TEKNIK OPTIMASI KODE PROGRAM

Tujuan dan Pembagian Optimasi

- Tujuan: menghasilkan kode program yang berukuran lebih kecil dan lebih cepat pada saat dieksekusi.
- Berdasarkan sifat ketergantungan dengan mesin maka optimasi dibagi menjadi dua:
  a. Machine Dependent Optimizer
  b. Machine Independent Optimizer

Machine Dependent dan Independent Optimizer

- Machine Dependent Optimizer
  Kode program dioptimasi agar lebih efisien saat dieksekusi untuk mesin tertentu. Proses optimasi memerlukan informasi feature yang ada pada mesin tujuan.
- Machine Independent Optimizer
  Kode program dioptimasi agar lebih efisien saat dieksekusi untuk lebih dari satu mesin tertentu. Machine Independent Optimizer terdiri dari dua jenis: optimasi lokal dan optimasi global.

Pembagian Optimasi

- Berdasarkan pengembangan program
  a. Penulisan baris program (*source code*)
  b. Waktu kompilasi oleh kompiler
- Berdasarkan letak atau posisi baris program (*source code*) di dalam file program, maka optimasi dibagi menjadi dua:
  a. Optimasi lokal
  b. Optimasi global

Optimasi lokal (1)

- Optimasi yang dilakukan pada suatu blok source code, dengan cara:
  1. Folding
  2. Redundant-Subexpression Elimination
  3. Optimasi dalam sebuah iterasi
  4. Strength Reduction

Optimasi lokal (2)
Folding

- Mengganti konstanta atau ekspresi yang bisa dievaluasi pada
  saat *compile time* dengan nilai komputasinya.
- Contoh:

A := 2 + 3 + B
Dapat diganti menjadi
A := 5 + B
dimana **5**, menggantikan ekspresi
2 + 3

Optimasi lokal (3)
Redundant-Subexpression Elimination

- Sebuah ekspresi yang sudah dihitung dapat digunakan kembali,
  sehingga tidak perlu menghitungnya kembali
- Contoh:

```
A := B + C
X := Y + B + C
Dapat dioptimasi menjadi
A := B + C
X := Y + A
```

Optimasi lokal (3)
Optimasi dalam sebuah iterasi

Terdiri atas dua:

- *Loop unrolling*
- *Frequency reduction*

Optimasi lokal (4)
Optimasi dalam sebuah iterasi

Loop unrolling, mengganti suatu *loop* (iterasi) dengan menuliskannya menjadi beberapa statement.

Dengan pertimbangan, sebuah iterasi akan mengerjakan serangkaian perintah sebagai berikut:

- ✓ Pemberian nilai awal (inisialisasi) untuk variabel iterasi, dilakukan sekali pada saat pertama kali mengerjakan iterasi.
- ✓ Mengetes apakah variabel iterasi telah mencapai kondisi terminasi, yaitu kondisi benar yang menyebaban iterasi selesai.
- ✓ Penyesuaian (*adjustment*) terhadap variabel iterasi dengan cara menambah atau mengurangi dengan jumlah tertentu terhadap nilai variabel tersebut.
- ✓ Mengerjakan perintah atau beberapa perintah yang ada di dalam blok iterasi.

Optimasi lokal (5)
Optimasi dalam sebuah iterasi

- Contoh

```
FOR I := 1 to 2 DO
A [I] := 0 ;
```

Dioptimasi menjadi:

```
A [1] := 0 ;
A [2] := 0 ;
```

Optimasi lokal (6)
Optimasi dalam sebuah iterasi

- *Frequency reduction*, memindahkan statement ke tempat yang lebih jarang dieksekusi, dengan pertimbangan untuk mengeksekusi sebuah statement memerlukan waktu.
- Contoh:

```
FOR I := 1 TO 10 DO
BEGIN
X := 5 ;
A := A + I ;
END ;
```

menjadi

```
X := 5 ;
FOR I := 1 TO 10 DO
BEGIN
A := A + I ;
END ;
```

Optimasi lokal (7)
Strength Reduction

- Mengganti suatu operasi dengan jenis operasi lain yang lebih cepat dieksekusi.
  Contoh:
- Operasi perkalian memerlukan waktu eksekusi lebih lama daripada operasi penjumlahan. Oleh karena itu dapat dilakukan penghematan waktu dengan mengganti operasi perkalian tertentu menjadi penjumlahan.
- A := A + 1
  dapat diganti dengan **INC (A)** ;