

④ Optimization & We can optimize the blocks with the help of given types -

i) Common Sub-Expression Elimination & In this, we don't need to find it over and over. Instead we can find it once and kept in store from where it referenced when encountered again. For example

$a = x + y$

$b = a - d$

$c = b + c$

In above example we are computing $x + y$ two times. Instead we can directly assign a to c

$a = x + y$

$b = a - d$

$c = a$

ii) Dead-Code Elimination & Sometimes it happens that a program contains dead code. It is the block of code which are not participating in the execution.

Suppose we have written $z = 4 * d$ but later on we are not using that part of code. So in order to optimize our code we can simply remove that part of code.

iii) Sometimes it happens that we use a temporary variable to assign a simple multiplication and then we will use that variable for assignment. Instead we can directly assign the multiplication to the main variable. For example.

$temp = 5 * z$

$a = b + temp$

Instead of that we can use

$a = b + (5 * z)$, It will help to optimize the code.

ii) Sometimes we are using the loops more than one time in order to perform same action. Then instead of using more loops we can reduced to one and perform the same action. For example

for ($i=0; i < 5; i++$)

$a = z * i$

for ($i=0; i < 5; i++$)

$b = z * i + 5$

In order to optimize we can use one loop instead of two loops

for ($i=0; i < 5; i++$)

$a = z * i$

$b = z * i + 5$

i) Some times we are computing some statement n number of times so in order to optimize we can use that one time.

For example

for ($i=0; i < 50; i++$)

$z = 2 * c$

if ($z > i$)

$d = i + 5$

Use this

$z = 2 * c$

for ($i=0; i < 50; i++$)

if ($z > i$)

$d = i + 5$