BINUS UNIVERSITY ONLINE

People
Innovation
Excellence

Tugas Personal ke-1

Week 2

Soal.

1. Jelaskan peran compiler dalam proses penerjemahan bahasa pemrograman. Apa perbedaan hasil akhir dari proses kompilasi dibandingkan dengan proses interpretasi? (LO 1 : 30%)

   Compiler adalah program yang berfungsi untuk menerjemahkan kode sumber atau source code yang ditulis dalam bahasa pemrograman tingkat tinggi menjadi kode mesin atau program target yang dapat dijalankan oleh komputer. Dalam proses ini, compiler melakukan beberapa tahap analisis dan sintesis terhadap program sumber.

   Peran compiler dalam penerjemahan bahasa penerjemahan :

   - Lexical Analysis
     Pada tahap ini, compiler membaca karakter dari program sumber dan mengelompokkannya menjadi unit-unit yang disebut token. Token dapat berupa identifier, keyword, operator, maupun konstanta. Informasi mengenai token tersebut kemudian disimpan dalam symbol table.

   - Syntax Analysis
     Pada tahap ini, compiler memeriksa apakah urutan token yang dihasilkan oleh analisis leksikal sesuai dengan aturan tata bahasa (grammar) dari bahasa pemrograman. Proses ini biasanya menghasilkan parse tree atau syntax tree yang menggambarkan struktur program.

   - Semantic Analysis
     Tahap ini memastikan bahwa program memiliki makna yang benar secara semantik, misalnya kesesuaian tipe data, penggunaan variabel yang telah dideklarasikan, dan aturan lain yang berkaitan dengan makna program.

   - Intermediate Code Generation
     Setelah analisis selesai dilakukan, compiler menghasilkan representasi perantara dari program sumber. Representasi ini dapat berupa syntax tree atau three-address code yang memudahkan proses optimasi dan penerjemahan ke kode target.

   - Optimization
     Compiler melakukan optimasi agar program lebih efisien dalam penggunaan memori dan waktu eksekusi.

- Code Generation
  Pada tahap akhir, compiler menghasilkan kode target atau kode mesin yang dapat dijalankan oleh komputer.

Perbedaan Kompilasi dan Interpretasi

Aspek	Kompilasi	Interpretasi
Cara kerja	Seluruh program diterjemahkan sekaligus	Program diterjemahkan dan dijalankan baris per baris
Hasil akhir	Menghasilkan file executable	Tidak menghasilkan file executable
Kecepatan eksekusi	Lebih cepat karena sudah menjadi kode mesin	Lebih lambat karena diterjemahkan saat dijalankan
Contoh bahasa	C, C++, Go	Python, JavaScript

2. Bahasa pemrograman C tidak memiliki tipe boolean. Jelaskan bagaimana *C compiler* dapat menerjemahkan *if-statement* menjadi *three-address code*! (LO 1 : 30%)

Bahasa pemrograman C tidak memiliki tipe data boolean. Dalam C, kondisi logika direpresentasikan menggunakan nilai integer, dimana:

- Nilai 0 dianggap false
- Nilai selain 0 dianggap true

Ketika compiler menerjemahkan sebuah if-statement, ekspresi kondisi terlebih dahulu dihitung dan hasilnya disimpan dalam variabel sementara. Nilai tersebut kemudian digunakan untuk menentukan alur percabangan program.

Contoh code c

```
int a, b, c, max;

if (a + b > c) {
  max = a + b;
}
```

Pada kode tersebut, kondisi if memeriksa apakah hasil penjumlahan a + b lebih besar dari c. Karena bahasa C tidak memiliki tipe boolean khusus, maka hasil dari ekspresi (a + b > c) akan berupa nilai integer, dimana:
0 = false
selain 0 = true

Three-Address Code yang Dihasilkan Compiler
Compiler akan menerjemahkan ekspresi tersebut ke bentuk three-address code menggunakan variabel sementara:
t1 = a + b
t2 = t1 > c
if t2 goto L1

goto L2
L1: max = t1
L2:

Penjelasan :

- t1 = a + b
  Compiler menghitung penjumlahan a + b dan menyimpannya pada variabel sementara t1
- t2 = t1 > c
  Dilakukan operasi perbandingan antara t1 dan c.
- if t2 goto L1
  Jika hasil perbandingan bernilai true, maka eksekusi program berpindah ke label L1.
- goto L2
  Jika kondisi bernilai false, program langsung menuju L2.
- L1: max = t1
  Pernyataan yang dijalankan jika kondisi pada if bernilai benar.
- L2:
  Menandai akhir dari percabangan if.

3. Buatlah DAG untuk ekspresi berikut: (LO 2 : 40%)

(a + b) * (c + d) + (a + b) * (e - f) - (c + d)

Dalam DAG sub-ekspresi yang berulang harus dihindari maka sub ekspresi yang muncul lebih dari satu kali hanya dihitung satu kali
sub-ekspresi yang berulang
a + b
c + d

Representasi Three-Address Code
t1 = a + b
t2 = c + d
t3 = e - f
t4 = t1 * t2
t5 = t1 * t3
t6 = t4 + t5
t7 = t6 - t2

Struktur DAG