Operatory bitowe - operują na bitach, podstawowych jednostkach informacji przechowywanych w komputerze za pomocą impulsów elektrycznych (01010)

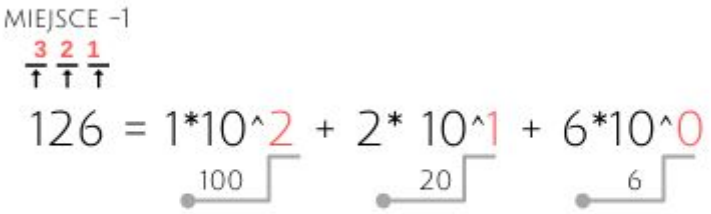
OPERATORY BITOWE

0 - false
1 - true

OPERATORY BITOWE	ROLA OPERATORA
&	iloczyn bitowy
|	suma bitowa
^	XOR e**X**clusive**OR**
x << 1	przesunięcie w lewo o 1
x >> 2	przesunięcie w prawo o 2
~	negacja bitowa

Wyróżniamy

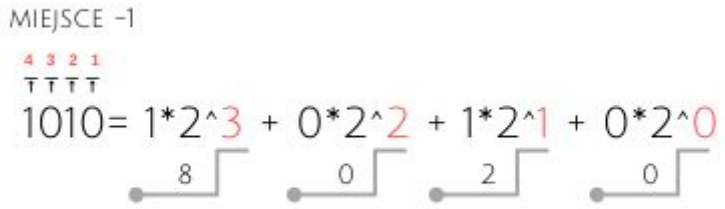
- ❏ system dziesiętny (od 0 do 9)



Mnożymy pierwszą liczbę (1) przez system dzięsietny, czyli 10 następnie podnosimy go do potęgi (podajemy miejsce liczby). Przy podaniu **miejsca** odejmujemy 1 czyli jeśli liczba występuje na pozycji nr 3 odejmujemy od niej 1 w tym momencie jej pozycja to 2. Kolejne liczby obliczamy identycznie zgodnie z regułą.

- ❏ system dwójkowy (od 0 do 2)

Przeliczamy liczbę 1010 systemem dwójkowym, aby sprawdzić jaka to liczba w systemie dziesiętnym



8 + 0 + 2 + 0 = 10

W systemie dwójkowym zapis **1010** to w postaci dziesiętnej **liczba 10.**

Aby ułatwić sobie sposób przeliczania liczb na system dziesiętny

miejsce	4	3	2	1
miejsce -1	**3**	**2**	**1**	**0**
	1	0	1	0

- ❏ Jeżeli występuje 0 - pomijamy tą wartość
- ❏ Miejsce każdej liczby 1 symbolizuje liczbę dwa (2) podniesioną do potęgi miejsca w którym się znajduję -1 czyli np.

1 0 **1** 0 to inaczej

2 ^ 3 = 8
0 ^ 2 = 0
2 ^ 1 = 2
0 ^ 0 = 0

8 + 2 = 10

Ostatnia liczba symbolizuje czy dana wartość jest parzysta bądź nie
np. 1010

- jeśli na końcu występuje **1 - nieparzysta**
- jeśli na końcu występuje **0 - parzysta**

Operacje bitowe :

★ **iloczyn bitowy -** operator ten działa jak koniunkcja, różnicą jest to że nie operuje na prawdzie i fałszu lecz na zerach i jedynkach. Koniunkcja jest prawdziwa wtedy, gdy oba warunki są spełnione.

1 1 1 0 // 14
1 0 1 1 // 11
---
1 0 1 0
8 + 2 = 10

```
System.out.println(14 & 11);
```

Otrzymany wynik to 10.

★ **suma bitowa -** działa jak alternatywa. Alternatywa jest fałszywa tylko wtedy, gdy oba argumenty są fałszywe.

1 0 1 0 // 10
1 0 1 1 // 11
---
1 0 1 1
8 + 2 + 1 = 11

```
System.out.println(10 | 11);
```

Otrzymany wynik to 11.

★ **XOR eXclusive OR (albo) -** gdy obie wartości są prawdziwe lub fałszywe to wynik jest fałszywy.

1 1 - 0 wartość fałszywa
0 0 - 0 wartość fałszywa
1 0 - 1 wartość prawdziwa
0 1 - 1 wartość prawdziwa

1 0 1 0 // 10
1 0 1 1 // 11
---
0 0 0 1 // 1

```
System.out.println(10 ^ 11);
```

Otrzymany wynik to 1.

XOR gdy ma podane te same wartości to zawsze zwróci 0, np.

```
System.out.println(5 ^ 5);
```

Natomiast, gdy już wartości będą się różnić zwróci liczby różne od 0.

★ Przesunięcie w prawo

Gdy napiszemy np.

0 00 000 1010 i przesuniemy w prawo o 1 otrzymamy :
0 00 00 0101

4 +1 = 5

```
System.out.println(10 >> 1);
```

Wynik otrzymany to 5, nastąpiła **operacja dzielenia** przez 2.

★ Przesunięcie w lewo

```
System.out.println(10 << 1);
```

Przemnożyliśmy 10 * 2
Nasz wynik to 20.

`1 -` liczba (*2) która mówi do której potęgi mamy podnieść liczbę, która znajduje się przed strzałkami

★ Operator negacji - zmienia wartość

1 0 1 0 **zmienia na**
0 1 0 1

```
System.out.println(~10);
```

Otrzymany wynik to -11

-