Introduction to Computing

Mathematics for Computing

Malay Bhattacharyya

Associate Professor

MIU, CAIML, TIH
Indian Statistical Institute, Kolkata

August, 2025

1. The Number Systems
2. Binary Arithmetic
3. Boolean Algebra

Decimal, Binary, Octal, Hexadecimal

Given the number $x_k x_{k-1} \cdots x_1 x_0$ represented in base b, its decimal value is:

$$\sum_{i=0}^{k} x_i * b^i.$$

Number conversions

	To decimal	**Multiplier**
Binary [0-1]	$b_k \cdots b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0$	
	$\cdots 2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$	Power of 2
Octal [0-7]	$b_k \cdots \overline{b_7\ b_6}\ \overline{b_5\ b_4\ b_3}\ \overline{b_2\ b_1\ b_0}$	
	$\cdots o_1 * 8^1 \quad o_0 * 8^0$	Power of 8
Hexadecimal [0-F]	$b_k \cdots \overline{b_7\ b_6\ b_5\ b_4}\ \overline{b_3\ b_2\ b_1\ b_0}$	
	$\cdots h_1 * 16^1 \quad h_0 * 16^0$	Power of 16

Number conversions

Decimal 6.625 equals to 110.101 in binary.

Converting the integer part 6 to binary:

Integer	Operation	Quotient (Integer)	Remainder
6	6 / 2	3	0
3	3 / 2	1	1
1	1 / 2	0 [STOP]	1

↑

Converting the fractional part 0.625 to binary:

Fraction	Operation	Fraction	Integer
0.625	0.625 * 2	0.25	1
0.25	0.25 * 2	0.5	0
0.5	0.5 * 2	0 [STOP]	1

↓

Arithmetic operations in binary

Addition	**Subtraction**	**Multiplication**	**Division**
$0 + 0 = 0$	$0 - 0 = 0$	$0 \times 0 = 0$	$0 \ / \ 0 =$ NA
$0 + 1 = 1$	$0 - 1 = 1$ (borrow 1)	$0 \times 1 = 0$	$0 \ / \ 1 = 0$
$1 + 0 = 1$	$1 - 0 = 1$	$1 \times 0 = 0$	$1 \ / \ 0 =$ NA
$1 + 1 = 0$ (carry 1)	$1 - 1 = 0$	$1 \times 1 = 1$	$1 \ / \ 1 = 1$

Note: The carry bit and borrow bit are required whenever an overflow and underflow happen, respectively.

Representation of unsigned integers

Binary	Decimal
00000000 00000000	0
00000000 00000001	+1
00000000 00000010	+2
00000000 00000011	+3
...	...
01111111 11111111	+32767
10000000 00000000	+32768
10000000 00000001	+32769
...	...
11111111 11111111	+65535

Note: The number of bits is fixed (depends upon the architecture).

Representation of signed integers

Binary	Decimal
1 0000000 00000000	-32768
...	...
1 1111111 11111101	-3
1 1111111 11111110	-2
1 1111111 11111111	-1
0 0000000 00000000	0
0 0000000 00000001	+1
0 0000000 00000010	+2
0 0000000 00000011	+3
...	...
0 1111111 11111111	+32767

Note: (1's complement of x) $+ 1 =$ 2's complement of $x = -x$, wherein 1's complement flips the bits in x.

Bitwise shift

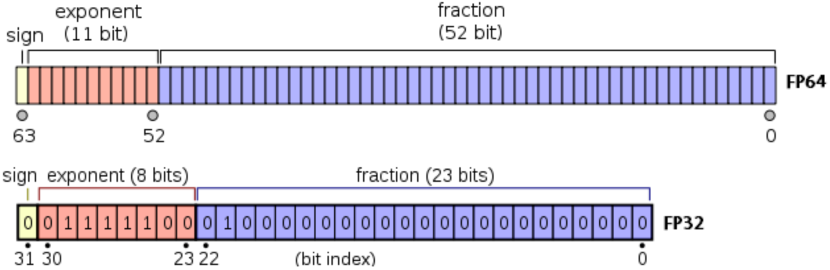
```
i = 11
i >> 1 = 5
i <<2 = 44
```

Bitwise shift

```
i = 11
i >> 1 = 5
i <<2 = 44
```

Decimal	Binary							
11	0	0	0	0	1	0	1	1
5	0	0	0	0	0	1	0	1
44	0	0	1	0	1	1	0	0

Floating point representation



Basics

Boolean algebra is a mathematical area that deals with operations on logical values with binary variables.

Boolean variables are represented as binary numbers to represent truthfulness, i.e., 1 denotes TRUE and 0 denotes FALSE.

De Morgan's theorems

- not (B_0 and B_1 and ... and B_n) = (not B_0) or (not B_1) or ... or (not B_n), where each B_i is a Boolean expression.
- not (B_0 or B_1 or ... or B_n) = (not B_0) and (not B_1) and ... and (not B_n), where each B_i is a Boolean expression.

Note: The original theorems are based on the union and intersection of sets.

Let's solve problems

Prove that $(n << 3) + (n << 1)$ will perform left rotation of the decimal digits in n by one position with zero padding for positive integers n. Note that $<<$ denotes bitwise left shift operation.

Let's solve problems

Prove that $(n << 3) + (n << 1)$ will perform left rotation of the decimal digits in n by one position with zero padding for positive integers n. Note that $<<$ denotes bitwise left shift operation.

Proof.

The expression $(n << 3) + (n << 1)$ reduces to the following:

$$(n * 2^3) + (n * 2^1) = 10 * n.$$

This is nothing but left rotation of the decimal digits in n by one position with zero padding. □

Let's solve problems

Prove that if n is a power of 2 then n & $(n-1)$ is 0. Note that & denotes bitwise AND operation.

Let's solve problems

Prove that if n is a power of 2 then n & $(n–1)$ is 0. Note that & denotes bitwise AND operation.

Proof.

If n is a power of 2, it can be written as follows:

$$1 * 2^k + 0 * 2^{k-1} + 0 * 2^{k-2} + \cdots + 0 * 2^1 + 0 * 2^0.$$

So, $n - 1$ can be written as follows:

$$1 * 2^{k-1} + 1 * 2^{k-2} + \cdots + 1 * 2^1 + 1 * 2^0.$$

Hence, n & $(n–1)$ will reduce to 0 as every pair of bits will differ. □

Let's solve problems

Prove that for any arbitrary pair of bits A and B, the following will hold: not (A and B) = (not A) or (not B).

Let's solve problems

Prove that for any arbitrary pair of bits A and B, the following will hold: not (A and B) = (not A) or (not B).

Proof.

This simply follows as a special case of one of the De Morgan's theorems. However, we can also prove this with truth tables as follows.

A	B	not (A and B)	(not A) or (not B)
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

Hence, not (A and B) = (not A) or (not B). □

Homework

Prove or disprove the following statements:

- The operation $n = n \wedge (1 << p)$ will toggle (flip) the bit in the position p of the integer n. Note that $\wedge$ and $<<$ denote bitwise XOR and bitwise left shift operation, respectively.
- The binary number $[xy] \ldots 3k$ times is always divisible by 7, where x and y are any arbitrary bit (0 or 1).
- The Boolean expression (A and (A or B)) and (not A) gets always evaluated as False.