Module -I

Introduction to Compiling:

1.1 INTRODUCTION OF LANGUAGE PROCESSING SYSTEM

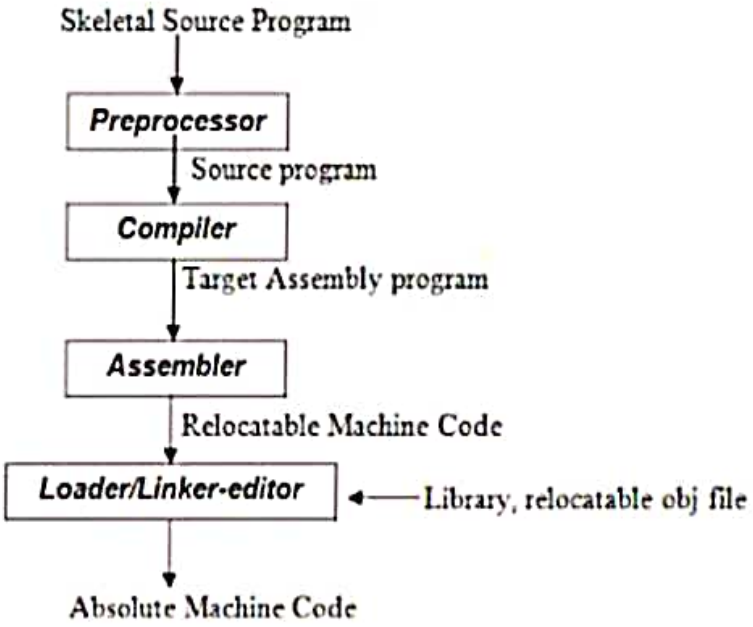

Fig 1.1: Language Processing System

Preprocessor

A preprocessor produce input to compilers. They may perform the following functions.

1. *Macro processing:* A preprocessor may allow a user to define macros that are short hands for longer constructs.
2. *File inclusion:* A preprocessor may include header files into the program text.
3. *Rational preprocessor:* these preprocessors augment older languages with more modern flow-of-control and data structuring facilities.
4. *Language Extensions:* These preprocessor attempts to add capabilities to the language by certain amounts to build-in macro

COMPILER

Compiler is a translator program that translates a program written in (HLL) the source program and translate it into an equivalent program in (MLL) the target program. As an important part of a compiler is error showing to the programmer.

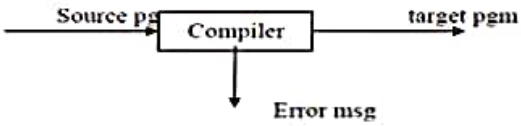

Fig 1.2: Structure of Compiler

Executing a program written n HLL programming language is basically of two parts. the source program must first be compiled translated into a object program. Then the results object program is loaded into a memory executed.

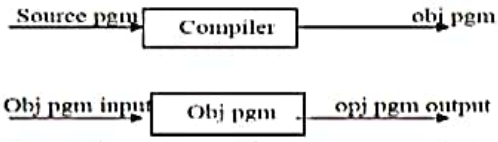

Fig 1.3: Execution process of source program in Compiler

ASSEMBLER

Programmers found it difficult to write or read programs in machine language. They begin to use a mnemonic (symbols) for each machine instruction, which they would subsequently translate into machine language. Such a mnemonic machine language is now called an assembly language. Programs known as assembler were written to automate the translation of assembly language in to machine language. The input to an assembler program is called source program, the output is a machine language translation (object program).

INTERPRETER

An interpreter is a program that appears to execute a source program as if it were machine language.

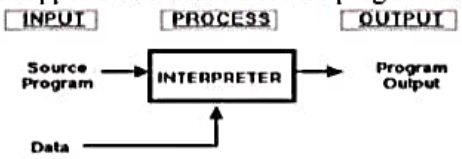

Fig1.4: Execution in Interpreter

Languages such as BASIC, SNOBOL, LISP can be translated using interpreters. JAVA also uses interpreter. The process of interpretation can be carried out in following phases.

1. Lexical analysis
2. Synatx analysis
3. Semantic analysis
4. Direct Execution

Advantages:

Modification of user program can be easily made and implemented as execution proceeds.
Type of object that denotes a various may change dynamically.
Debugging a program and finding errors is simplified task for a program used for interpretation.
The interpreter for the language makes it machine independent.

Disadvantages:

The execution of the program is *slower*.
Memory consumption is more.

LOADER AND LINK-EDITOR:

Once the assembler procedures an object program, that program must be placed into memory and executed. The assembler could place the object program directly in memory and transfer control to it,

thereby causing the machine language program to be execute. This would waste core by leaving the assembler in memory while the user's program was being executed. Also the programmer would have to retranslate his program with each execution, thus wasting translation time. To over come this problems of wasted translation time and memory. System programmers developed another component called loader

"A loader is a program that places programs into memory and prepares them for execution." It would be more efficient if subroutines could be translated into object form the loader could"relocate" directly behind the user's program. The task of adjusting programs o they may be placed in arbitrary core locations is called relocation. Relocation loaders perform four functions.

1.2 TRANSLATOR

A translator is a program that takes as input a program written in one language and produces as output a program in another language. Beside program translation, the translator performs another very important role, the error-detection. Any violation of d HLL specification would be detected and reported to the programmers. Important role of translator are:

1 Translating the HLL program input into an equivalent ml program.

2 Providing diagnostic messages wherever the programmer violates specification of the HLL.

1.3 LIST OF COMPILERS

1. Ada compilers
2. ALGOL compilers
3. BASIC compilers
4. C# compilers
5. C compilers
6. C++ compilers
7. COBOL compilers
8. Common Lisp compilers
9. ECMAScript interpreters
10. Fortran compilers
11. Java compilers
12. Pascal compilers
13. PL/I compilers
14. Python compilers
15. Smalltalk compilers

1.4 STRUCTURE OF THE COMPILER DESIGN

Phases of a compiler: A compiler operates in phases. A phase is a logically interrelated operation that takes source program in one representation and produces output in another representation. The phases of a compiler are shown in below

There are two phases of compilation.

a. Analysis (Machine Independent/Language Dependent)

b. Synthesis(Machine Dependent/Language independent)

Compilation process is partitioned into no-of-sub processes called **'phases'**.

Lexical Analysis:-

LA or Scanners reads the source program one character at a time, carving the source program into a sequence of automic units called **tokens.**