[5870]-1127 SPOS (2022)

Q1 b) Based on various functionalities there are various types of loaders

1) compile & go
2) General Loader Scheme
3) Absolute Loader
4) Subroutine Linkage
5) Relocating Loaders.
6) Direct Linkage Loader

Compile & go Loader

→ In this type of loader instruction is read line by line its machine code is obtained & is directly put in the main memory at some known address.

→ That means the assembler runs in one part of memory and the assembled instructions & data is directly put into their assigned memory locations

→ After completion of assembly process loader contains the instruction using which location counter is set to the start of newly assembled program.

→ typical example is WATFOR-77 its a FORTRAN compiler which uses "Load and go" scheme. This loading scheme is called as "assemble and go"

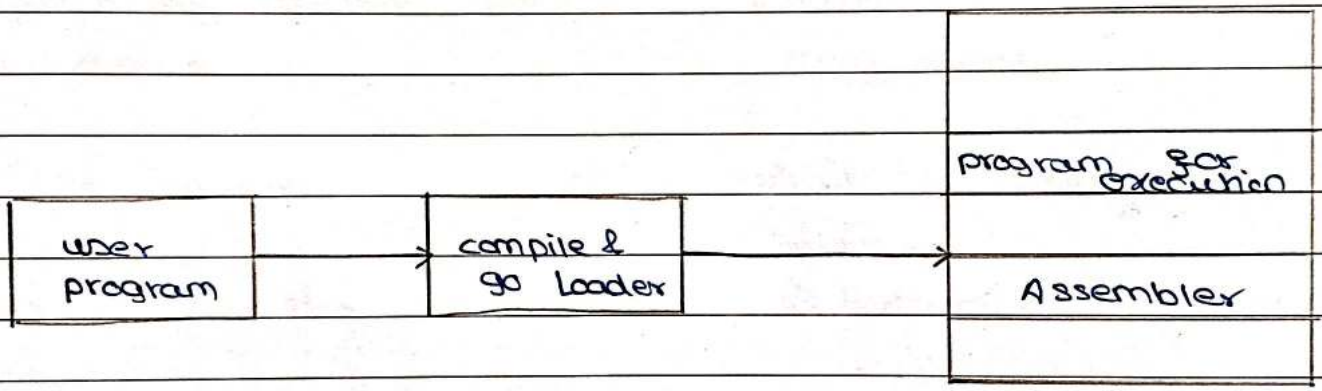
Advantages:

→ It is simple to implement.

Disadvantages:

→ Assembler occupies memory. Hence memory space remains occupied.

→ There is no production of .obj file the source code is directly converted to executable form



Compile and Go
Loading Scheme

Date :
Page :

Q1 @)

STATIC LINK LIBRARY	DYNAMIC LINK LIBRARY
1) For static library, actual code is extracted from library by linker & final executable code is built at the compilation time	1) For dynamic link library code need not be extracted and copied rather it is linked with executable code at run time.
2) Compatibility issues do not arise	2) Compatibility issues may arise.
3) Size is larger	3) it is compact in size.
4) Examples: All the .lib files	4) Examples: All the .dll files
5) Less portable as the code is embedded in executable form	5) More portable as DLL can be shared among different programs.
6) Produces larger executable file	6) Smaller executable file since library loaded seperately.
7) Version control can be difficult since every program needs to maintain its own copy in library	7) Allows version control as updates to library can affect all programs using it.

Q6a)

The scheduling policy determines when it is time for process to be removed from CPU and which ready process should be allocated the CPU next. Scheduling mechanism is composed of several different parts, depending on exactly how it is implemented in any particular O.S.

→ CPU scheduling is divided into 2 types: preemptive scheduling and non preemptive scheduling.

→ preemptive :
Scheduling method that interrupts the processing of process & transfers the CPU to another process is called preemptive CPU scheduling. process switches from running state to ready state and waiting state to ready state

preemptive scheduling incurs a cost associated with access shared data

It also affects design of operating system kernel.

it is more complex

ex: Round Robin method.

→ Non preemptive scheduling

usually proceeds towards completion uninterrupted once system has assigned a processor to process system cannot remove processor from process. process switches from running state to waiting state & termination of process.

Non preemptive scheduling does not increase the cost

It does not affect design of OS Kernel

Simple but very inefficient

ex: first come first serve method.

Date :
Page :

Q6 D)

First In First Out (FIFO)

→ also called first come first served scheduling method. The process are dispatched according to their arrival time at the ready queue. FIFO is non preemptive CPU scheduling algorithm.

→ FIFO is simple to implement because it uses FIFO queue. This algorithm is fine for most of batch operating system. useful in scheduling interactive processes because it cannot guarantee short response time.

→ when new process enters into the system its process control block is linked to the end of ready queue & it is removed from front of queue.

→ Turnaround time is unpredictable with FIFO algorithm. Average waiting time of FCFS is often quite long. Real life analogy is buying tickets.

→ Convoy effect : To reduce I/O device utilisation all I/O bound processes will be waiting excessively long for processor bound one this is called convoy effect.

If one process monopolises system, extent of overall effect of system performance depends on scheduling polices & whether process is processor bound or I/O bound.

2) priority scheduling:

→ It is preemptive or non preemptive priority of process can be defined either internally or externally. internally defined priority considers time limits, number of open files, use of memory and use of I/O devices.

→ External priorities are set by using external parameter of process like importance of process, cost of process etc.

→ In this CPU selects higher priority process first if priority of two process is same then FCFS scheduling algorithm is applied for solving the problem

→ A non preemptive priority algorithm will simply put new process at head of ready queue. In non preemptive currently executing process will not change. state.

Drawback of priority scheduling

→ waiting time is more for lower priority process. even if these required CPU burst time is less. It faced starvation problem.

→ starvation problem can be solved by aging.

Q7 a)

Virtual memory management :
is a technique that allows an OS to use hardware and software to provide the illusion of larger memory space than physically available.

It seperates the logical memory perceived by programs from the physical memory enabling more efficient & flexible memory use.

Address translation in paging.

→ In paging OS provides divides each incoming programs into pages of equal size. The section of disk called block / sectors. section of main memory are called frames. one sector hold one page of job instructions & fit into one page frame of memory.

→ In paging logical address space of program can be non contigious. It solves external fragmentation problem.

→ In relation between virtual address and physical memory address given by page table.

→ fixed size blocks are called frames & breaking of logical memory into blocks of same size called pages.

→ memory manage prepared following things.

1) find no. of pages in program.

2) free space in main memory.

3) loading of all programs into memory.

pages are not loaded continuously in main memory. each page can be stored in any available page frame anywhere in main memory memory manager keeps track of pages of program. paging provides external fragmentation and need for compaction.

Memory manager keeps track of

1) job table

2) page map table.

3) memory map table.

processor hardware performs logical to physical address translation. logical address contains page no. & offset. physical address contains frame no. & offset.

Q7 b)

Best fit:

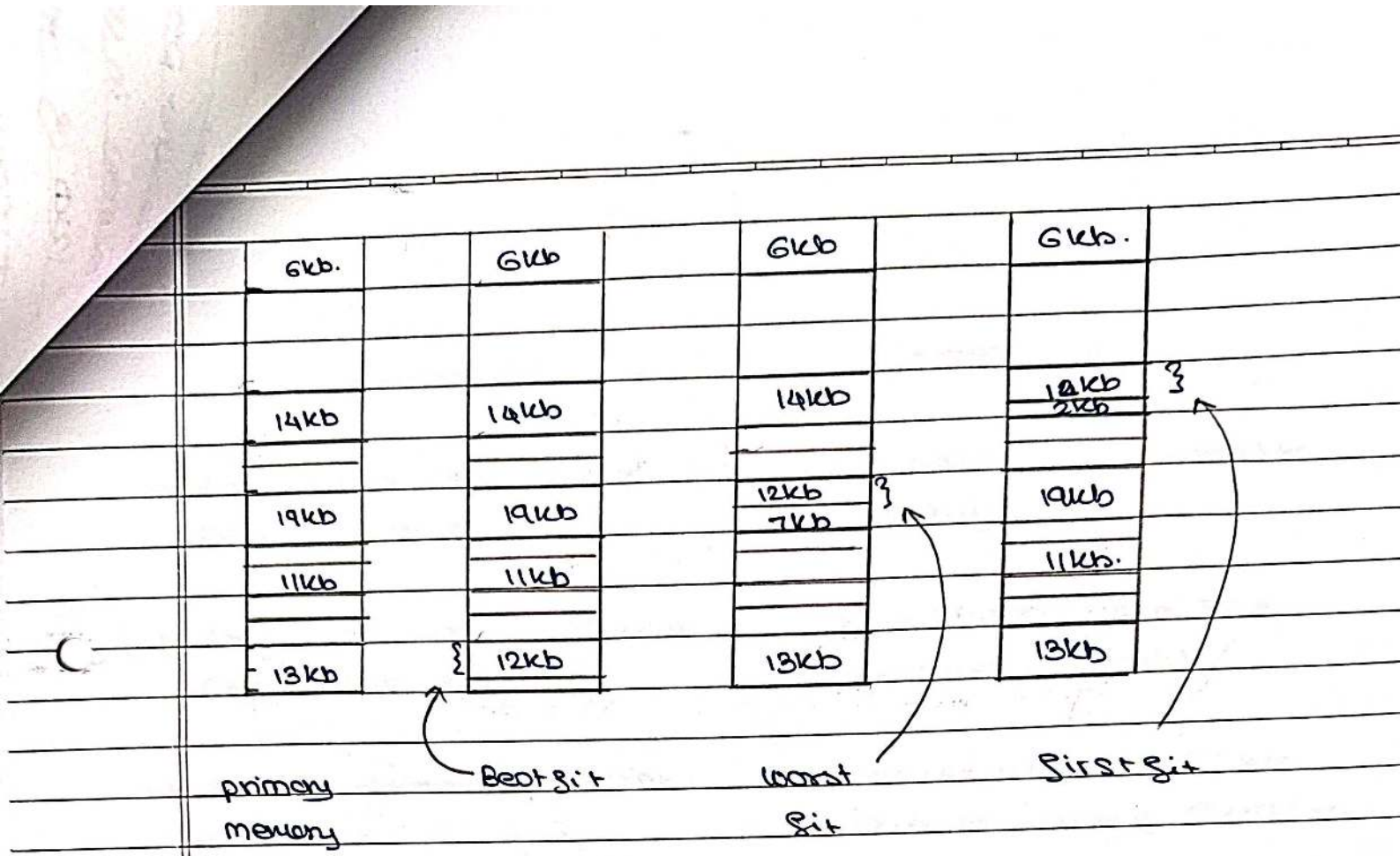
allocater placed process in smallest block of unallocated memory in which it will fit. suppose process requests 12kb of memory & memory manager currently has list of unallocated blocks of 6kb, 14kb, 19kb, 11kb, 13kb blocks best fit will allocate 12kb of 13kb block to the process. chooses closest in size to request.

Worst fit:

memory manage placed process in largest block of unallocated memory available idea is that this placement will create largest hold after allocations, thus increasing the possibility that compared to best fit another process can use remaing space worst fit will allocate 12kb of 19kb block to process leaving 7kb block for future use.

First Fit:

There may be many holes in memory so as reduce the amount of time it spends analysing available spaces begins at start of primary memory & allocates memory from first hole it encounters large enough to satisfy request. first fit allocates 12kb of 14kb block of process.



Best fit.

Advantages & disadvantages.

→ minimises wasted space by allocating smallest available partition.

→ Time consuming: scans entire list to find smallest partition

Worst fit:

→ Reduced chance of creating small, unusable memory blocks

→ requires searching through entire memory list.

First fit

→ Allocates first available partition that fits

→ leads to fragmentation.

Q3a)

COMPILERS	INTERPRETER
→ translates the entire program at once.	→ Translates & executes line by line
→ Generates an intermediate executable file	→ No intermediate file executes directly.
→ faster execution after compilation.	→ slower as translation happens during execution.
→ shows errors after analysing whole program	→ stops at first error encountered.
→ Requires more memory to store compiled code	→ Requires less memory as no intermediate code is stored

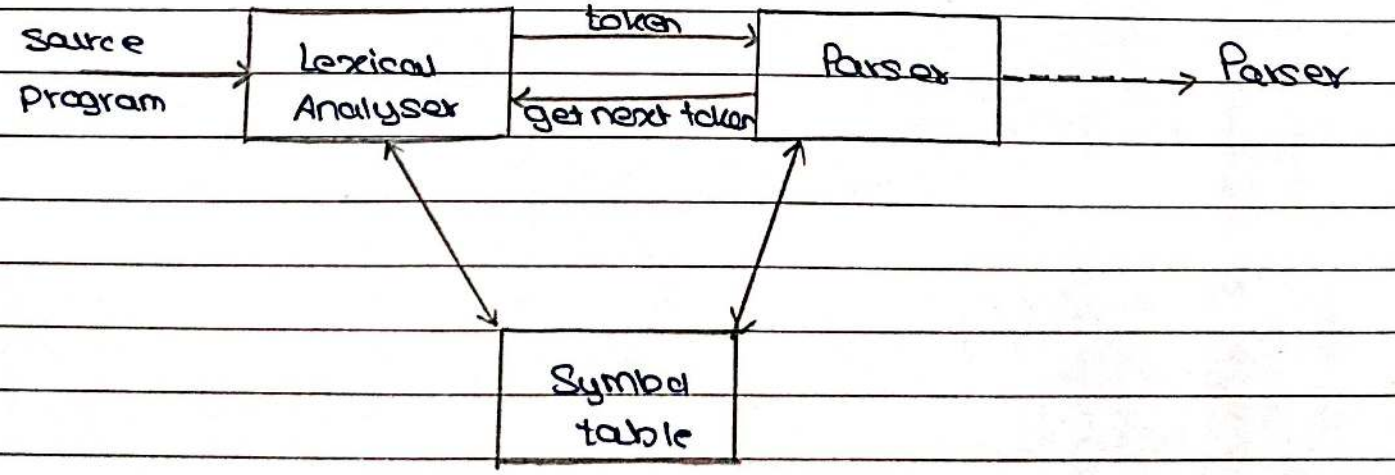
Q5b

LEX is lexical analyzer generator / tool used for converting a sequence of characters (source code) into sequence of tokens (lexical units)

→ typically used as first phase of compiler for transforming input code into meaningful tokens like keywords, operators, identifiers, literals.

Working:

→ LEX takes set of patterns, called regular expressions and generates a C program to perform lexical analysis based on those patterns. It matches input with these patterns to identify tokens & generates corresponding actions



Lexical Analysis.