Compiler Construction (CS-636)

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk Whatsapp# 0346-5100010

(Week 1) Lecture 1 & 2

Objectives: Learning objectives of these lectures are

Students will able to understand:

- What is Compiler?
- What is Interpreter?
- What is Hybrid Compiler?
- What is Language Processor?
- What is Language Processor System?
- What are the Components of Language Processor System?
  - Preprocessor
  - Compiler
  - Assembler
  - Linker
  - Loader

Text Book & Resources:

Compilers Principles Techniques and Tools (2nd Edition) by Alfread V. Aho, Ravi Sethi.

Videos Links:

https://youtu.be/e3fyo3uBGQw (Part 1)
https://youtu.be/h_xWnDN5Kls (Part 2)
https://youtu.be/6vcZdFk2nL8 (Part 3)

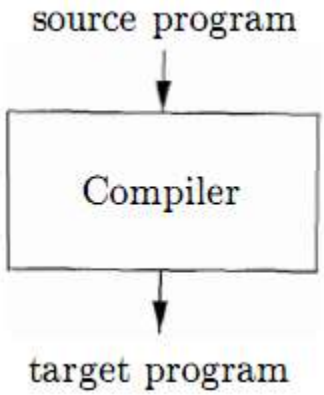
Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk Whatsapp# 0346-5100010

Introduction ot Compiler Construction

Computers are a balanced mix of software and hardware. Hardware is just a piece of mechanical device and its functions are being controlled by compatible software. Hardware understands instructions in the form of electronic charge, which is the counterpart of binary language in software programming. Binary language has only two alphabets, 0 and 1. To instruct, the hardware codes must be written in binary format, which is simply a series of 1s and 0s. It would be a difficult and cumbersome task for computer programmers to write such codes, which is why we have compilers to write such codes.

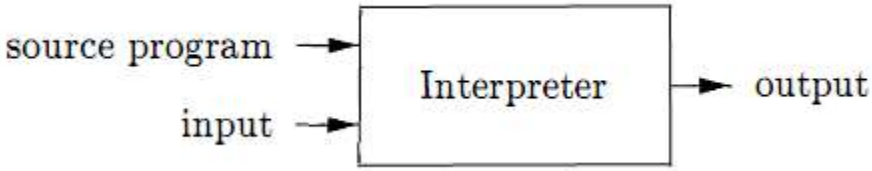
Language Processors

Simply stated, 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.



If the target program is an executable machine-language program, it can then be called by the user to process inputs and produce outputs.

An interpreter is another 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.

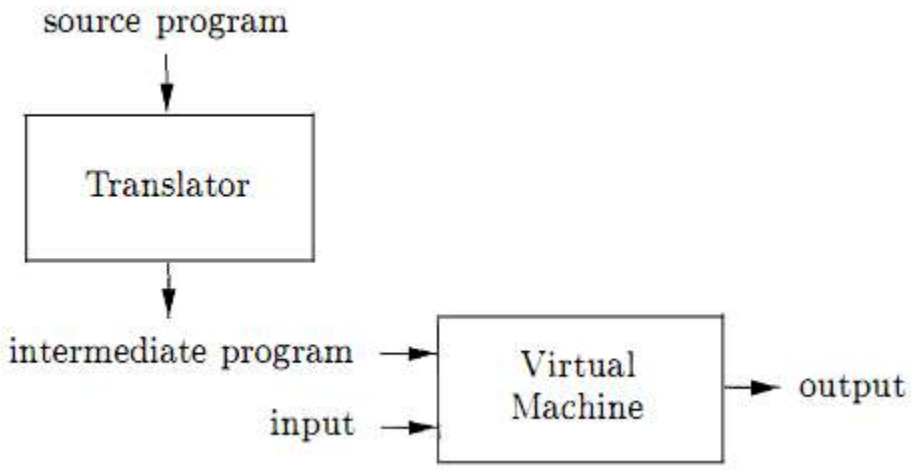


The machine-language target program produced by a compiler is usually much faster than an interpreter at mapping inputs to outputs. An interpreter, however, can usually give better error diagnostics than a compiler, because it executes the source program statement by statement.

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk Whatsapp# 0346-5100010

Example

Java language processors combine compilation and interpretation, as shown in Fig. A Java source program may first be compiled into an intermediate form called byte codes. The byte codes are then interpreted by a virtual machine. A benefit of this arrangement is that byte codes compiled on one machine can be interpreted on another machine, perhaps across a network. In order to achieve faster processing of inputs to outputs, some Java compilers, called **just-in-time** compilers, translate the byte codes into machine language immediately before they run the intermediate program to process the input.

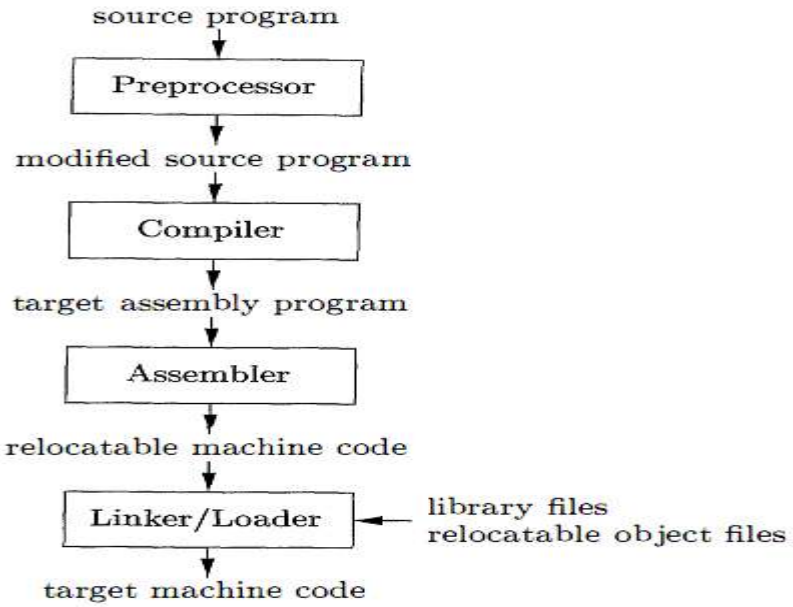


In addition to a compiler, several other programs may be required to create an executable target program. A source program may be divided into modules stored in separate files. The task of collecting the source program is sometimes entrusted to a separate program, called a preprocessor. The preprocessor may also expand shorthand, called macros, into source language statements.

The modified source program is then fed to a 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 the entire executable object files into memory for execution.

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk Whatsapp# 0346-5100010



Language Processor System

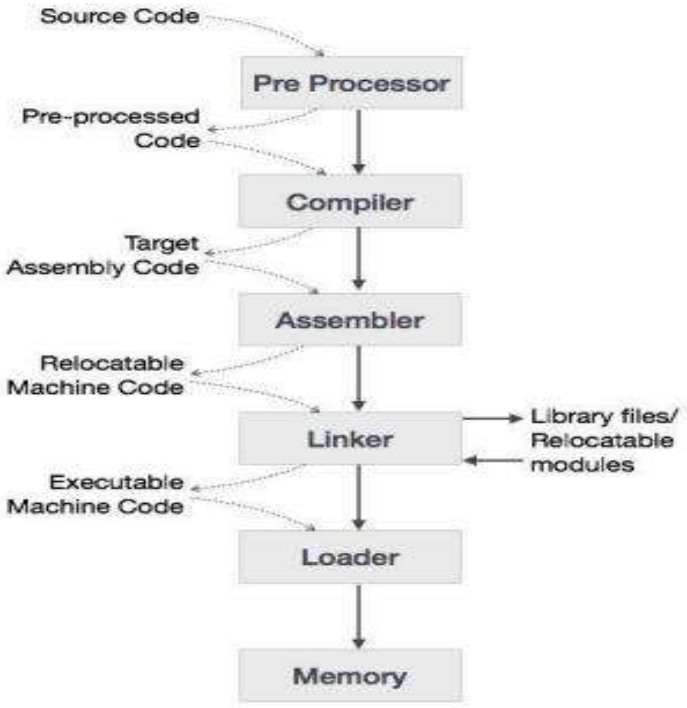
We have learnt that any computer system is made of hardware and software. The hardware understands a language, which humans cannot understand. So we write programs in high-level language, which is easier for us to understand and remember. These programs are then fed into a series of tools and OS components to get the desired code that can be used by the machine. This is known as Language Processing System.

The high-level language is converted into binary language in various phases. A **compiler** is a program that converts high-level language to assembly language. Similarly, an **assembler** is a program that converts the assembly language to machine-level language.
Let us first understand how a program, using C compiler, is executed on a host machine.

- User writes a program in C language (high-level language).
- The C compiler compiles the program and translates it to assembly program (low-level language).
- An assembler then translates the assembly program into machine code (object).
- A linker tool is used to link all the parts of the program together for execution (executable machine code).
- A loader loads all of them into memory and then the program is executed.

Before diving straight into the concepts of compilers, we should understand a few other tools that work closely with compilers.

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk Whatsapp# 0346-5100010



Preprocessor

A preprocessor, generally considered as a part of compiler, is a tool that produces input for compilers. It deals with macro-processing, augmentation; file inclusion, language extension, etc.

It operates on any line that begins with a pound/ hash sign. The preprocessor can only cut, copy and paste the text. It is part of compiler that runs before any code is compiled.
The symbol # includes macros and compiler specific directives e.g.

define
include

(i) File Inclusion:

It includes the library in your program and brings file before compilation.
e.g # include<stdlib.h>
include<math.h>

(ii) Macro Definition:

Macros provide a way of performing textual substitutions inline in your code.
Perform some duties as templates and inline functions. e.g

```
# define X 3
# define Y    A*B+C
 # define SQUARED (x)       x*x
main()
{
int y=2;
int S= SQUARED(y);
}
```

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk Whatsapp# 0346-5100010

Interpreter

An interpreter, like a compiler, translates high-level language into low-level machine language. The difference lies in the way they read the source code or input. A compiler reads the whole source code at once, creates tokens, checks semantics, generates intermediate code, executes the whole program and may involve many passes. In contrast, an interpreter reads a statement from the input converts it to an intermediate code, executes it, then takes the next statement in sequence. If an error occurs, an interpreter stops execution and reports it; whereas a compiler reads the whole program even if it encounters several errors.

Assembler

An assembler translates assembly language programs into machine code. The output of an assembler is called an object file, which contains a combination of machine instructions as well as the data required to place these instructions in memory.

Assembly Code

Assembly code is a mnemonic version of machine code in which names are used for operations and memory addresses instead of binary codes.
A mnemonic (memory aid) is any kind of trick we use to help us remember.

Example:
C=a+b

Assembly code

```
LD    r1 , a
LD    r2 , b
ADD   r3, r1, r2
ST    C , r3
```

Now what assembler will do with this code can be understand by two pass assembler.

Two Pass Assembler

Simplest form of assembler makes two passes over the input.

(i) First Pass:

In the first pass all the identifiers that denote memory location are found and stored in symbol table by giving them address of some bytes.

Identifier	Address
a	0
b	4
c	16

All identifiers are assigned to memory addresses.

(ii) Second Pass:

In the second pass assembler scans the input again. This time it translates each op code into sequence of bits, representing that operation in machine language.
It also translates each identifier into address of memory location.
So second pass produces relocatable machine code.

Operations	Registers	Tag	Addresses
0 0 0 1	01	00	0 0 0 0 0 0 1 0
0 0 0 1	10	00	0 0 0 0 0 1 0 0
0 0 1 1	11	00	0 0 0 0 0 0 0 0
0 0 1 0	11	00	0 0 0 1 0 0 0 0

Linker

Linker is a computer program that links and merges various object files together in order to make an executable file. All these files might have been compiled by separate assemblers. The major task of a linker is to search and locate referenced module/routines in a program and to determine the memory location where these codes will be loaded, making the program instruction to have absolute references.

Loader

Loader is a part of operating system and is responsible for loading executable files into memory and executes them. It calculates the size of a program (instructions and data) and creates memory space for it. It initializes various registers to initiate execution.

Cross-compiler

A compiler that runs on platform (A) and is capable of generating executable code for platform (B) is called a cross-compiler.

Source-to-source Compiler

A compiler that takes the source code of one programming language and translates it into the source code of another programming language is called a source-to-source compiler.