**Code Optimization Methods**

A transformation of program is called " *Local* " if it can performed by looking only at the statements in a *Basic Block*, otherwise, it is called " *Global* " .

**Local Transformations:**

1. Structure-Preserving Transformations:-
	* Common Sub expression Elimination
	* Dead Code Elimination
2. Algebraic Transformations:-This transformations uses to change the set of expressions ,computed by a basic block, with an algebraically equivalent set. The useful ones are those that simplify expressions or replace expensive operations by cheaper one, such as:

*x:=x+0*

*x:=x\*1*

*x:=x/1*

*Eliminated*

**

*x:= y^2  x:=y\*y*

Another class of algebraic transformations is ***Constant Folding*** ,that is, we can evaluate constant expressions at compiler time and replace the constant expressions by their values, for example, the expression 2\*3.14 would be replaced by 6.28.

**Global Transformations:**

**1.** Common Subexpression Elimination

***a=b+c*** ***a=b+c***

******

***c=b+c*** ***c=a***

***d=b+c*** ***d=b+c***

1. Dead Code Elimination**:** Variable is *dead* if never used

***x=y+1***

******

***y=1*** ***y=1***

***x=2\*z*** ***x=2\*z***

***With My Best Wishes*** *54* ***Esam***

***Compilers***

***Principle ,Techniques, and Tools***

***Anbar University – Computer College***

1. Copy Propagation

|  |  |  |
| --- | --- | --- |
| **Origin** | **Copy Propagation** | **Dead Code** |
| ***x=t3*** |  | ***x=t3*** |  |  |
| ***a[t2]=t5*** | ***a[t2]=t5*** | ***a[t2]=t5*** |
| ***a[4]=x*** | ***a[4]=t3*** | ***a[4]=t3*** |
| ***Goto B2*** | ***Goto B2*** | ***Goto B2*** |

1. Constant Propagation

|  |  |  |
| --- | --- | --- |
| **Origin** | **Copy Propagation** | **Dead Code** |
| ***x=3*** |  | ***x=3*** |  |  |
| ***a[t2]=t5*** | ***a[t2]=t5*** | ***a[t2]=t5*** |
| ***a[4]=x*** | ***a[4]=3*** | ***a[4]=3*** |
| ***Goto B2*** | ***Goto B2*** | ***Goto B2*** |

1. Loop Optimization
	* **Code Motion:** An important modification thatdecreases the amount of code in a loop is *Code* *Motion.* If result of expression does not changeduring loop( *Invariant Computation* ),can hoist its computation out of the loop.

***For(i=0;i<n;i++)***

******

***A[i]=a[i]+( x\*x )/( y\*y );***

***c=( x\*x )/( y\*y );***

***For(i=0;i<n;i++)***

***A[i]=a[i]+c;***

***With My Best Wishes*** *55* ***Esam***

***Compilers***

***Principle ,Techniques, and Tools***

***Anbar University – Computer College***

* **Strength Reduction:** Replaces expensive operat--ions (Multiplies, Divides)by cheap ones ( Adds, Subs )**.**For example, suppose the following expression**:**

***For(i=1;i<n;i++){v=4\*i;s=s+v;}*** ***i*** *is induction variable*

Then:

***v=0;***

***For(i=1;i<n;i++){ v=v+4; s=s+v; }***

***Induction Variable*** : is a variable whose value changes by aconstant amount on each loop iteration.

***With My Best Wishes*** *56* ***Esam***