UNIT-V

Machine-Independent Optimization

Machine independent optimization attempts to improve the intermediate code to get a better target code. The part of the code which is transformed here does not involve any absolute memory location or any CPU registers.

Code Optimization can perform in the following different ways:

(1) Compile Time Evaluation:

(a) z = 5*(45.0/5.0)*r

Perform 5*(45.0/5.0)*r at compile time.

(b) x = 5.7

y = x/3.6

Evaluate x/3.6 as 5.7/3.6 at compile time.

(2) Variable Propagation:

Before Optimization the code is:

```
c = a * b
x = a
till
d = x * b + 4
```

After Optimization the code is:

```
c = a * b
x = a
till
    d = a * b + 4
```

(3) Dead code elimination:

Before elimination the code is:

```
c = a * b
x = b
till
d = a * b + 4
```

After elimination the code is:

```
c = a * b
till
d = a * b + 4
```

(4) Code Motion:

It reduces the evaluation frequency of expression.

It brings loop invariant statements out of the loop.

```
do
{
   item = 10;
   valuevalue = value + item;
} while(value<100);
```

//This code can be further optimized as

```
item = 10;
do
{
   valuevalue = value + item;
} while(value<100);
```

(5) Induction Variable and Strength Reduction:

Strength reduction is used to replace the high strength operator by the low strength.

An induction variable is used in loop for the following kind of assignment like i = i + constant.

Before reduction the code is:

After Reduction the code is:

```
i = 1
t = 4
{
   while( t<40)
  y = t;
  t = t + 4;
}
```

Data Flow Analysis

To efficiently optimize the code compiler collects all the information about the program and distribute this information to each block of the flow graph. This process is known as data-flow graph analysis.

Certain optimization can only be achieved by examining the entire program. It can't be achieve by examining just a portion of the program.

consider the following code:

```
x = a + b;
   x = 6 * 3
```

Some optimization needs more global information. For example, consider the following code:

```
a = 1;
   b = 2;
   c = 3;
   if (....) x = a + 5;
   else x = b + 4;
   c = x + 1;
```