Teknik Optimasi

Teknik Kompilasi

Dosen: Utami Dewi W.,S.Kom

Dependensi Optimasi

- Tahapan optimasi kode yang bertujuan untuk menghasilkan kode program yang berukuran lebih kecil dan lebih cepat eksekusinya.
- Berdasarkan ketergantungannya pada mesin, optimasi dibagi menjadi :
  - *Machine Dependent Optimizer*

    Kode dioptimasi sehingga lebih efisien pada mesin tertentu.
  - *Machine Independent Optimizer*

    Strategi optimasi yang bisa diaplikasikan tanpa tergantung pada mesin tujuan tempat kode yang dihasilkan akan dieksekusi nantinya. Optimasi yang dilakukan meliputi optimasi lokal dan optimasi global.

Optimasi Lokal (1)

- Optimasi Lokal adalah optimasi yang dilakukan hanya pada suatu blok dari *source code*.
- Caranya sebagai berikut :
  1. Folding
  2. *Redudant-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.
- Misalnya :

  A := 2 + 3 + B

  Bisa diganti menjadi

  A := 5 + B

  dimana **5**, menggantikan ekspresi **2 + 3**

Optimasi Lokal (3)
Redundant-Subexpression Elimination

- Menggunakan hasil komputasi terdahulu daripada melakukan komputasi ulang
- Misalnya terdapat urutan instruksi :

  A := B + C

  X := Y + B + C

  Kemunculan **B + C** yang bersifat redundan, bisa memanfaatkan hasil komputasi sebelumnya selama tidak ada perubahan nilai variabel.

Optimasi Lokal (3)
Optimasi dalam sebuah iterasi

- Terdiri dari 2 macam
  1. *Loop Unrolling*
  2. *Frequency Reduction*

Optimasi Lokal (4)
Optimasi dalam sebuah iterasi

- ***Loop Unrolling***

✓ Menggantikan suatu *loop* dengan menulis statement dalam *loop* beberapa kali

✓ Hal ini karena sebuah iterasi pada implementasi level rendah akan memerlukan operasi :

→ Inisialisasi/pemberian nilai awal pada variabel loop. Dilakukan sekali pada saat permulaan eksekusi loop.

→ Pengetesan, apakah variabel loop telah mencapai kondisi terminasi.

→ Adjustment, yaitu penambahan atau pengurangan nilai pada variabel loop dengan jumlah tertentu.

→ Operasi yang terjadi pada tubuh perulangan (*loop body*)

Optimasi Lokal (5)
Optimasi dalam sebuah iterasi

- ***Loop Unrolling***
- ✓ Contoh instruksi :

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

Terdapat instruksi untuk inisialisasi I menjadi 1. Serta operasi penambahan nilai/*increment* 1 dan pengecekan nilai variabel I pada setiap perulangan. Sehingga untuk perulangan saja memerlukan 5 instruksi, ditambah dengan instruksi assignment pada tubuh perulangan menjadi 7 instruksi.

Dioptimasi menjadi

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

- Sehingga haya memerlukan 2 instruksi assignment saja

Optimasi Lokal (6)
Optimasi dalam sebuah iterasi

- ***Frequency Reduction***
- ✓ Pemindahan statement ke tempat yang lebih jarang dieksekusi.
- ✓ Contoh :

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

- Variabel X dapat dikeluarkan dari iterasi 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.
- Misalnya, pada beberapa computer operasi perkalian memerlukan waktu lebih banyak dari pada operasi penjumlahan, oleh karena itu bias dilakukan pengehematan waktu dengan mengganti operasi perkalian tertentu dengan penjumlahan.
- Contoh lainnya :

  A := A + 1

  dapat diganti dengan **INC (A)** ;

Optimasi Global (1)

- Optimasi global biasanya dilakukan dengan analisis flow, yaitu suatu graph berarah yang menunjukkan jalur yang mungkin selama eksekusi program.
- Ada 2 kegunaan optimasi global, yaitu
  1. Bagi programmer
  2. Bagi kompilator itu sendiri

Optimasi Global (2)
Bagi Programmer menginformasikan

- ***Unreachable / dead code :***
  - ✓ kode yang tidak akan pernah dieksekusi.
  - ✓ Misalnya :

```
X := 5
IF X = 0 THEN
    A := A + 1
```

Instruksi A := A + 1 tidak akan pernah dieksekusi

Optimasi Global (3)
Bagi Programmer menginformasikan

- ***Unused parameter* pada prosedur**
  - ✓ Parameter yang tidak pernah digunakan di dalam prosedur
  - ✓ Misalnya :

```
Procedure Jumlah (a,b,c : integer)
var x : integer
begin
        x := a + b
end ;
```

Parameter c tidak pernah digunakan di dalam prosedur, sehingga seharusnya tidak perlu diikutsertakan.

Optimasi Global (4)
Bagi Programmer menginformasikan

- ***Unused variabel***
  - ✓ Variabel yang tidak pernah dipakai di dalam program
  - ✓ Misalnya :

```
Program pendek;
var a, b : integer;
begin
        a := 5;
end ;
```

Variabel b tidak pernah dipergunakan di dalam program, sehingga bias dihilangkan.

Optimasi Global (5)
Bagi Programmer menginformasikan

- **Variabel yang dipakai tanpa nilai awal**
  - ✓ Misalnya :

```
Program awal;
var a, b : integer;
begin
        a := 5;
       a := a + b;
 end ;
```

Variabel b digunakan tanpa memiliki nilai awal/belum di-*assign*

Optimasi Global (6)
Bagi Kompilator

- Meningkatkan efisiensi eksekusi program
- Menghilangkan useless code/ kode yang tidak terpakai.