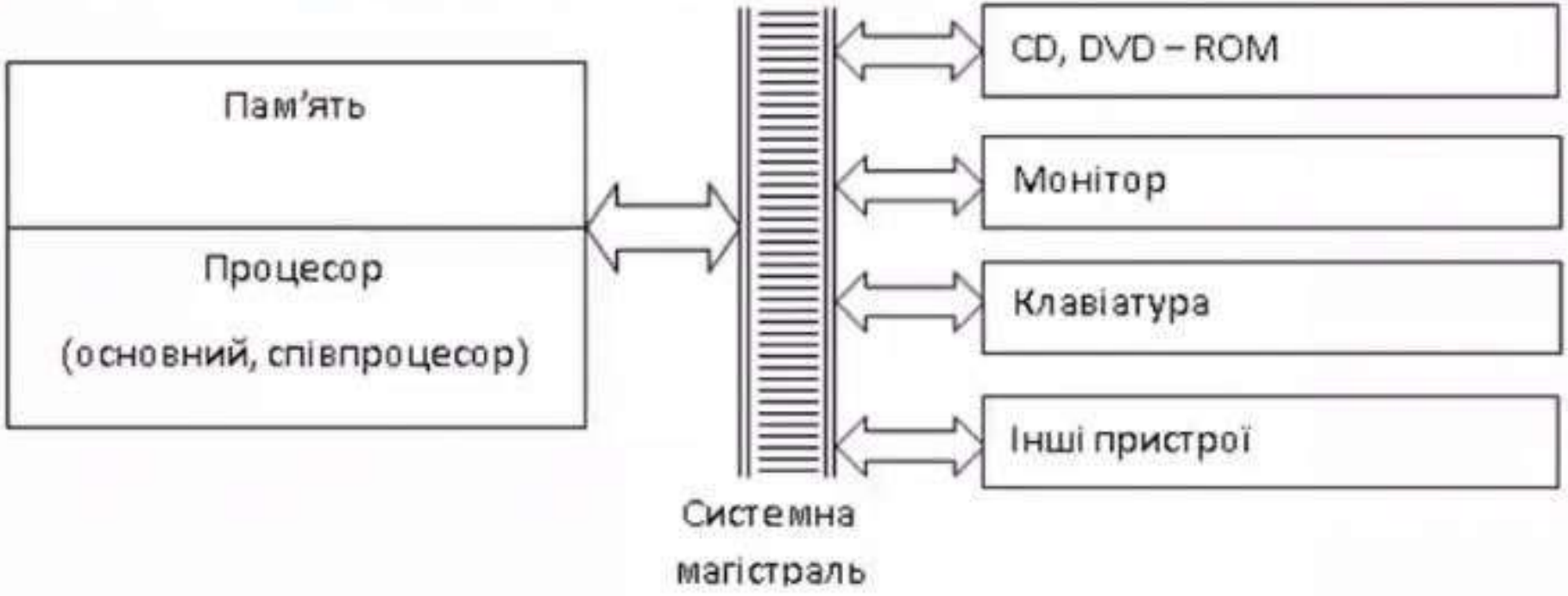
ТЕМА: Структура комп'ютера. Пам'ять, величини, операції

Структурна схема сучасного комп'ютера

Комп'ютер – це є спеціальний електронний пристрій, який вміє виконувати елементарні операції. Сукупність елементарних операцій складає алгоритм розв'язку задачі. Всі сучасні комп'ютери побудовані за однаковими структурними принципами. Типова структурна схема комп'ютера виглядає так:



Під час роботи процесор постійно взаємодіє з пам'яттю, виконуючи елементарні операції. Це є головна змістова частина роботи комп'ютера.

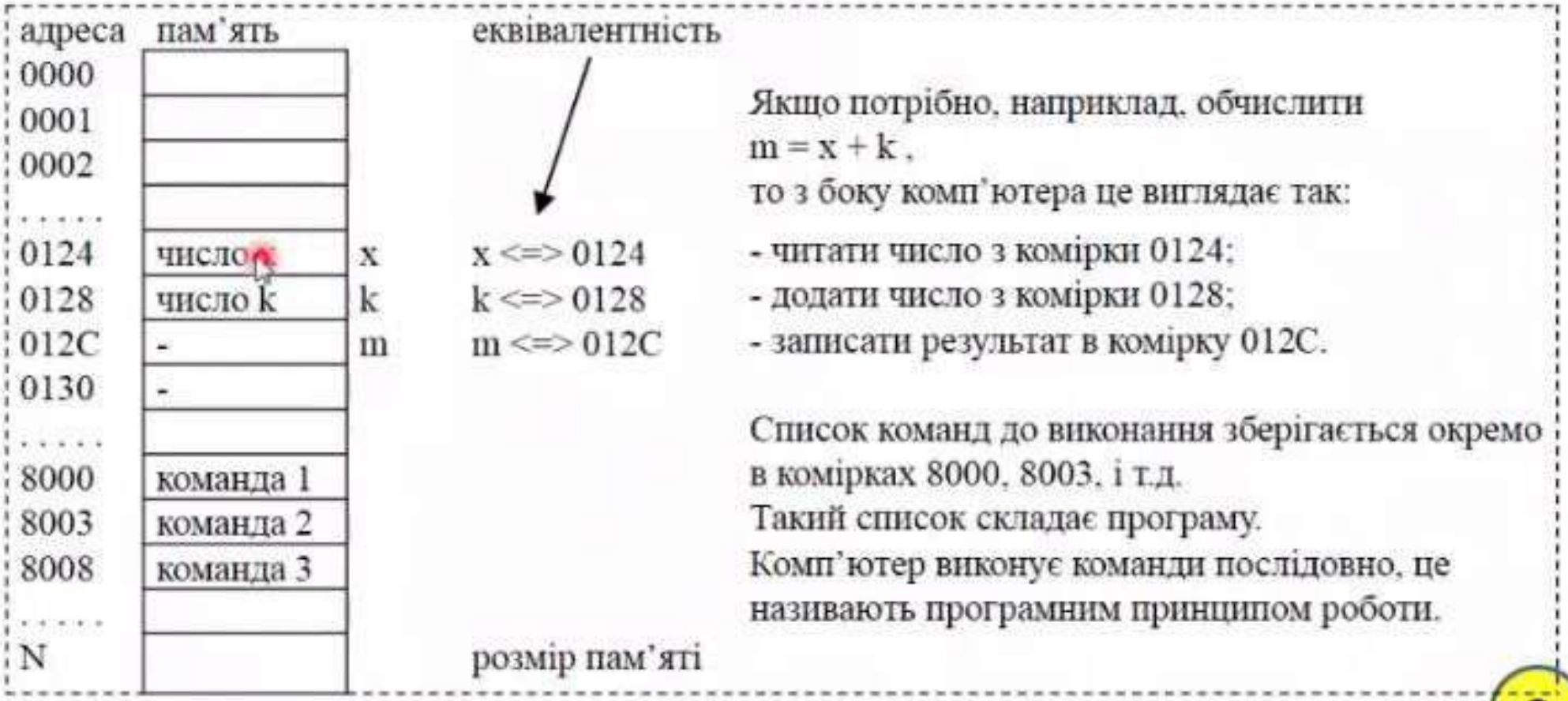
Адресована пам'ять

Пам'ять комп'ютера використовують для двох цілей:

1) зберігання даних на час їх опрацювання програмою;

2) зберігання самої програми як послідовності елементарних операцій процесора.

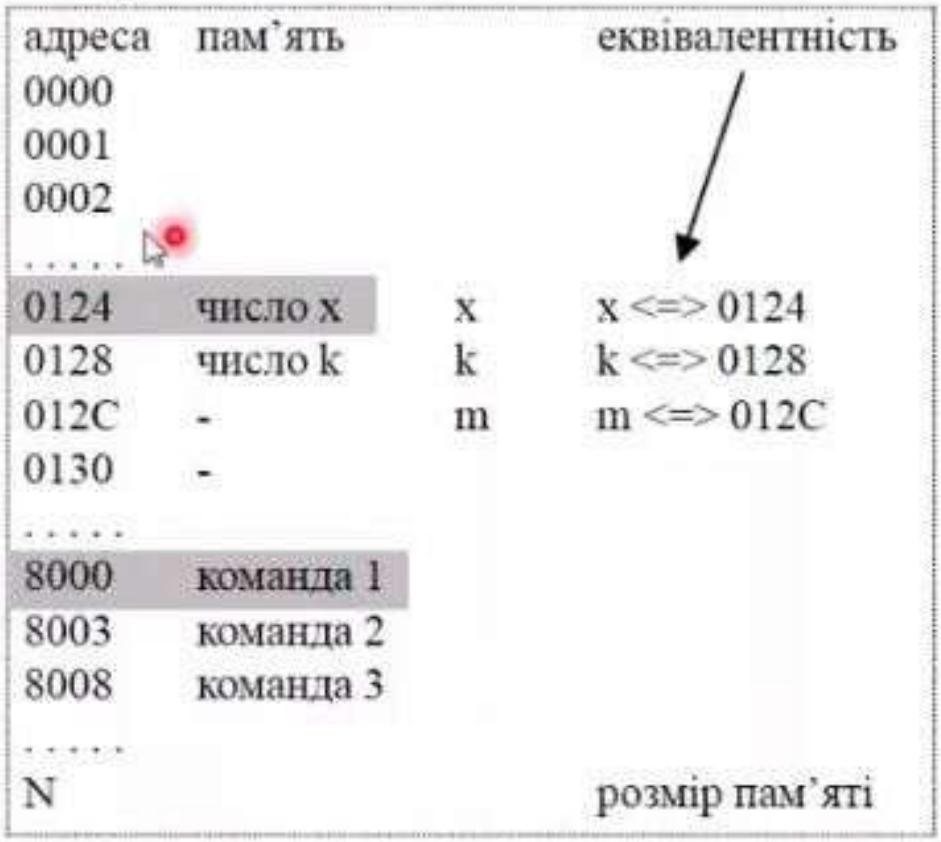
Пам'ять складається з окремих комірок і є адресованою, тобто кожна комірка має свою власну адресу – номер:



З боку комп'ютера адреса комірки – це її номер. В програмуванні адреси комірок позначають ідентифікаторами. Переведення ідентифікаторів в номери комірок автоматично виконує компілятор алгоритмічної мови.

Фізична пам'ять адресується з точністю до окремого байта.

Елементи даних і окремі команди мають різний розмір – від одного байта і більше, для їх розташування в пам'яті завжди виділяють суміжні групи байтів.

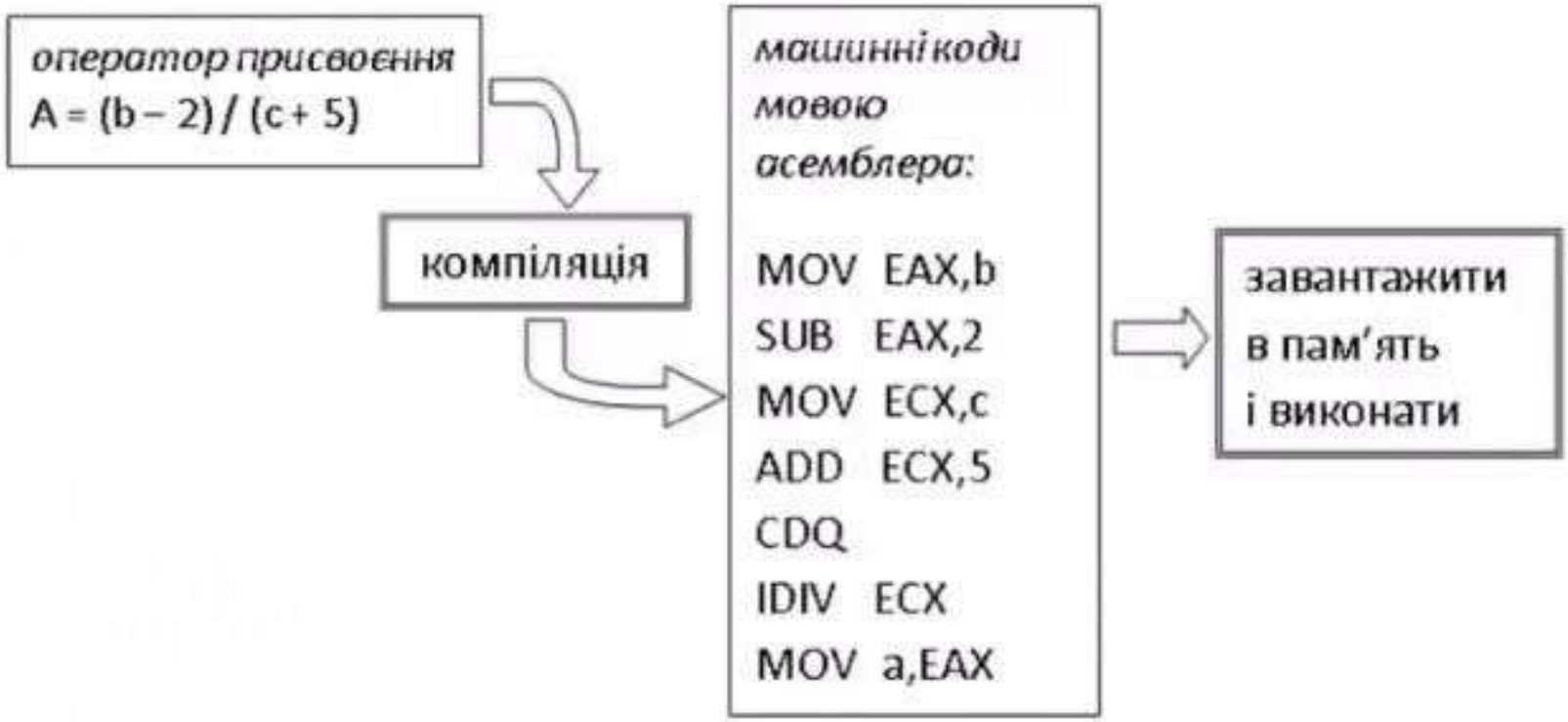


Проте потрібно пам'ятати, що в мові C++ все зберігається у фізичній пам'яті на час виконання програми.

Компіляція і виконання

Після того, як програма складена мовою C++, потрібно організувати її компіляцію і виконання. Компіляція полягає в автоматичному переведенні тексту програми в машинні команди – послідовність елементарних операцій, які вміє виконувати процесор. Програма компілюється повністю ціла, тому під час компіляції є можливість перевірити всі необхідні зв'язки окремих операторів між собою. Після компіляції отримаємо еквівалентну програму в машинних кодах, яку процесор безпосередньо виконує.

Наприклад, для деякого оператора присвоєння процедура отримання результату виглядає так:



Змінні величини і типи даних

Змінні величини позначають ідентифікаторами. Змінна величина є іменем комірки пам'яті, тобто адресою. Ідентифікатори можна записувати одним або більше символами. Перший символ має бути буквою або знаком підкреслення, а наступні можуть бути буквами, цифрами, знаками підкреслення. Малі і великі букви вважаються різними. Приклади ідентифікаторів:

```
data1, data2, data3, x, test, TEST, Test, plus_minus
```

В мові C++ дуже важливе значення має поняття типу даних. Тип даних – це неявна форма визначення розміру пам'яті для зберігання окремих елементів і списку допустимих операцій над такими даними. Тип всіх даних, використаних в програмі, є обов'язковим до визначення – від цього залежить вибір машинних команд для опрацювання, а заодно й контроль коректності програми.

Мова C++ визначає такі елементарні типи даних:

```
char, int, float, double, bool, void
```

Крім того, можна записувати модифікатори для точнішого визначення:

```
signed, unsigned, long, short
```

[Зміст і орієнтовні межі допустимих значень]

Зауваження щодо типу `bool`. Змінна логічного типу може приймати два значення: `true`, `false`. `true` та `false` є стандартними літералами, при цьому `true` трактують як число 1, а `false` трактують як число 0.

Правило: всі змінні мають бути оголошені до найпершого використання

Приклад оголошення змінних:

```
#include <iostream>
using  namespace  std;
int  main()
{
  int  alfa, beta;  // спочатку оголошення
  short  k,m,p;
  float  sum, ave, cnt, res;
  bool  isrd = true;
  //. . . . . .
  alfa = 10;  beta = 3 * alfa;  // пізніше операції
  cin >> cnt >> res ;
  //. . . . . .
  return 0;
}
```

Загальний формат оголошення змінних:

```
тип  список_змінних ;
```

Звернемо увагу, що тут і у всіх подальших випадках тип будь-якого елемента програми завжди записують попереду операцій з таким елементом.

Основні операції та їх особливості

1) Арифметичні операції

```
-     +     *     /     % (остача ділення)
-- (декремент)      ++ (інкремент)
```

`+, -, *` виконують звичайно

/ задає операцію ділення, причому, коли обидва операнди є цілими числами, то результатом буде ціла частина частки: 17 / 3 дорівнює 5

% задає ділення по модулю, тобто знаходження остачі від ділення першого операнда на другий: 17 % 3 дорівнює 2. Обидва операнди цілочисельні.

Операція *декременту* зменшує операнд на 1, а *інкременту* – збільшує на 1.

Форми операцій декременту та інкременту: префіксна, постфіксна.

Префіксну **++x** виконують до опрацювання операнда, а постфіксну **x++** після:

```
int  x = 5 ;
int  y = ++x ;    // y=6, x=6
```

Однак, при записі в постфіксній формі результат буде іншим:

```
int  x = 5 ;
int  y = x++ ;    // y=5, x=6
```

В загальному випадку треба бути уважним до вибору форми таких унарних операцій.

2) Операції порівняння

`<` `<=` `==` `>=` `>` `!=`

Операції порівняння мають менший пріоритет, ніж арифметичні:
`x + 3 > y - 2` еквівалентно `(x + 3) > (y - 2)`

Типова помилка:
`mira == 5` // це є *порівняння*: `true` або `false`
`mira = 5` // це є *присвоєння*: значення виразу дорівнює 5

3) Логічні операції

Логічне АБО : ||

Логічне І : &&

Логічне НЕ : !

```
ch=='y'  ||  ch=='Y'
```
дужок не потрібно

```
i < ArrSize  &&  temp >= 0
!(x>5)
```

Варто занотувати пріоритети операцій порівняння і логічних.

В порядку від вищого пріоритету до нижчого список такий:

```
!
>     >=     <     <=
==     !=
&&
||
```

Порядок обчислень можна змінити за допомогою дужок.

Зауваження до логічних операцій:

операції логічного І та логічного АБО мають нижчий пріоритет від порівняння:

`x > 5 && x < 10` означає `( x > 5 ) && ( x < 10 )`

операція логічного НЕ має вищий пріоритет від порівняння та арифметичної:

`!x > 5` означає `( !x ) > 5,` отже треба писати `!(x>5)`

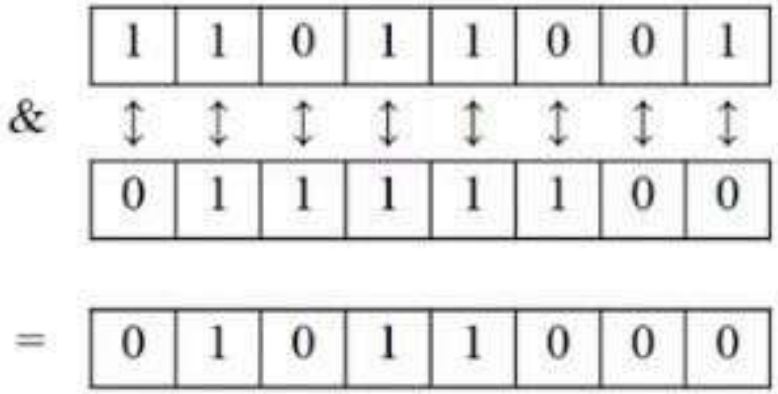
4) Побітові операції

На відміну від інших мов, C++ має повний перелік побітових операцій. Побітові операції застосовують до окремих бітів операндів типів `char` або `int`, як вони зображені в пам'яті. Зокрема можна перевіряти окремі біти, надавати бітам значення (0 або 1), зсувати біти, з яких складаються байти і машинні слова.

Список побітових операцій:

Операція	Дія
&	І
|	АБО
^	виключне АБО
~	НЕ
>>	зсув направо
<<	зсув наліво

Побітові операції І, АБО, НЕ мають такі самі таблиці істинності, як їх логічні аналоги, але всі порівняння виконують порозрядно до окремих бітів. Наприклад:

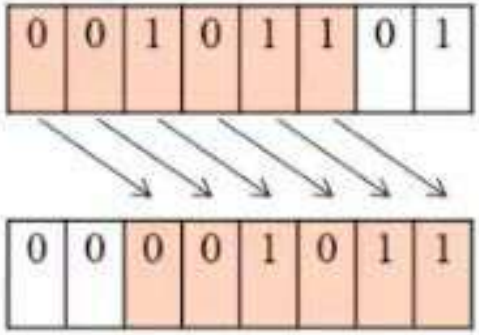


Підкреслимо, що результатом операцій порівняння і логічних операцій завжди є істинне або хибне значення, в той час аналогічні побітові операції можуть утворювати будь-які цілі числа.

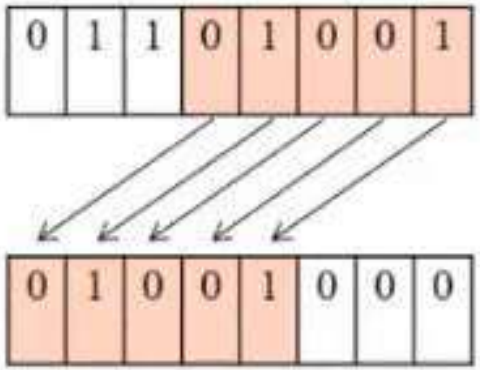
Виключне АБО дає істинне значення тоді і лише тоді, коли один операнд істинний, а другий хибний. Якщо обидва операнди одночасно істинні або хибні, то результат хибний.

Операції зсуву бітів ">>" і "<<" змішують всі біти двійкового зображення операнда направо або наліво на задану кількість розрядів. Загальний вигляд побітового зсуву:

1) `значення >> кільк. розрядів`



2) `значення << кільк. розрядів`



Висновки

1) Мова C++ є мовою компільованого типу, коли спершу цілу програму перекладають на машинну мову, після чого виконують процесором (на відміну від Python – мови інтерпретованого типу – виконує інтерпретатор).

2) Мова C++ в багатьох аспектах пов'язана з архітектурою комп'ютера, зокрема, для визначення типів даних і допустимих операцій.

3) Мова C++ вимагає оголошення всіх необхідних величин та їх типів, що є потрібним для компіляції і вибору машинних команд.

4) Мова визначає перелік типів даних, які можна застосувати в програмах.

5) Поняття типу застосовують до змінних величин, але не до зображень об'єктів в пам'яті. Дані, зображені в адресованій пам'яті, самі по собі типу не мають. (На відміну від Python, де тип є характеристикою об'єкта, але не величини).