Page No. 08
Date 25 01 22

TUTORIAL 3

U19CS012

1.) Describe in detail about any 4 system software.

(a) OPERATING SYSTEM - Harness communication between hardware, system programs and other applications.

(b) TRANSLATOR - Translates high-level languages to low-level machine code.

(c) DEVICE DRIVERS - Enable [Device] communication with OS

(d) UTILITY - Ensures optimum functionality of devices and application.

(A) OPERATING SYSTEM

① OS is a type of system software that manages computer's hardware and software resources. [Acts as a link between software and hardware]

② It controls and keeps a record of the execution of all other programs that are present in the computer, including application programs & other system software.

③ Task's performed by OS

(i) Memory Management: The OS keep tracks of the primary memory and allocates the memory when a process requests it.

(ii) Processor Management: Allocates the main memory (RAM) to a process and de-allocates it when it is no longer required.

(iii) File management: Allocates and de-allocates the resources and decides who get the resources.

(iv) Security: Prevents unauthorised access to programe and data by means of passwords.

(v) Error detecting Aids: Production of dumps, traces, error messages

(vi) Scheduling : The OS schedules proces through its scheduling algorithms.

④ Examples of OS

• Windows 10	Network/server OS	Internet/web OS	mobile OS
• Mac OS X	→ Ubuntu server	→ Chrome OS	• iPhone OS
• Ubuntu	→ Windows server	→ Club Linux	• Android OS
	→ Red Hat enterprise	→ Remix OS	• Windows Phone OS

(B) TRANSLATORS

① These are intermediate programs relied on by software programmers to translate high-level language source code to machine language code.

② ~~Pro~~ Popular translator Languages = Compilers, Assemblers and Interpreters. They are usually designed by computer manufacturers.

③ Machine code is written in number system of base-2, [0's & 1's] This is lowest level language possible, seeming meaningless to humans, zeros and ones are actually sequenced intelligently by the processor to refer to every conceivable human code.

④ Translators helps in various design tasks :-

(i) Identify syntax errors during translation, thus allowing changes to be made to the code

(ii) Provide diagnostic reports whenever the code rules are not followed.

(iii) Allocate data storage for the program.

(iv) List both source code & program details.

U19CS012

(C) DEVICE DRIVERS

① Devia Drivers is a type of system software which brings computer devices and peripherals to life.
② Drivers are very essential for a computer system to work properly because without device driver the particular hardware fails to work accordingly.
③ Drivers make it possible for all connected components and external add ons to perform their intended tasks and directed by OS.

④ Examples of Devices which requirce driver:
- Mouse
- Keyboard
- Sound card
- Display card
- Printer.

(D) UTILITIES

① Utility are types of system software which sits between system and application software. These are programs intended for diagnostic and maintenance tasks for the computer.
② Examples and Feature of utility software include:
① Antivirus and Security Software for security of files & application
② Disk partition services [Windows disk partioner]
③ Disk defragmentation to organize scattered files on drive
Eg Disk defragmenter
④ File compression to optimize disk space (WinRAR)
⑤ Data Backup & Data Recovery
⑥ Firewall for protection eg: Windows Firewall.

2> Define assembler, compiler and interpreter.

2>

1> COMPILER

source code → Compiler → Object code

① The Language Processor that reads the complete source program written in high-level language as a whole in one-go and translates it into an equivalent program in machine language is called compiler.

② In compiler, the source code is translated to object code successfully, if it is free of errors. [Eg: C, C++, C#, Java]

2> ASSEMBLER

Assembly code → Assembler → Machine Code

① Assembler is a program to convert instructions in low-level assembly code into relocatable machine code.

② It generates instructions by evaluating the mnemonics (symbols) in the operation field and find the value of symbols and literals to produce machine code. [ADD, MUL, SUB, DIV, MOV]

③ Mnemonics depends upon the architecture of the machine.

for eg, the architecture of intel 8085 & intel 8086 are different

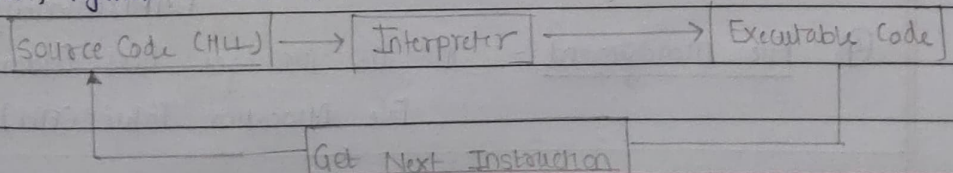
3> INTERPRETER

① The software by which the conversion of high level instructions is performed line-by-line to machine level language, other than compiler and assembler is known as interpreter.

② If an error is found on any line, the execution stops till it is corrected.

③ It translates source code into some efficient intermediate representation and immediately executes this.

Eg: Perl, Python & matlab.



3.) What is System Software? Give some application of operating system.

3) ① System Software is a set of programs that control and manage the operations of computer hardware. It also helps application programs to execute correctly.

② System Software are designed to control the operation and extend the processing functionalities of a computer system.

③ System Software makes the operation of computer more fast, effective and secure. Eg: operating system, Drivers, Utility S/w.

Applications of operating system

① Processor Management - management of CPU + allotment of CPU time to different processes + Scheduling techniques like SJF, ~~RRS~~ Round Robin, Priority Based Scheduling.

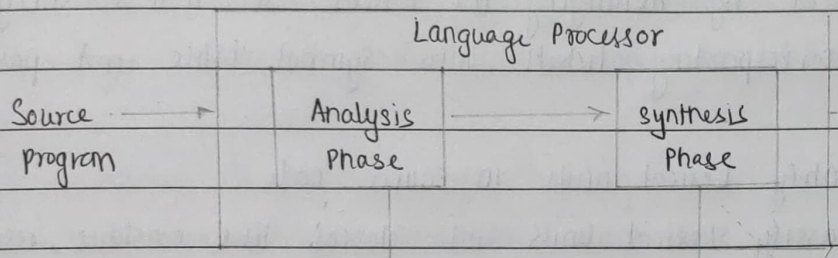
② Device Management - CPU processing speed is much higher than I/O devices. The OS communicates betwn hardware and attached devices and maintains balance between them and CPU by using 2 techniques ① Buffering ② Spooling (Simultaneous Peripheral operation on line)

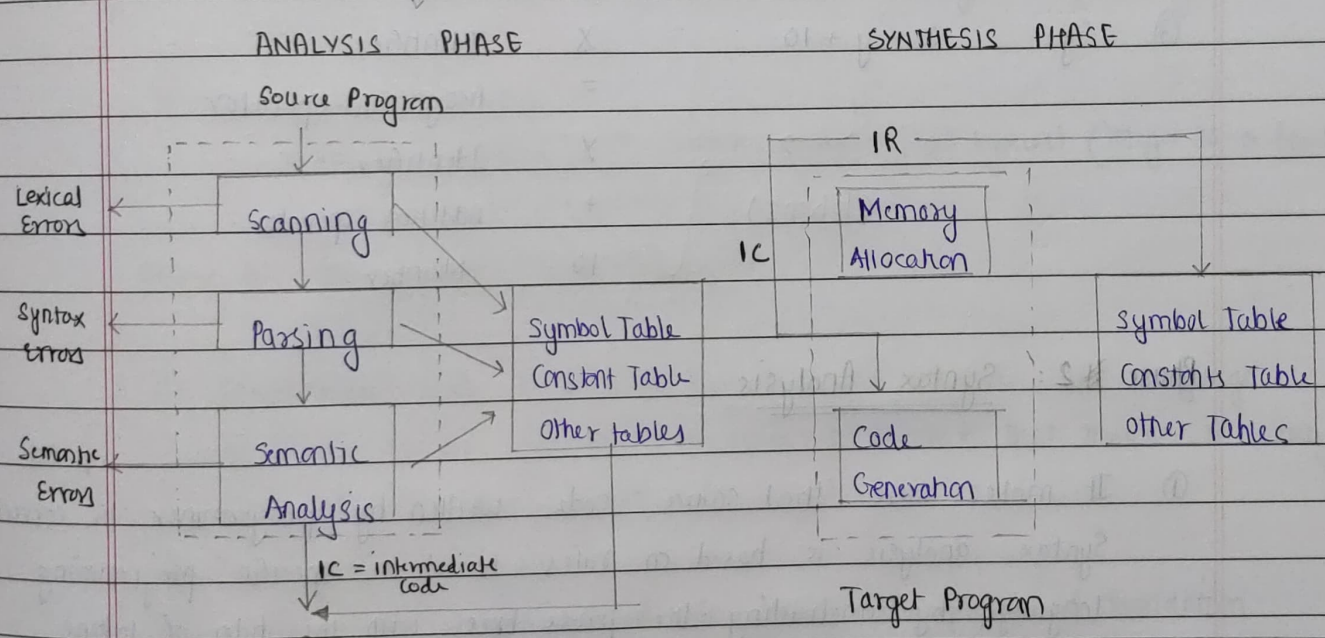
③ Memory Management - In computer, both CPU & I/O devices interact with memory. When program needs to be executed, it is loaded into main memory till the execution is completed. (2 techniques → [•] Partitioning • Virtual Memory)

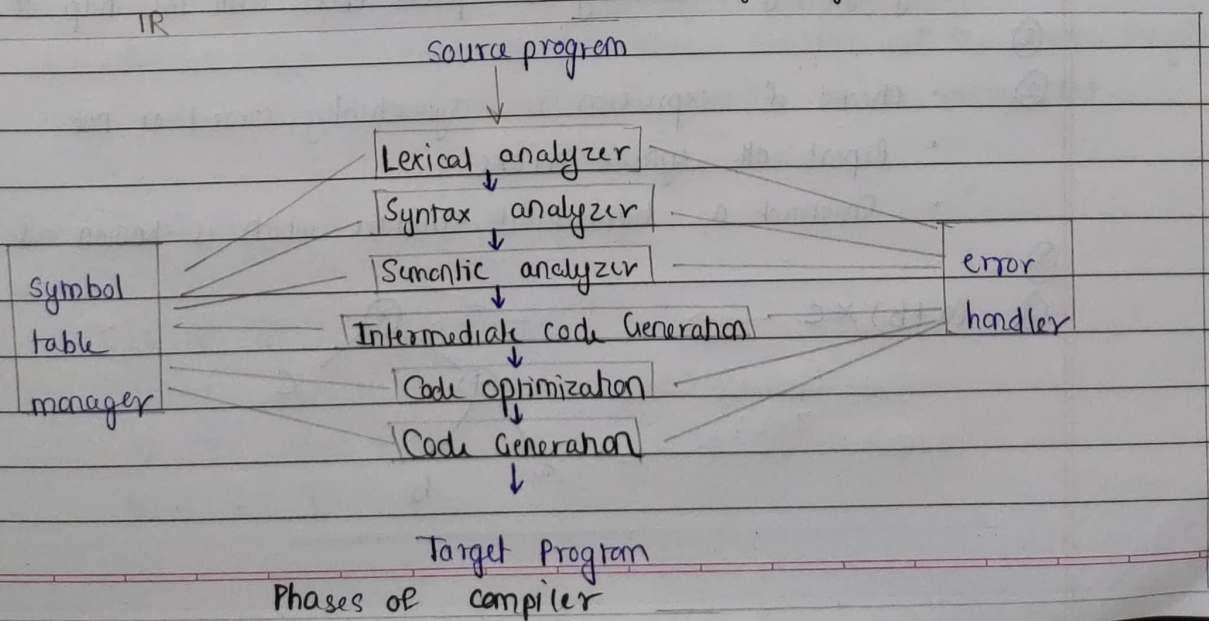
④ File Management - The OS manages files, folder and directory System. File Allocation Table (FAT)

4.) What are two part of compilation? Explain the various phases of compiler.

4.) Analysis and Synthesis are two parts of compilation.







Phases of compiler

Phase 1 : Lexical Analysis

① Character stream from source program is grouped in meaningful sequences by identifying the tokens. It makes entry of the corresponding tokens into symbol table and pass token to next phase.

② · Identify Lexical units in source code
· Classify Lexical units into classes like constants, reserved word
· Identify token which is not part of language. (ie ignore comments)

③ Eg: $x = y + 10$ → (Tokens)

X	Identifier
=	Assignment operator
Y	Identifier
+	Addition operator
10	Number

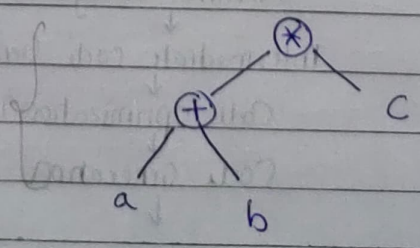
Phase 2 : Syntax Analysis

① It makes sure that source code written by programmer is correct or not. Syntax analysis is based on rules based on specific programming language by constructing the parse tree with the help of tokens.

②
· Checks if expression is syntactically correct or not
· Report all syntax errors
· Construct a hierarchical structure which is known as parse tree

③

$(a+b) * c$ → (Parse Tree)



Phase 3: Semantic Analysis

① It checks the semantic consistency of code. It also checks whether code is conveying an appropriate meaning. It will check for Type mismatch, incompatible operands, function called with improper arguments

② • Allows to perform type checking
• In case of type mismatch, where there are no exact type correction rules which satisfy the desired operation, a semantic error is shown.
• Collects type information and checks for type compatibility

③ float x = 20.2;
float y = (x) * 30; → Semantic Analyzer will typecast (integer 30 → float 30.0)

Phase 4: Intermediate Code Generation

① Intermediate code is between high level and machine level language. It needs to be generated in such a manner that makes it easy to translate it into target code.

② • Holds the values computed during the process of translation
• Allows you to maintain precedence ordering of the source language
• holds the correct number of operands of the instructions.

③ total = count + rate * 5

Intermediate Code:
t1 := int_to_float(5)
t2 := rate * t1
t3 := count + t2
total := t3

U19CS012

Phase 5: Code Optimization

① This phase removes unnecessary code line and arranges the sequence of statements to speed up the execution of the program without wasting resources. Goal = Improve (I.C.) { to run faster + occupy less space }

② • Improves the running time of target program
• Generates streamline code still in intermediate representation
• Removing unreachable code & getting rid of unused variable
• Removing statements which are not altered from loop.

③

```
a = intofloat(10)          (Code optimized)
b = c * a          }       b = c * 10.0
d = e + b          }       f = e + b
f = d
```

Phase 6: Code Generation

① It takes input from code optimization and produces page code/ object code as a result. Objective is to allocate storage and generate relocatable machine code.

② All memory locations and registers are also selected and allotted during this phase

Eg: `a = b + 60.0`
would be translated to registers

```
MOVF  a, R1
MULF  #60.0, R2
ADDF  R1, R2
```

5.> Differentiate Tokens, Patterns, Lexeme.

5.>

Token : ① a name for a set of input strings with related structure.

② * A sequence of characters that have collective meaning.

③ Treated as single logical entity.

⑤ constants

eg : Typical tokens are ① Identifier ② keywords ③ operators ④ special symbols

Pattern : ① a rule describing the set of strings associated with a token.

② A set of string in the input for which same token is produced as output. This set of string is described by a rule called a pattern associated with the token.

③ Example " a letter followed by zero or more letters/digits, or underscores "

Lexeme : ① the actual input string that matches a pattern.

② It is sequence of characters in the source program that is matched by pattern for a token.

③ example : "count"

Example : C language statement [printf ("Total = %d \n", score);]

① printf & score are lexemes match the pattern for token id, and "Total = %d \n" → lexeme matching literal.