1. What is system programming?
Systems programming, or system programming, is the activity of programming
computer system software. System programming involves designing and writing
computer programs that allow the application software to interact with computer
hardware to interact with computer hardware for effective execution.

2. What is system software?
The system software is a computer program, which is designed to run computer
hardware and applications. The system software makes your system faster, effective,
and safe. System software is actually a group of programs that regulate and manage
the various operations of computer hardware, like the hardware itself, and other
software installed on the computer. It also assists application software in running
properly. System Software helps make the performance of a computer faster, safer,

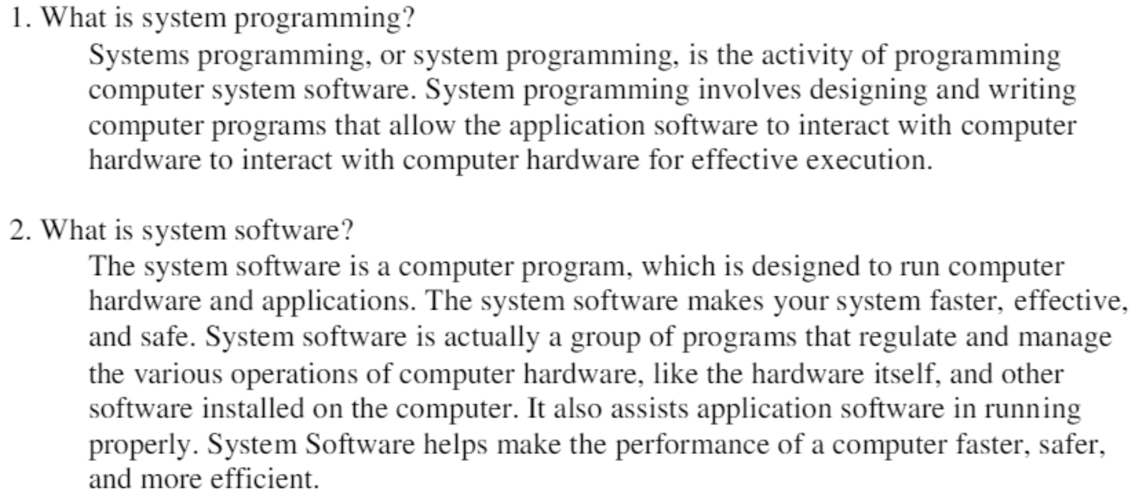
1. What is system programming?
Systems programming, or system programming, is the activity of programming
computer system software. System programming involves designing and writing

computer programs that allow the application software to interact with computer
hardware to interact with computer hardware for effective execution.

2. What is system software?
The system software is a computer program, which is designed to run computer
hardware and applications. The system software makes your system faster, effective,
and safe. System software is actually a group of programs that regulate and manage
the various operations of computer hardware, like the hardware itself, and other
software installed on the computer. It also assists application software in running
properly. System Software helps make the performance of a computer faster, safer,
1. What is system programming?
Systems programming, or system programming, is the activity of programming
computer system software. System programming involves designing and writing
computer programs that allow the application software to interact with computer
hardware to interact with computer hardware for effective execution.

2. What is system software?
The system software is a computer program, which is designed to run computer

hardware and applications. The system software makes your system faster, effective,
and safe. System software is a group of programs that regulate and manage
the various operations of computer hardware, like the hardware itself, and other
software installed on the computer. It also assists application software in running
properly. System Software helps make the performance of a computer faster, safer,

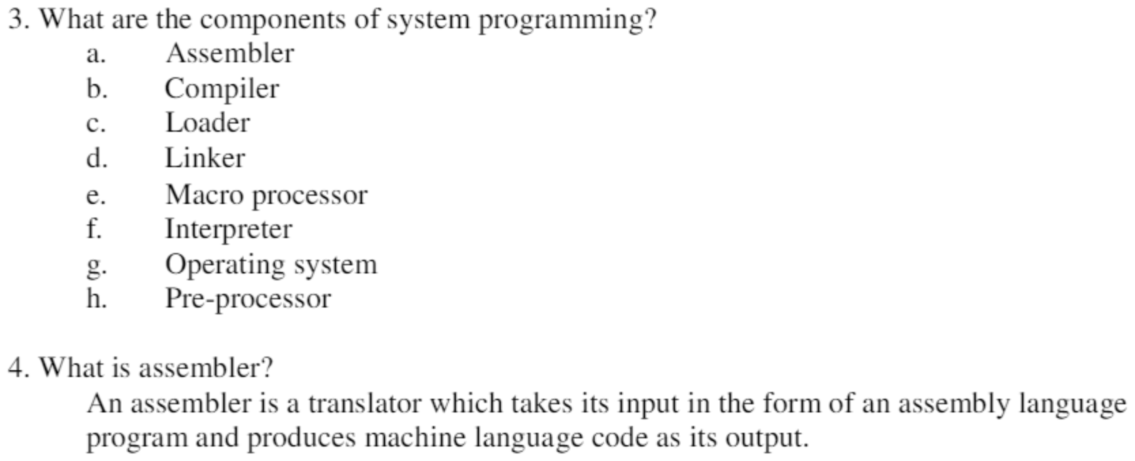


1. What is system programming?

   Systems programming, or system programming, is the activity of programming computer system software. System programming involves designing and writing computer programs that allow the application software to interact with computer hardware to interact with computer hardware for effective execution.

2. What is system software?

   The system software is a computer program, which is designed to run computer hardware and applications. The system software makes your system faster, effective, and safe. System software is actually a group of programs that regulate and manage the various operations of computer hardware, like the hardware itself, and other software installed on the computer. It also assists application software in running properly. System Software helps make the performance of a computer faster, safer, and more efficient.



3. What are the components of system programming?
   a. Assembler
   b. Compiler
   c. Loader
   d. Linker
   e. Macro processor
   f. Interpreter
   g. Operating system
   h. Pre-processor

4. What is assembler?

   An assembler is a translator which takes its input in the form of an assembly language program and produces machine language code as its output.

5. Define a macro?

A macro name is an abbreviation, which stands for some related lines of code. Macros are useful for the following purposes:

· To simplify and reduce the amount of repetitive coding
· To reduce errors caused by repetitive coding
· To make an assembly program more readable.

6. Difference between compiler and Interpreter?

Compliers and interpreters are programs that help convert the high-level language (Source Code) into machine codes to be understood by the computers.

S.No.	Compiler	Interpreter
1.	Compiler scans the whole program in one go.	Translates program one statement at a time.
2.	As it scans the code in one go, the errors (if any) are shown at the end together.	Considering it scans code one line at a time, errors are shown line by line.
3.	Main advantage of compilers is its execution time.	Due to interpreters being slow in executing the object code, it is preferred less.
4.	It converts the source code into object code.	It does not convert source code into object code instead it scans it line by line
5	It does not require source code for later execution.	It requires source code for later execution.
Eg.	C, C++, C# etc.	Python, Ruby, Perl, SNOBOL, MATLAB, etc.

7. What is role of linker?

Linker is a program in a system which helps to link an object module of program into a single object file. It performs the process of linking. In computing, a **linker** or **link editor** is a computer system program that takes one or more object files (generated by a compiler or an assembler) and combines them into a single executable file, library file, or another "object" file.

8. What is loader?

The loader is special program that takes input of object code from linker, loads it to main memory, and prepares this code for execution by computer. Loader allocates memory space to program. Even it settles down symbolic reference between objects. It is in charge of loading programs and libraries in operating system.

9. What is text editor?

A text editor is a computer program that lets a user enter, change, store, and usually print text (characters and numbers, each encoded by the computer and its input and output devices, arranged to have meaning to users or to other programs). Typically, a text editor provides an "empty" display screen (or "scrollable page") with a fixed-line length and visible line numbers. You can then fill the lines in with text, line by line.

10. Explain different component of text editor?

1. Line editor: In this, you can only edit one line at a time or an integral number of lines. You cannot have a free-flowing sequence of characters. It will take care of only one line.
   Ex: Teleprinter, edlin, teco
2. Stream editors: In this type of editors, the file is treated as continuous flow or sequence of
   characters instead of line numbers, which means here you can type paragraphs.
   Ex: Sed editor in UNIX
3. Screen editors: In this type of editors, the user is able to see the cursor on the screen and
   can make a copy, cut, paste operation easily. It is very easy to use mouse pointer.
   Ex: vi, emacs, Notepad
4. Word Processor: Overcoming the limitations of screen editors, it allows one to use some
   format to insert images, files, videos, use font, size, style features. It majorly focuses on Natural language.
5. Structure Editor: Structure editor focuses on programming languages. It provides features
   to write and edit source code.
   Ex: Netbeans IDE, gEdit.

11.Explain the different functions of loader.

Loader Function: The loader performs the following functions:

1) Allocation

2) Linking

3) Relocation

4) Loading

Allocation:

• Allocates the space in the memory where the object program would be loaded for Execution.

• It allocates the space for program in the memory, by calculating the size of the program. This activity is called allocation.

• In absolute loader allocation is done by the programmer and hence it is the duty of the programmer to ensure that the programs do not get overlap.

• In reloadable loader allocation is done by the loader hence the assembler must supply the loader the size of the program.

Linking:

• It links two or more object codes and provides the information needed to allow references between them.

• It resolves the symbolic references (code/data) between the object modules by assigning all the user subroutine and library subroutine addresses. This activity is called linking.

• In absolute loader linking is done by the programmer as the programmer is aware about the runtime address of the symbols.

• In relocatable loader, linking is done by the loader and hence the assembler must supply to the loader, the locations at which the loading is to be done.

Relocation:

• It modifies the object program by changing the certain instructions so that it can be loaded at different address from location originally specified.

• There are some address dependent locations in the program, such address constants must be adjusted according to allocated space, such activity done by loader is called relocation.

• In absolute Loader relocation is done by the assembler as the assembler is aware of the starting address of the program.

• In relocatable loader, relocation is done by the loader and hence assembler must supply to the loader the location at which relocation is to be done.

Loading:

• It brings the object program into the memory for execution.

• Finally, it places all the machine instructions and data of corresponding programs and subroutines into the memory. Thus, program now becomes ready for execution, this activity is called loading.

• In both the loaders (absolute, relocatable) Loading is done by the loader and hence the assembler must supply to the loader the object program.

12. Explain all types of loaders?

Absolute loader.

Bootstrap loader.

Relocating loader.

Linking loader.

Absolute loader is a kind of loader in which relocated object files are created, loader accepts these files and places them at specified locations in the memory. This type of loader is called absolute because no relocation information is needed; rather it is obtained from the programmer or assembler.

The relocating loader will load the program anywhere in memory, altering the various addresses as required to ensure correct referencing. The decision as to where in memory the program is placed is done by the Operating System, not the programs header file.

Performs all linking and relocation operations, including automatic library search, and loads the linked program into memory for execution. Linking loaders perform four functions:

1. Allocation: allocating space in memory for programs

2. Linking: resolving external references between modules

3. Relocation: adjusting all address references

4. Loading: physically placing machine instructions and data in memory

13. What is input and output for pass1?

14. What is input and output for pass2?

15. Explain the following

1. LTORG

The LTORG directive instructs the assembler to assemble the current literal pool immediately. A literal pool contains the literals you specify in a source module either after the preceding LTORG instruction, or after the beginning of the source module.

2. DC

You use the DC instruction to define the data constants you need for program execution. The DC instruction causes the assembler to generate the binary representation of the data constant you specify into a particular location in the assembled source module; this is done at assembly time. DC simply creates initial data in an area of the program. The contents of that area might be modified during program execution, so the original data is not truly "constant".

3. DS

DS is a data definition statement. The DS assigns storage, but does not initialise the space.

16. what are types of statement in assembly?

An assembly program contains following three kinds of statements:

- 1. **Imperative statements**:
  - These indicate an action to be performed during execution of the assembled program. Each imperative statement typically translates into one machine instruction.
- 2. **Declaration statements**: The syntax of declaration statements is as follows:
  - [Label] DS<constant>
  - [Label] DC '<value>'
  - The DS statement reserves areas of memory and associates names with them. The DC statement constructs memory words containing constants.

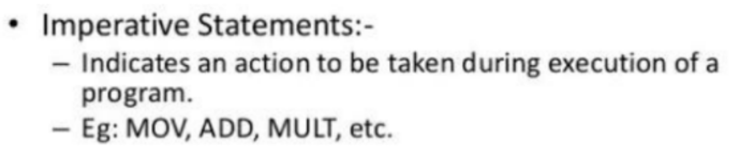
The DC statement constructs memory words containing constants.

- 3. **Assembler directives**: These instruct the assembler to perform certain actions during the assembly of a program. For example
  - START <constant> directive indicates that the first word of the target program generated by the assembler should be placed in the memory word with address <constant>

17. Example of Imperative statement?

Imperative Statements :- An imperative statement is instruction in assembly program. Every imperative statement generates one machine instruction.

e.g . MOVER AREG , BREG



- Imperative Statements:-
  - Indicates an action to be taken during execution of a program.
  - Eg: MOV, ADD, MULT, etc.

18. What is assembler directive?

Assembler directives are the **instructions** used by the assembler at the time of assembling a source program. More specifically, we can say, assembler directives are the commands or instructions that control the operation of the assembler.

19. Explain the following

1. EQU

The EQU instruction **assigns absolute or relocatable values to symbols**. Use it to: Assign single absolute values to symbols. Assign the values of previously defined symbols or expressions to new symbols, thus letting you use different mnemonics for different purposes

2. ORIGIN

ORIGIN tells the assembler about the starting-address of memory-area to place the data block. Ex: ORIGIN 204; Instructs assembler to initiate data-block at memory-locations starting from 204.

3. START

The START instruction can be used to **initiate the first or** only executable control section of a source module, and optionally to set an initial location counter value.

4. END

Directive tells the assembler that this is the end of the source-program text.

20. Give the data structure used for Pass1 and Pass2 of assembler?

- Operation Table (OPTAB) and
- Symbol Table (SYMTAB).

For each mnemonic N the OPTAB contains:

1. Mnemonic type and its Machine language expression;

2. The pattern M (N) generated by P ass2 when a statement with mnemonic N is encountered;

3. The function called by P ass1 to translate the assembly language statements whose mnemonic is N;

4. The function called by P ass2 to instantiate the pattern M (N) when a

statement with mnemonic N is encountered.

• SYMTAB, that holds the translation of user defined symbols

21. What is symbol table, literal table, pool table?

Symbol Table (SYMTAB): - Each entry in symbol table has two primary fields name and address. The symbol table uses concept of forward reference to achieve address of symbols.

Literal Table (LITTAB): - Literal table contains all literals and address of all literals. For literals LTORG directives places the all constants at consecutive memory locations. If we are not using LTORG then all literals are placed after END in memory

Pool Table (POOLTAB): - Awareness of different literal pools is maintained using the auxiliary table POOLTAB. At any Stage, the current literal pool is the last pool in LITTAB

22.Explain absolute loader scheme with its advantages and disadvantages.

- The absolute loader is a kind of loader in which relocated object files are created, loader accepts these files and places them at a specified location in the memory.
- This type of loader is called absolute loader because no relocating information is needed, rather it is obtained from the programmer or assembler

Advantages:

1. It is simple to implement.
2. This scheme allows multiple programs or the source programs written in different languages. If there are multiple programs written in different languages then the respective language assembler will convert it to the language and common object file can be prepared with all the ad resolution.
3. The task of loader becomes simpler as it simply obeys the instruction regarding where to place the object code to the main memory.
4. The process of execution is efficient.

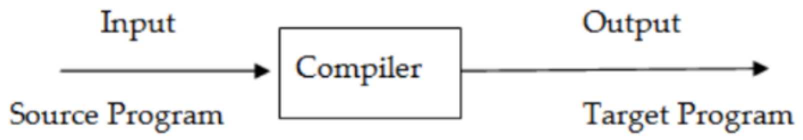
Disadvantages:

1. In this scheme, it's the programmer's duty to adjust all the inter-segment addresses and manually do the linking activity. For that, it is necessary for a programmer to know the memory management.

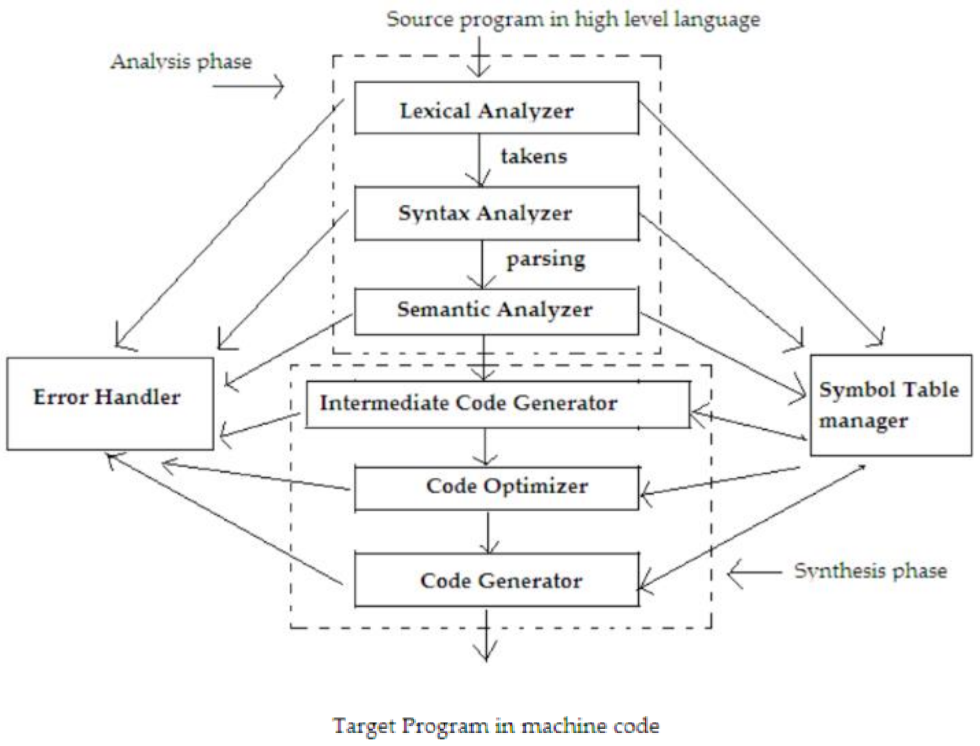
2. If at all any modification is done to some segment the starting address of immediate next segments may get changed the programmer has to take care of this issue and he/she needs to update the corresponding starting address on any modification in the source

23.What is compiler?

A compiler is a program that reads a program written in source language and translates it into an equivalent program in machine code. During compilation, it reports the presence of errors in the source program.



24.Explain different phases of compiler?



The analysis of a source program is divided into mainly three phases. They are:

1. **Linear Analysis-**
   This involves a scanning phase where the stream of characters is read from left to right. It is then grouped into various tokens having a collective meaning.
2. **Hierarchical Analysis-**
   In this analysis phase, based on a collective meaning, the tokens are categorized hierarchically into nested groups.
3. **Semantic Analysis-**
   This phase is used to check whether the components of the source program are meaningful or not.

25.What is conditional macro?

A *conditional expression* defines the selection criteria for the different functions to be carried out in a macro group. A conditional expression is an expression that evaluates to a true or false value, and the appropriate function is processed based on that value.

Conditional expressions are required parameters for the following macros:

- #IF
- #ELIF
- #DOEX
- #EXIF.

You can also use conditional expressions with the following macros:

- #DO
- #GOTO.

26.How to define macro?

Macro is a unit of specification of program generation through expansion. It is single line abbreviation for group of instructions Typically MACRO is defined at start of program or at end of program.

Macro Definition Syntax :-

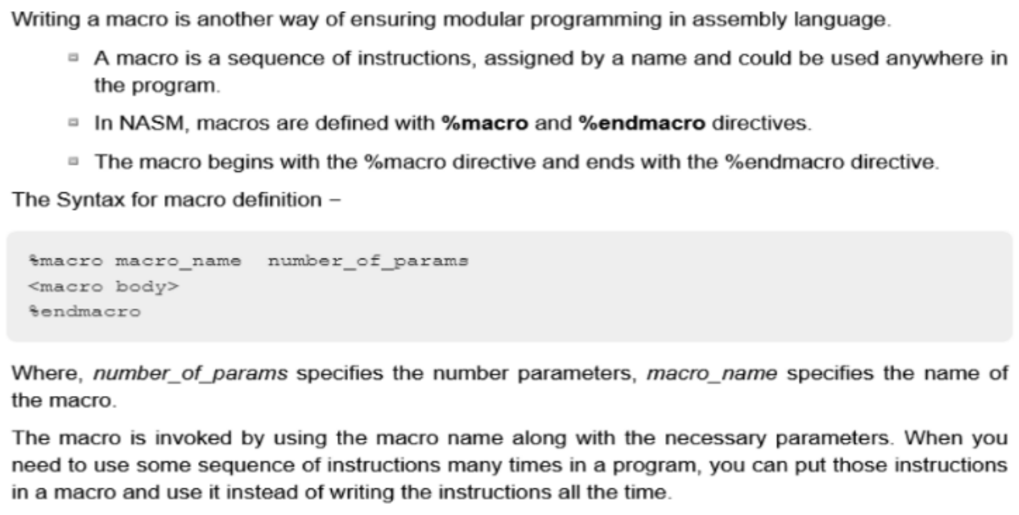
1) Macro header :- It contains keyword 'MACRO'.

2) Macro prototype statement syntax :-

< Macro Name > [& < Formal Parameters >]

3) Model Statements :- It contains 1 or more simple assembly statements, which will replace MACRO CALL while macro expansion.

4) MACRO END MARKER :- It contains keyword 'MEND'.



Writing a macro is another way of ensuring modular programming in assembly language.

- A macro is a sequence of instructions, assigned by a name and could be used anywhere in the program.
- In NASM, macros are defined with **%macro** and **%endmacro** directives.
- The macro begins with the %macro directive and ends with the %endmacro directive.

The Syntax for macro definition −

```
%macro macro_name  number_of_params
<macro body>
%endmacro
```

Where, *number_of_params* specifies the number parameters, *macro_name* specifies the name of the macro.

The macro is invoked by using the macro name along with the necessary parameters. When you need to use some sequence of instructions many times in a program, you can put those instructions in a macro and use it instead of writing the instructions all the time.

27.How to call macro?

The macro is invoked by using the macro name along with the necessary parameters. When you need to use some sequence of instructions many times in a program, you can put those instructions in a macro and use it instead of writing the instructions all the time

28.Explain how to design of two pass macro processor?

Macro processor takes a source program containing macro definitions and macro calls. It converts it into an assembly language program without macro definition or calls

29.What is input for pass1 and pass2 of two pass macro processor?

30.What is dynamic linking?

In dynamic linking, the names of the external libraries /shared libraries are copied into the final executable; thus, the real linking occurs at run time when the executable file and libraries load to the memory. **The operating system performs** dynamic linking.

31.Explain following

1. Compile and Go,

In this type of loader, the instruction is read line by line, its machine code is obtained and it is directly put in the main memory at some known address. That means the assembler runs in one part of memory and the assembled machine instructions and data is directly put into their assigned memory locations. This scheme is simple to implement.

Disadvantages: -

1) Wastage of memory as assembler present in memory.

2) There is no production of object file.

3) Multiple programs can't get handed.

4) Execution time will be more in this scheme as every time program is assembled and then executed.

2. General Loader Scheme,

In this loader scheme, the source program is converted to object program by translator. The loader accepts these object codes and puts machine instructions and data in an executable form at their assigned memory. The loader occupies some portion of main memory. The program need not be re translated each time while running it. There is no wastage of memory, because assembler is not placed in the memory, instead of it, loader occupies some portion of the memory.

3. Absolute Loaders,

Absolute loader is a kind of loader in which relocated object files are created, loader accepts these files and places them at specified locations in the memory. This type of loader is called absolute because no relocation information is needed; rather it is obtained from the programmer. In this scheme, the programmer must have knowledge of memory management.

4. Relocating Loaders,

To avoid possible assembling of all subroutines when a single subroutine is change and to perform the tasks of allocation and linking for the programmer, the general class of relocating loader was introduced. The output of a relocating loader is the program and information about all other programs it references. In addition, there is information (relocation information) as to location in this program that must be change if it is to be loaded in an arbitrary location in memory.

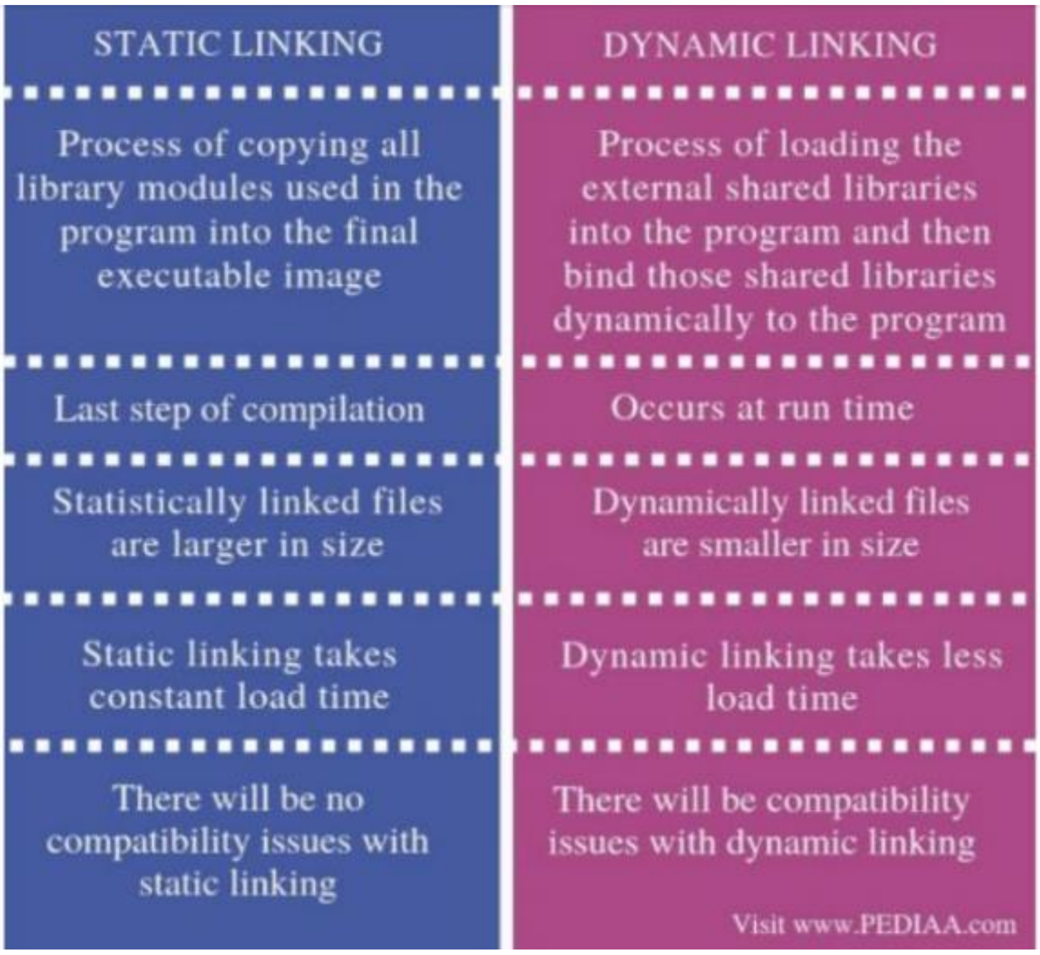
5. Direct linking Loaders,

To avoid possible assembling of all subroutines when a single subroutine is change and to perform the tasks of allocation and linking for the programmer, the general class of relocating loader was introduced. The output of a relocating loader is the program and information about all other programs it references. In addition, there is information (relocation information) as to location in this program that must be change if it is to be loaded in an arbitrary location in memory.

32.Difference between absolute loader and relocating loader?

The biggest difference between these two loaders is that absolute loaders will load files into a specific location and a relocating loader will place the data anywhere in the memory.

33.Differnce between static and dynamic linking?



STATIC LINKING	DYNAMIC LINKING
Process of copying all library modules used in the program into the final executable image	Process of loading the external shared libraries into the program and then bind those shared libraries dynamically to the program
Last step of compilation	Occurs at run time
Statistically linked files are larger in size	Dynamically linked files are smaller in size
Static linking takes constant load time	Dynamic linking takes less load time
There will be no compatibility issues with static linking	There will be compatibility issues with dynamic linking

Visit www.PEDIAA.com

34.What is OS?

An operating system is an interface between users and hardware of a computer system. An operating system manages resources of a computer.

These resources include: -

a) Memory b) Processor c) Input / Output devices d) File system

OS keeps track of each resource and decides who will get control over resources, for how long.

Operating system makes computer more convenient to use. All resource are used in very efficient way.

35.What is function of OS?

Operating system works as an interface between user and the computer.

Functions of Operating System: -

1) Program Development: - Operating system provide a set of utility program which are necessary to program development. (e.g., Editor, linker, etc)

2) Program Execution: - Operating system handles loading of programs from secondary memory to primary memory. It makes program available for execution.

3) I/O Operations: - A running program may require I/O. I/O operations are handled by operating system.

4) File System Manipulations: - An operating system provides various system calls for manipulation of files.

5) Communication: - Under many situations a process has to communicate with other process. Communication is required for exchange of information. Communication done through message pipes, mailbox, etc.

6) Resource Sharing and Protection: - Resources are shared properly with all processes as well as protection is also provided.

7) Error Detection: - Different types of errors may occur while a computer system is running.

8) Accounting: - Operating system keeps track of every action. For how much duration user has a log, what modifications he did, etc.

36.Explain process state model?

Process: - A process is basically a program while it is being executed. A process is active part of program. Program is passive part. A process is dynamic entity and program is static entity.

Every process has following state: -

a) Create / New: - Operating system creates new process by using **fork** () system call. These processes are newly created process and resources are not allocated.

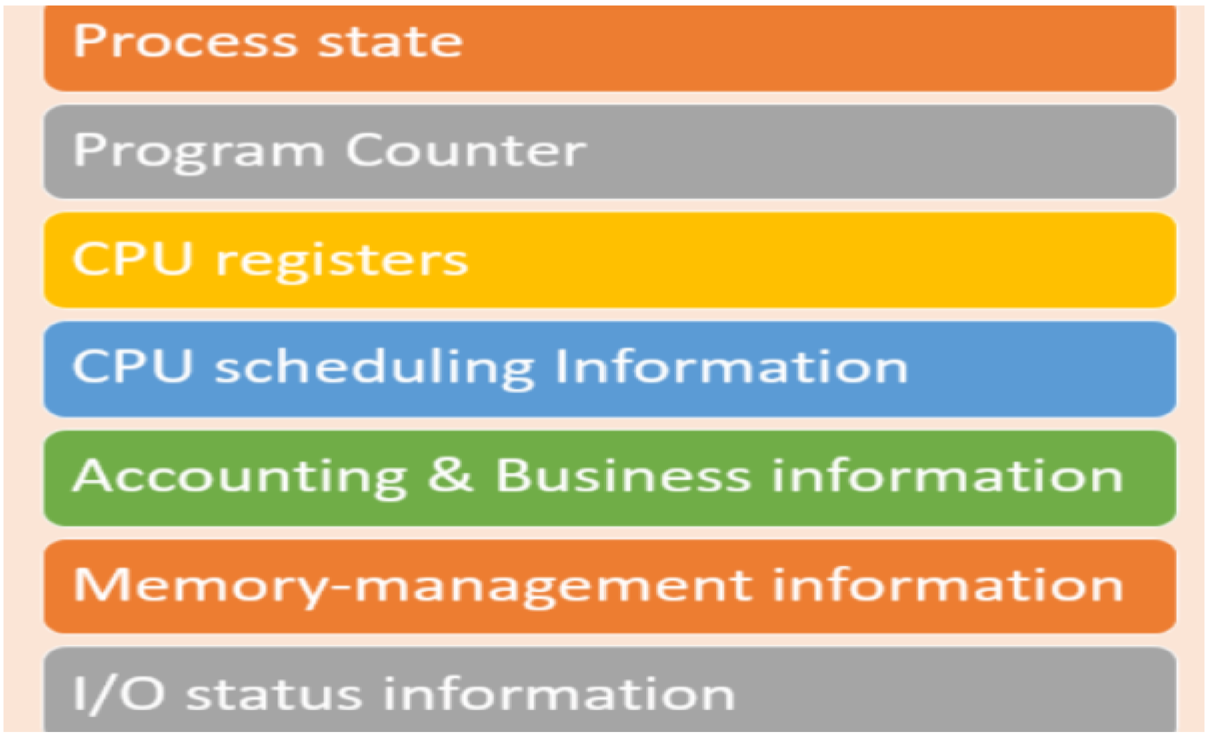
b) Ready: - The process is competing for CPU. There are number of processes in ready state.

c) Running: - The process that is having CPU is considered as process in running state.

37.What is Process control block?

PCB stands for Process Control Block. It is a data structure that is maintained by the Operating System for every process. The PCB should be identified by an integer Process ID (PID). It helps you to store all the information required to keep track of all the running processes.

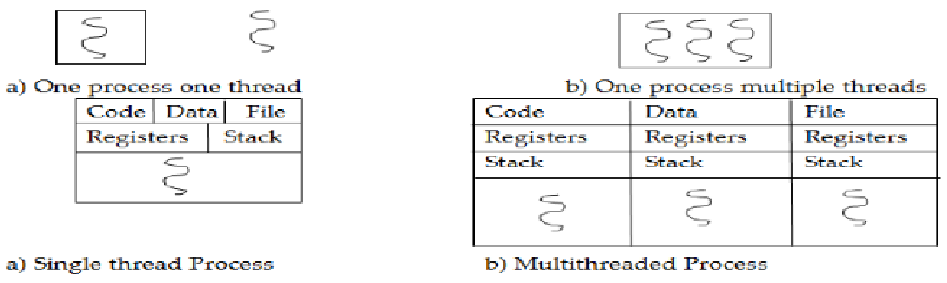
It is also accountable for storing the contents of processor registers. These are saved when the process moves from the running state and then returns back to it. The information is quickly updated in the PCB by the OS as soon as the process makes the state transition.


Process state
Program Counter
CPU registers
CPU scheduling Information
Accounting & Business information
Memory-management information
I/O status information

38.What is thread?

Thread is called as light weight process. It is used to handle burden of process or to simplify process. Thread is a dispatch able unit of work. It consists of thread ID, program counter, stack and register set.

Traditionally there is single thread of execution per process. In multithreading environment, O.S. supports multiple threads of execution within a single process.



39.Difference between process and thread?

S.N.	Process	Thread
1	Process is heavy weight or resource intensive.	Thread is light weight, taking lesser resources than a process.
2	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
3	In multiple processing environments, each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
4	If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.
5	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.

6	In multiple processes each process operates independently of the others.	One thread can read, write or change another thread's data.

40.Explain thread life cycle?

A process is divided into several light-weight processes, each light-weight process is said to be a **thread**. The thread has a program counter that keeps the tracks of which instruction to execute next, if process registers, which hold its current working variables. It has a stack which contains the executing thread history. The life cycle of thread is **Born state**, **Ready state**, **Running state**, **Blocked State**, **Sleep**, **Dead**.

1. Born State: A thread that has just created.
2. Ready State: The thread is waiting for the processor (CPU).
3. Running: The System assigns the processor to the thread means that the thread is being executed.
4. Blocked State: The thread is waiting for an event to occur or waiting for an I/O device.

5. Sleep: A sleeping thread becomes ready after the designated sleep time expires.
6. Dead: The execution of the thread is finished.

41.What is multithreading?

Multithreading is similar to multitasking, but enables the processing of multiple threads at one time, rather than multiple processes. Since threads are smaller, more basic instructions than processes, multithreading may occur within processes.

42.What is preemptive scheduling?

Preemptive Scheduling: - Cooperative scheduling.

A scheduling method that interrupts the processing of a process and transfers the CPU to another process is called a preemptive CPU scheduling. The running process will switch to ready state and new process will come from ready to running state.

A process can be temporarily suspended due to:

1) Request for I/O.

2) Time slab is over.

43.What is non-preemptive scheduling?

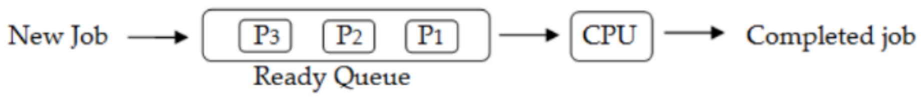
Non-Preemptive Scheduling: - Non cooperative scheduling.

Non- Preemptive operation usually proceeds towards completion uninterrupted. Once the system has assigned a processor to a process, the system cannot remove that processor from the process.

Sr. No.	**Preemptive Scheduling**	**Non-Preemptive Scheduling**
i)	It is cooperative scheduling	It is non cooperative scheduling
ii)	It is process gives control to other process completion.	One process complete itself and then give before its controls to other process.
iii)	It increases the cost and it has higher overhead.	It does not increase cost and it has less overhead.
iv)	It is more complex.	It is simple.
v)	Efficient	Non efficient
vi)	Ex. Round Robin scheduling method	Ex. First Come First Serve (FCFS), SJF (Shortest Job First

44.Explain in short following Scheduling Algorithms:
1. FCFS,

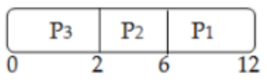
FCFS (First Come First Serve) – It is non preemptive scheduling. It is one of the simplest scheduling algorithms.



In FCFS jobs are executed in order of their arrival. As from figure it is clear that P1, P2 and P3 are executed in sequence first P1 is completed, then P2 and lastly P3 will be completed. Average turnaround time and waiting time are very high.

2. SJF,

In shortest job first (Non-Preemptive Scheduling) selected for execution. If two processes have the same CPU time, FCFS is used. Consider the following example of three process: Using SJF we will get



Process	Execution
P1	6
P2	4
P3	2

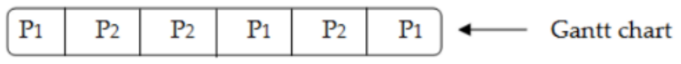
Initially P3 will be completed for 2, then P2 for – 4 and lastly P1 = 6.

SJF will be optimal scheduling algorithm in terms of minimizing. i) Average turnaround time. ii) Average waiting time.

3. RR,

Round Robin Scheduling: - It is used in time sharing system. It provides good response time. In RRS the CPU time is divided in slices of time (Quantum) Each process is given one time slice for which CPU is allocated to process for execution. Consider following example with time slice = 2

Using RRS we will get



Process	Execution
P1	6
P2	4
P3	2

Round Robin scheduling utilize the system resources in an equitable manner. Small processes may be executed in a single time slice giving good response time where as long processes may require several time slices. It is preemptive scheduling.

4. Priority

In this scheduling, processes are scheduled according to their priorities, i.e., highest priority process is scheduled first. If priorities of two processes match, then schedule according to arrival time. Here starvation of process is possible.

45.What is Semaphore, Mutex?

It is a synchronization tool used to solve complex critical section problems. A semaphore is an integer variable that, apart from initialization, is accessed only through two standard atomic operations: Wait and Signal.

46.Explain

1. Reader writer problem,

The readers-writers problem is used to manage synchronization so that there are no problems with the object data. For example - If two readers access the object at the same time there is no problem. However, if two writers or a reader and writer access the object at the same time, there may be problems.
To solve this situation, a writer should get exclusive access to an object i.e. when a writer is accessing the object, no reader or writer may access it. However, multiple readers can access the object at the same time.

2. Producer Consumer problem,

The Producer-Consumer problem is a classical multi-process synchronization problem, that is we are trying to achieve synchronization between more than one process.

There is one Producer in the producer-consumer problem, Producer is producing some items, whereas there is one Consumer that is consuming the items produced by the Producer. The same memory buffer is shared by both producers and consumers which is of fixed-size.

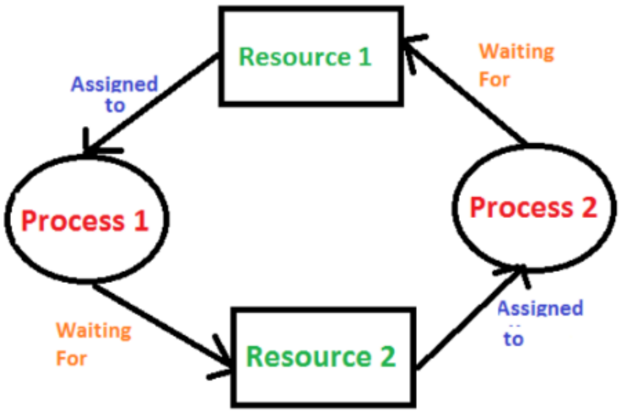
The task of the Producer is to produce the item, put it into the memory buffer, and again start producing items. Whereas the task of the Consumer is to consume the item from the memory buffer.

3. Dining Philosopher problem.

The dining philosopher's problem is the classical problem of synchronization which says that Five philosophers are sitting around a circular table and their job is to think and eat alternatively. A bowl of noodles is placed at the center of the table along with five chopsticks for each of the philosophers. To eat a philosopher needs both their right and a left chopstick. A philosopher can only eat if both immediate left and right chopsticks of the philosopher is available. In case if both immediate left and right chopsticks of the philosopher are not available then the philosopher puts down their (either left or right) chopstick and starts thinking again.

47. What is deadlock?

Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.



48.Explain the

1. Deadlock prevention,

We can prevent Deadlock by eliminating any of the above four conditions.

1. Mutual Exclusion
2. Hold and Wait
3. No Preemption
4. Circular Wait

2. Deadlock avoidance,

In deadlock avoidance, the request for any resource will be granted if the resulting state of the system doesn't cause deadlock in the system. The state of the system will continuously be checked for safe and unsafe states.

In order to avoid deadlocks, the process must tell OS, the maximum number of resources a process can request to complete its execution.

The simplest and most useful approach states that the process should declare the maximum number of resources of each type it may ever need. The Deadlock avoidance algorithm examines the resource allocations so that there can never be a circular wait condition.

3. Deadlock detection,

- If resources have a single instance – In this case for Deadlock detection, we can run an algorithm to check for the cycle in the Resource Allocation Graph. The presence of a cycle in the graph is a sufficient condition for deadlock
- In the above diagram, resource 1 and resource 2 have single instances. There is a cycle R1 → P1 → R2 → P2. So, Deadlock is Confirmed.
- If there are multiple instances of resources – Detection of the cycle is necessary but not sufficient condition for deadlock detection, in this case, the system may or may not be in deadlock varies according to different situations.

4. Deadlock recovery.

A traditional operating system such as Windows doesn't deal with deadlock recovery as it is a time and space-consuming process. Real-time operating systems use Deadlock recovery.

1. **Killing the process –**
   Killing all the processes involved in the deadlock. Killing process one by one. After killing each process check for deadlock again keep repeating the process till the system recovers from deadlock. Killing all the processes one by one helps a system to break circular wait condition.
2. **Resource Preemption –**
   Resources are preempted from the processes involved in the deadlock, preempted resources are allocated to other processes so that there is a possibility of recovering the system from deadlock. In this case, the system goes into starvation.

49.Explain the memory management?

In a multiprogramming computer, the operating system resides in a part of memory and the rest is used by multiple processes. The task of subdividing the memory among different processes is called memory management. Memory management is a method in the operating system to manage operations between main memory and disk during process execution. The main aim of memory management is to achieve efficient utilization of memory.

Why Memory Management is required: (**optional**)

- Allocate and de-allocate memory before and after process execution.
- To keep track of used memory space by processes.
- To minimize fragmentation issues.
- To proper utilization of main memory.
- To maintain data integrity while executing of process.

50.What is Fixed Partitioning, Dynamic Partitioning?

Fixed Partitioning: -

This is the oldest and simplest technique used to put more than one process in the main memory. In this partitioning, the number of partitions (non-overlapping) in RAM is fixed but the size of each partition may or may not be the same. As it is a contiguous allocation, hence no spanning is allowed.

Dynamic Partitioning

Dynamic partitioning tries to overcome the problems caused by fixed partitioning. In this technique, the partition size is not declared initially. It is declared at the time of process loading.

The first partition is reserved for the operating system. The remaining space is divided into parts. The size of each partition will be equal to the size of the process. The partition size varies according to the need of the process so that the internal fragmentation can be avoided.

51.Explain First Fit, Best Fit, Next Fit methods?

1. First Fit: In the first fit, the partition is allocated which is first sufficient from the top of
   Main Memory.
2. Best Fit: Best fit allocates the process to a partition which is the smallest sufficient partition among the free available partitions.
3. Next fit: Next fit is a modified version of 'first fit'. It begins as the first fit to find a free
   partition but when called next time it starts searching from where it left off, not from the beginning. This policy makes use of a roving pointer. Next fit is a very fast searching algorithm and is also comparatively faster than First Fit and Best Fit Memory Management Algorithms.

52.What is paging?

In Operating Systems, Paging is a storage mechanism used to retrieve processes from the secondary storage into the main memory in the form of pages. The main idea behind the paging is to divide each process in the form of pages. The main memory will also be divided in the form of frames.

53.What is segmentation?

In Operating Systems, Segmentation is a memory management technique in which the memory is divided into the variable size parts.

Segmentation is another way of dividing the addressable memory. It is another scheme of memory management and it generally supports the user view of memory. The Logical address space is basically the collection of segments. Each segment has a name and a length.

54.What is page table?

A page table is the data structure used by a virtual memory system in a computer operating system to store the mapping between virtual addresses and physical addresses. The page table is a key component of virtual address translation that is necessary to access data in memory.

55.What is page size?

With computers, page size refers to the size of a page, which is a block of stored memory. Page size affects the amount of memory needed and space used when running programs. Most operating systems determine the page size when a program begins running. A page size includes all of the files that create the web page.

56.What is frame?

It is the smallest unit of data for memory management in a virtual memory operating system. A frame refers to a storage frame or central storage frame. In terms of physical memory, it is a fixed sized block in physical memory space, or a block of central storage.

57.What are page replacement policy?

Page replacement algorithms are an important part of virtual memory management and it helps the OS to decide which memory page can be moved out, making space for the currently needed page. However, the ultimate objective of all page replacement algorithms is to reduce the number of page faults.

58.what are advantages of First In First Out (FIFO), Last Recently Used (LRU), Optimal.

FIFO: - The FIFO method has four major advantages:
(1) it is easy to apply,
(2) the assumed flow of costs corresponds with the normal physical flow of goods,
(3) no manipulation of income is possible, and
(4) the balance sheet amount for inventory is likely to approximate the current market

LRU: -
1. It is open for full analysis.
2. In this, we replace the page which is least recently used, thus free from Belady's Anomaly.
3. Easy to choose page which has faulted and hasn't been used for a long time.

OPR: -
1. Complexity is less and easy to implement.
2. Assistance needed is low i.e. Data Structure used are easy and light.

59.What is Thrashing.

Thrashing is a computer activity that makes little or no progress. Usually, this happens either of limited resources or exhaustion of memory. It arises when a page fault occurs. Page fault arises when memory access of virtual memory space does not map to the content of RAM.
The effect of Thrashing is: -

1.CPU becomes idle.
2.Decreasing the utilization increases the degree of multiprogramming and hence bringing more processes at a time which in fact increases the thrashing exponentially.

60.What is translation Look aside Buffer?

A translation lookaside buffer (TLB) is a memory cache that is used to reduce the time taken to access a user memory location. The TLB stores the recent translations of virtual memory to physical memory and can be called an address-translation cache.