Moving between bases:

Base 10 $\rightarrow$ Base 2

Example:

Convert $25_{10} \rightarrow$ binary

Method:

Continually divide by 2 and read the remainders in reverse order.

25/2	$\rightarrow$	12	(1)
12/2	$\rightarrow$	6	(0)
6/2	$\rightarrow$	3	(0)
3/2	$\rightarrow$	1	(1)
1/2	$\rightarrow$	0	(1)

Answer = 11001_2

Each digit in a binary number is called a bit (a Binary digit).

Typically more digits are needed in a smaller base to represent the same information than in a higher base. However, we need fewer symbols (states) in a lower base.
This is a tradeoff. Manipulating many states is complex, but using many digits is resource-intensive (i.e. uses many transistors).

Converting from Binary Back to Decimal

Method:

Starting from the least significant bit, multiply each bit by successive powers of 2, beginning with 2^0, and add each product together.

Example:

11001_2

	1	x	2^0
+	0	x	2^1
+	0	x	2^2
+	1	x	2^3
+	1	x	2^4

Answer: $1 + 0 + 0 + 8 + 16 = 25_{10}$

The methods above will work for converting decimal to any base and back. Simply use that base as divisor when converting forward and use powers of that base when converting back.

Example: Convert $96_{10} \rightarrow$ base 4

96/4	$\rightarrow$	24	(0)
24/4	$\rightarrow$	6	(0)
6/4	$\rightarrow$	1	(2)
1/4	$\rightarrow$	0	(1)

Answer = 1200_4

Convert back:

1200_4

	0	x	4^0
+	0	x	4^1
+	2	x	4^2
+	1	x	4^3

Answer: 0 + 0 + 32 + 64 = 96_{10}

Counting in Binary (positional number system)

Binary	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

With four bits, we can represent 16 values.

Remember we can always compare numbers in counting order. This sometimes saves us converting.

In general,	*the first column changes every*	*$base^0$ times*
	the 2^{nd} …	*$base^1$ …*
	the 3^{rd} …	*$base^3$ …*

We can use this rule to remember how to count in any base.

Note: A limited number of bits means a limited number of combinations of those bits.

4 bits $\Rightarrow 2^4$ (=16) combinations

In general, n bits give 2^n combinations.

While in practice you can engineer methods to give you larger numbers than possible with a given number of bits, but in theory you can't get bigger number. E.g. a 32-bit machine is in theory limited to approximately 4 billion numbers.