



3.1 programming Language Processing

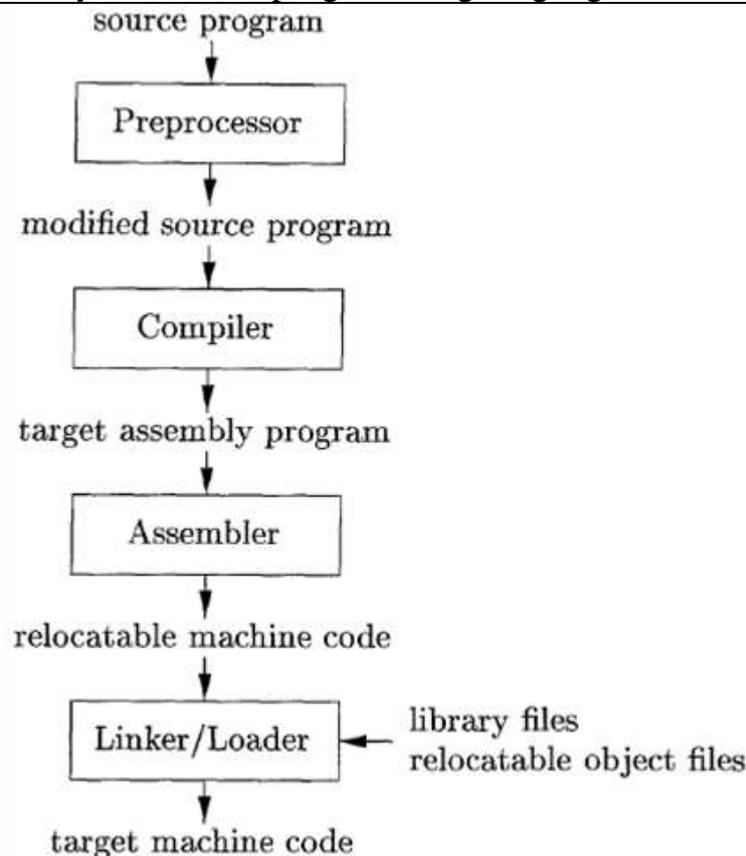


source language :

Likes traditional programming languages such as C# and Java.

target language :

May be another programming language, or machine language.



تعريف الـ

Preprocessor

والـ

Assembler

والـ

Linker/loader

A source program may be divided into **modules** stored in separate files. The task of **collecting** the source program is sometimes entrusted to a separate program, called a *preprocessor*.

Preprocessor may expand shorthands, called **macros**, into source language statements. Modified source program is fed to compiler.

The **compiler** may produce an assembly-language program as its output, because **assembly language is easier to produce** as output and is **easier to debug**. The assembly language is then processed by a program called an **assembler** that produces **relocatable** machine code as its output.

Large programs are often **compiled in pieces**, so the relocatable machine code may have to be linked together with other relocatable object files and **library files** into the code that actually runs on the machine. The **linker** resolves external memory addresses, where the code in one file may refer to a location in another file. The **loader** then puts together all of the executable object files **into memory** for execution.

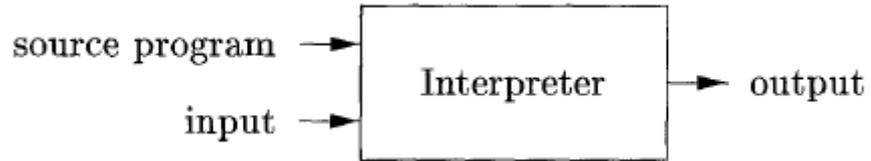


تعريف الـ
interpreter

An **interpreter** is common kind of language processor.

Instead of producing a target program as a translation, an interpreter appears to directly **execute** the operations specified in the **source program** on **inputs** supplied by the user.

بينفذ على طوووول ..



صحح للكلام المفيد

بميزة الكوموبايلر

The machine-language **target program** produced by a compiler is usually much **faster** than an interpreter at mapping inputs to outputs .

بميزة الانتربرطر

An interpreter, can give **better error diagnostics** than a compiler, because it executes the source program statement by statement.

What is the difference between a **compiler** and an **interpreter**?

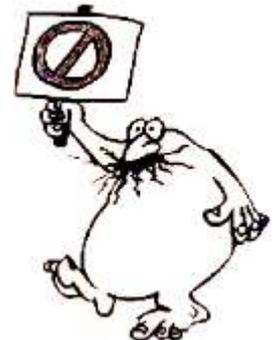
What are the advantages of

- (a) a **compiler over an interpreter**
- (b) an **interpreter over a compiler**?

What advantages are there to a language-processing system in which the **compiler produces assembly** language rather than machine language?

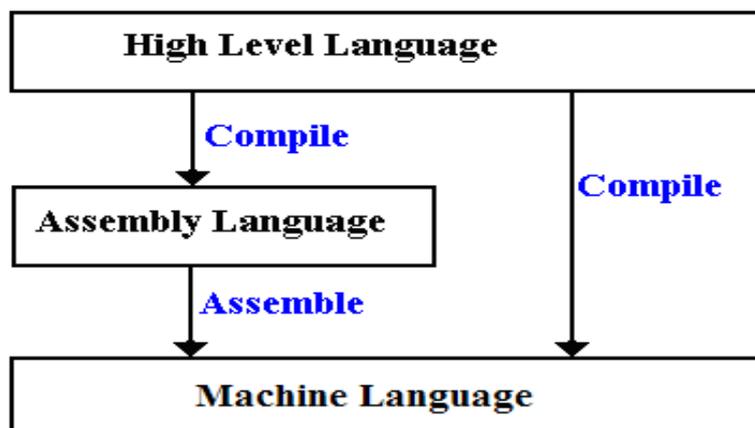
A compiler that translates a high-level language into another high-level language is called a *source-to-source* translator. What advantages are there to **using C as a target language** ?

Describe some of the **tasks** that an **assembler** needs to perform.



مع المفتش كورومبو
أسئلة على اللي فات

To summarize



Phases of Compiler

position = initial + rate * 60

Lexical Analyzer

$\langle \text{id}, 1 \rangle \langle = \rangle \langle \text{id}, 2 \rangle \langle + \rangle \langle \text{id}, 3 \rangle \langle * \rangle \langle 60 \rangle$

Syntax Analyzer

$\langle \text{id}, 1 \rangle$ — = — $\langle \text{id}, 2 \rangle$ — + — $\langle \text{id}, 3 \rangle$ — * — 60

Semantic Analyzer

$\langle \text{id}, 1 \rangle$ — = — $\langle \text{id}, 2 \rangle$ — + — $\langle \text{id}, 3 \rangle$ — * — inttofloat — 60

Intermediate Code Generator

```
t1 = inttofloat(60)
t2 = id3 * t1
t3 = id2 + t2
id1 = t3
```

Code Optimizer

```
t1 = id3 * 60.0
id1 = id2 + t1
```

Code Generator

```
LDF R2, id3
MULF R2, R2, #60.0
LDF R1, id2
ADDF R1, R1, R2
STF id1, R1
```

1	position	...
2	initial	...
3	rate	...

SYMBOL TABLE

References

Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman, Compilers: Principles, Techniques, and Tools