Binary Number Systems

By dividing by 2 repeatedly we are representing each bit of the number.

We write the numbers backwards because we are increasing the power of 2 that the number is divided by. So 0 is not the number divided by 2, rather it is the number divided by x. An example can be seen below:

284731	Bit	Divided by
142365	1	2
71182	1	$2^2 = 4$
35591	0	$2^3 = 8$
17795	1	
8897	1	
4448	1	
2224	0	
1112	0	
556	0	
278	0	
139	0	
69	1	
34	1	
17	0	
8	1	
4	0	
2	0	
1	0	
0	1	

This is how the data is represented within memory. The excess space is filled with 0's. So for the above example, if written in 32 bit format would have 13 leading 0s.

Octal digits are formed with groupings of digits, So the octal of above is

00	000	000	000	001	000	101	100	000	111	011
0	0	0	0	1	0	5	4	0	7	3

Hexidecimal representation is done in the same way, however it is grouped in 4's

0000	0000	0000	0100	0101	1000	0011	1011
0	0	0	4	5	8	3	B

- Starting from the least significant digit bit, multiply the digits of the binary number with increasing powers of 2.
- The least signifcant bit (LSB) is multiplied with 2^0, the next bit by 2^1, and so on.
- Then add the products

0	1	1	0	0	1	1	0	1	0	1	1	0	0	0	1
0^{15}	0^{14}	2^{13}	0^{12}	0^{11}	2^{10}	2^9	0^8	2^7	0^6	2^5	2^4	0^3	0^2	0^1	2^0

For binary addition we need to carry if the product is greater than 1. We shift so they allign on the left side of the table

For subtraction we need to borrow, the borrowed number is 10, at a lower bit.

```
0 - 0 = 0
1 - 0 = 1
1 - 1 = 0
10 - 1 = 1
```

a = 01110011 (115)
b = 00011111 (31)

So...

$$01110011 - 00011111 = 01010100(84)$$

-	-	-	-
1	0	1	0
-	1	0	1
0	0	0	1

- Two's complement in n bits:
  - The negative of a twos complement is given by inverting each bit (0 to 1 or 1 to 0) and then adding one, ignoring any carry between n bits.
- If numbers are represented in n bits:
  - the positive numbers are from 0000..01 to 0111..11,
  - 0 is all 0: 000..00
  - the negative numbers are from 1000..00 to 1111..11
  The leading bit sign 0 means non negative, the leading 1 means negative.

bit 1	bit 2	bit 3	bit4
0	0	0	0
1	1	1	1

now if we add one to the twos compliment shown of `0000` (compliment is 1111) we get the result of `1 0000`, however because the 1 is out of range $0 = 0$

Binary	Decimal	Signed Binary
NA	-3	100
NA	-2	101
NA	-1	110
0	0	000
001	1	001
010	2	010
011	3	011

Example: Calculating Two's Complement

- Given number: 01010101
- One's Complement: 10101010
- Two's Complement: 10101010 + 1 = 10101011
- Applications of Two's Complement:
  - Arithmetic operations (addition, subtraction, multiplication, division)
  - Negative number representation in computers
  - Overflow detection
- Benefits of Two's Complement:
  - Simplifies arithmetic operations
  - Eliminates the need for separate addition and subtraction circuits
  - Only one representation of zero (no positive and negative zero)

$-89_{10} =$

-	1	1	0	1	0	1
43	21	10	5	2	1	0

Twos compliment as a formula is therefore:

$$2^n - x$$

- Also in 2's complement notation, ^{there} is the only 1 representation for 0.
- We often need to convert a number in n bits to a number represented with more than n bits
- an example of this would be char to int for example
- This can be done by taking the most significant bit from the sorter one and replicating it to fill the new bits of the larger one

- Existing bits are simply copied

So we duplicate the most significant digit, so
`char x = -107 != 0000000000010101`
`if short y = x ... its = 111110010101

we DON'T add 0's, we duplicate the most significant digit