SYSTEM SOFTWARE

1. Describe different types of Assemblers.

Ans. Type of Assembler

1. Single pass assemblers
2. Two pass assemblers and
3. Multi pass assemblers

Pass is a terminology used in system software. Each reading of a program can be called as a pass. The assembly process can be done within a pass, if so, such assemblers are single pass assemblers. If the assembly is done two passes, then those assemblers are called two pass assemblers. The translation of the assembly language program can be done in several passes. Such assemblers are called multi pass assemblers.

***Single* Pass Assemblers:**

In single pass assemblers, the entire translation of assembly language program into object program is done in only one pass. The source program is read only once. These are also called as 'one pass assembler'. These assemblers suffer the problem of forward reference. Handling the forward reference in a single pass assembler is difficult. The object code can be produced in the single pass assemblers in two different ways. In the first way, the object code is directly loaded into the main memory for execution. Here, no loader is required. This type of loading scheme is called 'compile and loading scheme'. In the second way, the object program will be loaded into the main memory for execution later as necessity arises. Here, a separate loader program is necessary.

Two Pass Assemblers:

The two pass assemblers are widely used and the translation process is done in two passes. The two pass assemblers resolve the problem of forward references conveniently. An assembler, which goes through an assembly language program twice, is called a two pass assembler. During the first pass it collects all labels. During the second pass it produces the machine code for each instruction and assigns address to each of them. It assigns addresses to labels by counting their position from the starting address. It provides many features than the single pass assembler. It is widely used.

Multi Pass Assemblers:

If the assembly process is done more than two passes then those assemblers are called as multi pass assemblers. One reading of a program (source program or any program in concern) can be called as a pass. In assemblers, the passes are necessary to solve the problem of forward references. In order to process the entire symbol definitions several passes are made over the source program.

2. What is Language Processor? Explain the two Language processing activities.

Ans. A ***language processor*** is Software which bridges a specification or execution gap. The activity performed by the language processor is called Language processing. The program which inputs to the language processor is called source program and to its output as the target program. The languages in which these programs are written are called source language and destination language respectively. A language processor abandons the generation of target program if it detects errors in the source program. Some examples of language processors are,
1) A *language translator* bridges the execution gap to the machine language of a computer system. (E.g: Assembler, compiler, interpreter etc.)

2) The *detranslator* bridges the same execution gap as language translator but in reverse direction.

3) *Preprocessor* is another language processor which bridges execution gap.

4) *The language* migrator bridges the specification gap between two programming languages.

Language processing activities are those that bridge the specification gap and those that bridge the execution gap. These can be divided into two,
1) Program generation activities.

2) Program execution activities.

Program generation activity aims at automatic generation of a program. A program execution activity organizes the execution of a program written in a programming language in a computer system.

3. What is an assembly language? Explain its basic features. State the advantages and disadvantages of coding in assembly language.

Ans. Assembler is a program, which translates Assembly Language Program (ALP) into machine language program (object program). It places the object program in the secondary memory.
An assembly language is a machine dependent, low level programming language which is specific to a certain computer system (or a family of computer systems). Compared to the machine language of a computer system, it provides three basic features which simplify programming:

1. ***Mnemonic operation codes:*** Use of mnemonic operation codes (also called *mnemonic opcodes)* for machine instructions eliminates the need to memorize numeric operation codes. It also enables the assembler to provide helpful diagnostics, for example indication of misspell of operation codes.
2. ***Symbolic operands:*** Symbolic names can be associated with data or instructions. These symbolic names can be used as operands in assembly statement. The assembler performs memory bindings to

these names; the programmer need not know any details of the memory bindings performed by the assembler.

3. ***Data declarations:*** Data can be declared in a variety of notations, including the decimal notation. This avoids manual conversion of constants into their internal machine representation, for example, conversion of -5 into (11111010)2 or 10.5 into (41A80000)16.

An assembler translates a file of assembly language statements into a file of binary machine instructions. The translation process has two major parts. The first step is to find memory locations with labels so that the relationship between symbolic names and addresses is known when instructions are translated. The second step is to translate each assembly statement by combining the numeric equivalents of opcodes, register Specifies, and labels into a legal instruction. The assembler produces an output file, called an object file, which contains the machine instructions, data, and book keeping (reference) information.

The assembler receives the assembly language program as input and converts into corresponding object program. It also provides the information to the loader. The loader places this object code into the main memory during execution.

Advantages of coding in assembly language are:

- Provides more control over handling particular hardware components
- May generate smaller, more compact executable modules
- Often results in faster execution

Disadvantages:

- Not portable
- More complex
- Requires understanding of hardware details (interfaces)

An assembler does the following:

1. Generate machine instructions
   - evaluate the mnemonics to produce their machine code
   - evaluate the symbols, literals, addresses to produce their equivalent machine addresses
   - convert the data constants into their machine representations
2. Process pseudo operations.

4. List and explain the various issues which must be considered to make device drivers portable across CPU architectures.

Ans. A device driver should be designed so that it can accommodate peripheral devices to operate on more than one CPU architecture. The following issues must be considered to make your drivers portable across CPU architectures:

i. Control status register (CSR) access
ii. I/O copy operation
iii. Direct memory access (DMA) operation

iv. Memory mapping
v. 64-bit versus 32-bit
vi. Memory barriers

i. ***Control status register (CSR) access:*** Many device drivers based on the UNIX operating system access a device's control status register (CSR) addresses directly through a device register structure. Some CPU architectures do not allow to access the device CSR addresses directly. You can write one device driver with the appropriate conditional compilation statements so that device driver can operate on both types of CPU architectures.

ii. ***I/O copy operation issues:*** *I/O copy operations can differ from one device driver to another due to the differences in CPU architectures. If you use techniques other than the generic kernel interfaces that digital technology provides for performing I/O copy operations, you may not be able to write one device driver to operate on more than one CPU architecture or more than one CPU type within the same architecture.*

iii. ***Direct memory access (DMA) operation issues: Direct*** *memory access (DMA) operations can differs from one device driver to another because of the different types of the buses used and its DMA hardware support features.*

iv. ***Memory mapping issues:*** *Some CPU architectures do not support an application's use of memory map section. You should design the device driver suitable for different types of memory map section deployed to handle applications in different types of OS and for the one that do not use memory map section also*

v. ***64-bit versus 32-bit:*** *You have to consider while declaring data types for 32-bit and 64-bit CPU architectures. Pay careful attention to data types to make your device drivers work on both 32-bit and 64-bit systems.*

vi. ***Memory barriers issues:*** *In certain architecture or a model of computer system, a CPU may actually perform the memory operations in any order it likes, provided program causality appears to be maintained. Independent memory operations or a read and write operations can be performed in random order without any problem, but this can be a problem for CPU-CPU interaction and for I/O. So some way of intervening to instruct CPU to restrict the order is required. Memory barriers are such interventions that impose a perceived partial ordering over the memory operations. Memory barrier (mb) acts as an interface that guarantees ordering of operations. The mb interface is derived from the MB instruction. The MB instruction must be used in any system to guarantee correctly ordered access to I/O registers or memory that can be accessed via off-board DMA.*

5. Explain briefly Android Architecture Libraries.

Ans. Libraries in Android architecture include the Surface Manager, Media Framework, WebKit, SQLite, OpenGL/ES, FreeType, SGL, SSL,

Libc. These native libraries run as processes within the underlying Linux kernel.
Libc: Libc is a standard C library tuned for embedded devices.
SSL: It is a Secure Sockets Layer cryptographic protocol for secure internet communications.
SGL: SGL is a scalable graphic library or also called clear graphic library that gives the underlined 2D graphic engine. In Android graphics platform,you can combine 3D and 2D graphics in the same application.
OpenGL/ES: OpenGL/ES is a 3D library. OpenGL supports the CD graphics which is based on OpenGL1.O, so these libraries either use hardware 3D accelerator if they are available or they use the highly optimized software. They have a software implementation that is hardware acceleratable if the device has a 3D chip on it.
FreeType: Freetype is used for bitmap and vector font rendering. FreeType is a free, high quality and portable font engine.
WebKit: It is the open source browser engine, used as a core of Android's browser. It acts as the webrendering engine that powers the default browser. It is the same browser that powers Safari from Apple and it renders well on small screens and on mobile devices.
SQLite: It is the basic datastore technology for the Android platform and is very lightweight relational database engine that manages access to display subsystem. It is used as the core of most of its data storage.
Media Frame work: The Media Framework was provided by Packet Video, one of the members of the open handset alliance and that contains the entire codex that make up the core of the media experiences. So, in there you'll find all the audio and video codes you need to build a rich media experience like IMPEG4, H.264, MP3, and AAC.
Surface Manager: The surface manager is responsible for composing different drawing surfaces on to the screen or you can say that it is responsible for graphics on the device's screen. It manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications. Surface manager seamlessly monitor the 2D and 3D graphic.
LibWebCore: LibWebCore is a modern web browser engine that gives us embeddable web view.
Media Libraries: It supports playback and recording of many popular audio and video formats, as well as static image file. Media libraries are based on **PackerVideosOpenCORE**. So these libraries support playback as well as
recording of many popular audio and video formats like MPEG4, H264, MP3 etc.
Android Runtime
The Android runtime layer also includes set of core java libraries and DVM (Dalvik Virtual Machine) located in same layer. The Android runtime layer includes set of base libraries that are required for java

libraries. Every android application runs in its own process, with its own instance in the Dalvik Virtual Machine.

Core Libraries: Core Libraries are written in the Java programming language.The core library contains all of the collection classes, utilities, I/O, all the utilities and tools that you use.

Dalvik Virtual Machine: Android based systems utilize their own virtual machine (VM), which is known as the Dalvik Virtual Machine (DVM). It is extremely low-memory based virtual machine, which was designed especially for Android to run on embedded systems and work well in low power situations. It is also tuned to the CPU attributes. Applications for Android are developed in Java and executed in a virtual machine, called Dalvik VM. The DVM uses special byte-code, hence native Java bytecode cannot directly be executed on Android systems. The Dalvik VM creates a special file format (.DEX) that is created through build time post processing.

6. Write the Steps for addressing UPnP device with proper flowchart.

Ans. All UPnP devices must follow the same steps in acquiring an IP address. The steps, as specified by the UPnP device architecture, are presented in the flowchart is shown below.