Program

Retno Mumpuni

```
const LOCALE = globalThis.navigator.language

const div = document.body.appendChild(document.createElement('div'))
const list = div.appendChild(document.createElement('ol'))

const dayNames = new Map()

for (let i = 0; i < 7; ++i) {
    const d = Temporal.PlainDate.from({
        year: Temporal.Now.plainDateISO().year,
        month: 1,
        day: i + 1,
    })

    dayNames.set(d.dayOfWeek, d.toLocaleString(LOCALE, { weekday: 'long' }))
}

for (const num of [...dayNames.keys()].sort((a, b) => a - b)) {
    list.appendChild(Object.assign(
        document.createElement('li'),
        { textContent: dayNames.get(num) },
    ))
}
```

Program = Kumpulan Instruksi

- Program ditulis dengan menggunakan kaidah / aturan bahasa pemrograman tertentu
- Sama halnya komunikasi antar manusia, komputer bisa menjalankan pekerjaannya sesuai dengan instruksi yang mengikuti kaidah/aturan.

- Secara garis besar, bahasa pemrograman dibagi :
  - High-level language : Bahasa beraras-tinggi
  - Low-level language : Bahasa beraras-rendah

Low-level language

- Bahasa beraras rendah : bahasa pemrograman yang berorientasi pada mesin.
- Menggunakan kode biner (0 atau 1), atau sebuah kode sangat sederhana untuk menggantikan kode tertentu dalam sistem biner.
- Sangat sulit dipahami oleh orang awam, bahkan programmer sekalipun.
- Sangat bergantung pada mesin (machine dependent)
- Sangat cepat dieksekusi – komputer tidak perlu menerjemahkan terlalu jauh
- MISAL :
  - B402 atau 1011 0100 0000 0100 artinya : inputkan angka 2 ke register AH
  - B22A atau 1011 0010 0010 1010 artinya : muatlah angka A2 hex ke register DL
  - CD21 atau 1100 1101 0010 0001 artinya : jalankan interupsi 21 heksadesimal

  3 perintah diatas harus dieksekusi berurutan untuk sekedar menampilkan karakter * pada layar

High-level language

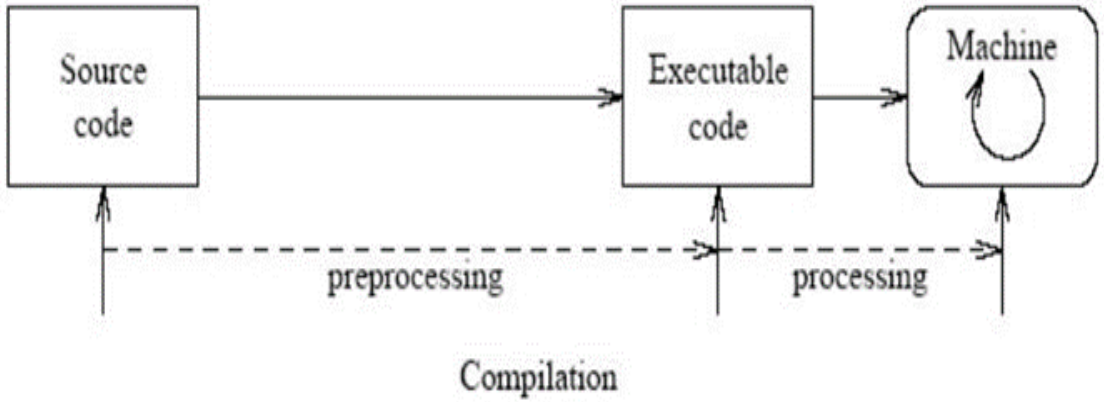
- Bahasa beraras tinggi adalah bahasa pemrograman yang berorientasi pada bahasa manusia.
- Dibuat dengan menggunakan bahasa yang mudah dipahami oleh manusia
- Contoh : Java, C, C++, Pascal, Basic, PHP, dll

- Misal :
  - Write (‘*’) -- PASCAL
  - Display ‘*” – COBOL
  - PRINT “*” – BASIC
  - Printf (“*”) -- C
  - System.out.println(“*”) – JAVA
  - Echo (“*”) – PHP

  Semua baris di atas adalah contoh instruksi untuk menampilkan karakter * pada layar, dengan masing-masing contoh bahasa.

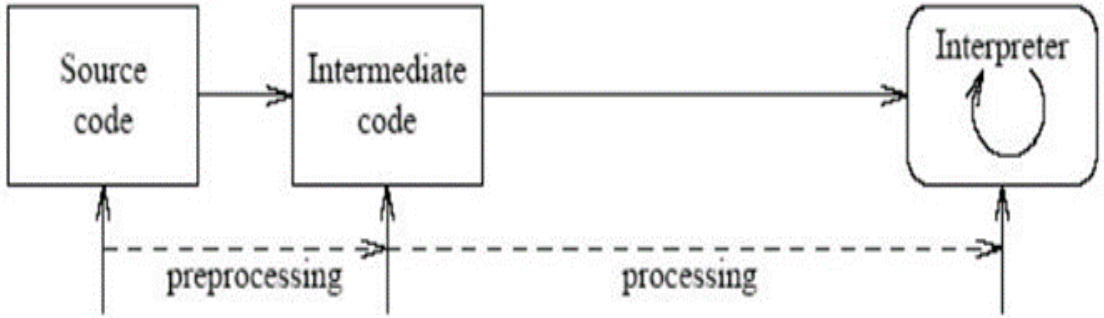
Translator

- Sebelum bisa dieksekusi oleh mesin (komputer), high level language / bahasa beraras tinggi harus diubah / diterjemahkan terlebih dahulu ke bahasa mesin.
- Proses penerjemahan ini memerlukan **translator.**
- Berdasarkan urutan kerjanya, terdapat dua macam translator :
  - Compiler : C, C++, Pascal
  - Interpreter : Python, Matlab, JAVA



Compilation

Di compiler, semua instruksi dalam high level harus diterjemahkan secara utuh terlebih dahulu dalam suatu executable code, untuk kemudian dieksekusi mesin. – misal : .exe file



Interpretation

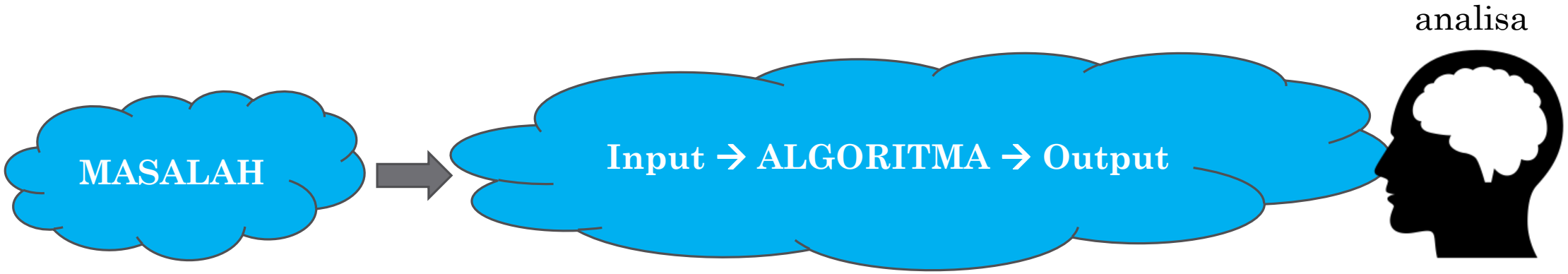
Di interpreter, tiap baris instruksi dalam high level diterjemahkan menjadi intermediate code per baris, untuk kemudian tiap baris intermediate code tersebut dieksekusi oleh mesin. – konsekuensinya : lebih lambat

Interpreter vs Compiler

Interpreter	Compiler
Kelebihan : Mudah untuk mencari kesalahan seandainya terdapat kesalahan pada program (debugging). Karena program dapat terus berjalan, hingga akhirnya komputer menemukan kesalahan.	Kelebihan : 1. Pengerjaan instruksi dilakukan sangat cepat, karena telah ditranslate ke bahasa mesin secara utuh. 2. Kode sumber tidak perlu didistribusikan ke pengguna yang menjalankannya, sehingga kerahasiaan terjamin.
Kekurangan : 1. Kode sumber harus selalu tersedia – isu pencurian hak cipta 2. Eksekusi berjalan lambat	Kekurangan : Kode sumber program yang ditulis harus benar seluruhnya secara sintaks, baru kemudian program dapat berjalan.

Membuat Program

- Program dibuat karena adanya masalah atau tugas yang akan diselesaikan
- Tahapan pembuatan program :
  - Menganalisis masalah
  - Membuat algoritma
  - Mengimplementasikan algoritma ke dalam instruksi program
  - Mengeksekusi dan menguji program
- Menganalisis masalah adalah tahapan yang penting. Diperlukan pengalaman, pengetahuan, imajinasi, kreativitas, dan kecerdasan untuk menganalisa informasi apa yang akan menjadi input dan output, serta bagaimana mengolahnya.



Contoh Algoritma

- Menghitung Luas Lingkaran
  - 3.14 x jari-jari ^{2}
- Algoritma / runutan langkahnya adalah sebagai berikut :
  - Dapatkan jari-jari lingkaran
  - Hitung luas lingkaran dengan menggunakan rumus 3.14 x jari-jari ^{2}
  - Tampilkan Luas
- Selain dengan kalimat, algoritma juga bisa dinyatakan dalam **pseudocode**

Mengubah Algoritma menjadi Pseudocode

- Algoritma / runutan langkahnya adalah sebagai berikut :
  - Dapatkan jari-jari lingkaran → Jari-jari ← 20
  - Hitung luas lingkaran dengan Menggunakan rumus 3.14 x jari-jari ^{2} → Luas ← 3.14 * (Jari-jari^2)
  - Tampilkan Luas → Print (Luas)

Contoh Program

```
#include <stdio.h>

int main ( )
{
        double jari_jari;
        double luas;

        jari_jari = 20;

        luas = 3.14 * jari_jari * jari_jari;

        printf("Luas lingkaran = %lf", luas);

        return 0;
}
```

Pseudocode

Jari-jari ← 20

Luas ← 3.14 * (Jari-jari2)

Print (Luas)

Terimakasih