M.E.S. COLLEGE OF ENGINEERING, PUNE-01

T.E. (Computer Engineering)

Semester I AY 2021-2022

ORAL QUESTION BANK (SPOSL)

---

1. **What is system programming?**

   Ans : 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.

2. **What is system software?**

Ans : System software is a type of computer program that is designed to run a computer's hardware and application programs. If we think of the computer system as a layered model, the system software is the interface between the hardware and user applications.

3. **What are the components of system programming?**

Ans : Components of system programming are: 1) Loader 2) Assembler 3) Compiler 4) Macro 5) Interpreter

4. **What is assembler?**

   Ans : The Assembler is a Software that converts an assembly language code to machine code. It takes basic Computer commands and converts them into Binary Code that Computer's Processor can use to perform its Basic Operations. These instructions are assembler language or assembly language.

5. **Define a macro?**

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

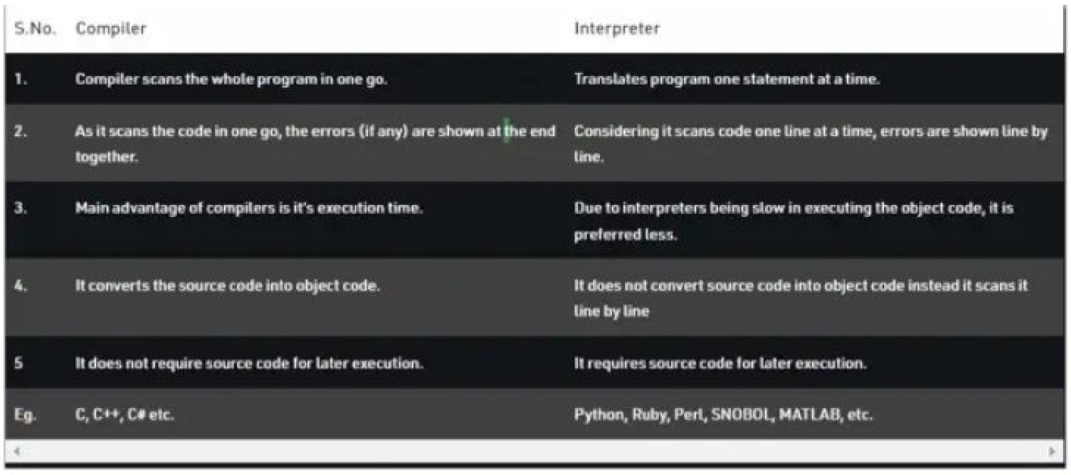
   · To simplify and reduce the amount of repetitive coding

   · To reduce errors caused by repetitive coding

   · To make an assembly program more readable.

6. **Difference between compiler and Interpreter?**

Ans:



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 it's 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?**

Ans : Linker is a program in a system which helps to link a object modules of program into a single object file. It performs the process of linking. Linking is performed at both compile time, when the source code is translated into machine code and load time, when the program is loaded into memory by the loader

8. **What is loader ?**

Ans: A loader is a system program, which takes the object code of a program as input and prepares it for execution.

9. **What is text editor?**

Ans: A text editor is a tool that allows a user to create and revise documents in a computer.

10. **Explain different component of text editor?**

1. Ans : 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.**

Ans: 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 loader?

Ans: Types of Loader :-
There are eight(8) general loader schemes available. But generally main loader scheme is four(1,2,3,4). Those are –

1. Absolute Loader.
2. Relocating Loader.

3. Direct Linking Loader.
4. Dynamic Loader.
5. Assemble – and – go or Compile – and – go loader.
6. Boot Strap Loader.
7. Linking Loader.
8. Relocation Loader.

Absolute Loader :- It is a sim placed type of loader scheme. It this scheme the loader simply accepts the machine language code produced by assembler and place it into main memory at the location specified by the assembler. The task of an absolute loader is virtually trivial. Absolute loader is simply to implemented but it has several disadvantage –

First : The programmer must specify to the assembler the address in main memory where the program is to be loaded.
Second : The programmer must remember the address of multiple sub – programs and there performance.

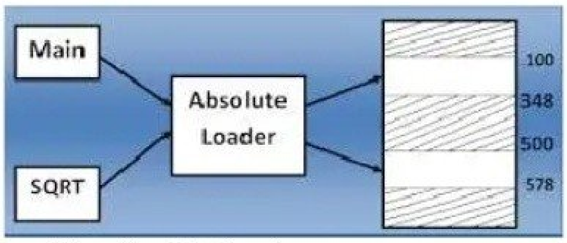


Fig - Absolute Loader

Relocating Loader :- To avoid possible reassembling of all subroutines , When a single subroutines is changed and to performed the task of allocation and linking of the programmer. The general class of relocating loader was introduced.

An example of relocating loader is Binary Symbolic Subroutine (BSS) loader, IBM 7294, IBM 1130, GE635 are the BSS loader.
Output of a relocating loader is the object program and information about all other programs its references. In addition ,there is relocating information has two location in this program that need to be changed. If it is to be loaded in an arbitrary location in memory.

Direct Linking Loader :- Direct linking loader is a general relocatable loader and perhaps the most popular loading scan presently used. It has the advantage of allowing the programmer multiple procedure segments and multiple data segments. It has also an advantage that has given to the programmer complete freedom in referencing data or instructions content in other segments. This provides flexibility of inter segment referencing and accessing while at the same time allowing independent translation of programs.

Dynamic loader :- Dynamic loader is the Persian of loader that actually intersect the "calls" and loads the necessary procedure is called over lay super visor or simply flipper. This over all scheme is called Dynamic Loading or Load on Call.
An advantage id Dynamic Loader is that, no over head is in corrected unless the procedure to be called or referenced is actually used. Also the system can be dynamically re – configured.
The major draw back is occurred due to the fact that we here postponed most of the binding process until execution time.

Assemble – and – go or Compile – and – go loader :-

Any method of performing the loader function is to have the assembler run in one of the part of memory and place the machine instructions in data directly into assigned memory location. This is a simply solution, involving mode any extra procedure. It is used by the WATFOR FORTAN compiler and other several language processor (PL/I). Such a loading scheme is commonly called compile – and – go and assemble – and – go loader.

Boot Strap Loader :- When a computer is first turned on or restarted a special type of absolute loader is executed , called Boot Strap Loader. It loads the operation system into the main memory and executes the related programs. It is added to the beginning of all object programs that are to be loaded into and empty ideal system.

Linking Loader :- The need for linking a program with other programs assign because a program written by a programmer or its translated version is really of a stand alone nature. That is a program generally cannot executed on its own. Without requiring the presents of some other programs is computer memory. The standard function must reside into the main memory. The linking function next address of program known to each other, So that such transfers can take place during the execution.

Relocation Loader: - Another function commonly performed by a loader is that of program re – location. Relocation is simply moving a program from one area to another in the storage. It referred to adjustment of address field and not to movement of a program. The task of relocation is to add some constant value to each relative address in the segment the part of a loader which performed relocation is called re – location loader.

13. What is input and output for pass1?

Ans : The assembler takes the same code for pass 1 and pass 2, but during pass 1 it builds the symbol table, so it fills out the values during the next pass.

```
BULLET_X EQU 10
BULLET_Y EQU 15

ldx #BULLET_X
ldy #BULLET_Y
jsr shoot
....
shoot:
...
```

So for example in the assembly language fragment above (6502 code), that values of BULLET_X and BULLET_Y are known when the assembler gets to the instruction, but the address of shoot is not known. The simplest way is for 2 passes and on the first pass, only the current address is calculated (so ldx #BULLET_X will add 2 to PC, since it takes 2 bytes to code), and the jsr shoot will take 3 bytes, and it saves the current pc when it gets to the definition of the shoot label. During the next pass it the outputs the actual byte codes, there were some assemblers that actually output jsr 0000 and then patched that value when the shoot value is encountered.

Ouput of pass1 assembler : Multipass assembler means more than one pass is used by assembler. Multipass assembler is used to eliminate forward references in symbol definition. ... In pass one we find out all the Symbols and Literals. And in pass two we will perform assembling of code and the data (generating instruction and generating data).

14. What is input and output for pass2?

Ans :**Input :** Implement Pass-II of two pass assembler for pseudo-machine in Java using object oriented features. The output of assignment-1 (intermediate file and symbol table) should be input for this assignment.

Output : Pass-2 of assembler generates machine code by converting symbolic machine-opcodes into their respective bit configuration(machine understandable form). It stores all machine-opcodes in MOT table (op-code table) with symbolic code, their length and their bit configuration.

15. Explain the following

1. **LTORG**

Ans : Use the LTORG instruction so that the assembler can collect and assemble literals into a literal pool. 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. If a control section has not been established, LTORG initiates an unnamed (private) control section.

2. **Dc**

Ans : 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. The DC instruction's name — Define Constant — is misleading: DC simply creates initial data in an area of the program.

3. **DS**

Ans : DS is called data segment register. It points to the segment of the data used by the running program. You can point this to anywhere you want as long as it contains the desired data. ES is called extra segment register.

16. what are types of statement in assembly?

Ans : Kinds of statements in assembly language
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.
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 ?

Ans : Imperative Statements :- An imperative statement is instruction in assembly program. Every imperative statement generates one machine instruction. e.g A. ... START < constant > :- This directive indicates that the first word of the target program will start on ROM memory location with address < constant >

18. What is assembler directive?

Ans : Assembler directives are instructions that direct the assembler to do something. ... This is used to set the program or register address during assembly. For example, ORG 0100h tells the assembler to assemble all subsequent code starting at address 0100h. DS. Defines an amount of free space.

19. Explain the following

1. **EQU**

   Ans: EQU is used as a substitute for labels or symbols.

2. **ORIGIN**

   Ans : The origin directive tells the assembler where to load instructions and data into memory. The syntax of this directive is. ORIGIN < address spec > This directive indicates that LC (location counter) should be set to the address given by. < address spec >. The ORIGIN statement is useful when the target program does not consist of consecutive memory words

3. **START**

   Ans : This instruction starts the execution of program from location 200 and label with START provides name for the program

4. **END**

Ans : The END statement is the last line in the main program and in all subroutines or functions. Since an END statement is a non-executable statement, the program cannot branch to an END statement.

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

Ans: **Pass 1 Data Structures**

1. Input source program

2. A Location Counter (LC), used to keep track of each instruction's location.

3. A table, the Machine-operation Table (MOT), that indicates the symbolic mnemonic, for each instruction and its length (two, four, or six bytes)

4. A table, the Pseudo-Operation Table (POT) that indicates the symbolic mnemonic and action to be taken for each pseudo-op in pass 1.

5. A table, the Symbol Table (ST) that is used to store each label and its corresponding value.

6. A table, the literal table (LT) that is used to store each literal encountered and its corresponding assignment location.

7. A copy of the input to be used by pass 2.

Pass 2 Data Structures

1. Copy of source program input to pass1.

2. Location Counter (LC)

3. A table, the Machine-operation Table (MOT), that indicates for each instruction, symbolic mnemonic, length (two, four, or six bytes), binary machine opcode and format of instruction.

4. A table, the Pseudo-Operation Table (POT), that indicates the symbolic mnemonic and action to be taken for each pseudo-op in pass 2.

5. A table, the Symbol Table (ST), prepared by pass1, containing each label and corresponding value.

6. A Table, the base table (BT), that indicates which registers are currently specified as base registers by USING pseudo-ops and what the specified contents of these registers are.

7. A work space INST that is used to hold each instruction as its various parts are being assembled together.

8. A work space, PRINT LINE, used to produce a printed listing.

9. A work space, PUNCH CARD, used prior to actual outputting for converting the assembled instructions into the format needed by the loader.

10. An output deck of assembled instructions in the format needed by the loader.

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

Ans **Symbol Table** : A symbol table is a data structure used by a language translator such as a compiler or interpreter, where each identifier (or symbol) in a program's source code is associated with information relating to its declaration or appearance in the source.

Literal Table : Literal table is used for keeping track of literals that are encountered in the programs. • We directly specify the value, literal is used to give a location for the value. • Literals are always encountered in the operand field of an instruction.

Pool Table : Awareness of different literal pools is maintained using the auxiliary table POOLTAB.

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

Ans: The advantage of absolute loader is simple and efficient. But the disadvantages are, the need for programmer to specify the actual address, and, difficult to use subroutine libraries.

23.What is compiler?

Ans : A compiler is a program that translates a high-level language (for example, C, C++, and Java) into a low-level language (object program or machine program).

24.Explain different phases of compiler?

Ans : The compilation process contains the sequence of various phases. Each phase takes source program in one representation and produces output in another representation. Each phase takes input from its previous stage.

There are the various phases of compiler:

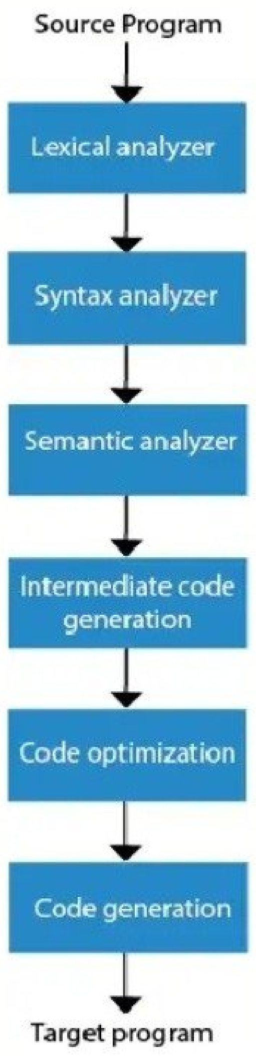


Fig: phases of compiler

Lexical Analysis:

Lexical analyzer phase is the first phase of compilation process. It takes source code as input. It reads the source program one character at a time and converts it into meaningful lexemes. Lexical analyzer represents these lexemes in the form of tokens.

Syntax Analysis

Syntax analysis is the second phase of compilation process. It takes tokens as input and generates a parse tree as output. In syntax analysis phase, the parser checks that the expression made by the tokens is syntactically correct or not.

Semantic Analysis

Semantic analysis is the third phase of compilation process. It checks whether the parse tree follows the rules of language. Semantic analyzer keeps track of identifiers, their types and expressions. The output of semantic analysis phase is the annotated tree syntax.

Intermediate Code Generation

In the intermediate code generation, compiler generates the source code into the intermediate code. Intermediate code is generated between the high-level language and the machine language. The intermediate code should be generated in such a way that you can easily translate it into the target machine code.

Code Optimization

Code optimization is an optional phase. It is used to improve the intermediate code so that the output of the program could run faster and take less space. It removes the unnecessary lines of the code and arranges the sequence of statements in order to speed up the program execution.

Code Generation

Code generation is the final stage of the compilation process. It takes the optimized intermediate code as input and maps it to the target machine language. Code generator translates the intermediate code into the machine code of the specified computer.

25.What is conditional macro?

Ans : Conditional assembly are frequently considered to be mechanisms that allow a single version of the source code for a program to be used to generate multiple versions of the executable. ... It is also referred to as conditional macro expansion. Conditional Assembly can be achieved with the help of AIF and AGO statements.

26 How to define micro?

Ans 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'.

Example of MACRO :-

```
        Start of definition  - MACRO

Macro name   ( Prototype ) – My MACRO

Macro body   (Model Statements ) – ADD AREG, X

                                   ADD BREG, X

End of Macro Definition      -     MEND
```

27.How to call macro?

Ans < MACRO NAME > [<ACTUAL Parameters >]

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

Ans : https://www.wikinote.org/mod/page/view.php?id=350

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

30.What is dynamic linking?

Ans: Dynamic linking consists of compiling and linking code into a form that is loadable by programs at run time as well as link time. The ability to load them at run time is what distinguishes them from ordinary object files.

31. Explain following

a. **Compile and Go,**
b. **General Loader Scheme,**
c. **Absolute Loaders,**
d. **Relocating Loaders,**
e. **Direct linking Loaders,**

32. Difference between absolute loader and relocating loader?

ans: 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?

Ans: The main difference between static and dynamic linking is that static linking copies all library modules used in the program into the final executable file at the final step of the compilation while in dynamic linking, the linking occurs at run time when both executable files and libraries are placed in the memory

34. What is OS?

Ans: An operating system is a program that acts as an intermediary between the user and the computer hardware. The purpose of an OS is to provide a convenient environment in which user can execute programs in a convenient and efficient manner. It is a resource allocator responsible for allocating system resources and a control program which controls the operation of the computer h/w.

35. What is function of OS?

Ans: An operating system has three main functions:

i. manage the computer's resources, such as the central processing unit, memory, disk drives, and printers,
ii. establish a user interface, and
iii. execute and provide services for applications software.

36. **Explain process state model?**

Ans: There are various types of process state model. Process state defines current state of a process. The process states are new, ready, running, waiting, suspended waiting, terminated etc.

New state: The process is being created.
Ready state: The process is ready to run, but waiting to be assigned a processor.
Running state: Program is running in other words instructions are being executed.
Waiting state: Process preempted and switched from running state to waiting state because of process is waiting for some event to occur such as I/O completion.
Terminated state: This state is final state, when process is completed or finished execution it moves from running state to terminated state.

37.What is Process control block,?

Ans: Process Control Block is a data structure that contains information of the process related to it. The process control block is also known as a task control block, entry of the process table, etc.
It is very important for process management as the data structuring for processes is done in terms of the PCB. It also defines the current state of the operating system.

38.What is thread?
Ans: A thread is a path of execution within a process. A process can contain multiple threads.

39.Difference between process and thread?

Ans: The primary difference is that threads within the same process run in a shared memory space, while processes run in separate memory spaces.
Threads are not independent of one another like processes are, and as a result threads share with other threads their code section, data section, and OS resources (like open files and signals). But, like process, a thread has its own program counter (PC), register set, and stack space.

40.Explain thread life cycle?
Ans: A thread goes through various stages in its lifecycle. For example, a thread is born, started, runs, and then dies. The following diagram shows the complete life cycle of a thread.

1. **NEW** – a newly created thread that has not yet started the execution
2. **RUNNABLE** – either running or ready for execution but it's waiting for resource allocation
3. **BLOCKED** – waiting to acquire a monitor lock to enter or re-enter a synchronized block/method
4. **WAITING** – waiting for some other thread to perform a particular action without any time limit
5. **TIMED_WAITING** – waiting for some other thread to perform a specific action for a specified period
6. **TERMINATED** – has completed its execution

41.What is multithreading?

Multithreading in Java is a process of executing multiple threads simultaneously.
however, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process.
Java Multithreading is mostly used in games, animation, etc

42.What is preemptive scheduling?

Ans: In this, a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state. When scheduling takes from either of below circumstances it is *preemptive scheduling* –

- When a process switches from the running state to the ready state (for example, when an interrupt occurs).
- When a process switches from the waiting state to the ready state (for example, at completion of I/)).

43. What is nonpreemptive scheduling?

Ans: In this, once a process enters the running state it cannot be preempted until it completes it's allocation time. When scheduling takes place only under below circumstances we say that the scheduling scheme is *non-preemptive* or *cooperative*. –

- When a process switches from the running state to the waiting state (for example, as the result of an I/O request or an invocation of wait() for the termination of a child process).
- When a process terminates.

44. Explain in short following Scheduling Algorithms:

1. **FCFS:**

   The FCFS is implemented with the help of a FIFO queue. The processes are put into the ready queue in the order of their arrival time. The process that arrives first becomes the head of the queue while the others that arrive after are added to the rear of the queue. In First Come First Served (FCFS) algorithm, the process that arrives first, is sent first for execution by the CPU when CPU is free

2. **SJF:**

   In this algorithm, the process with the least burst time is processed first. The burst time of only those processes is compared that are present or have arrived until that time. It is also non-preemptive in nature. Its preemptive version is called Shortest Remaining Time First (SRTF) algorithm.

3. **RR:**

   Round Robin(RR) scheduling algorithm is mainly designed for time-sharing systems. This algorithm is similar to FCFS scheduling, but in Round Robin(RR) scheduling, preemption is added which enables the system to switch between processes.

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?

Ans: 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,**

2. **Producer Consumer problem,**
3. **Dining Philosopher problem.**

47. What is deadlock ?

Ans: 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,**
2. **Deadlock avoidance**
3. **Deadlock detection,**
4. **Deadlock recovery.**

Ans: 1) Deadlock prevention: The idea is to not let the system into a deadlock state. Prevention is done by negating one of above mentioned necessary conditions for deadlock.
2) Deadlock avoidance is kind of futuristic in nature. By using strategy of "Avoidance", we have to make an assumption. We need to ensure that all information about resources which process will need are known to us prior to execution of the process. We use Banker's algorithm (Which is in-turn a gift from Dijkstra) in order to avoid deadlock.

3) Deadlock detection and recovery: Let deadlock occur, then do pre emption to handle it once occurred. Ignore the problem altogether: If deadlock is very rare, then let it happen and reboot the system. This is the approach that both Windows and UNIX take.

49. Explain the memory management?

50. What is Fixed Partitioning, Dynamic Partitioning?

Ans: **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. Here partitions are made before execution or during system configure.
Variable Partitioning :
It is a part of Contiguous allocation technique. It is used to alleviate the problem faced by Fixed Partitioning. In contrast with fixed partitioning, partitions are not made before the execution or during system configure.

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

- **First Fit:** In the first fit, the partition is allocated which is first sufficient from the top of Main Memory.
- **Best Fit:** Best fit allocates the process to a partition which is the smallest sufficient partition among the free available partitions.
- **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 ?

Ans : 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.

53.what is segmnetation?

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

54.what is page table

Ans : 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

Ans: 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?

Ans : 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

Ans : 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 page replacement(OPR).

Ans : **FIFO** : Advantages of 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. LRU : 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?

Ans : thrashing occurs when a computer's virtual memory resources are overused, leading to a constant state of paging and page faults, inhibiting most application-level processing.[1] This causes the performance of the computer to degrade or collapse. The situation can continue indefinitely until either the user closes some running applications or the active processes free up additional virtual memory resources.

60. What is translation Look aside Buffer?

Ans : 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.