Name: M. N. KHAN
ID : 2216080041
8th Batch (EV)
CSE
course : compiler
Course code: CSI411

Ans to the Q: NO: 01

(a) Ans: Types of Complier:

There are various types of compliers which are as follows: -

Traditional Compilers (C, C++ & Pascal):

These compilers transform a source program in an HLL into its similar in negative machine program or object programme.

Interpreters (LISP, SNOBOL, & Java 1.0) :-

These compilers first convert source code into intermediate code, & then interprets it to equivalent machine code.

Cross compilers: These are the compilers that run on one machine & make code for another machine.

Incremental compailers: Incremental compailer is a compailer, which executes that recompailation of only a changed source instead of compiling the complete source code.

converters(eg, COBOL to c++): These programs will be compiling from one high-level language to another.

Just-In-Time compilers (Java, Microsoft .NET): These are the run time compilers from intermediate language (byte code, MSIL) to executable code or negative machine code.

Single pass compiler: In a single pass compiler when a line source in processed it is scanned & the tokens are extracted.

Multi-pass compiler: The compiler scans the input source once & makes the first modified structure, therefore scans the first produced form & makes a second modified structure.

Ahead of time (AOT) compilers: These are the pre-Compilers to the native code ~~from~~ Java & NET.

Binary compilation: These compilation will be compiling the object code of another platform.

Phases of a Compiler:

A compiler is a software program that converts the high-level source code written in a programming language into low-level machine code that can be executed by the computer hardware.
The typical phases of a compiler are:

1. Lexical Analysis:-

The first phase of a compiler is lexical analysis also known as scanning. This phase reads the source code & breaks it into a stream of tokens which are the basic units of the programming language.

2. Syntax Analysis:

The second phase of a compiler is syntax analysis also known as parsing. The output of this phase is usually an Abstract syntax tree (AST)

3. Semantic Analysis:

The third phase of a compiler is semantic analysis. This phase is checked whether the code is semantically correct. The compiler also checks for other semantic errors, such as undeclared variables & incorrect function calls.

4. Intermediate code Generation:

This fourth phase of a compiler is intermediate code generation. This phase generation an intermediate representation of the source code that can be easily translate into machine code.

5. Optimization:

The fifth phase of a compiler is optimization. this phase are Applies various optimizations techniques to the intermediate code to improve the performance of the generated machine code.

6. Code Generation:

The final phase of a compiler is code generation this phase takes the optimized intermediate code & generates the actual machine code that can be executed by the target hardware to be installed in their system, for executing the shared executable file of your source code.

(1/b) Advantage of compiler: There are several advantages to using a compiler:-

1. Improve performance: Compiled code tends to run faster than interpreted code because it has been translated into machine code that can be directly executed by the computer's processor.

2. Portability: Compilers allow programmers to write code in a high-level programming language that can be easily translated into machine code for a variety of different platforms.

3. Increased security: Compilers can help improve the security of software by performing a number of checks on the source code.

4. debugging support: Most compilers include a number of debugging tools that can help programmers find & fix errors in their code.

5. No dependencies: Your client or anyone else doesn't need any compiler, interpeter or third party programme.

(1/b) Disadvantage of compiler:

There are a few potential disadvantage of using a compiler in software development:

1. Compilation time: Depending on the size & complexity of the source code, compilation can take a significant amount of time.

2. Error detection: Compilers scan only detect syntax error & certain semantic errors & may not catch all errors in the source code.

3. Portability: Programs compiled from high level language may not be as fast able to run other platform

4. Execution speed: Programs compiled from high level language may not be as fast as programs written in low-level additional instructions for the compiler to interpret.

5. Lack of flexibility: Compilers can limit the flexibility of programme since changes often requires recompilation.

(1/c) Why to learn compiler design:

Compiler design helps full implimentation of high level progra mming languages. support optimization for computer architecture parallelism. It translates a high-level language into a low-level language.

Top down parsing:

Top Down parsing searches for a production rule to be used to contract a string.

Top down approach starts evaluating the parse tree from the top & move downwards for parsing other nodes.

Top down parsing uses leftmost derivation.

Bottom - up parsing: Bottom up parsing searches for a production rule to be used to reduce a string to get a starting symbol of grammer.

Bottom - up parsing attempts to reduce the input string to first symbol of the grammar.

Bottom up parsing uses the rightmost derivation

Ans: to the Q:NO: 02

(a) Language Processing System: The Computer is an intelligent combination of software & hardware. The hardware consider considers instructions as electronic charge, which is equivalent to the binary language in software programming. The binary language has only os & 1s. To enlighten, the hardware code has to be written in binary format, which is just a series of os & 1s. this is known as a language processing system.

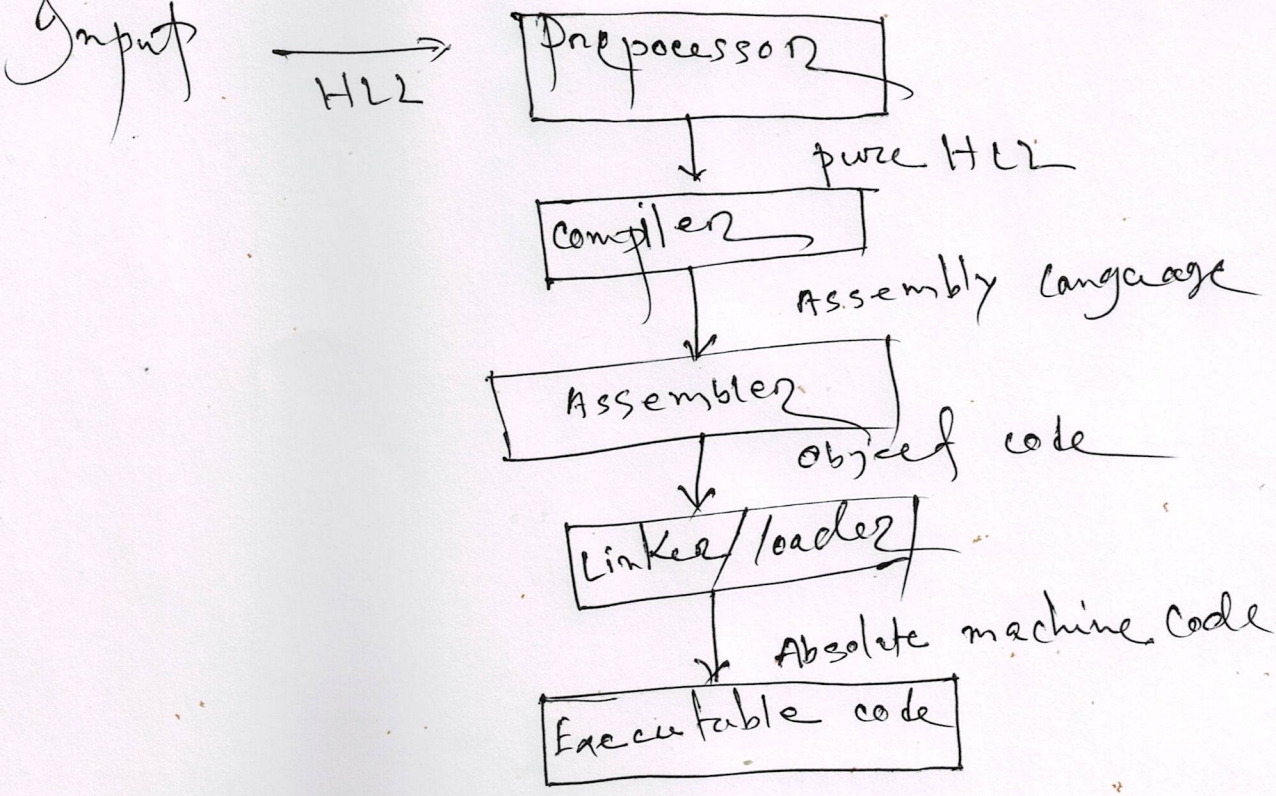
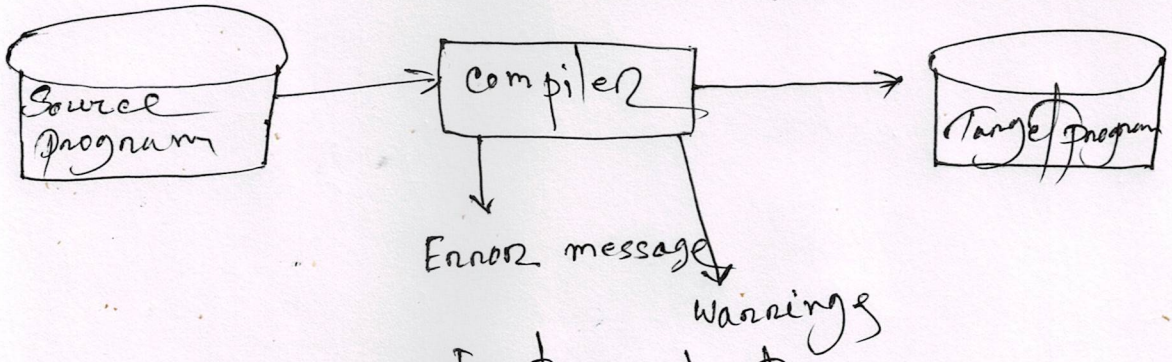


Fig: Language Processing System.

Preprocessor:
It includes all header files & also evaluates whether a macro is a piece of code that is given a name.

Compiler: the compiler takes the modified code as input & produces the target code as input & produces the target modified code as ~~input~~ output.



Input - Output

Assembler: The assembler takes the target code as input & produces real locatable machine as output.

linker: Linker or link editor is a program that takes a collection of objects & combines them into an executable program.

Loader: The loader keeps the linked program in the main memory.

executable code: It is low level & machine-specific code that the machine can easily understand.

(b) Cross compiler: A Cross compiler is a compiler capable of creating executable code for a platform other than the one on which the compiler is running. For example, a compiler that runs on a PC but generates code that runs on an Android smartphone is a cross compiler.

(c) Source compiler: A source to source compiler (S2S compiler) is also referred to by three name, the first is the source to source translator, the second is trans compiler & the third one is transpiler.

A S2S compiler is given as input the source code of a program to which it returns a source code with the overall same functionality in the same or different programming language.