TEKNIK KOMPILASI

PROGRAM STUDI : TEKNIK INFORMATIKA

UNIVERSITAS INDRAPRASTA PGRI

JL. Nangka No. 58 C Tanjung Barat Jagakarsa, Jakarta Selatan
JL. Raya Tengah, Kelurahan Gedong, Pasar Rebo, Jakarta Timur
Website : http://www.unindra.ac.id

DAFTAR PUSTAKA

- Practice and principles of Compiler building with C, Henk Alblas, Albert Nymeyer, Prentice Hall, 1996
- Introduction to The theory of computation, Michael sipser, PWS publishing Company, 1997
- The Essence of Compilers, Robin Hunter,Prentice Hal Europe, 1999
- Modern Compiler Design, Dick Grune, Henri E. Bal, Et all, JohnWiley & Son, 2000

T U J U A N

- Mengetahui Penerapan konsep ilmu komputer pada perilaku komputer yaitu algoritma, arsitektur komputer, stuktur data maupun penerapan teori bahasa dan automata

- Compiler adalah merupakan konstruksi inti dari ilmu komputer

ARTI KATA TEKNIK KOMPILASI

- **Teknik** :
  - Metode atau Cara

- **Kompilasi** :
  - Proses mengabungkan serta menterjermahkan sesuatu (source program) menjadi bentuk lain

- **Compiler** :
  - To translate a program written in a high-level programming language into machine language.

Translator : Compiler & Interpreter

Translator :

- Adalah suatu program dimana mengambil input sebuah program yang ditulis pada satu bahasa program (source language) ke bahasa lain (The object on target language)

- Jika source language adalah high level language, seperti cobol, pascal, fortran maka object language adalah low-level language atau mesin language.Translator seperti ini disebut COMPILER

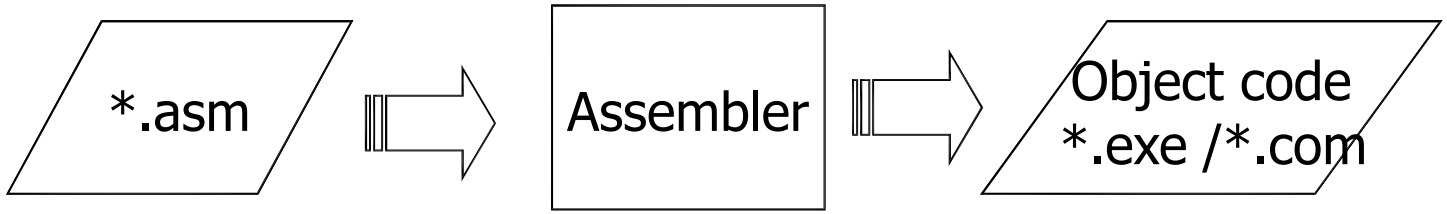
Kenapa perlu Translator ?

- Dengan bahasa mesin adalah bahasa bentuk bahasa terendah komputer, berhubungan langsung dengan bagian bagian komputer seperti bits, register & sangat primitive
- Jawaban atas pertanyaan ini akan membingungkan bagi programmer yang membuat program dengan bahasa mesin.
- Bahasa mesin adalah tidak lebih dari urutan 0 dan 1
- Instruksi dalam bahasa mesin bisa saja dibentuk menjadi micro-code, semacam prosedur dalam bahasa mesin
- Bagaimana dengan orang tidak mengerti bahasa mesin

Ada Beberapa Translator

1. Assembler

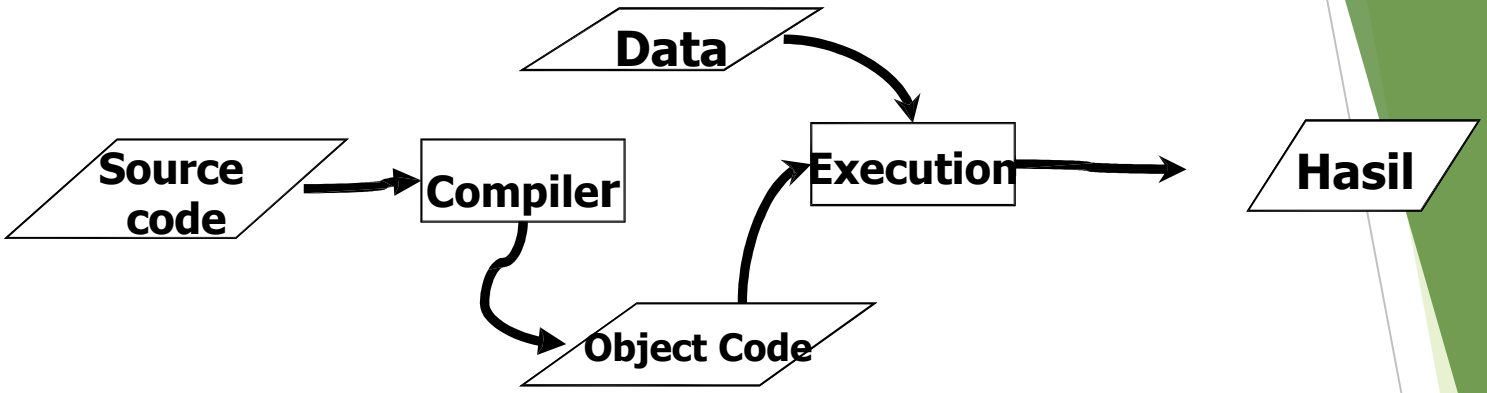
Source code adalah bahasa assembly, Object code adalah bahasa mesin



2. Compiler

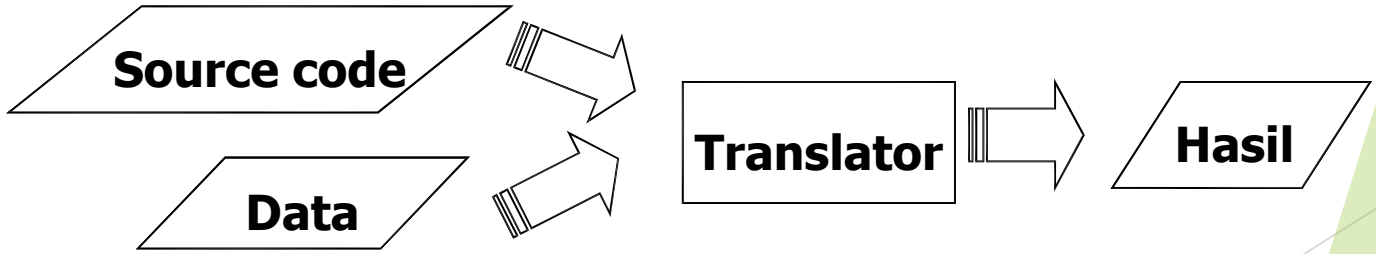
Source code adalah bahasa tingkat tinggi, object code adalah bahasa mesin atau bahasa assembly. Source code dan data diproses berbeda

compiler

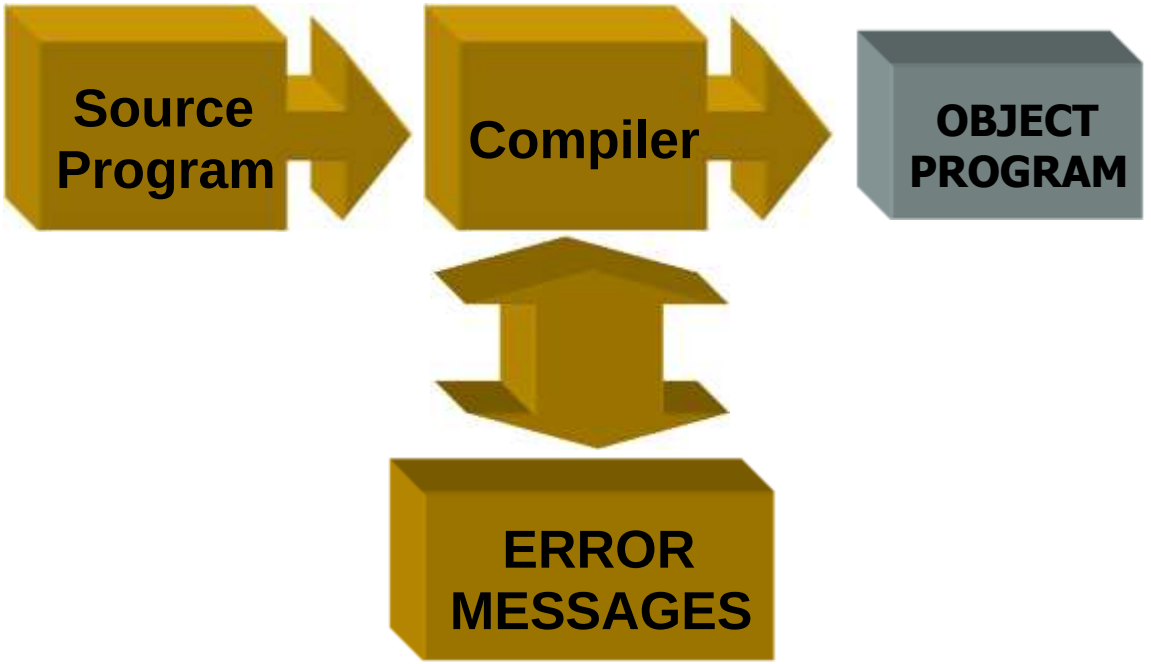


3. Interpreter

Interpreter tidak menghasilkan bentuk object code, tetapi hasil translasinya hanya dalam bentuk internal, dimana program induk harus selalu ada-berbeda dengan compiler



Translator : Compiler & Interpreter



COMPILER vs INTERPRETER

- Compiler bisa menangkap berbagai kesalahan dalam 1 program kode sumber secara sekaligus. Kalau Interpreter cuma bisa menangkap beberapa kesalahan pada 1 baris kode sumber pada suatu saat
- Biasanya program yang dihasilkan compiler lebih cepat dari waktu pelaksanaan program dengan interpreter.
- Kalau compiler menghasilkan kode antara (misal object code) dan harus digabungkan / dilink menjadi bentuk yang dapat dijalankan mesin / komputer (executable). Kalau Interpreter biasanya tidak menghasilkan kode antara.
- Kalau hendak menjalankan program hasil kompilasi bisa dilakukan tanpa kode sumber. Kalau interpreter butuh kode sumber.

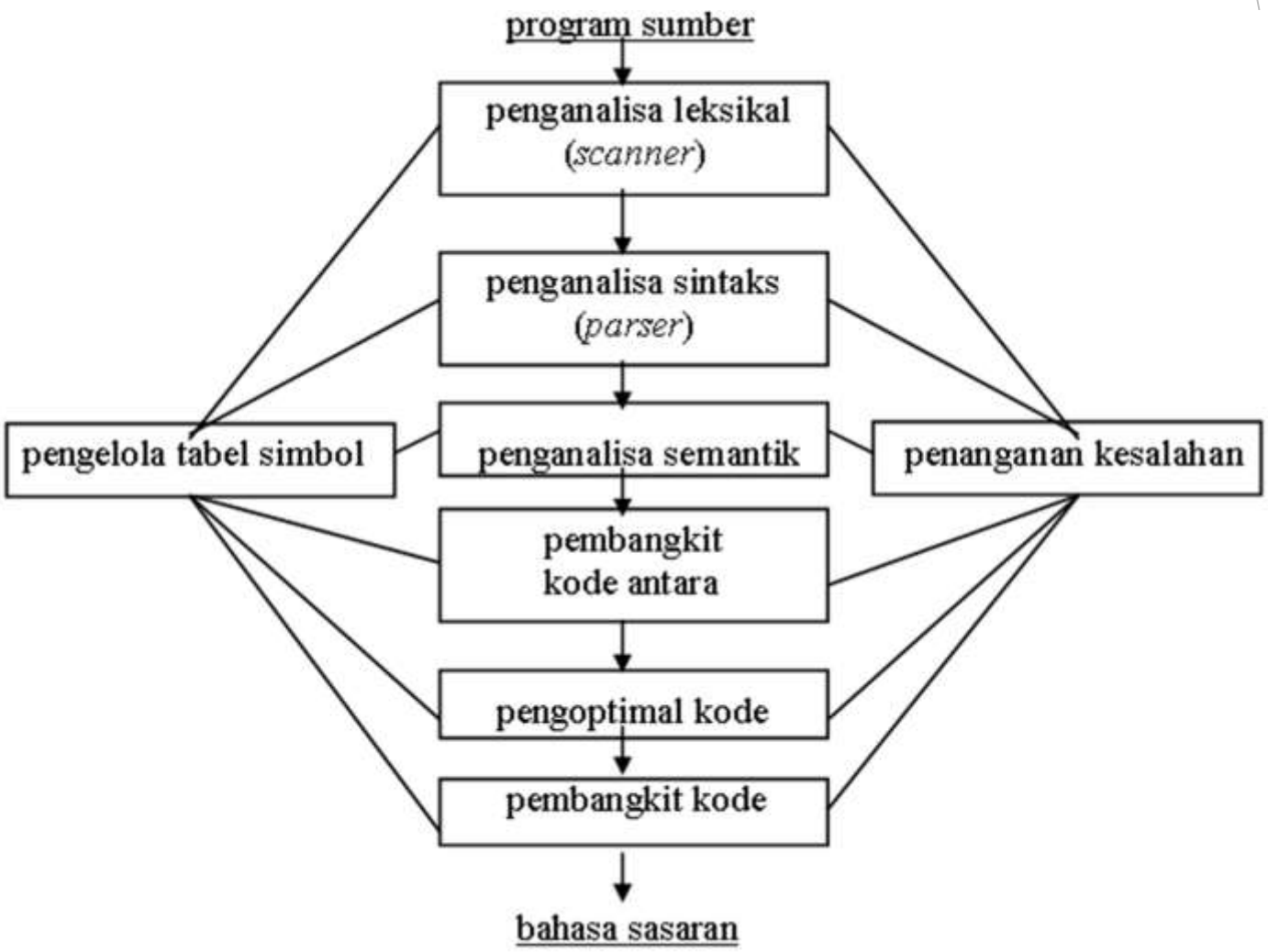
COMPILER vs INTERPRETER

- Kalau dengan kompiler, maka pembuatan kode yang bisa dijalankan mesin dilakukan dalam 2 tahap terpisah, yaitu parsing / pembuatan kode objek dan linking / penggabungan kode objek dengan library. Kalau interpreter tidak ada proses terpisah.
- Kalau compiler membutuhkan linker untuk menggabungkan kode objek dengan berbagai macam library demi menghasilkan suatu kode yang bisa dijalankan oleh mesin. Kalau interpreter tidak butuh linker.
- Interpreter cocok untuk membuat / menguji coba modul / sub-routine / program-program kecil. Kalau compiler agak repot karena untuk mengubah suatu modul / kode objek kecil, maka harus dilakukan proses linking / penggabungan kembali semua objek dengan library yang diperlukan.
- Pada kompiler bisa dilakukan optimisasi / peningkatan kwalitas kode yang bisa dijalankan.Ada yang dioptimasi supaya lebih cepat, ada yang supaya lebih kecil, ada yang dioptimasi untuk sistem dengan banyak processor. Kalau interpreter susah / tidak bisa dioptimasikan.

Proses kompilasi dikelompokkan ke dalam dua kelompok besar :

1. *analisa* : program sumber dipecah-pecah dan dibentuk menjadi bentuk antara (*inter-mediate representation*)
2. *sintesa* : membangun program sasaran yang diinginkan dari bentuk antara

Fase-fase proses sebuah kompilasi



Penganalisa Leksikal

- membaca program sumber, karakter demi karakter. Sederetan (satu atau lebih) karakter dikelompokkan menjadi satu kesatuan mengacu kepada *pola kesatuan kelompok karakter* (*token*) yang ditentukan dalam *bahasa sumber*. Kelompok karakter yang membentuk sebuah token dinamakan *lexeme* untuk token tersebut. Setiap token yang dihasilkan disimpan di dalam *tabel simbol*. Sederetan karakter yang tidak mengikuti pola token akan dilaporkan sebagai *token tak dikenal* (*unidentified token*)

Penganalisa Sintaks

- memeriksa kesesuaian *pola deretan token* dengan aturan sintaks yang ditentukan dalam *bahasa sumber*. Sederetan token yang tidak mengikuti aturan sintaks akan dilaporkan sebagai *kesalahan sintaks* (*sintax error*). Secara logika deretan token yang bersesuaian dengan sintaks tertentu akan dinyatakan sebagai pohon parsing (*parse tree*)

Penganalisa Semantik

- memeriksa token dan ekspresi dari batasan-batasan yang ditetapkan. Batasan-batasan tersebut misalnya :

a. panjang maksimum token *identifier* adalah 8 karakter,

b. panjang maksimum ekspresi tunggal adalah 80 karakter,

c. nilai bilangan bulat adalah -32768 s/d 32767,

d. operasi aritmatika harus melibatkan operan-operan yang bertipe sama

Pembangkit Kode Antara

- membangkitkan kode antara (*intermediate code*) berdasar-kan pohon parsing. Pohon parse selanjutnya diterjemahkan oleh suatu penerjemah yang dinamakan *penerjemah berdasarkan sintak* (*syntax-directed translator*). Hasil penerjemahan ini biasanya merupakan *perintah tiga alamat* (*three-address code*) yang merupakan representasi program untuk suatu *mesin abstrak*. Perintah tiga alamat bisa berbentuk *quadruples* (*op, arg1, arg2, result*), *tripels* (*op, arg1, arg2*). Ekspresi dengan satu argumen dinyatakan dengan menetapkan *arg2* dengan - (strip, *dash*)

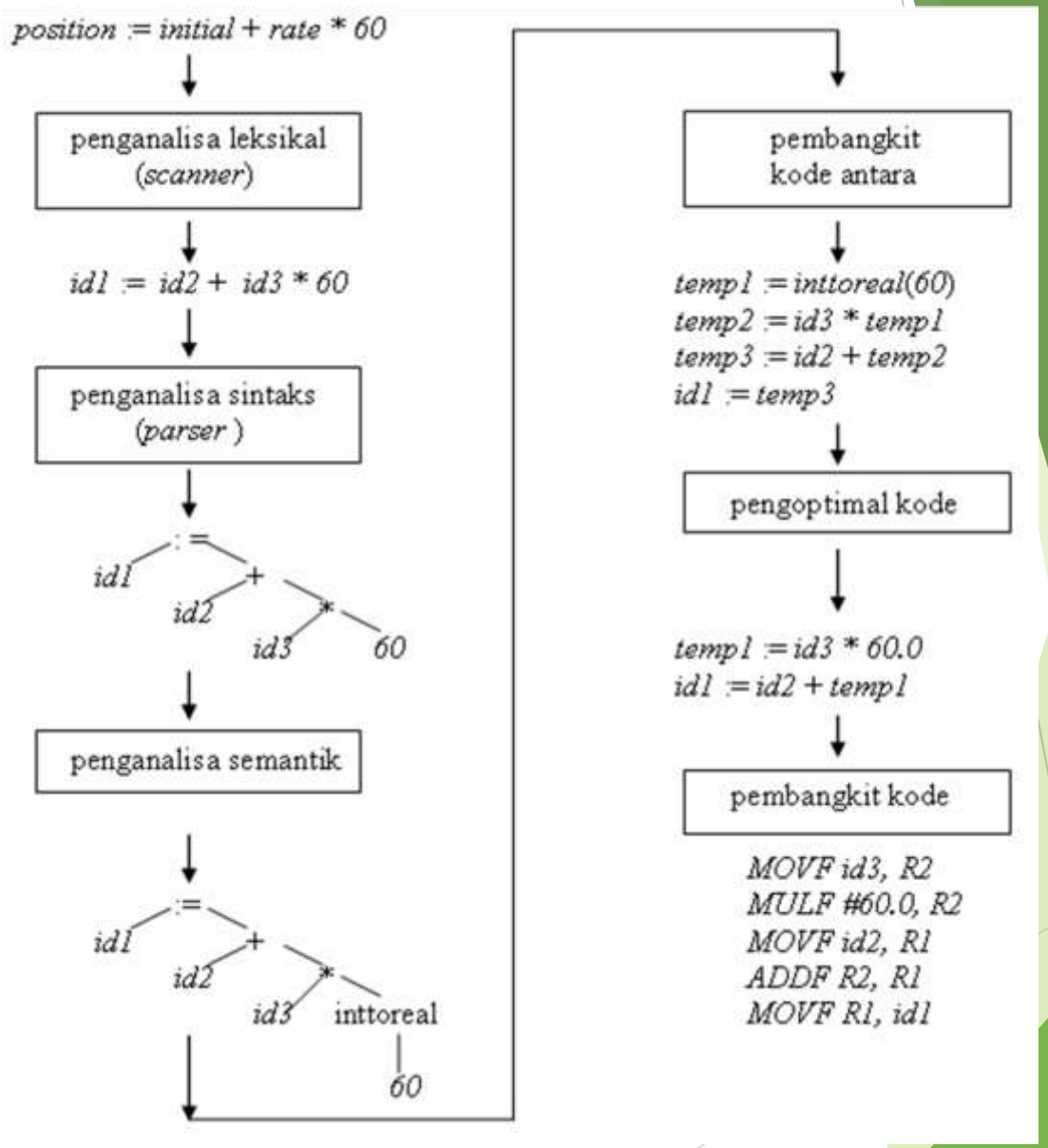
Pengoptimal kode

- melakukan optimasi (penghematan space dan waktu komputasi), jika mungkin, terhadap kode antara

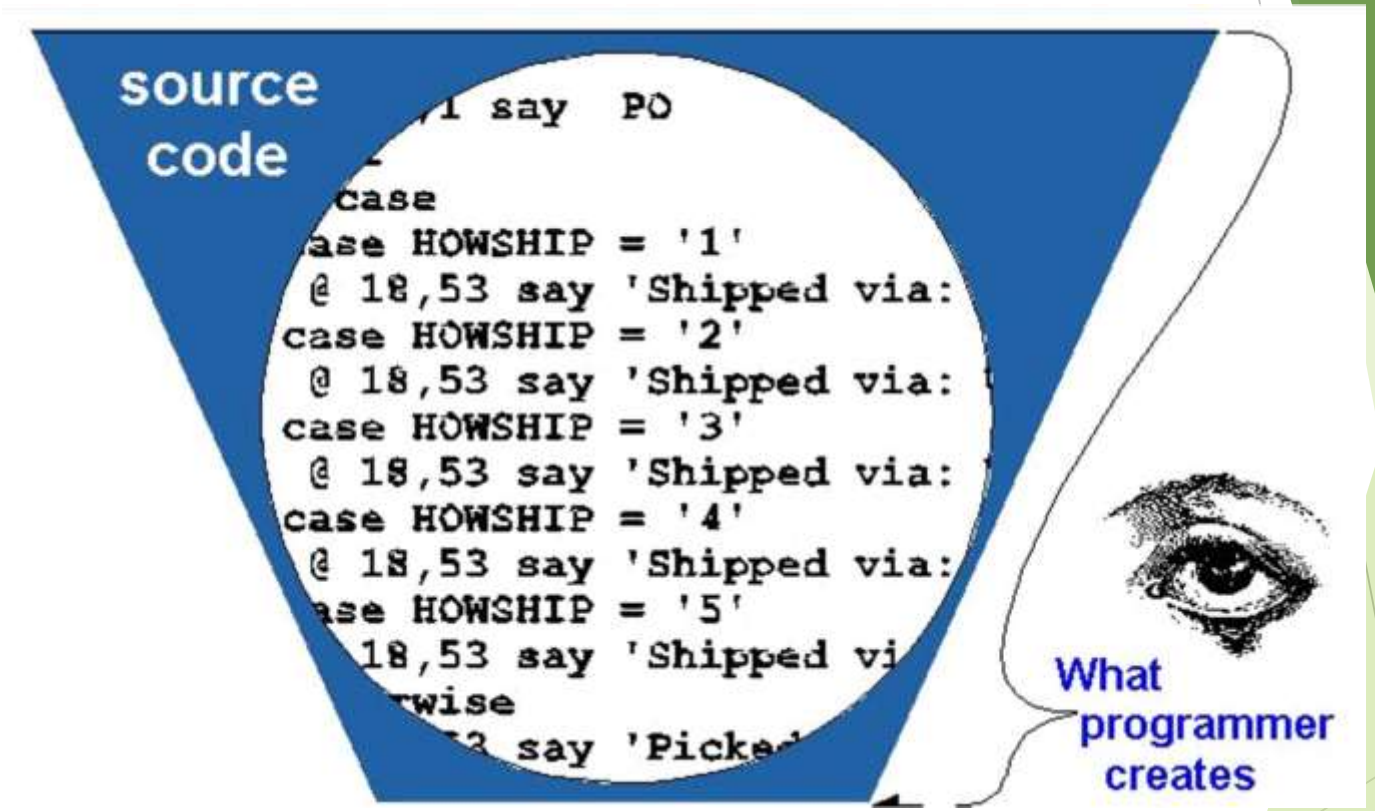
Pembangkit Kode Mesin

- membangkitkan kode dalam bahasa target tertentu (misalnya bahasa mesin)

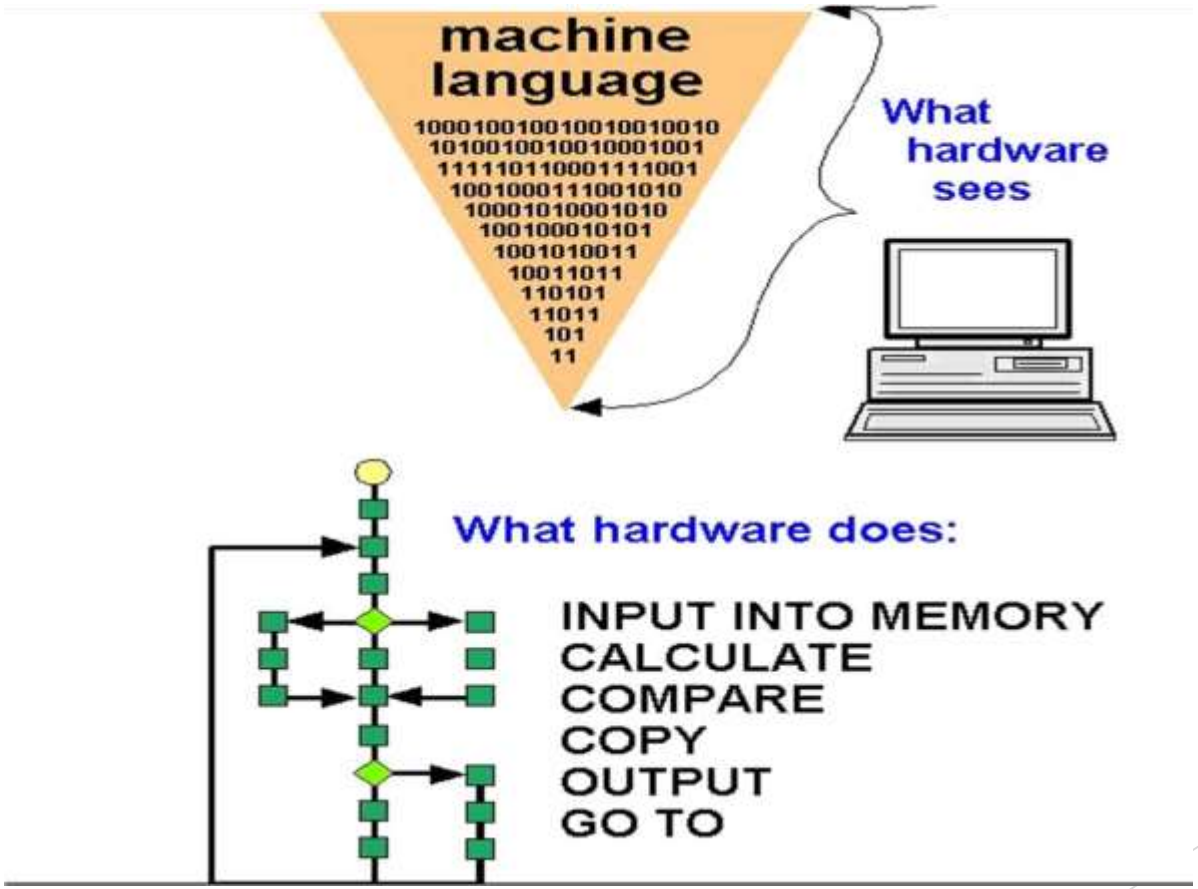
Contoh Kompilasi



View dari programmer



Mesin View


machine language
10001001001001001001
10100100100100011001
11111011000111111001
10010001110010101
10001010001010
10010001010
1001010011
10011011
110101
11011
101
11
What hardware sees
What hardware does:
INPUT INTO MEMORY
CALCULATE
COMPARE
COPY
OUTPUT
GO TO

Pembuatan compiler

Bahasa mesin

- Sangat sukar dan sangat sedikit kemungkinannya untuk membuat compiler dengan bahasa ini, karena manusia susah mempelajari bahasa mesin,
- Sangat tergantung pada mesin,
- Bahasa Mesin kemungkinan digunakan pada saat pembuatan Assembler

Pembuatan compiler

Assembly

- Hasil dari program mempunyai Ukuran yang relatif kecil
- Sulit dimengerti karena statement/perintahnya singkat-singkat, butuh usaha yang besar untuk membuat
- Fasilitas yang dimiliki terbatas

Pembuatan compiler

Bahasa Tingkat Tinggi (high level language)

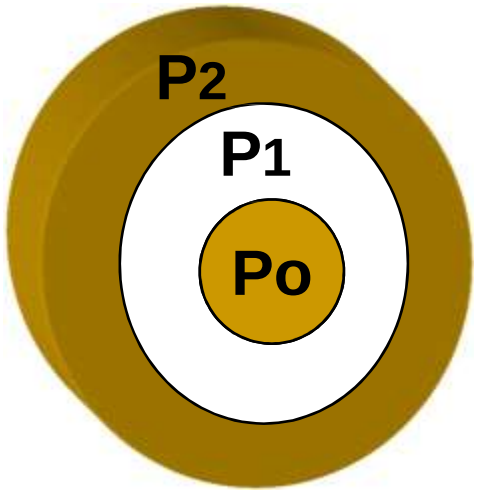
- Lebih mudah dipelajari
- Fasilitas yang dimiliki lebih baik (banyak)
- Memiliki ukuran yang relatif besar, misal membuat compiler pascal dengan menggunakan bahasa C
- Untuk mesin yang berbeda perlu dikembangkan tahapan-tahapan tambahan.
- Misal membuat compiler C pada Dos bedasarkan compiler C pada unix

Pembuatan compiler

BootStrap

- Untuk membangun sesuatu yang besar, dibangun/dibuat dulu bagian intinya (niklaus Wirth - saat membuat pascal compiler)

BootStrap



- PO dibuat dengan assembly,
- P1 dibuat dari P0, dan
- P2 dibuat dari P1, jadi compiler untuk bahasa P dapat dibuat tidak harus dengan menggunakan assembly secara keseluruhan

Contoh dari source program ke dalam kode mesin

IF COUNT =10
GOTO DONE
ELSE
GOTO AGAIN
ENDIF
	Compare A to B
	If equal go to C
	Go to D
		Compare 3477 2883
		If = go to 23883
		Go to 23343

1 001 01 01 001 01 000
1 01 01 00
1 01 01 01 001 01 01 00
1 001 01 0
1 01 001 01 01 0001 01
001 001 0