Bitwise operations used in C language

Imre Varga PhD

University of Debrecen
Faculty of Informatics
Department of IT Systems and Networks

13 June 2026

Would you like to create a bit pattern?

Use **hexadecimal** notation.

Use, utilization:

For example, creating **mask**s.

Expected value:

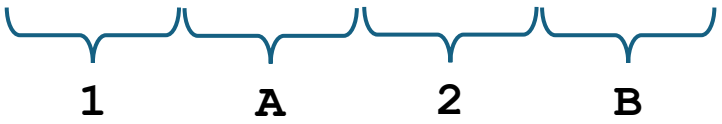
```
0001101000101011
```

Required code:

```
short int N = 0x1A2B;
// decimal: 6699
// octal: 015053
```

Mathematical background:

All hexadecimal digits mean a group of 4 bits.



Would you like to set a few bits without changing others?
Use **bitwise or** operation!

Initial value:

`0101001100001111`

Expected value:

`0101001101101111`

Required code:

```
short int N = 0x530F;
short int M = 0x0060;
// 0000 0000 0110 0000
N = N|M;
```

Mathematical background:

$x \vee F = x$ unchanged

$x \vee T = T$ set to 1

Would you like to clear a few bits without changing others?

Use **bitwise and** operation!

Initial value:

```
0101001100001111
```

Expected value:

```
0101000001101111
```

Required code:

```
short int N = 0x530F;
short int M = 0x0300;
// 0000 0011 0000 0000
N = N&M;
```

Mathematical background:

$x \wedge F = F$ cleared to 0

$x \wedge T = x$ unchanged

Would you like to invert some bits without changing the rest?

Use **bitwise exclusive or** operation!

Initial value:

`0101001100001111`

Expected value:

`0010100001101111`

Required code:

```
short int N = 0x530F;
short int M = 0x7800;
// 0111 1000 0000 0000
N = N^M;
```

Mathematical background:

$x \oplus F = x$ unchanged

$x \oplus T = \bar{x}$ inverted

Would you like to invert all bits?

Use **bitwise negation** operation!

Initial value:

`0101001100001111`

Expected value:

`1010110011110000`

Required code:

```
short int N = 0x530F;
N = ~N; //0xACF0
```

Mathematical background:

$\neg x = \bar{x}$ all bits reversed

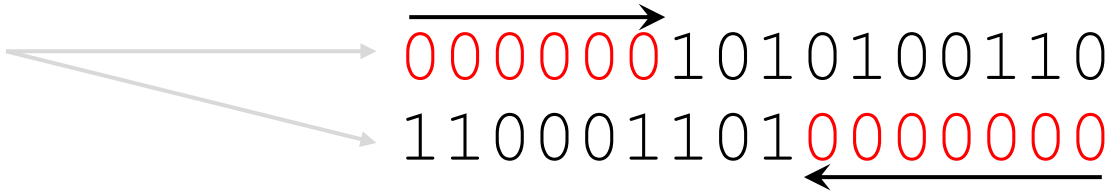
Would you like to shift the entire bit pattern so that no new 1 bits appear?

Use a **logical shift** operation!

Initial value:

1010100110001101

Expected values:



Required code:

```
unsigned short int N,A,B;
N = 0xA98D;
A = N>>6;
B = N<<7;
```

Mathematical background:

All bits move.
0 bits are placed in the vacant bit positions if the shift is performed on an unsigned value.

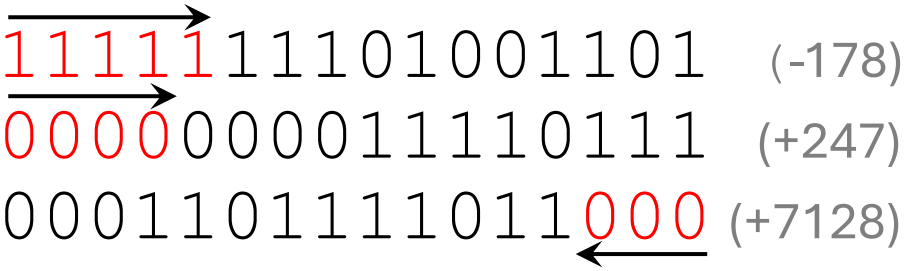
Would you like to shift the entire bit pattern so that it results in multiplication/division?

Use **arithmetic shift** operation!

Initial values:

1110100110100110 (-5721)
0000111101111100 (+3964)
0000001101111011 (+891)

Expected values:



Required code:

```
short int X,Y,Z;
X=0xE9A6>>5; // -5721/32
Y=0x0F7C>>4; // +3964/16
Z=0x027B<<3; // +891*8
```

Mathematical background:

All bits move.
Sign bit copies are added from the left, 0s from the right. When shifting to the right: the sign is unchanged, the quotient is truncated to an integer.

Complex example

Initial value:

Expected value:

Required code:

```
unsigned short int N = 0x5B4D;
N = N|0xC000;
N = N&0x1800;
N = N^0x0100;
N = ((N&0x0070)>>4) | ((N&0x0007)<<4) | (N&0xFF88);
```