Converting Binary (base 2) to Denary (base 10)

Say we are converting the binary number 1101 0110 into denary; add the following headings over each number:

128	64	32	16	8	4	2	1
1	1	0	1	0	1	1	0

Then simply multiply to find the total:

(1x128) + (1x64) + (0x32) + (1x16) + … = <u>214</u>

Converting Denary (base 10) to Binary (base 2)

The reverse process is to take out the largest number (power of 2) you can, like this:

214 – can we take out 128?	Yes	1	Remainder = 86
86 – can we take out 64?	Yes	1	Remainder = 22
22 – can we take out 32?	No	0	Remainder = 22
22 – can we take out 16?	Yes	1	Remainder = 6
6 – can we take out 8?	No	0	Remainder = 6
6 – can we take out 4?	Yes	1	Remainder = 2
2 – can we take out 2?	Yes	1	Remainder = 0
0 – can we take out 1?	No	0	Remainder = 0

Answer = <u>1101 0110</u>

Converting Binary (base 2) to Hexadecimal (base 16)

Say we are converting the binary number 1101 0110 into hexadecimal; split the number into two 4-bit nibbles and convert them into denary *(if the number does not have the right number of digits, simply add zeros to the LHS).*

1101 = 13 0110 = 6

Then, convert each denary number into a single hex digit (where 10 = A, 11 = B, 12 = C, 13 = D, 14 = E, 15 = F)

13 = D 6 = 6

Therefore:

<u>1101 0110 = D6</u>

Converting Hexadecimal (base 16) into Binary (base 2)

Lets convert D6 back into binary. First convert each hex digit in to a denary number and then convert that into binary:

D6 = 13 6

13 = (1x8) + (1x4) + (0x2) + (1x1) = 1101

6 = (0x8) + (1x4) + (1x2) + (0x1) = 0110

Therefore:

<u>D6 = 1101 0110</u>

Adding binary numbers

Lets add 0110 1010 to 00101101 Note: = 106 + 45 = 151

First, write them out like this:

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

Just like denary adding, add the two digits together and carry the 1 if necessary:

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

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

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

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

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

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

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

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

Note: = 151

Negative binary numbers – 2s complement

With 8 bits we can store any **positive integer** from 0 to 255. But what about **negative integers**?

The answer is to change the range from -128 to +127, using the first bit to indicate the sign; thus **1**000 000 would indicate that the number is **negative** and the lowest possible number (-128) and **0**111 1111 would be both **postivie** and the highest possible number (+127).

This means that all the positive numbers still work as expected (simply ignoring the leading 0) and you don't end up with two 0 value (positive 0 and negative 0).

To convert a negative denary number into binary using 2s complement

- Convert the positive number into binary
- Invert each bit
- Add 1

e.g. -127

Convert to binary:	0111 1111
Invert each bit:	1000 0000
Add 1:	1000 0001

e.g. -37

Convert to binary:	0010 0101
Invert each bit:	1101 1010
Add 1:	1101 1011

To convert a negative binary number into denary using 2s complement

Simply reverse the process:

e.g. 1000 0001

Subtract 1:	1000 0000
Invert each bit:	0111 1111
Convert to denary:	-127

e.g. 1101 1011

Subtract 1:	1101 1010
Invert each bit:	0010 0101
Convert o denary	-37

Subtracting binary numbers

Rather than subtracting a positive number, try adding a negative number.
+106 - +45 = +106 + -45

Lets work out 0110 1010 - 00101101 Note: = 106 - 45 = 61

First, use the 2s complement to invert the second number:

-0010 1101 = +(1101 0010 + 1) = 1101 0011

Then do the addition:

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

Discard any leading digits (remember the first digit just indicates the sign - +/-)

0011 1101 = 61