Edo State polytechnic, usen

School of Applied Sciences

Computer Science Department

Com212 systems programming

Question 1

(a) Explain in your own term systems programming and give three example

(b) Explain the concept of application programming, give five examples

Question 2

(a) Explain in details Operating Systems, list ten operating systems

(b) Explain why operating system is referred to as a resource manager

Question 3

(a) Explain in details an assembler

(b) Give four reasons why assembly language is important to learn and also give four advantages of assembly language

Question 4

(a) Write the format of an assembly language

(b) How do you handle errors in assembly language

Question 5

(a) What is a compiler?

(b) Explain the compilation process

Question 6

With the aid of a diagram, explain the phases of a compiler

Question 7

(a) What does a semantic analysis do?
(b) List the various error recovery strategies for a lexical analysis.
(c) Explain the concept of YACC

Edo State polytechnic, usen
School of Applied Sciences
Computer Science Department
Com212 systems programming

SOLUTION

Question 1

(a) System programming involves designing and writing computer programs that allow the computer hardware to interface with the programmer and the user, leading to the effective execution of application software on the computer system. Typical system programs include the operating system and firmware, programming tools such as compilers, assemblers, I/O routines, interpreters, scheduler, loaders and linkers as well as the runtime libraries of the computer programming languages.

(b)

An application program is a comprehensive, self-contained program that performs a particular function directly for the user.

- Examples: Email
- Web browsers
- Games
- Word processors
- Enterprise software
- Accounting software

Graphics software

Question 2

(a) An **operating system** is the most important **software** that runs on a computer. It manages the computer's memor**y** and **processes**, as well as all of its **software** and **hardware**. It also allows you to **communicate** with the computer without knowing how to speak the computer's language. Without an operating system, a computer is useless.

Examples: windows NT, windows Millennium, windows Vista, windows 7, windows 8, windows 10, etc.

(b)

The operating system is resources manager; it manages all the activities of the OS, like processing scheduling, batching, the kernel process, memory allocation, IN/OUT operations, sending jobs to the printer, etc.

Question 3

(a) An assembler is a program that converts assembly language into machine code. It takes the basic commands and operations from assembly code and converts them into binary code that can be recognized by a specific type of processor.Assemblers are similar to compilers in that they produce executable code. However, assemblers are more simplistic since they only convert low-level code (assembly language) to machine code. Since each assembly language is designed for a specific processor, assembling a program is performed using a simple one-to-one mapping from assembly code to machine code. Compilers, on the other hand, must convert generic high-level source code into machine code for a specific processor.

(b)

- Assembly language lets you talk to a computer in its "native tongue"
- All computer programs actually run in machine language
- High level languages such as C must be translated (compiled) into machine language
- Assembly language translates directly into machine language

Advantages of Assembly Language

- Low-level access to the computer
- Higher speed
- Total control over CPU
- (Must know what you are doing in order to make these advantages work)

Question 4

(a)

Assembly language programs divide roughly into five sections

- header
- equates

- data
- body
- closing

(b)

Dealing with Errors

- **TASM** will report the line number and give an error message for each error it finds
- Sometimes it is helpful to have a listing file (**.lst**), created by using **TASM** with the **-l** option
- The **.lst** file contains a complete listing of the program, along with line numbers, object code bytes, and the symbol table

Using the Debugger

- Useful for logic errors that the assembler misses
- See the text for a complete tutorial
- You do **not** need to use the TDH386.SYS driver or the TD386.EXE debugger with the latest version of the assembler
- To use the debugger on myprog.asm

Question 5

(a)

A compiler is a special program that processes statements written in a particular programming language and turns them into machine language or "code" that a computer's processor uses. Typically, a programmer writes language statements in a language such as Pascal or C one line at a time using an editor. The file that is created contains what are called the source statements. The programmer then runs the appropriate language compiler, specifying the name of the file that contains the source statements.

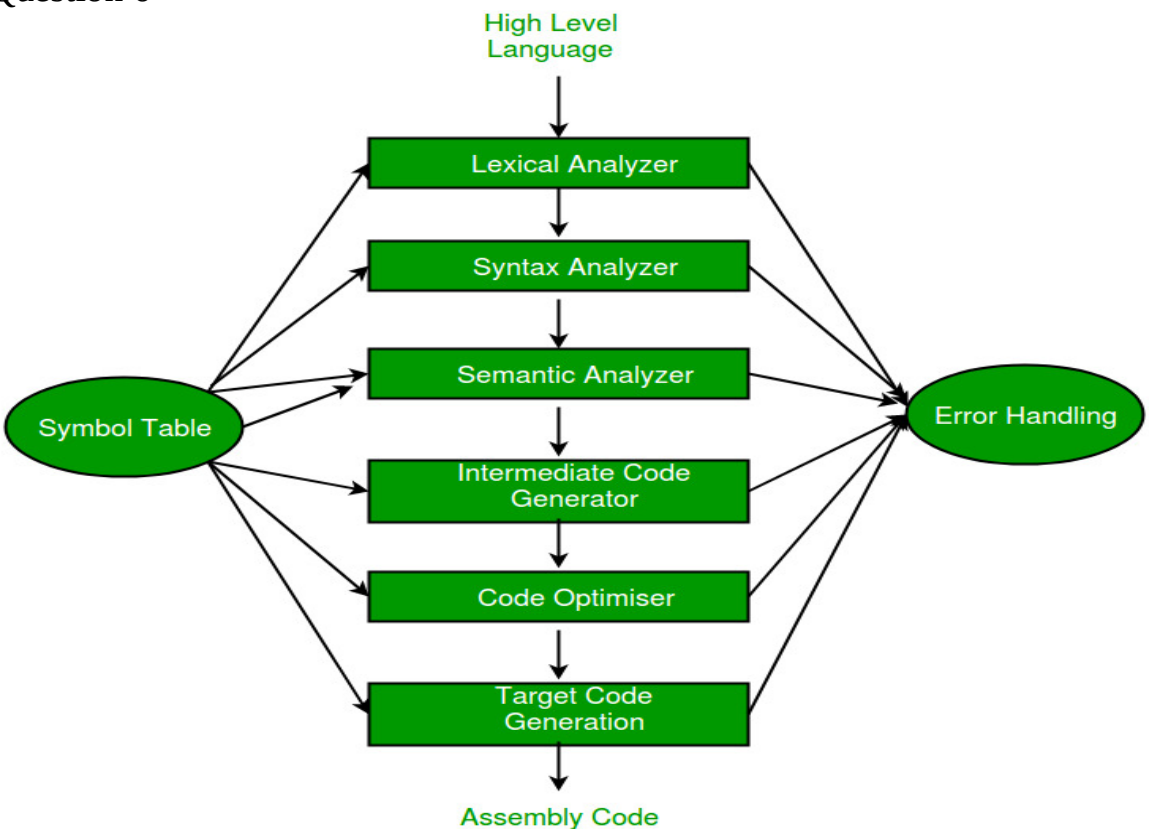
(b)

↓
Preprocessor
↓
Source program
↓
Compiler

↓
Target assembly program
↓
Assembler
↓
Relocatable machine code
↓
Loader/ link editor ←library, relocatable object files
↓
Absolute machine code

Analysis and Synthesis are the two parts of compilation.
The analysis part breaks up the source program into constituent pieces and creates an intermediate representation of the source program.
The synthesis part constructs the desired target program from the intermediate representation.

Question 6



Lexical Analyzer – It reads the program and converts it into tokens. It converts a stream of lexemes into a stream of tokens. Tokens are defined by regular expressions which are understood by the lexical analyzer. It also removes white-spaces and comments.
Syntax Analyzer – It is sometimes called as parser. It constructs the parse tree. It takes all the tokens one by one and uses Context Free Grammar to construct the parse tree.
The rules of programming can be entirely represented in some few productions. Using these productions we can represent what the program actually is. The input has to be checked whether it is in the desired format or not.Syntax error can be detected at this level if the input is not in accordance with the grammar.
Semantic Analyzer – It verifies the parse tree, whether it's meaningful or not. It furthermore produces a verified parse tree.It also does type checking, Label checking and Flow control checking.
Intermediate Code Generator – It generates intermediate code, that is a form which can be readily executed by machine We have many popular intermediate codes. Example – Three address code etc. Intermediate code is converted to machine language using the last two phases which are platform dependent..
Code Optimizer – It transforms the code so that it consumes fewer resources and produces more speed. The meaning of the code being transformed is not altered. Optimisation can be categorized into two types: machine dependent and machine independent.
Target Code Generator – The main purpose of Target Code generator is to write a code that the machine can understand and also register allocation, instruction selection etc. The output is dependent on the type of assembler. This is the final stage of compilation.

Question 7

(a) What does a semantic analysis do?

Semantic analysis is one in which certain checks are performed to ensure that components of a program fit together meaningfully.
Mainly performs type checking.

(b)List the various error recovery strategies for a lexical analysis.

Possible error recovery actions are:

- · Panic mode recovery
- · Deleting an extraneous character
- · Inserting a missing character
- · Replacing an incorrect character by a correct character
- Transposing two adjacent characters

(c).YACC is an automatic tool for generating the parser program.
YACC stands for Yet Another Compiler Compiler which is basically the utility available from UNIX.
Basically YACC is LALR parser generator.
It can report conflict or ambiguities in the form of error messages.

IHAMA E.I.