Compiler Design

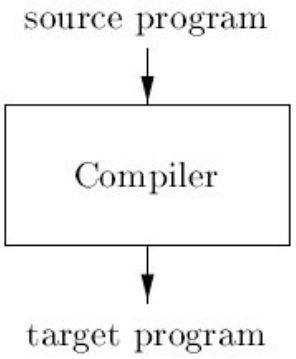
Lecture -1

Introduction

- The world as we know it depends on programming languages.
- All the software running on the computers is written in some programming language.
- So before a program runs, it first must be translated into a form in which it can be executed by a computer.
- The system software that do this translation is called *Compiler.*

Language Processors

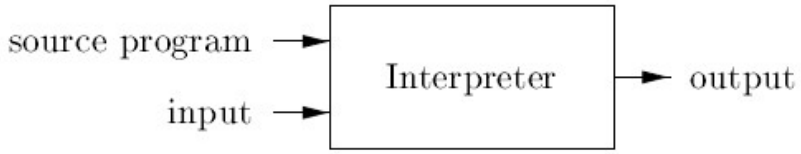
- A ***compiler*** is a program that can read a program in one language *(the source language)* and translate it into an equivalent program in another language *(the target language).*



- *An important role of the compiler is to report any errors in the source program that it detects during the translation process.*

Language Processors

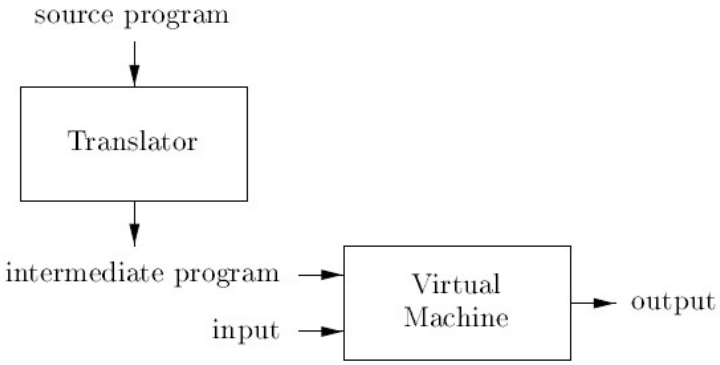
- An interpreter is another common kind of language processor.
- An interpreter appears to directly execute the operations specified in the source program on inputs supplied by the user.



- The machine-language target program produced by a compiler is usually much faster than an interpreter at mapping inputs to outputs .
- An interpreter,however, can usually give better error diagnostics than a compiler, because it executes the source program statement by statement.

Java Language Processor

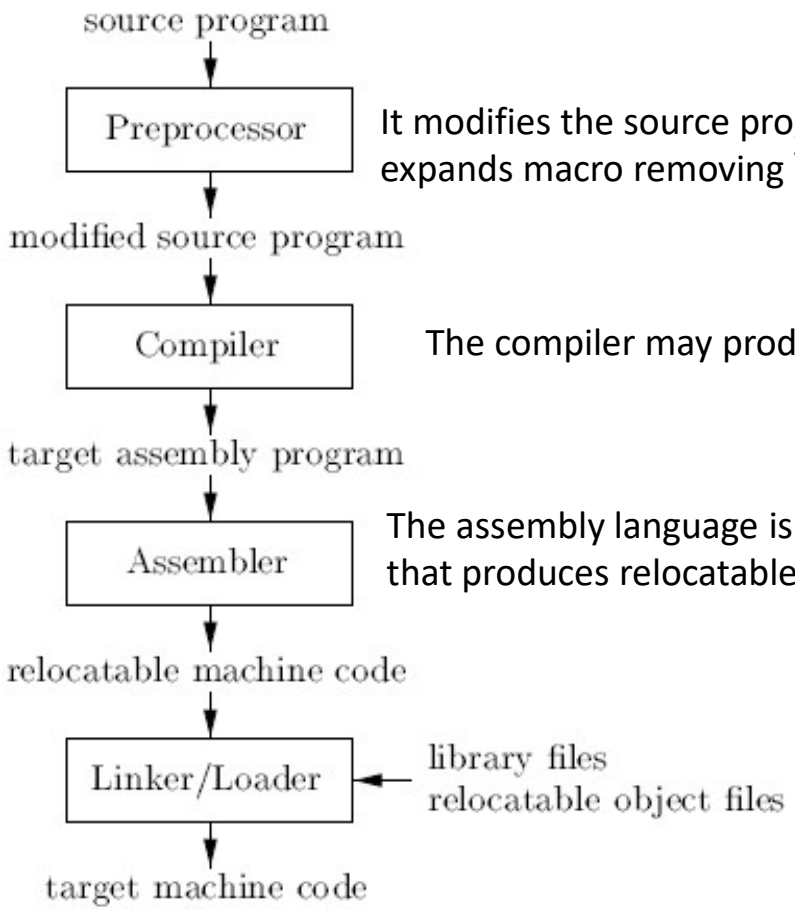
- Java language processors combine compilation and interpretation.



- A Java source program may first be compiled into an intermediate form called bytecodes.
- The bytecodes are then interpreted by a virtual machine.
- Advantage of this is that bytecodes compiled on one machine can be interpreted on another machine, perhaps across a network

COMPARISON	COMPILER	INTERPRETER
Input	It takes an entire program at a time.	It takes a single line of code or instruction at a time.
Output	It generates intermediate object code.	It does not produce any intermediate object code.
Working mechanism	The compilation is done before execution.	Compilation and execution take place simultaneously.
Speed	Comparatively faster	Slower
Memory	Memory requirement is more due to the creation of object code.	It requires less memory as it does not create intermediate object code.
Errors	Display all errors after compilation, all at the same time.	Displays error of each line one by one.
Error detection	Difficult	Easier comparatively
Pertaining Programming languages	C, C++, C#, Scala, typescript uses compiler.	PHP, Perl, Python, Ruby uses an interpreter.

A Language Processing System



It modifies the source program by removing comments,
expands macro removing \n and \t, etc

The compiler may produce an assembly-language program as its output,

The assembly language is then processed by a program called an ***assembler*** that produces relocatable machine code as its output.

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 int

Introduction

- **Why compiler?**

- Programming problems are easier to solve in high-level languages
  - Languages closer to the level of the problem domain, e.g.,
    SmallTalk: OO programming

    JavaScript: Web pages
- Solutions are usually more efficient (faster, smaller) when written in machine language

  -Language that reflects to the cycle-by-cycle working of a processor

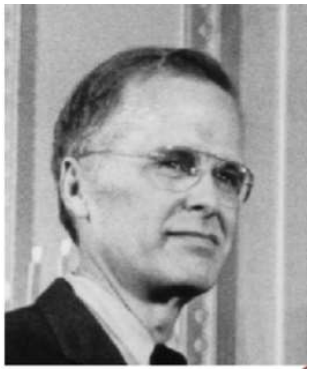
- Compilers are the bridges
  -Tools to translate programs written in high-level languages to efficient executable code

What is a Compiler?

- A system software to convert source language program to target language program.
- Validates input program to the source language specification – produces error messages/warnings.
- Primitive systems did not have Compilers, programs assembly language, hardcoded into machine code
- Compiler design started with FORTAN in 1950s
- Many tools have been developed for compiler design automation.

What, When and Why of Compilers

- What:
  - A compiler is a program that can read a program in one language and translates it into an equivalent program in another language.
- When
  - 1952, by Grace Hopper for A-0.
  - 1957, Fortran compiler by John Backus and team.



- Why? Study?
  - A programming language is an artificial language designed to communicate instructions to a machine, particularly a computer.
  - For a computer to execute programs written in these languages, these programs need to be translated to a form in which it can be executed by the computer.

Application of Compiler technology

- Parsers for HTML in web browser
- Interpreters for javascript/flash
- Machine code generation for high level languages
- Software testing
- Program optimization
- Malicious code detection
- Design of new computer architectures

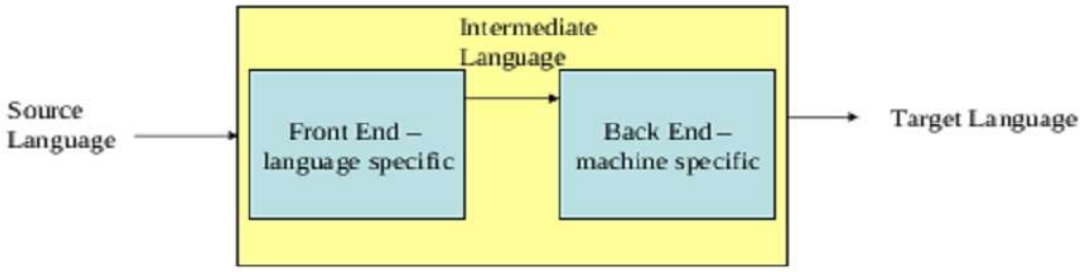
How to translate?

- Requirement:
  - In order to translate statements in a language, one needs to understand both the structure of the language:
    - the way “sentences" are constructed in the language
    - the meaning of the language: what each “sentence" stands for.

- Terminology:
  - Structure ≡ Syntax
  - Meaning ≡ Semantics

How to translate?

- Analysis-Synthesis model of compilation :
- Two parts:
  - Analysis part(Front-End)
  - Synthesis part (Back-End)



- <u>Analysis Part</u>
  - Breaks up the source program into constituent pieces and imposes a grammatical structure on them.
  - It then uses this structure to create an intermediate representation of the source program.
  - It detects that the source program is either **syntactically ill formed or semantically unsound**, then it must provide informative messages, so the user can take corrective action.
  - The analysis part also collects information about the source program and stores it in a data structure called a symbol table, which is passed along with the intermediate representation to the synthesis part.
- <u>Synthesis part</u>
  - Constructs the desired target program from the intermediate representation and the information in the symbol table.