|  |  |  |
| --- | --- | --- |
| unikom | **TEORI BAHASA DAN OTOMATA**  **Program Studi Teknik Informatika** | **IF** |
| **PENDAHULUAN**  **TEKNIK KOMPILASI** |

1. **Bahasa Pemrograman**

* Menjembatani antara pemikiran manusia yang sering tidak terstruktur dengan kepastian yang diperlukan oleh Komputer untuk melakukan eksekusi.
* Suatu solusi untuk suatu masalah akan menjadi lebih mudah bila bahasa pemrograman yang dipergunakan lebih dekat dengan permasalahan tersebut.
* Bahasa pemrograman berdasarkan tingkat ketergantungannya dengan mesin bias meliputi :
  1. Bahasa Mesin

Merupakan bentuk terendah dari bahasa Komputer. Setiap instruksi dalam program direpresentasi dengan kode numeric, yang secara fisik berupa deretan angka 0&1.

* 1. Bahasa Assembly

Merupakan bentuk simbolik dari bahasa mesin.

* 1. Bahasa Tingkat Tinggi
     + Disebut tingkat tinggi karena lebih dekat dengan manusia
     + Memberikan fasilitas yang lebih banyak, kontrol program yang terstruktur, block dan prosedur.
     + Contoh : Pascal dab BASIC
  2. Bahasa yang problem oriented
* Memungkinkan penyelesaian untuk suatu masalah atau aplikasi yang spesifik.
* Contoh : SQL
* Keuntungan **bahasa tingkat tinggi** dibandingkan **bahasa tingkat rendah** sebagai berikut :

1. Kemudahan untuk dipelajari, tidak membutuhkan latar belakang pengetahuan perangkat keras.
2. Lebih mendekati permasalahan yang akan diselesaikan.
3. Pemrogram tidak perlu mengetahui bagaimana representasi data kedalam bentuk internal memory.
4. Memberikan banyak pilihan struktur kontrol seperti : kondisional, looping, dan struktur blok
5. Program lebih mudah di-debug
6. Kemampuan struktur data yang lebih, sehingga memfasilitasi pengekspresian suatu solusi dari masalah tertentu.
7. karena ketersediaan *features* seperti prosedur, bahasa tingkat tinggi memungkinkan suatu deskripsi modular dan hirarkis dalam pemrograman.
8. kompatibilitas dan dokumentasi yang lebih baik dalam pengembangan program.
9. tidak bergantung pada mesin sehingga memiliki portabilitas tinggi.
10. **Translator**
    * + - Sebuah translator melakukan pengubahan source code/source program (program sumber) ke dalam target code/object code/object program(program object).
        - Source code ditulis dalam bahasa sumber sedang object code bias berupa suatu bahasa pemrograman lain atau bahasa mesin pada suatu computer.
        - Ada beberapa macam translator :
    1. **Assembler**

* Source Code adalah bahasa assembler, object code adalah bahasa mesin
* Contohnya : Turbo Assembler dan macro assembler



* 1. **Kompilator(Compiler)**
* Source code adalah bahasa tingkat tinggi (misal bahasa Pascal)
* Object Code adalah bahasa mesin atau bahasa assembly.
* Source code dan data diproses pada saat yang berbeda.
* Contohnya : Turbo Pascal, proses kompilasi :



* Compile time : saat pengubahan source ke object code
* Run time : Saat eksekusi object
  1. **Interpreter**
* Interpreter tidak membangkitkan object code, hsil translasi hany dalam bentuk internal.
* Contoh interpreter : BASIC, LISP, SMALL TALK, MATLAB.
* Source Code data diproses dalam waktu yang sama.
* Proses interpretasi dapat dilihat pada gambar :



1. **Model Kompilator**

* Pengembangan kompilator untuk sebuah bahasa merupakan pekerjaan yng kompleks.
* Sebuah kompilator pada umumnya memiliki 2 tugas pokok :
  1. Fungsi Analisis (Front End)

Tugasnya melakukan dekomposisi program sumber menjadi bagian-bagian dasarnya.

* 1. Fungsi Sintesis (Back End)

Tugasnya melakukan pembangkitan dan optimasi program objek.

* Model sebuah kompilator :



* Keterangan gambar :
  + Scanner(analisis leksikal) : memecah program sumber menjadi besaran leksik/token
  + Parser(analisis Sintaks) : memeriksa kebenaran dan urutan kemunculan token.
  + Analisis Semantik : melakukan analisa semantik, biasanya dalam realisasi akan digabungkan dengan *intermediate code generator* (bagian yang berfungsi membangkitkan kode antara)
  + Kode generator : membangkitkan kode objek.
  + Kode Optimasi : memperkecil hasil dan mempercepat proses
  + Tabel Simbol : menyimpan semua informasi yang berhubungan dengan proses kompilasi.

Tahap kompilasi adalah tahap mengerti bahasa.

Interaksi antara scanner dengan Parser terdapat 2 kemungkinan :

1. Scanner menghasilkan suatu token untuk diproses oleh parser. Parser akan memanggil scanner bila token berikutnya dibutuhkan.
2. Scanner menghasilkan semua token yang berhubungan dengan source program sebelum meneruskan ke parser. Pada kasus ini scanner telah memeriksa keseluruhan souce program, disebut *separate pass.*