, because we have ten fingers, binary is natural for computers. As we will see later, hexadecimal is also used for computer notation (and sometimes octal as well) because there is a straightforward conversion between binary and hexadecimal.

The conversion between decimal and binary, however is more complicated. It is based on repeated division, and what is important is not the quotient, but the remainder.

Suppose I want to convert the number 457,253 into binary. I could write out a table like the one above, but this would take half a life time. Repeated division is much more efficient. Here is the first step in the conversion. I write the division like this. I divide by two each time, so I don't bother writing down the divisor. I write the quotient at the bottom and the remainder at the right.

$$\begin{array}{r} 457,253 \overline{)1} \\ 228,626 \end{array}$$

I continue with this in the following manner.

457,253|1
228,626|0
114,313|1
57,156|0
28,578|0
14,289|1
7,144|0
3,572|0
1,786|0
893|1
446|0
223|1
111|1
55|1
27|1
13|1
6|0
3|1
1|1
0

Now, I read the remainders from bottom to top. **THIS IS IMPORTANT! BOTTOM TO TOP!**

And I get 1101111101000100101. This is the binary representation of 457,253 in binary. Now, how do I convert back from binary into decimal? By multiplying, of course. Suppose I have the binary number 110101010011101 and I want to convert it into decimal. I have to know the powers of 2 to do this. Here is a table. Use your calculator if you need more.

1	16	256	4,096	65,536	1,048,576
2	32	512	8,192	131,072	2,097,152
4	64	1,024	16,384	262,144	4,194,304
8	128	2,048	32,768	524,288	8,388,608

Starting at the right hand side we create the following formula using the powers of two and the binary number.

$$(1 \times 16,384) + (1 \times 8,192) + (0 \times 4,096) + (1 \times 2,048) + \\ (0 \times 1,024) + (1 \times 512) + (0 \times 256) + (1 \times 128) + (0 \times 64) + \\ (0 \times 32) + (1 \times 16) + (1 \times 8) + (1 \times 4) + (0 \times 2) + (1 \times 1)$$

As we move from right to left, we use the next digit and the next highest power of 2. We can eliminate those terms with zeros in them and get this:

$$(1 \times 16,384) + (1 \times 8,192) + (1 \times 2,048) + (1 \times 512) + (1 \times 128) + \\ (1 \times 16) + (1 \times 8) + (1 \times 4) + (1 \times 1)$$

Next, we can get rid of the 1's and just leave the powers of 2.

$$16,384 + 8,192 + 2,048 + 512 + 128 + 16 + 8 + 4 + 1$$

We can use our calculator to add this up and get 27,963. (Check my arithmetic on EVERYTHING.)

Converting between binary and hexadecimal is much easier. To do this we use the following table. You should commit this table to memory.

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

To convert the binary number 11010001010100111101010101101 to hexadecimal, first group the number into four-bit groups starting at the right. (Bit is short for Binary Digit.)

$$1,1010,0010,1010,0111,1010,1010,1101$$

We add leading zeros at the left to make a four bit group at the left, like this.

0001,1010,0010,1010,0111,1010,1010,1101

Now we use the table and substitute hexadecimal digits for the four-bit groups.

1A2A7AAD. This is the representation of the number in hexadecimal. (Computer people usually say “hex” instead of hexadecimal.)

To convert from hexadecimal to binary we reverse the process. To convert the number 1D3C09D to binary, just use the table to substitute a four bit group for each hexadecimal digit. This will give you the following.

0001,1101,0011,1100,0000,1001,1101

We then eliminate the commas and the leading zeros to give us a binary number.

1110100111100000010011101

The process of converting between octal and binary is the same except we use the following table, and three-bit groupings.

0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111