TEORI KOMPUTASI

PRIO HANDOKO, S.KOM., M.T.I.

Program Studi Informatika Universitas Pembangunan Jaya

BAB 1: PENGANTAR TEORI KOMPUTASI

Capaian Pembelajaran

- Mahasiswa memahami konsep Teori Komputasi (Teknik Komputasi dan Teknik Kompilasi).
- Mahasiswa jenis-jenis *translator* (penterjemah).
- Mahasiswa memahami bahasa pemrograman dalam pembuatan *compiler.*
- Mahasiswa memahami pengukuran mutu kompiltor.
- Mahasiswa memahami proses dan tahapan proses kompilasi.

Bab 1: Pendahuluan

Agenda.

- Teori Komputasi (Teori Komputasi dan Teknik Kompilasi)
- Translator
- Pembuatan Compiler
- Mengukur Mutu Compiler
- Proses dan Tahapan Kompilasi

Teori Komputasi

- **Teori Komputasi** merupakan gabungan dari teknik komputasi dan teknik kompilasi
- **Teknik**: metode atau cara
- **Komputasi**: proses/pengolahan data yang dilakukan oleh komputer.
- **Teknik Komputasi**:

  "suatu metode/cara dalam melakukan proses/pengolahan data yang dilakukan oleh komputer."

Teori Komputasi dan Teknik Kompilasi

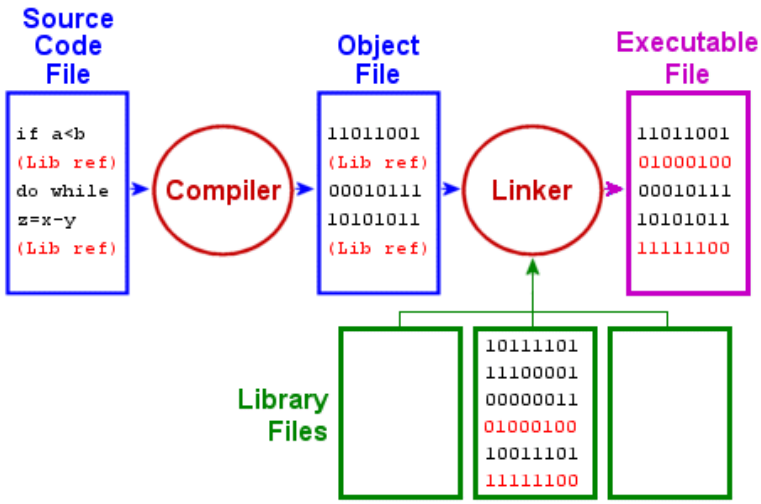
- ***to compile...***

  "to translate a program written in high-level programming language into machine programming language (low-level programming language)."

- **Teknik Kompilasi**:

  "suatu metode/cara dalam menggabungkan serta menterjemahkan program sumber (source program/source code) menjadi bentuk lain (machine language).:

Teori Komputasi dan Teknik Kompilasi



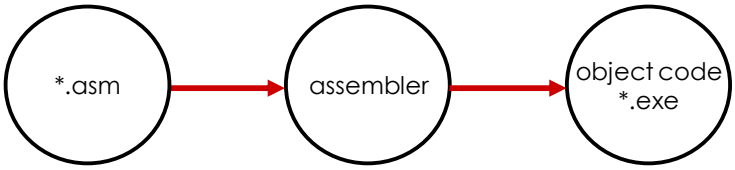
Translator

- **Translator**:
  "suatu program yang mengambil input sebuah program yang ditulis pada satu bahasa pemrograman (source language) ke dalam bahasa lain (the object on target language)."

- *Mengapa dalam proses komputasi dibutuhkan translator?*
  1. bahasa mesin adalah bentuk bahasa yang terendah pada komputer;
  2. bahasa mesin tidak lebih dari urutan bit **0** dan **1**;
  3. instruksi dalam bahasa mesin dapat dibentuk menjadi sebuah *microcode*; dan
  4. user tidak memahami bahasa mesin

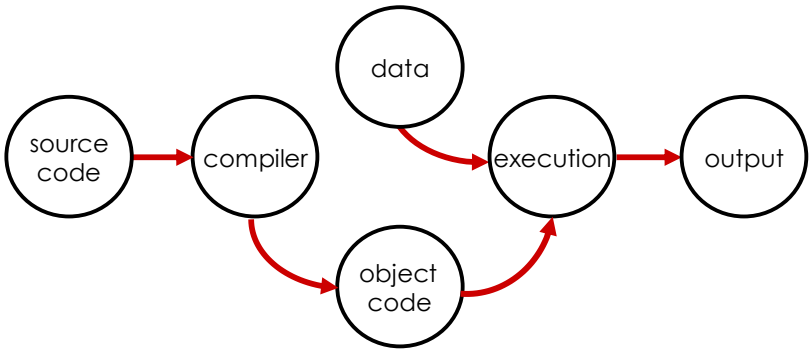
Jenis Translator

1. **Assembler**, *source code* bahasa *assembly*, *object code* adalah bahasa mesin.



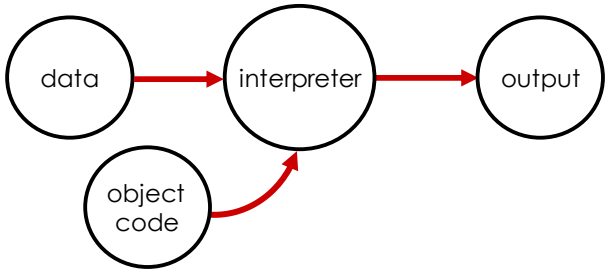
Jenis Translator

2. **Compiler**, *source code* bahasa *high-level*, *object code* adalah bahasa mesin. *Source code* dan data diproses terpisah.



Jenis Translator

3. **Interpreter**, *interpreter* tidak menghasilkan bentuk *object code*, hasil translasinya dalam bentuk internal, dimana *source code* harus selalu ada.



Pembuatan Compiler

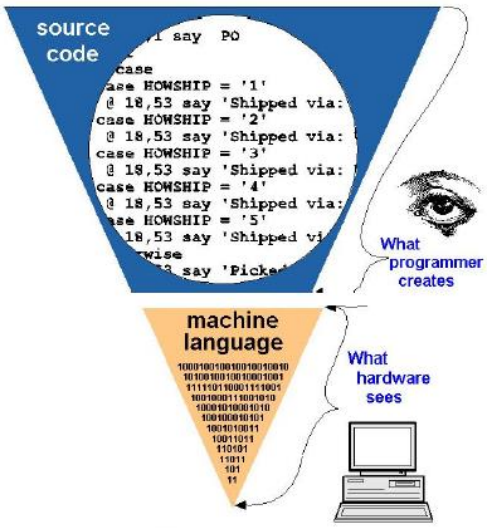
- **Bahasa Mesin**
  - sukar;
  - tergantung pada jenis mesin (semakin baik semakin rumit); dan
  - lebih besar kemungkinan digunakan saat pembuatan *assembler*
- **Bahasa Rakitan (Assembly)**
  - hasil program memiliki ukuran yang sangat kecil;
  - sulit dipahami karena perintahnya singkat-singkat;
  - fasilitas terbatas.

Pembuatan Compiler

- **High-Level Language**
  - bahasa yang lebih dikenal dan lebih dapat dipahami oleh manusia karena *statement* (pernyataan) dibuat dalam bahasa yang mudah dimengerti oleh manusia, yaitu bahasa Inggris;
  - memberikan fasilitas dan kemudahan yang lebih banyak bagi *user*;
  - program mudah untuk diperbaiki dan dikoreksi (*debug*); dan
  - tidak bergantung pada salah satu jenis mesin komputer dan masih membutuhkan *translator*.
- **Bootstrap**

  "Untuk membangun sesuatu yang besar maka dibuat dahulu bagian intinya" (Niklaus Wirth).

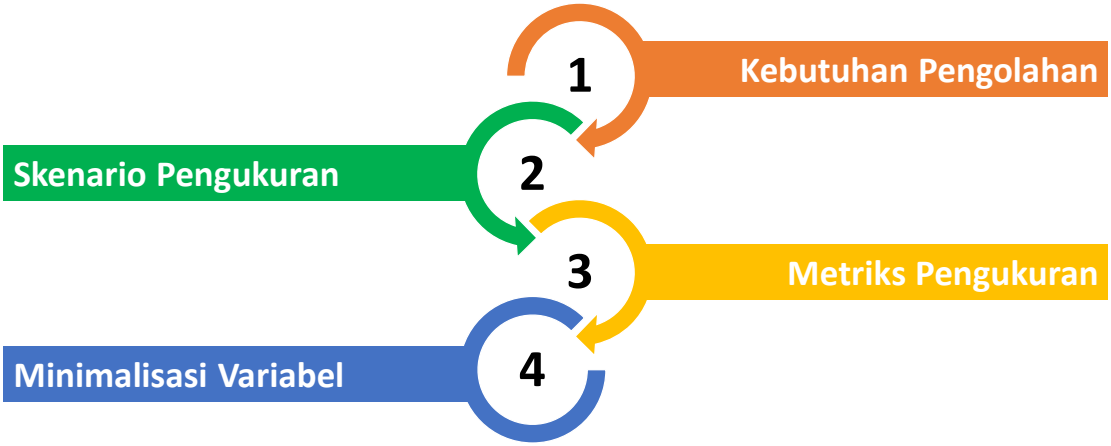
Pembuatan Compiler



Statement yang dibuat oleh manusia pada sebuah bahasa pemrograman (*source language*) dilihat sebagai barisan bit 0 dan 1 oleh mesin (*machine language*)

Mengukur Mutu Kompilator

Brainstorming



Mengukur Mutu Kompilator

Atribut *Compiler*.

- Kompilator memiliki waktu eksekusi yang singkat.
- Tidak ada interaksi pengguna selama eksekusi.
- Kompilator mengeksekusi dalam sejumlah fase di mana output satu fase menjadi input untuk fase berikutnya. Setiap fase memiliki hasil yang berdiri sendiri, sehingga menyulitkan untuk melakukan *debugging*.

Mengukur Mutu Kompilator

- Secara umum kompiler melakukan analisis di seluruh level program, sehingga data seluruh set input tersimpan di memori dan semuanya dapat dioperasikan sekaligus.
- Dengan beberapa pengecualian, keluaran kompiler tidak dapat dengan mudah diverifikasi dengan inspeksi atau alat uji jenis lain.

Mengukur Mutu Kompilator

Bagaimana melakukan pengujian mutu kompilator?

1. **Mendokumentasikan semua hasil pengujian.**
   Melakukan pencatatan terkait ***exception handling*** (EH) – penanganan pengecualian.

Pengecualian

"Suatu keadaan tidak normal yang terjadi ketika eksekusi suatu program sehingga membutuhkan proses khusus yang bahkan dapat mengubah alur program itu sendiri".

Mengukur Mutu Kompilator

2. **Tes Permutasi**

   Tes yang dilakukan untuk menentukan seberapa luas jangkauan sebuah objek (string) yang diuji dengan cara menyusun kembali suatu kumpulan objek dalam urutan yang berbeda dari urutan yang semula.

3. **Real-World Code**

   Menambahkan dokumentasi hasil pengujian yang telah dilakukan dengan hasil pengujian orang lain atau penguji lain

Mengukur Mutu Kompilator

4. **Tes Performa**
   1. **Kualitas**.

      Pengujian yang dilakukan untuk mengukur kecepatan kompilator dalam menghasilkan *object code*.

   2. **Ukuran**.

      Besarnya ukuran kompilator yang dihasilkan. Pengukuran di sisi ini sangat berkaitan erat dengan pengukuran kualitas kompilator.

   3. **Throughput**.

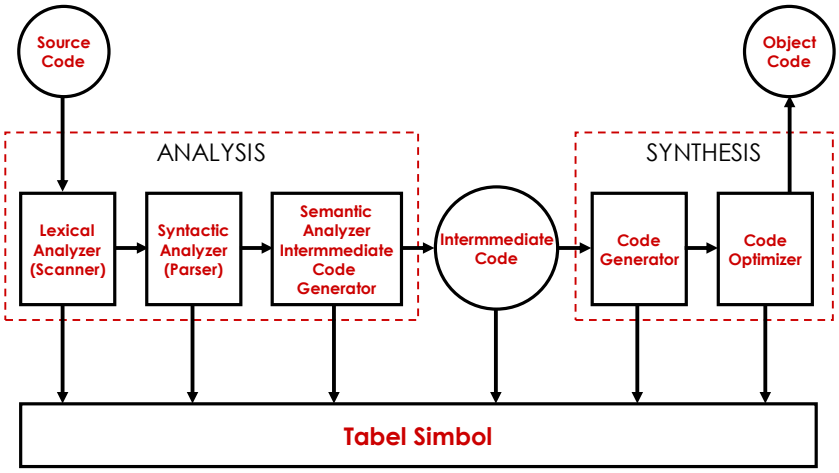
      Pengujian yang dilakukan untuk mengukur seberapa cepat kompiler bekerja.

Proses dan Tahapan Kompilasi

Terbagi ke dalam 2 kelompok besar:

1. **Analisa (*Front End*)**
   Melakukan dekomposisi program sumber (*source code*) menjadi bagian-bagian dasarnya (*intermediate representation*).
2. **Sintesa (*Back End*)**
   Melakukan pembangkitan dan optimasi program objek.

Tahapan Proses Kompilasi

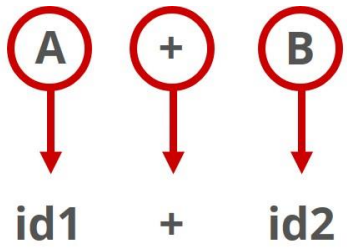


Tahapan Proses Kompilasi

1. **Scanner**

   Memecah program sumber menjadi besaran **leksikal/token** yaitu, satuan terkecil dari bahasa, yaitu deretan karakter terpendek yang mengandung arti (*identifier*, *keyword*, label, operator aritmatika dan relasional, dan operator penugasan).

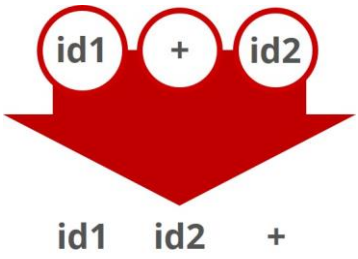
Contoh: Perhitungan aritmatika A + B



Tahapan Proses Kompilasi

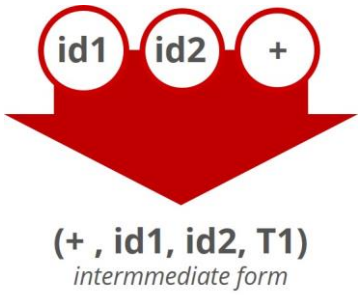
2. **Parser**

   Memeriksa kebenaran dan urutan kemunculan ***token*** (notasi ***postfix***)



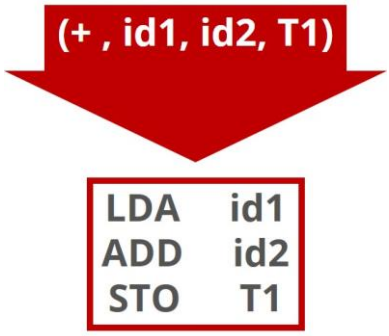
3. **Semantic Analyzer (SA)**

   Analisis yang membangkitkan kode antara **(*intermmediate form*)**

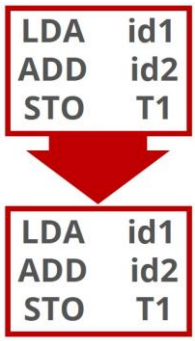


Tahapan Proses Kompilasi

4. **Code Generator (CG)**
   Membangkitkan ***object code***

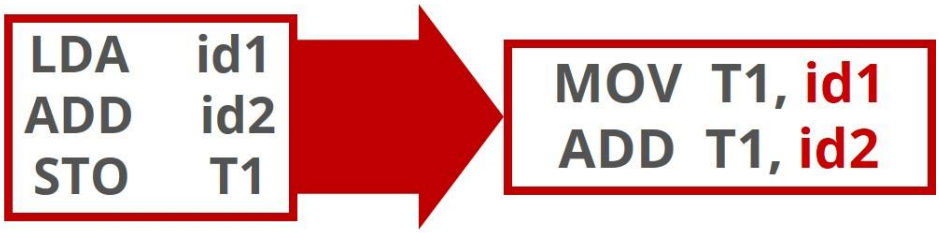


5. **Code Optimizer (CO)**
   Memperkecil hasil dan mempercepat proses



Tahapan Proses Kompilasi

6. **Object Code (OC)**
   Menuliskan dalam bahasa ***assembly***.




TEORI KOMPUTASI
good bye!
UNTIL NEXT WEEK...