Total No. of Questions : 10] SEAT No. : 237

P3984 [Total No. of Pages : 3

[5353]-587

T.E. (Computer Engineering) (Semester - II)

SYSTEM PROGRAMMING AND OPERATING SYSTEM

(2015 Pattern)

Time : 2½ Hours] *[Max. Marks : 70*

Instructions to the candidates:

1) *Solve Q.1 or Q.2, Q.3 or Q.4, Q.5 or Q.6, Q.7 or Q.8, Q.9 or Q.10.*
2) *Neat diagrams must be drawn whenever necessary.*
3) *Figures to the right indicate full marks.*
4) *Assume suitable data if necessary.*

SPOS MODEL ANSWER MAY – 2018

Q 1. a) Write Algorithm of pass I of two pass assembler. [5]

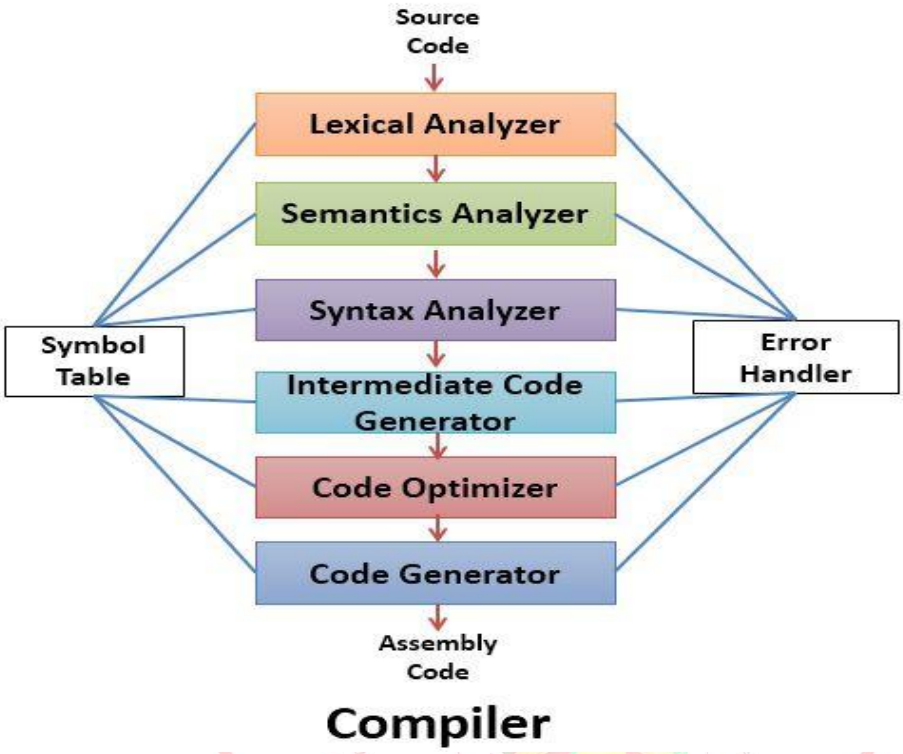
Ans :- begin

```
    if starting address is given
        LOCCTR = starting address;
    else
        LOCCTR = 0;
    while OPCODE != END do          ;; or EOF
        begin
        read a line from the code
        if there is a label
            if this label is in SYMTAB, then error
            else insert (label, LOCCTR) into SYMTAB
        search OPTAB for the op code
        if found
            LOCCTR += N         ;; N is the length of this instruction (4 for MIPS)
        else if this is an assembly directive
            update LOCCTR as directed
        else error
        write line to intermediate file
        end
    program size =  LOCCTR - starting address;
end
```

b) What is Compiler? Explain any two phases of compiler with suitable example [5]

Ans : Compiler is program which converts high level language in to the low level language as target program.

Phases of Compiler :



***Lexical Analysis Phase*:**

- The lexical phase reads the characters in the source program and groups them into a stream of tokens in which each token represents a logically cohesive sequence of characters, such as, An identifier, A keyword, A punctuation character.
- The character sequence forming a token is called the lexeme for the token.

***Syntax Analysis Phase*:**

- Syntax analysis imposes a hierarchical structure on the token stream. This hierarchical structure is called syntax tree.
- A syntax tree has an interior node is a record with a field for the operator and two fields containing pointers to the records for the left and right children.
- A leaf is a record with two or more fields, one to identify the token at the leaf, and the other to record information about the token.

***Error Detecting and Reporting*:**

- Each phase encounters errors.
- Lexical phase determine the input that do not form token.
- Syntax phase determine the token that violates the syntax rule.
- Semantic phase detects the constructs that have no meaning to operand.

OR

Q 2. a) Explain in brief imperative statements, declaration statements and assembler directives with examples for assembly language programming. [5]

Ans :-

a) **Imperative Statements :** It indicates an action to be performed during the execution of the assembled program. Each imperative statement typically translates into one machine instruction.

b) **Declaration Statements :** Two types of declaration statements is as follows

[Label] DS

[Label] DC

<constant>'<Value>'

The DS (Declare Storage) statement reserves areas of memory and associates names with them.

Eg)**A DS 1**

B DS 150

First statement reserves a memory of 1 word and associates the name of the memory as A.

Second statement reserves a memory of 150 word and associates the name of the memory as B

The DC (Declare Constant) Statement constructs memory word containing constants.

Eg) **ONE DC '1'**

Associates the name ONE with a memory word containing the value '1' . The programmer can declare constants in decimal,binary, hexadecimal forms etc., These values are not protected by the assembler. In the above assembly language program the value of ONE Can be changed by executing an instruction MOVEM BREG,ONE

c. **Assembler Directives :**

Assembler directives instruct the assembler to perform certain actions during the assembly of a program. Some Assembler directives are described in the following

START <Constant>

Indicates that the first word of the target program generated by the assembler should be placed in the memory word with address <Constant>

END [<operand spec>]

It Indicates the end of the source program

b) Explain Pass – I Direct linking loader with flowchart. [5]

Ans :-

Design of direct linking loader (DLL)

The direct linking loader has the advantage of allowing the programmer multiple procedure segments and multiple data segments and allows complete freedom in referencing data or instructions contained in other segments. This provides flexible intersegment referencing and accessing ability ,while at the same time allowing independent translation of programs.

The translator must give the loader the following information with each procedure or data segment.

1. The length of segment
2. A list of all symbols in the segment that may be referenced by other segment and their relative location with in the segment
3. A list of all symbols not defined in the segment but referenced in the segment
4. Information as to where address constants are located in the segment and a description of how to revise their values
5. The machine code translation of the source program and the relative address assigned.

There are four sections to the object desk

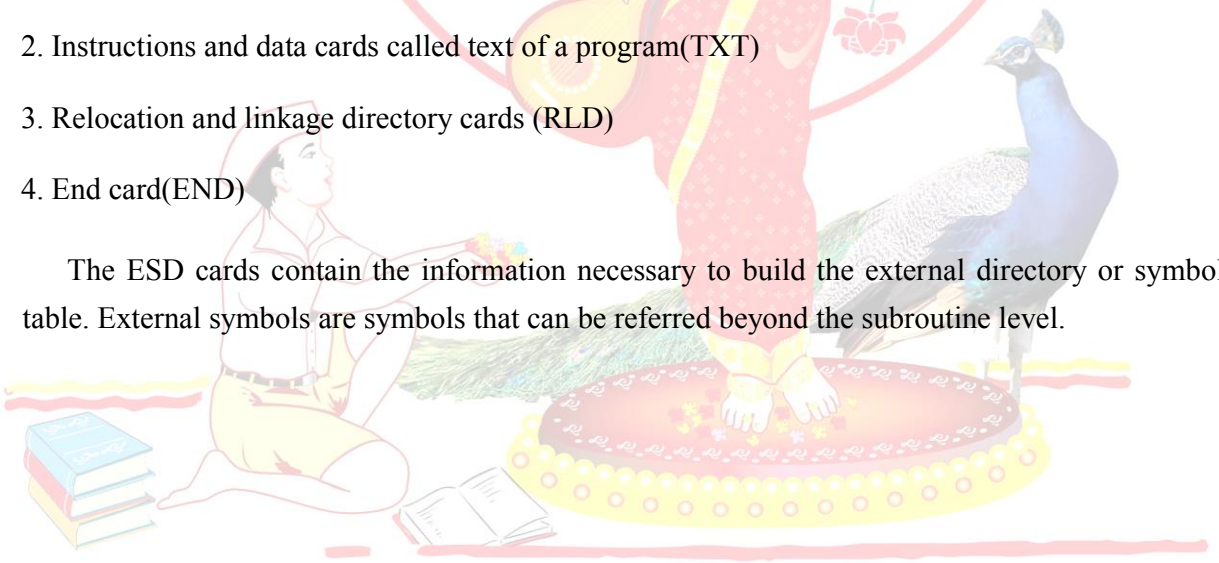
1. External symbol directory cards(ESD)

2. Instructions and data cards called text of a program(TXT)

3. Relocation and linkage directory cards (RLD)

4. End card(END)

The ESD cards contain the information necessary to build the external directory or symbol table. External symbols are symbols that can be referred beyond the subroutine level.



Direct Linking Loader Flowchart

Q 3. a) Whar are the data structure used in the design of macro processor? [6]

Ans :-

Pass 1

Uses the following databases :

1. Source program as input
2. Source program without macro definition as output of pass1 and input of pass2.
3. Macro definition table (MDT)
4. Macro name table (MNT)
5. Argument list array (ALA)
6. MNTP (macro name table pointer)
7. MDTP (macro definition table pointer)

A source program containing both macro definitions and macro calls is given as input to pass1 of microprocessor.

The MNT is used to store name of macros. The entire macro definition is stored in the MDT. The index into MDT (starting row number of this macro definition) is stored in MNT.

The argument list array (ALA) is used to substitute index markers for formal parameters before storing macro definition in MDT.

MNTP gives the next available entry in MNT.

MDTP gives the next available entry in MDT.

b) Explain Macro expansion with relevant example. **[4]**

.

Ans :-

Each call to a macro is replaced by its body.

During replacement, actual parameter is used in place of formal parameter.

- During macro expansion, each statement forming the body of the macro is picked up one by one sequentially.
- Each statement inside the macro may have :
  (1) An ordinary string, which is copied as it is during expansion.
  (2) The name of a formal parameter which is preceded by the character '&'.
- During macro expansion an ordinary string is retained without any modification. Formal parameters (string starting with &) is replaced by the actual parameter value.

```
MACRO
   INCR    &VARIABLE,
&INCR_BY,&USE_REG
   MOVER     &USE_REG, &VARIABLE
   ADD       &USE_REG, &INCR_BY
   MOVEM     &USE_REG, &VARIABLE
MEND
```

Fig. 2.2.2 : A macro definition

- Name of the macro is **INCR**
- There are three positional parameters
  These parameters are :
  (1) VARIABLE
  (2) INCR_BY
  (3) USE_REG
- The body of macro **INCR** contains three statements.

```
MACRO
      INCR &VARIABLE, &INCR_BY,&USE_REG
      MOVER     &USE_REG, &VARIABLE
      ADD       &USE_REG, &INCR_BY
      MOVEM     &USE_REG, &VARIABLE
MEND
 START        100
 READ         X
 READ         Y
 INCR         X, Y, AREG
 PRINT        X
 STOP
 X      DS     1
 Y      DS     1
 END
```

Fig. 2.2.3 : An assembly program with macro

The macro processor will process the program given in Fig. 2.2.3 as explained below.

(1) The statement **START 100** will be copied as it is.

(2) The statement **READ X** will be copied as it is.

(3) The statement **READ Y** will be copied as it is.

(4) The statement **INCR X, Y, AREG** is a call to macro. The macro **INCR** will be expanded there. Values of the formal parameters are :

Formal parameter	Value
VARIABLE	X
INCR_BY	Y
USE_REG	AREG

There are three statements in the body of the macro. During expansion of the macro, actual parameters will be used instead of formal parameters.

```
MOVER   &USE_REG, &VARIABLE        expansion      MOVER   AREG, X
ADD     &USE_REG, &INCR_BY                        ADD     AREG, Y
MOVEM   &USE_REG, &VARIABLE                       MOVEM   AREG, X
```

(5) Remaining statements

```
PRINT X
STOP
X   DS   1
Y   DS   1
END
```

will be retained without any modification.

OR

Q 4. a) Enlist different types of error handled by PASS I & PASS II of assembler. [5]

Ans :-

1. Syntax error like missing commas etc
2. Invalid opcode
3. Duplicate definition of symbol
4. Undefined symbol
5. Missing start statement
6. Missing end statement
7. Symbol defined but not used

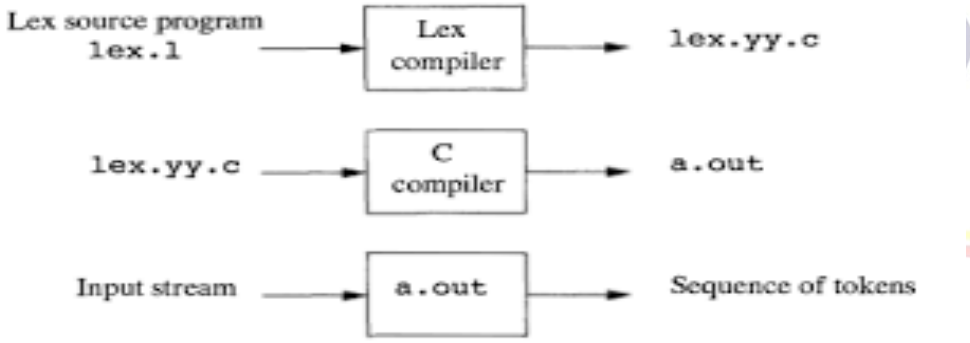
   Error like undefined symbol, missing start statement and missing end statement can only be reported at end of the source program.

b) What is LEX. Explain working of LEX? [5]

Ans : Lex is a compiler too which is used to scan source program from left to right to separate out the token from program**.**

Working of LEX :

Lex stands for Lexical Analyzer. Lex is a tool for generating Scanners. Scanners are programs that recognize lexical patterns in text. These lexical patterns (or regular Expressions) are defined in a particular syntax. A matched regular expression may have an associated action. This action may also include returning a token. When Lex receives input in the form of a file or text, it takes input one character at a time and continues until a pattern is matched, then lex performs the associated action (Which may include returning a token). If, on the other hand, no regular expression can be matched, further processing stops and Lex displays an error message.



It accept source program as lex.l file and lex compiler compile it to generate the taken and generate lex.yy.c file then lex.yy.c file is compiled by c compiler to generate a.out file

Lex scan whole program and compare source program with specific rules or pattern to separate tokens for yaac program.

OR

Q 5. Explain the following types of schedular. **[6]**

Ans :- Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types −

- Long-Term Scheduler
- Short-Term Scheduler
- Medium-Term Scheduler

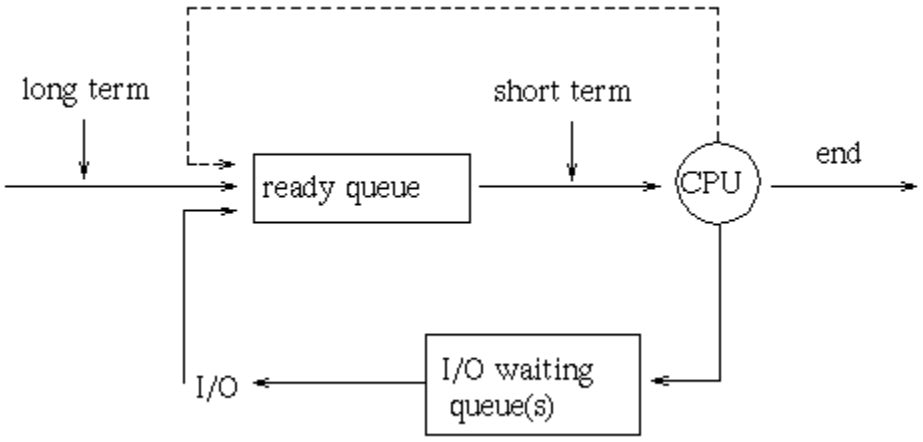
Long Term Scheduler :

It is also called a **job scheduler**. A long-term scheduler determines which programs are admitted to the system for processing. It selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling.

Short Term Scheduler :

It is also called as **CPU scheduler**. Its main objective is to increase system performance in accordance with the chosen set of criteria. It is the change of *ready state to running state* of the process. *CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.*

Short-term schedulers, also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are *faster* than long-term schedulers.



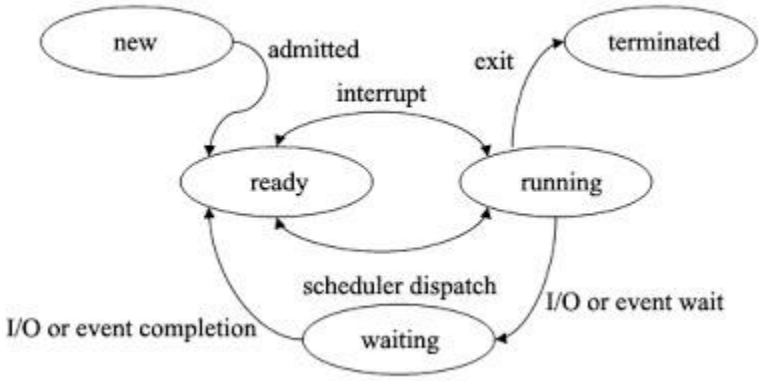
Medium Term Scheduler :

Medium-term scheduling is a part of **swapping**. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium-term scheduler is in-charge of handling the swapped out-processes.

A running process may become suspended if it makes an I/O request. A suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called **swapping**, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

b) Draw and explain Process State Transition diagrame. **[6]**

Ans :



1. New state: A process that just has been created but has not yet been admitted to the pool of execution processes by the operating system. Every new operation which is requested to the system is known as the new born process.

2. Ready state: When the process is ready to execute but he is waiting for the CPU to execute then this is called as the ready state. After completion of the input and output the process will be on ready state means the process will wait for the processor to execute.

3. Running state: The process that is currently being executed. When the process is running under the CPU, or when the program is executed by the CPU, then this is called as the running process and when a process is running then this will also provide us some outputs on the screen.

4. Waiting or blocked state: A process that cannot execute until some event occurs or an I/O completion. When a process is waiting for some input and output operations then this is called as the waiting state and in this process is not under the execution instead the process is stored out of memory and when the user will provide the input and then this will again be on ready state.

5. Terminated state: After the completion of the process, the process will be automatically terminated by the CPU. So this is also called as the terminated state of the process.

c) What is process. What is thread? List down benefit of using thread? **[6]**

Ans : A **process** is basically a program in execution. The execution of a process must progress in a sequential fashion.

A **thread** of execution is the smallest sequence of programmed instructions that can be managed independently by a scheduler, which is typically a part of the operating system.

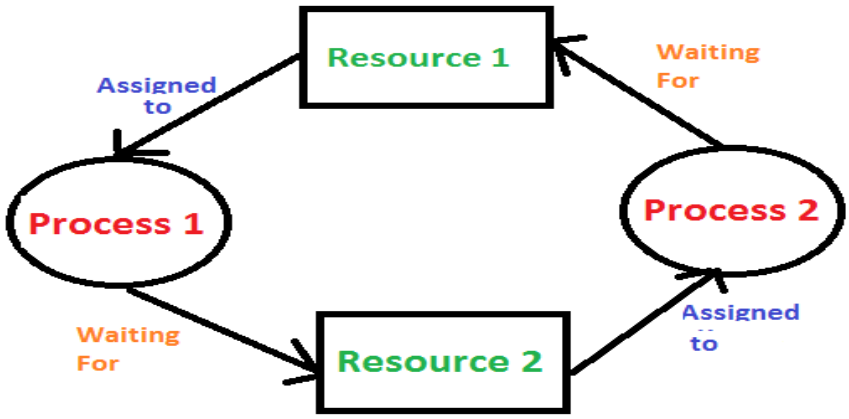
Benefit of thread :

- Threads minimize the context switching time.
- Use of threads provides concurrency within a process.
- Efficient communication.
- It is more economical to create and context switch threads.
- Threads allow utilization of multiprocessor architectures to a greater scale and efficiency.

OR

Q 6. a) What is deadlock? State and explain condition of deadlock. [8]

Ans : A **deadlock** is a situation in which two computer programs sharing the same resource are effectively preventing each other from accessing the resource, resulting in both programs ceasing to function. The earliest computer **operating systems** ran only one program at a time.



Deadlock conditions :

1. **Mutual Exclusion**
   Resources shared such as read-only files do not lead to deadlocks but resources, such as printers and tape drives, requires exclusive access by a single process.

2. **Hold and Wait**
   In this condition processes must be prevented from holding one or more resources while simultaneously waiting for one or more others.

3. **No Preemption**
   Preemption of process resource allocations can avoid the condition of deadlocks, where ever possible.

4. **Circular Wait**
   Circular wait can be avoided if we number all resources, and require that processes request resources only in strictly increasing(or decreasing) order.

b) Explain Process Control Block with suitable diagram. [6]

- **Ans : The process scheduling state:** The state of the process in terms of "ready", "suspended", etc., and other scheduling information as well, like priority value, the amount of time elapsed since the process gained control of the CPU or since it was suspended. Also, in case of a suspended process, event identification data must be recorded for the event the process is waiting for.

- **Process structuring information:** process's children id's, or the id's of other processes related to the current one in some functional way, which may be represented as a queue, a ring or other data structures.
- **Interprocess communication information:** various flags, signals and messages associated with the communication among independent processes may be stored in the PCB.
- **Process Privileges** in terms of allowed/disallowed access to system resources.
- **Process State:** State may enter into new, ready, running, waiting, dead depending on CPU scheduling.
- **Process Number (PID):** A unique identification number for each process in the operating system (also known as Process ID).
- **Program Counter (PC):** A pointer to the address of the next instruction to be executed for this process.
- **CPU Registers:** Indicates various register set of CPU where process need to be stored for execution for running state.
- **CPU Scheduling Information:** indicates the information of a process with which it uses the CPU time through scheduling.
- **Memory Management Information:** includes the information of page table, memory limits, Segment table depending on memory used by the operating system.
- **Accounting Information:** Includes the amount of CPU used for process execution, time limits, execution ID etc.
- **I/O Status Information:** Includes a list of I/O devices allocated to the process.

c. Explain interprocess communicatikon **[4]**

Ans : Inter Process Communication (IPC) is a mechanism that involves communication of one process with another process. This usually occurs only in one system.

Communication can be of two types −

- Between related processes initiating from only one process, such as parent and child processes.
- Between unrelated processes, or two or more different processes.

Following are some important terms that we need to know before proceeding further on this topic.

Pipes − Communication between two related processes. The mechanism is half duplex meaning the first process communicates with the second process. To achieve a full duplex i.e., for the second process to communicate with the first process another pipe is required.

FIFO − Communication between two unrelated processes. FIFO is a full duplex, meaning the first process can communicate with the second process and vice versa at the same time.

Message Queues − Communication between two or more processes with full duplex capacity. The processes will communicate with each other by posting a message and retrieving it out of the queue. Once retrieved, the message is no longer available in the queue.

Shared Memory − Communication between two or more processes is achieved through a shared piece of memory among all processes. The shared memory needs to be protected from each other by synchronizing access to all the processes.

Semaphores − Semaphores are meant for synchronizing access to multiple processes. When one process wants to access the memory (for reading or writing), it needs to be locked (or protected) and released when the access is removed. This needs to be repeated by all the processes to secure data.

Signals − Signal is a mechanism to communication between multiple processes by way of signaling. This means a source process will send a signal (recognized by number) and the destination process will handle it accordingly.

- **Shared memory**
- **Message passing**
  e.g. producer consumer problem
  reader-writer problem
  dining philosopher problem

.

Q 7. a) Explain the following terms in brief . **[8]**

Ans :

Virtual Memory :- Virtual Memory is a storage allocation scheme in which secondary memory can be addressed as though it were part of main memory. The addresses a program may use to reference memory are distinguished from the addresses the memory system uses to identify physical storage sites, and program generated addresses are translated automatically to the m/c adderess

The size of virtual storage is limited by the addressing scheme of the computer system and amount of secondary memory is available not by the actual number of the main storage locations.

It is a technique that is implemented using both hardware and software. It maps memory addresses used by a program, called virtual addresses, into physical addresses in computer memory.

1. All memory references within a process are logical addresses that are dynamically translated into physical addresses at run time. This means that a process can be swapped in and out of main memory such that it occupies different places in main memory at different times during the course of execution.
2. A process may be broken into number of pieces and these pieces need not be continuously located in the main memory during execution. The combination of dynamic run-time addres translation and use of page or segment table permits this.

If these characteristics are present then, it is not necessary that all the pages or segments are present in the main memory during execution. This means that the required pages need to be loaded into memory whenever required. Virtual memory is implemented using Demand Paging or Demand Segmentation.

Compaction :- Compaction is a process in which the free space is collected in a large memory chunk to make some space available for processes.

- In memory management, swapping creates multiple fragments in the memory because of the processes moving in and out.
- Compaction refers to combining all the empty spaces together and processes.
- Compaction helps to solve the problem of fragmentation, but it requires too much of CPU time.
- It moves all the occupied areas of store to one end and leaves one large free space for incoming jobs, instead of numerous small ones.
- In compaction, the system also maintains relocation information and it must be performed on each new allocation of job to the memory or completion of job from memory.

Thrasing :

process that is spending more time paging than executing is said to be thrashing. In other words it means, that the process doesn't have enough frames to hold all the pages for its execution, so it is swapping pages in and out very frequently to keep executing. Sometimes, the pages which will be required in the near future have to be swapped out.

Belady's anomaly :

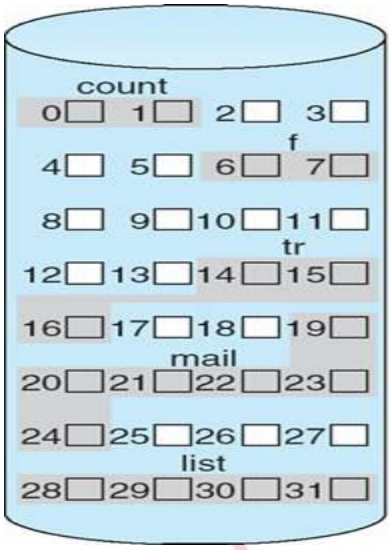
Also called FIFO anomaly. Usually, on increasing the number of frames allocated to a process virtual memory, the process execution is faster, because fewer page faults occur. Sometimes, the reverse happens, i.e., the execution time increases even when more frames are allocated to the process. This is Belady's Anomaly. This is true for certain page reference patterns.

b) Explain Contiguous and Non-contiguous memory allocation policies with suitable example . [8]

Ans :

Contiguous memory allocation is a classical memory allocation model that assigns a process consecutive memory blocks (that is, memory blocks having consecutive addresses).

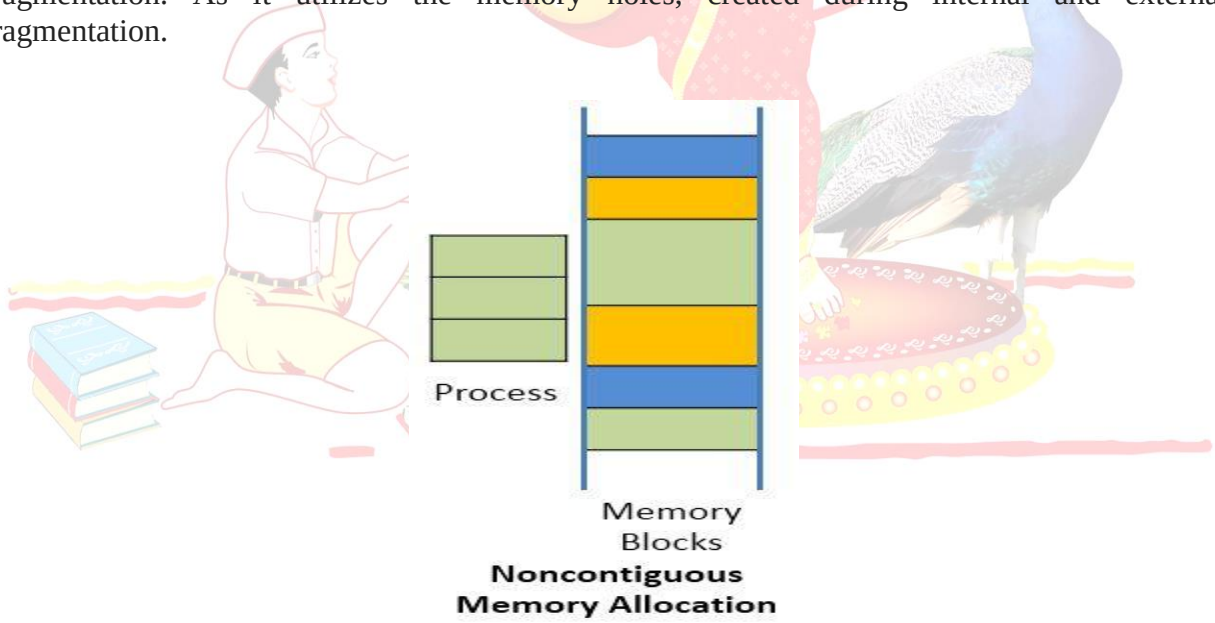
Contiguous memory allocation is one of the oldest memory allocation schemes. When a process needs to execute, memory is requested by the process. The size of the process is compared with the amount of contiguous main memory available to execute the process. If sufficient contiguous memory is found, the process is allocated memory to start its execution. Otherwise, it is added to a queue of waiting processes until sufficient free contiguous memory is available.



directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

The Non-contiguous memory allocation allows a process to **acquire the several memory blocks at the different location in the memory** according to its requirement. The noncontiguous memory allocation also **reduces** the **memory wastage** caused due to internal and external fragmentation. As it utilizes the memory holes, created during internal and external fragmentation.



Paging and segmentation are the two ways which allow a process's physical address space to be non-contiguous. In non-contiguous memory allocation, the process is divided into **blocks** (pages or segments) which are placed into the different area of memory space according to the availability of the memory.

The noncontiguous memory allocation has an advantage of reducing memory wastage but, but it **increases** the **overheads** of address translation. As the parts of the process are placed in a different location in memory, it **slows the execution** of the memory because time is consumed in address translation.

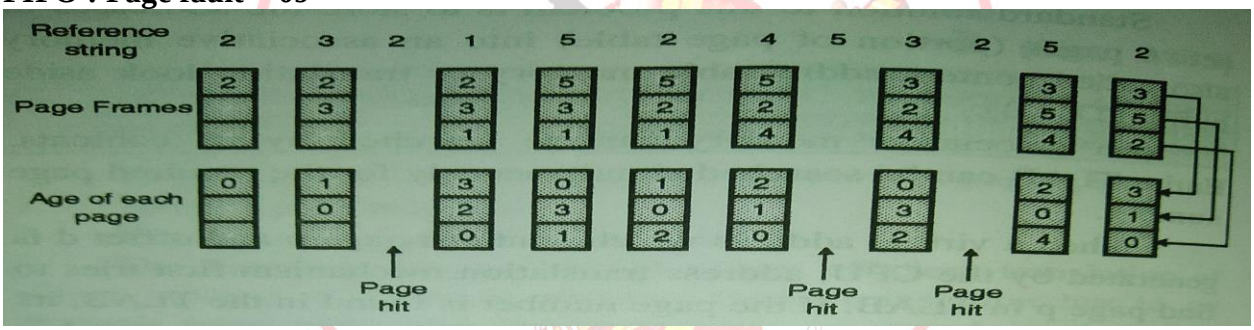
OR

Q 8. a) Consider page sequence 2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2 ans discuss working of following page replacement policies. Also count page faults (frame size – 3). [9]

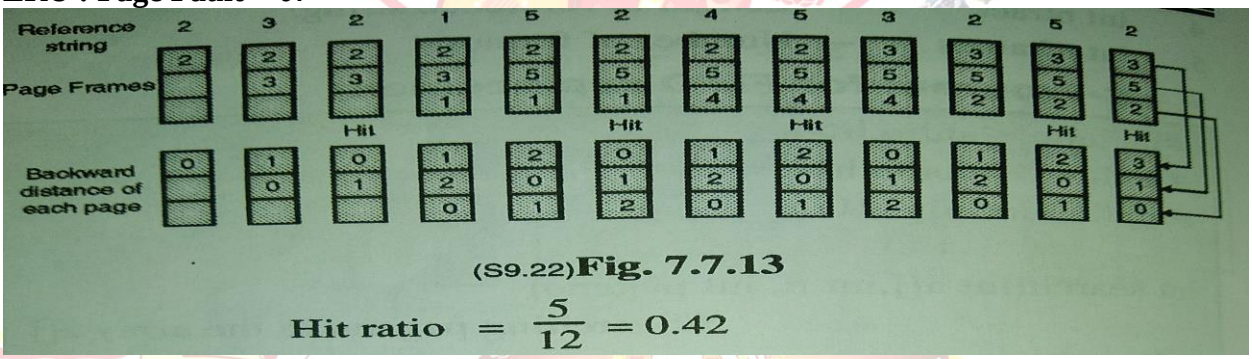
1. **FIFO**
2. **LRU**
3. **Optimal**

Ans :-

1. **FIFO : Page fault = 09**



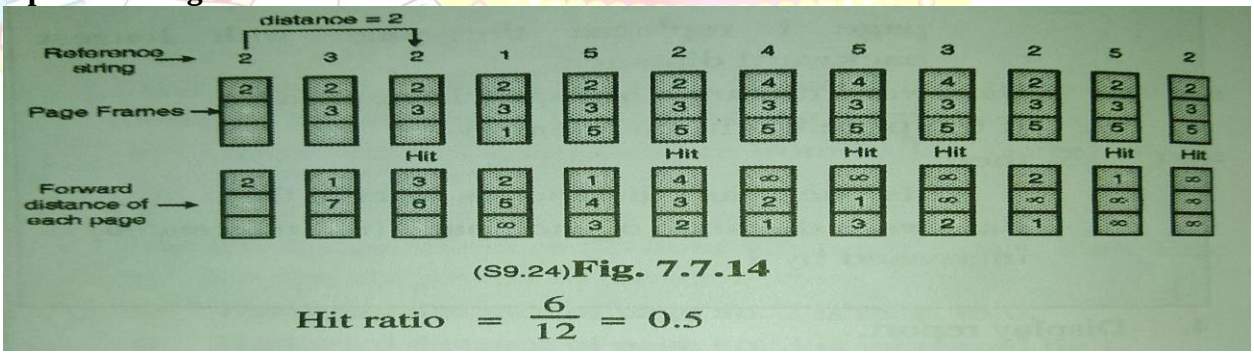
2. **LRU : Page Fault = 07**



(S9.22)**Fig. 7.7.13**

$$\text{Hit ratio} = \frac{5}{12} = 0.42$$

3. **Optimal : Page Fault = 06**



(S9.24)**Fig. 7.7.14**

$$\text{Hit ratio} = \frac{6}{12} = 0.5$$

b) Differentiate internal and external fragmentation. **[4]**

Ans :

The difference between the memory allocated and the required memory is called internal fragmentation.	The unused spaces formed between non-contiguous memory fragments are too small to serve a new process request. This is called external fragmentation.
It occurs when main memory is divided into fixed-size blocks regardless of the size of the process.	It occurs when memory is allocated to processes dynamically based on process requests.
It refers to the unused space in the partition which resides within an allocated region, hence the name.	It refers to the unused memory blocks that are too small to handle a request.
Internal fragmentation can be eliminated by allocating memory to processes dynamically.	External fragmentation can be eliminated by compaction, segmentation, and paging.

c) What is thrasing? **[3]**

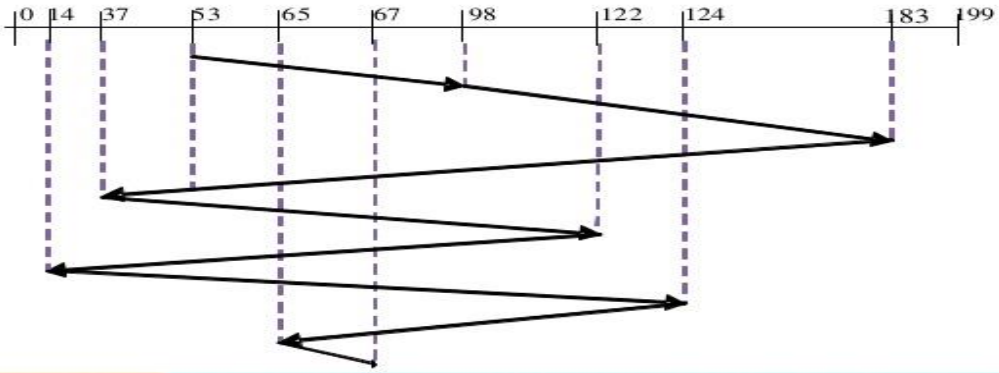
Ans : process that is spending more time paging than executing is said to be thrashing. In other words it means, that the process doesn't have enough frames to hold all the pages for its execution, so it is swapping pages in and out very frequently to keep executing. Sometimes, the pages which will be required in the near future have to be swapped out.

Q 9. a) Compare the performance of given schedulling policies like FCFS, SSTF, SCAN, C-SCAN considering content of queue as

Queue : 98, 183, 37, 122, 14, 124, 65, 67. Head starts at 53. **[12]**

Ans :

FCFS



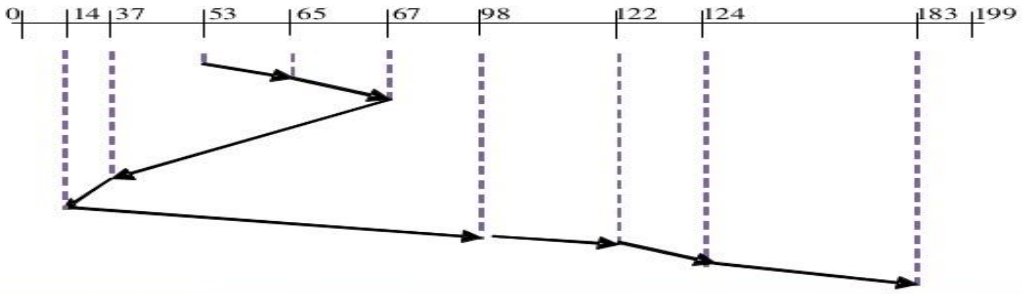
USING FCFS SCHEDULING TOTAL HEAD MOVEMENT IS:

= (98-53)+(183-98)+(183-37)+(122-37)+(122-14)+(124-14)+(124-65)+(67-65)

= 45+ 85+146+85+108+110+59+2

= 640 cylinders.

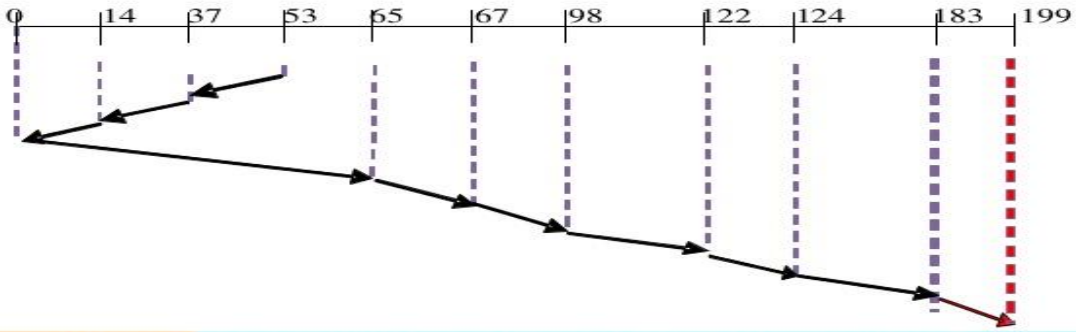
SSTF



USING SSTF SCHEDULING TOTAL HEAD MOVEMENT IS:

= (65-53)+(67-65)+(67-37)+(37-14)+(98-14)+(122-98)+(124-122)+(183-124)
= 12+2+30+23+84+24+2+59
= 236 cylinders.

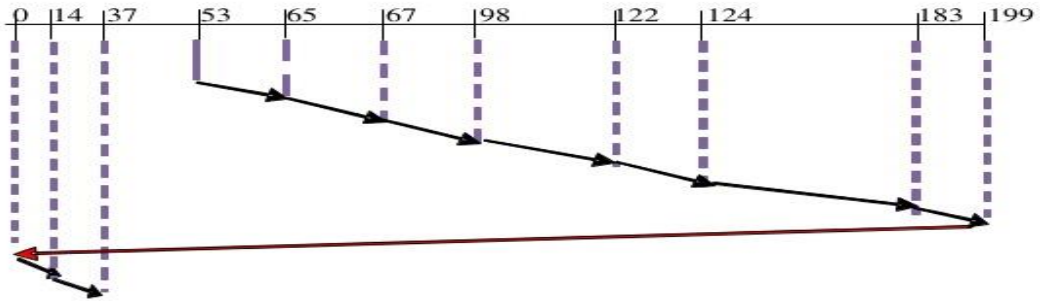
SCAN



USING SCAN SCHEDULING TOTAL HEAD MOVEMENT IS:

=(53-37)+(37-14)+(14-0)+(65-0)+(67-65)+(98-67)+(122-98)+(124-122)+(183-124)
= 16+23+14+65+2+31+24+2+59
= 236 cylinders.

C-SCAN



USING C-SCAN SCHEDULING TOTAL HEAD MOVEMENT IS:

= (65-53)+(67-65)+(98-67)+(122-98)+(124-122)+(183-124)+(199-183)+(14-0)+(37-14)
=12+2+31+24+2+59+169+14+23
=336 cylinders

b) List the method of allocating disk space. Explain any one of these methods. [4]

Ans :

There are 3 techniques for allocation of disk block :-

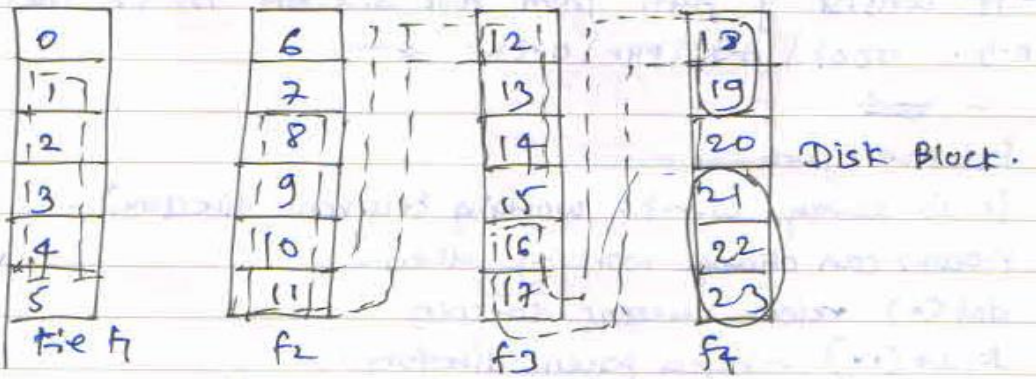
1) Contiguous allocation
2) Linked allocation
3) Indexed allocation

1) Contiguous Allocation :-

- In this, files are assigned to contiguous area of secondary storage. as shown in fig.

Directory

File	Start	Size
f_1	1	4
f_2	8	7
f_3	16	4
f_4	21	3



Contiguous Allocation of files.

OR

Q 10. a) What information is present in Directories? Explain the structre of Directory in details? [8]

Ans :

2) Directories:

- Directories are system files for maintaining the structure of the file system.

e.g

C:\ program files \ java \ jdk 1.8 . . .

* Directory Structure :

- It is data structure for storing detail about files. A directory typically contains a rws of record one per file.
- each record contains 1) file name
  2) file attribute.
  3) Addr. of disk block where data are st

Types of Directory structure :

1) Flat directory
2) Hierarchical directory.

1) Flat directory - all files are contained in root directory & there is no other subdirectory.

2) Hierarchical Directory :

it is organized in tree data structure., This is root directory. It can subdirectories and files.

- In this files can be grouped together in natural way
- file can be located quickly.

b) Explain File Management under Linux? **[4]**

Ans :

- Creating a file.
- Opening a file
- Renaming
- Reading data from file
- writing data to file.
- Random access g record
- Getting attribute g file
- Setting attribute g file.
- Appending data to file.
- Closing a file.

c) Describe any four types of File Organization. **[4]**

Ans :

1. Pile file
2. Sequential file
3. Indexed sequential file
4. Indexed file
5. Direct hash file

1. Sequential file :

- fixed format is used for record.
- All record are g same length
- posi^{n} g each field in record and length g field is fixed.
- Record are physically ordered on the value g one g the fields called the ordering fields.

Name	RollNO	Year
ABC	101	TE
XYZ	102	SE
PQR	103	BE.

↑ ordered data.

Sequential file.

2. Hash file :

Hashed (Direct) file Organization :-

- It is common tech used for fast accessing of records on secondary storage.
- Record of files are divided among bucket. A bucket is either one disk block or cluster of contiguous block.
- A hashing fu? map a key into a bucket number. a bucket are numbered from 0, 1, 2 - - - b-1.
- A hash fu? f maps each key value into one of integer thro. 0 to n-1.
- If x is key, f(x) is nos of bucket that contains the record with key x.
- The block making up each bucket could either be contiguous block or they can be chain together in linked list

$$= \text{bucket directory} + \frac{\text{Nos of Record}}{\text{Nos of bucket} \times \text{nos of record per block}}$$

Thus, the one is b time faster (b = nos of bucket) than unordered file.

3. Indexed file :

- Indexing is used to speed up retrieval of record. it is done with help of seperate sequential file.
- Each record of index file consist of two field and point into the main file.
- To find specific record for given key value, index is search for given key value.
- Binary search can be used to search in index file. After getting the address of record from index file. the record in main file. can easily be retrieved.
- Index file is ordered on the ordering key i.e. Roll_No. each record of index file points to the corresponding record. main file is not sorted.

********** THE END **********