1. Explain overview of translation process.

- A translator is a kind of program that takes one form of program as input and converts it into another form.
- The input is called *source program* and output is called *target program.*
- The source language can be assembly language or higher level language like C, C++, FORTRAN, etc...
- There are three types of translators,
    1. Compiler
    2. Interpreter
    3. Assembler

2. What is compiler? List major functions done by compiler.

- A compiler is a program that reads a program written in one language and translates into an equivalent program in another language.

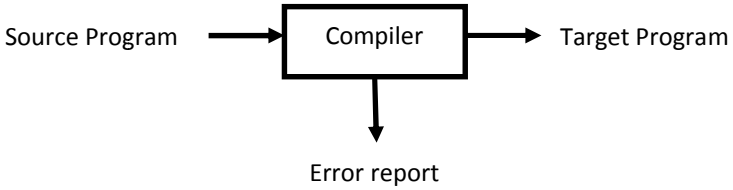


Fig.1.1. A Compiler

Major functions done by compiler:

- Compiler is used to convert one form of program to another.
- A compiler should convert the source program to a target machine code in such a way that the generated target code should be easy to understand.
- Compiler should preserve the meaning of source code.
- Compiler should report errors that occur during compilation process.
- The compilation must be done efficiently.

3. Write the difference between compiler, interpreter and assembler.

1. Compiler v/s Interpreter

No.	Compiler	Interpreter
1	Compiler takes entire program as an input.	Interpreter takes single instruction as an input.
2	Intermediate code is generated.	No Intermediate code is generated.
3	Memory requirement is more.	Memory requirement is less.
4	Error is displayed after entire program is checked.	Error is displayed for every instruction interpreted.
5	Example: C compiler	Example: BASIC

Table 1.1 Difference between Compiler & Interpreter

2. Compiler v/s Assembler

No.	Compiler	Assembler
1	It translates higher level language to machine code.	It translates mnemonic operation code to machine code.
2	Types of compiler, • Single pass compiler • Multi pass compiler	Types of assembler, • Single pass assembler • Two pass assembler
3	Example: C compiler	Example: 8085, 8086 instruction set

Table 1.2 Difference between Compiler & Assembler

4. Analysis synthesis model of compilation. OR Explain structure of compiler. OR Explain phases of compiler. OR Write output of phases of a complier. for a = a + b * c * 2; type of a, b, c are float

There are mainly two parts of compilation process.

1. **Analysis phase**: The main objective of the analysis phase is to break the source code into parts and then arranges these pieces into a meaningful structure.
2. **Synthesis phase**: Synthesis phase is concerned with generation of target language statement which has the same meaning as the source statement.

Analysis Phase: Analysis part is divided into three sub parts,

I. Lexical analysis
II. Syntax analysis
III. Semantic analysis

Lexical analysis:

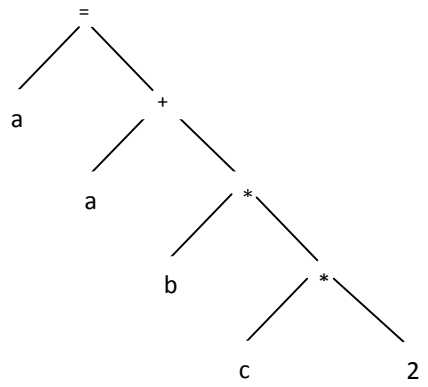
- Lexical analysis is also called linear analysis or scanning.
- Lexical analyzer reads the source program and then it is broken into stream of units. Such units are called token.
- Then it classifies the units into different lexical classes. E.g. id's, constants, keyword etc...and enters them into different tables.
- For example, in lexical analysis the assignment statement **a: = a + b * c * 2** would be grouped into the following tokens:

a	Identifier 1
=	Assignment sign
a	Identifier 1
+	The plus sign
b	Identifier 2
*	Multiplication sign
c	Identifier 3
*	Multiplication

	sign
2	Number 2

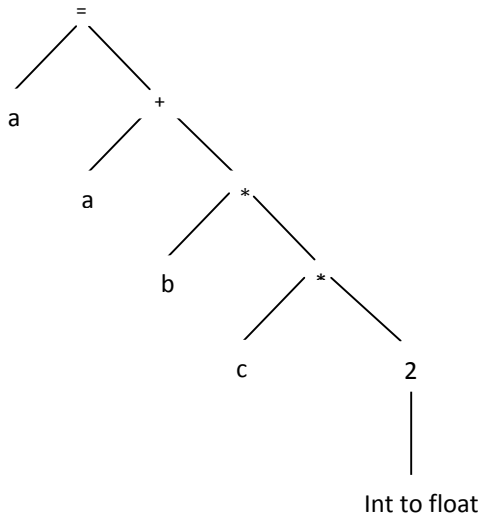
Syntax Analysis:

- Syntax analysis is also called hierarchical analysis or parsing.
- The syntax analyzer checks each line of the code and spots every tiny mistake that the programmer has committed while typing the code.
- If code is error free then syntax analyzer generates the tree.



Semantic analysis:

- Semantic analyzer determines the meaning of a source string.
- For example matching of parenthesis in the expression, or matching of if..else statement or performing arithmetic operation that are type compatible, or checking the scope of operation.



Synthesis phase: synthesis part is divided into three sub parts,

I. Intermediate code generation
II. Code optimization
III. Code generation

Intermediate code generation:

- The intermediate representation should have two important properties, it should be

easy to produce and easy to translate into target program.

- We consider intermediate form called “three address code”.
- Three address code consist of a sequence of instruction, each of which has at most three operands.
- The source program might appear in three address code as,

```
t1= int to real(2)
t2= id3 * t1
t3= t2 * id2
t4= t3 + id1
id1= t4
```

Code optimization:

- The code optimization phase attempt to improve the intermediate code.
- This is necessary to have a faster executing code or less consumption of memory.
- Thus by optimizing the code the overall running time of a target program can be improved.

```
t1= id3 * 2.0
t2= id2 * t1
id1 = id1 + t2
```

Code generation:

- In code generation phase the target code gets generated. The intermediate code instructions are translated into sequence of machine instruction.

```
MOV id3, R1
MUL #2.0, R1
MOV id2, R2
MUL R2, R1
MOV id1, R2
ADD R2, R1
MOV R1, id1
```

Symbol Table

- A **symbol table** is a data structure used by a language translator such as a compiler or interpreter.
- It is used to store names encountered in the source program, along with the relevant attributes for those names.
- Information about following entities is stored in the symbol table.
  - ✓ Variable/Identifier
  - ✓ Procedure/function
  - ✓ Keyword
  - ✓ Constant
  - ✓ Class name
  - ✓ Label name

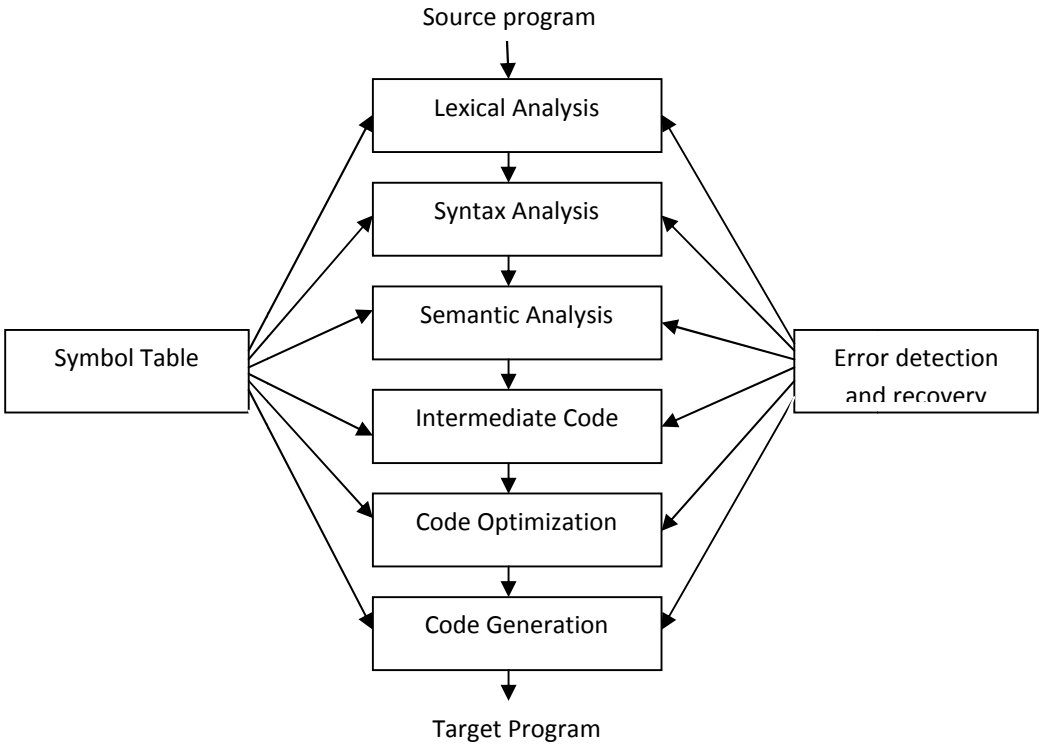


Fig.1.2. Phases of Compiler

5. The context of a compiler. OR Cousins of compiler. OR What does the linker do? What does the loader do? What does the Preprocessor do? Explain their role(s) in compilation process.

- In addition to a compiler, several other programs may be required to create an executable target program.

Preprocessor

Preprocessor produces input to compiler. They may perform the following functions,

1. Macro processing: A preprocessor may allow user to define macros that are shorthand for longer constructs.
2. File inclusion: A preprocessor may include the header file into the program text.
3. Rational preprocessor: Such a preprocessor provides the user with built in macro for construct like while statement or if statement.
4. Language extensions: this processors attempt to add capabilities to the language by what amount to built-in macros. Ex: the language equal is a database query language embedded in C. statement beginning with ## are taken by preprocessor to be database access statement unrelated to C and translated into procedure call on routines that perform the database access.

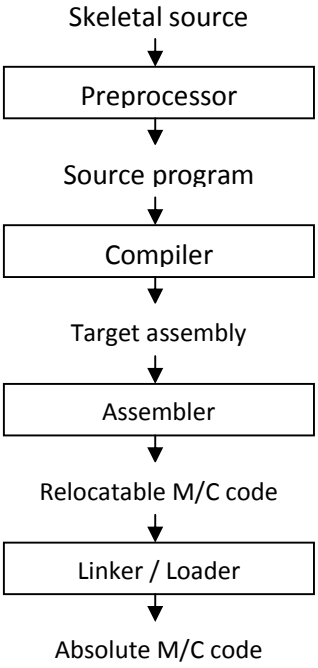


Fig.1.3. Context of Compiler

Assembler

Assembler is a translator which takes the assembly program as an input and generates the machine code as a output. An assembly is a mnemonic version of machine code, in which names are used instead of binary codes for operations.

Linker

Linker allows us to make a single program from a several files of relocatable machine code. These file may have been the result of several different compilation, and one or more may be library files of routine provided by a system.

Loader

The process of loading consists of taking relocatable machine code, altering the relocatable address and placing the altered instructions and data in memory at the proper location.

6. Explain front end and back end in brief. (Grouping of phases)

- The phases are collected into a front end and back end.

Front end

- The front end consist of those phases, that depends primarily on source language and largely independent of the target machine.
- Front end includes lexical analysis, syntax analysis, semantic analysis, intermediate code generation and creation of symbol table.
- Certain amount of code optimization can be done by front end.

Back end

- The back end consists of those phases, that depends on target machine and do not depend on source program.

- Back end includes code optimization and code generation phase with necessary error handling and symbol table operation.

7. What is the pass of compiler? Explain how the single and multi-pass compilers work? What is the effect of reducing the number of passes?

- One complete scan of a source program is called pass.
- Pass include reading an input file and writing to the output file.
- In a single pass compiler analysis of source statement is immediately followed by synthesis of equivalent target statement.
- It is difficult to compile the source program into single pass due to:
- **Forward reference:** a forward reference of a program entity is a reference to the entity which precedes its definition in the program.
- This problem can be solved by postponing the generation of target code until more information concerning the entity becomes available.
- It leads to multi pass model of compilation.
- In Pass I: Perform analysis of the source program and note relevant information.
- In Pass II: Generate target code using information noted in pass I.

Effect of reducing the number of passes

- It is desirable to have a few passes, because it takes time to read and write intermediate file.
- On the other hand if we group several phases into one pass we may be forced to keep the entire program in the memory. Therefore memory requirement may be large.

8. Explain types of compiler. OR Write difference between single pass and multi pass compiler.

Single pass compiler v/s Multi pass Compiler

No.	Single pass compiler	Multi pass compiler
1	A one-pass compiler is a compiler that passes through the source code of each compilation unit only once.	A multi-pass compiler is a type of compiler that processes the source code or abstract syntax tree of a program several times.
2	A one-pass compiler is faster than multi-pass compiler.	A multi-pass compiler is slower than single-pass compiler.
3	One-pass compiler are sometimes called narrow compiler.	Multi-pass compilers are sometimes called wide compiler.
4	Language like Pascal can be implemented with a single pass compiler.	Languages like Java require a multi-pass compiler.

Table 1.3 Difference between Single Pass Compiler & Multi Pass Compiler

9. Write the difference between phase and pass.

Phase v/s Pass

No.	Phase	Pass
1	The process of compilation is carried out in various step is called phase.	Various phases are logically grouped together to form a pass.
2	The phases of compilation are lexical analysis, syntax analysis, semantic analysis, intermediate code generation, code optimization and code generation.	The process of compilation can be carried out in a single pass or in multiple passes.

Table 1.3 Difference between Phase & Pass