3.1 Main Memory Management

 During execution, programs and data must be in main memory (at least partially).

 Furthermore, to improve both utilization of the CPU and speed of its response to users, the computer system must keep several processes in memory; that is, we must share memory.

 Since main memory is usually too small to accommodate all the data and programs permanently, the computer system must provide secondary storage to back up main memory.

 Memory management is the act of managing computer memory.

 This involves providing ways to allocate portions of memory to programs at request, and freeing it for reuse when no longer needed.

 The management of main memory is critical to the computer system.

Memory Management Basics

 In a uni-programming system, Main memory is divided into two parts:

– one part for the OS
– One part for the program currently being executed.

 In a multiprogramming system, the user part of the memory must be further subdivided to accommodate multiple processes.
 The task of subdivision is carried out dynamically by the Operating System and is known as Memory Management.
Monoprogramming Model

 Only one program at a time in main memory;
 Can use whole of the available memory.
 Since only one program or process resides in memory at a time, so sharing and protection is not an issue.

 However, the protection of OS program from
the user code is must otherwise the system will
crash down.
 This protection is done by a special hardware
mechanism such as a dedicated register, called as
a Fence Register. Also called Limit Register.
 The Fence Register is set to highest address
occupied by the OS code.
 A memory address generated by user program
to access certain memory location is first
compared with the fence register’s content.
 If the address generated is below the fence, it will
be trapped and denied permission.
 Since the modification of fence register is
considered as a privileged operation, therefore,
only OS is allowed to make any changes to it.

Multiprogramming with Fixed Partitions (Without
Swapping)
 Memory partitions scheme with a fixed number

of partitions was introduced to support
multiprogramming. this scheme is based on
contiguous allocation

 Each partition is a block of contiguous memory
 Memory is partitioned into a fixed number of
partition
 Each partition is of fixed size

Example: As shown in fig. memory is partitioned
into 5 regions the region is reserved for updating the
system the remaining four partitions are for the user
program

OS

P1

P2

P3

P4

Fixed Size Partitioning

Logical versus Physical Address Space

- An address generated by the CPU is commonly referred to as a logical address, whereas
- An address seen by the memory unit is commonly referred to as a physical address.
- The set of all logical addresses generated by a program is referred to as a logical address space;
- The set of all physical addresses corresponding to these logical addresses is referred to as a physical address space.
- The run-time mapping from virtual (logical) to physical addresses is done by the memory management unit (MMU), which is a hardware device.
- Relocation is a mechanism to convert the logical address into a physical address.
- To do this, there is a special register in CPU called relocation register (also known as base register).

 Every address used in the program is relocated
as:
 effective physical address= Logical address +
Contents of Relocation register

 MMU is a special hardware which performs
address binding, uses relocation scheme.

3.2 Swapping
 It is a memory management scheme that
temporarily swaps out an idle or blocked process
from the main memory to secondary memory
which ensures proper memory utilization and
memory availability for those processes which are
ready to be executed.
 The purpose of the swapping in operating
system is to access the data present in the hard disk
and bring it to RAM so that the application
programs can use it
 In secondary memory, the place where the

swapped-out process is stored is called swap space. The swapping method forms a temporary queue of swapped processes in the secondary memory.

 In the case of high-priority processes, the process with low priority is swapped out of the main memory and stored in swap space then the process with high priority is swapped into the main memory to be executed first.

 Although the process of swapping affects the performance of the system, it helps to run larger and more than one process. This is the reason why swapping is also referred to as memory compaction.

 Swap In: The method of removing a process from HDD and restoring it to the Main Memory.

 Swap Out: The method of bringing out a process from the main memory and sending it to the HDD so that the processes with higher priority will be executed.

Note: Swap In and Swap Out method is done by Medium Term Scheduler (MTS).

Advantages

 Swapping in OS helps in achieving the goal of Maximum CPU Utilization.

 Swapping ensures proper memory availability for every process that needs to be executed.

 Swapping helps in avoiding the problem of process starvation means a process should not take much time for execution so that the next process should be executed.

 CPU can perform various tasks simultaneously with the help of swapping so that processes do not have to wait much longer before execution.

 Swapping ensures proper RAM (main memory) utilization.

Disadvantages

 If the system deals with power-cut during bulky

swapping activity then the user may lose all the
information which is related to the program.

 If the swapping method uses an algorithm that is
not up to the mark then the number of page
faults can be increased and therefore this
decreases the complete performance.
 There may be inefficiency in a case when there
is some common resource used by the processes
that are participating in the swapping process.
Numerical
Suppose the user process&#39;s size is 2048KB and is a
standard hard disk where swapping has a data transfer
rate of 1Mbps. Now we will calculate how long it will take
to transfer from main memory to secondary memory.
User process size is 2048Kb
Data transfer rate is 1Mbps = 1024 kbps
Time = process size / transfer rate
= 2048 / 1024
= 2 seconds

= 2000 milliseconds

Now taking swap-in and swap-

out time, the process will take 4000 milliseconds.

3.3 Memory Allocation Strategies

Contiguous memory allocation

- Continuous memory allocation is a memory management technique in which memory is allocated to processes in contiguous blocks.
- This ensures that memory is utilized efficiently, with minimal fragmentation and wasted memory.
- This technique simplifies memory management by allowing the operating system to manage memory in larger blocks, leading to faster access to memory and improved system performance.
- Overall, continuous memory allocation is an important technique used by operating systems to efficiently manage memory resources and

ensure that memory is effectively utilized.

This allocation can be done in two ways:

 Fixed-size Partition Scheme

In here, each process is allotted a fixed size continuous block in the main memory. That means there will be continuous blocks of fixed size into which the complete memory will be divided, and each time a process comes in, it will be allotted one

of the free blocks. Because irrespective of the size of the process, each is allotted a block of the same size memory space. This technique is also called static partitioning.

 Variable-size Partition Scheme

In here, no fixed blocks or partitions are made in the memory. Instead, each process is allotted a variable-sized block depending upon its requirements. That means, whenever a new process wants some space in the memory, if available, this amount of space is allotted to it. Hence, the size of each block depends on the size and requirements of the process which occupies it.

Advantages of Contiguous Memory Allocation

 Efficient memory utilization: Contiguous memory allocation is efficient in terms of memory utilization, as there is no internal fragmentation within a process&#39;s allocated memory block.

 Simple and easy to manage: This technique is simple and easy to manage, as the operating system can quickly allocate and deallocate memory to processes by assigning contiguous blocks.

 Fast access: Since the memory is allocated in contiguous blocks, access to the memory is faster than other memory management techniques.

Disadvantages of Contiguous Memory Allocation

▯ External Fragmentation: One of the main disadvantages of contiguous memory allocation is external fragmentation, which occurs when small gaps of free memory are scattered throughout the memory space.

▯ Limited memory capacity: Contiguous memory allocation is limited by the size of the memory blocks available on the system, which may limit the total amount of memory that can be allocated to a process.

▯ Difficulty in sharing memory: This technique makes it difficult to share memory between multiple processes, as each process is assigned a contiguous block of memory that cannot be shared with other processes.

▯ Lack of flexibility: Contiguous memory allocation lacks flexibility in allocating and deallocating memory, as the operating system can only allocate memory in contiguous blocks.

Non-Contiguous Memory Allocation
 It allows a process to obtain multiple memory
blocks in various locations in memory based on
its requirements.
 The two methods for making a process&#39;s physical
address space non-contiguous are paging and
segmentation.
 Non-contiguous memory allocation divides the
process into blocks (pages or segments) that are

allocated to different areas of memory space
based on memory availability.
 The non-contiguous memory allocation also
reduces memory wastage caused by internal and
external fragmentation because it uses the
memory holes created by internal and external
fragmentation.
 Non-contiguous memory allocation can decrease
memory wastage, but it also raises address

translation overheads.

Advantages

 It has the advantage of reducing memory waste, but it increases overhead because of the address translation.

 It slows down the memory execution because time is consumed in address translation.

Disadvantages

 The downside of this memory allocation is that the access is slow because you must reach the other nodes using pointers and traverse them.

3.4 Paging

 Most virtual system uses a techniques called paging that permits the physical address space of a process to be non-contiguous.

 Program-generated addresses are called virtual addresses and form the virtual address space.

 When virtual memory is used, the virtual addresses requested by CPU do not go directly to the memory bus.

 Instead, they go to an MMU (Memory

Management Unit) that maps the virtual addresses

onto the physical memory addresses as illustrated

in Fig

Fig: Position and Function of MMU.

[In above fig, MMU is shown integrated with the CPU but it was separate chip

long time back.]

 The basic method for implementing paging involves

breaking physical memory into fixed size block called

frames and breaking logical memory into blocks of

the same size called pages.

 The virtual address space is divided up into units

called pages and corresponding units in the

physical memory called page frames. The pages

and pages frame are always of same size.

 The process of retrieving processes in the form

of pages from the secondary storage into the
main memory is known as paging.

Paging Example:

Fig: Relation between virtual address space and
physical memory address

In this example, we have a computer that can generate 16 bit addresses, from 0 up to 64K. These are the virtual addresses.

 This computer, however, has only 32 KB of physical memory, so although 64-KB programs

can be written, they cannot be loaded into memory in their entirety and run.

 64 KB of virtual address space and 32 KB of physical memory, we get 16(64/4KB, since page frame size is typically of 4KB) virtual pages and 8 page frames. Transfers between RAM and disk are always in units of a page.

 When the program tries to access address 0 (see the virtual address space), for example, using the instruction MOV REG,0
 Virtual address 0 is sent to the MMU.
 The MMU sees in page table that this virtual address falls in page 0 (0 to 4095), which according to its mapping is page frame 2 (8192 to 12287 in the physical address).
 It thus transforms the address to 8192 and outputs address 8192 onto the bus.
 The memory knows nothing at all about the MMU and just sees a request for reading or writing address 8192, which it honors.
 Thus, the MMU has effectively mapped all virtual addresses between 0 and 4095 onto physical addresses 8192 to 12287

PAGE Fault:
 A page fault is a trap to the software raised by the hardware when a program accesses a page

that is mapped in the virtual address space,
but not loaded in physical memory.

 In the typical case the operating system tries to handle the page fault by making the required page accessible at a location in physical memory or kills the program in the case of an illegal access.

 The hardware that detects a page fault is the memory management unit in a processor.

 The exception handling software that handles the page fault is generally part of the operating system.

 What happens if the program tries to use an unmapped page, for example, by using the instruction MOV REG,32780

Page Fault

 A page fault is a trap to the software raised by the hardware when a program accesses a page that is mapped in the virtual address space, but not loaded in physical memory.

- In the typical case the operating system tries to handle the page fault by making the required page accessible at a location in physical memory or kills the program in the case of an illegal access.
- The hardware that detects a page fault is the memory management unit in a processor.
- The exception handling software that handles the page fault is generally part of the operating system.
- What happens if the program tries to use an unmapped page, for example, by using the instruction MOV REG,32780

Which is within virtual page 8 (starting at 32768)

- The MMU notices that the page is unmapped (indicated by a cross in the figure) and causes the CPU to trap to the operating system. This trap is called a page fault.

 The operating system picks a little-used page frame and writes its contents back to the disk.
 It then fetches the page just referenced into the page frame just freed, changes the map, and restarts the trapped instruction.

 In computer storage technology, a page is a fixed-length block of memory that is used as a unit of transfer between physical memory and external storage like a disk,
 And a page fault is an interrupt (or exception) to the software raised by the hardware, when a program accesses a page that is mapped in address space, but not loaded in physical memory.
 An interrupt that occurs when a program requests data that is not currently in real memory.
 The interrupt triggers the operating system to fetch the data from a virtual memory and load

it into RAM.

 An invalid page fault or page fault error occurs when the operating system cannot find the data in virtual memory.

 This usually happens when the virtual memory area, or the table that maps virtual addresses to real addresses, becomes corrupt.

3.5 Structure Of The Page Table:

 The structure of the page table represents how many ways a page table can be structured

 For each process, page table stores the number of frame, allocated for each page.

 The purpose of the page table is to map virtual pages into page frames by the MMU.

 Mathematically the page table is a function with the virtual page number as argument and the physical frame number as result.

 Using the result of this function the virtual page field in a virtual address can be replaced by a page frame field, thus forming a physical memory address.

Characteristics Of Page Table

- It is stored in the main memory.
- Generally; the Number of entries in the page table = the Number of Pages in which the process is divided.
- PTBR means page table base register and it is basically used to hold the base address for the page table of the current process.

- Each process has its own independent page table.

Some of the common techniques used for structuring the Page table are as follows:

- Hierarchical Paging
- Hashed Page Tables

 Inverted Page Tables

1) Hierarchical Paging

 It is also called Multilevel Paging and it is a very simple methodology.

 When the page table is too big to fit in a contiguous space then this hierarchical paging system is used with several levels.

 In this, the logical address space is broken up into Multiple page tables.

 Hierarchical paging uses the following two types of page tables:

 Two-Level Page Table

 Three-Level Page Table

Two Level Page Table:

 On this page table, itself is paged. Therefore, here two-page tables&#39; inner page table and outer page table are present.

 Example: 32-bit logical address space and a page size of 4 KB is divided into A Page Number consisting of 20 bits and A Page Offset consisting of 12 bits.

 Since the Page table itself paged, the page number is further divided into, A 10-bit page number and A 10-bit page offset.

Therefore Logical Address looks as follows:

P1 P2 D

10 10 12

Outer Page Table Inner Page Table Offset

 Here, P1 &amp; P2 represent Page Numbers and d represents the Page Offset.

 In the above address, P1 is an index into the Outer Page table, and P2 indicates the displacement within the page of the Inner page Table.

Three Level Page Table

 A two-level paging table is not appropriate for the system with a 64-bit logical address space.

 Hence, for 64-bit logical address space, Three-
Level Page Table is used.
 In this, divide the outer page table first, and then
it will result in a Three-level page table as shown
below.
 Example: 64-bit logical address space and a page
size of 4 KB is divided.
P1 P2 P2 d
32 10 10 12

2 nd Outer Page
Table

Outer Page
Table

Inner Page
Table

Offset

2] Hashed Page Tables

 Hashed page tables are commonly used in address spaces greater than 32 bits.

 In a hashed page table, the virtual addresses are hashed into the hash table.

 Each element in the table comprises a linked list of elements to avoid collisions.

 For each element in the hash table, there are three fields available,

 The virtual Page Number (hash value, all bits are not part of the page offset)

 The value of the mapped page frame

 A pointer to the next element in the linked list

 The virtual page number is matched against the first field (virtual address), and if a match is found, the corresponding mapped address in the second field is used to form the desired memory address.

 If a match is not found, the linked list is traversed using the next pointer until a match is found.

 The CPU generates a logical address for the page it needs.

 Then logical address needs to be mapped to the physical address. This logical address has two entries, page number (P3) and an offset.

 The page number from the logical address is directed to the hash function.

 The hash function produces a hash value corresponding to the page number.

 This hash value directs to an entry in the hash table.

 As each entry in the hash table has a link list. Here the page number is compared with the first element’s first entry if a match is found then the second entry is checked.

 In the example mentioned in the above image, the logical address includes page number P3 which does not match the first element of the link list as it includes page number P1.

 So it will move ahead and check the next element.

 Now, this element has a page number entry (P3) so further it will check the frame entry of the element which is fr5.

 To this frame number, it will append the offset provided in the logical address to reach the page’s physical address.

 This is how the hashed page table works to map the logical address to the physical address.

3] Inverted Page Tables

 The inverted page table consists of a one-page table entry for every frame of the main memory.

 The inverted page table combines a page table and a frame table into one data structure.
 So the number of page table entries in the Inverted Page Table reduces to the number of frames in physical memory.
 A single-page table represents the paging information of all the processes.
 One entry for each virtual page number and real page of memory.

 This provides the solution to the wastage of memory problem.
 The overhead of storing an individual page table for every process gets eliminated through the inverted page table.
 Only a fixed portion of memory is required to store the paging information of all the processes together.
 This technique is called inverted paging, as the indexing is done concerning the frame number

instead of the logical page number.

 Each entry in the page table contains the fields such as

 Page number

 Process ID

 Control bits

 Chained Pointer.

 As inverted page tables decrease the need for storage for each table, but it increases the time needed to search the table when a page reference occurs.

 The CPU generates the logical address for the page it needs to access.

 This time the logical address consists of three entries process id, page number, and the offset.

 The process id identifies the process, of which the page has been demanded, the page number indicates which page of the process has been asked for and the offset value indicates the displacement required.

 The match of process id along with associated page number is searched in the page table and says if the search is found at the i th entry of page table then i and offset together generates the physical address for the requested page.
 This is how the logical address is mapped to a physical address using the inverted page table.

3.6 Segmentation

 In Operating Systems, Segmentation is a memory management technique in which the memory is divided into the variable size parts. Each part is known as a segment which can be allocated to a process.
 Segmentation is the Memory management scheme that supports user view of memory.
 A segment is a logical unit such as: main program, procedure, function, method, object, local variables, global variables, common block, stack,

symbol table, arrays.

 The details about each segment are stored in a table called a segment table. Segment table is stored in one (or many) of the segments.

Fig: Segmentation

 Segment table contains mainly two information about segment:

 Base: It is the base address of the segment. The segment base contains the starting physical address of the segments residing in the memory.

 Limit: The segment limit is also known as segment offset. It is the length of the segment.

Fig: Segment Table

 STBR: STBR stands for Segment Table Base Register. STBR stores the base address of the segment table.

 STLR: STLR stands for Segment Table Length Register. STLR stores the number of segments used by a program.

 There are two types of segmentation:

 Virtual memory segmentation – Each process is divided into a number of segments, not all of which are resident at any one point in time.

 Simple segmentation – Each process is divided into a number of segments, all of which are loaded into memory at run time, though not necessarily contiguously.

Advantages of Segmentation –

 No Internal fragmentation.

 Segment Table consumes less space in comparison to Page table in paging.

 As a complete module is loaded all at once, segmentation improves CPU utilization.

 The user’s perception of physical memory is quite similar to segmentation. Users can divide user programs into modules via segmentation. These modules are nothing more than the separate processes’ codes.
 The user specifies the segment size, whereas in paging, the hardware determines the page size.
 Segmentation is a method that can be used to segregate data from security operations.
 Flexibility: Segmentation provides a higher degree of flexibility than paging. Segments can be of variable size, and processes can be designed to have multiple segments, allowing for more fine-grained memory allocation.
 Sharing: Segmentation allows for sharing of memory segments between processes. This can

be useful for inter-process communication or for sharing code libraries.
 Protection: Segmentation provides a level of

protection between segments, preventing one process from accessing or modifying another process's memory segment. This can help increase the security and stability of the system.

Disadvantages of Segmentation –

- As processes are loaded and removed from the memory, the free memory space is broken into little pieces, causing External fragmentation.
- Overhead is associated with keeping a segment table for each activity.
- Due to the need for two memory accesses, one for the segment table and the other for main memory, access time to retrieve the instruction increases.
- Fragmentation: As mentioned, segmentation can lead to external fragmentation as memory becomes divided into smaller segments. This can lead to wasted memory and decreased performance.

 Overhead: The use of a segment table can
increase overhead and reduce performance.
Each segment table entry requires additional
memory, and accessing the table to retrieve
memory locations can increase the time needed
for memory operations.
 Complexity: Segmentation can be more
complex to implement and manage than paging.
In particular, managing multiple segments per
process can be challenging, and the potential for
segmentation faults can increase as a result.

Paging Vs Segmentation

S.N
O

Paging Segmentation

1. In paging, the

program is divided
into fixed or mounted
size pages.

In segmentation, the
program is divided into
variable size sections.

2. For the paging
operating system is
accountable.

For segmentation
compiler is accountable.

3. Page size is
determined by
hardware.

Here, the section size is

given by the user.

4. It is faster in
comparison to
segmentation.

Segmentation is slow.

5. Paging could result in

internal
fragmentation.

Segmentation could
result in external
fragmentation.

6. In paging, the logical
address is split into a
page number and

page offset.

Here, the logical address is split into section number and section offset.

7. Paging comprises a page table that encloses the base address of every page.

While segmentation also comprises the segment table which encloses segment number and segment offset.

8. The page table is employed to keep up

the page data.

Section Table maintains
the section data.

9. In paging, the

operating system must
maintain a free frame
list.

In segmentation, the
operating system
maintains a list of holes
in the main memory.

10. Paging is invisible to

the user.

Segmentation is visible to
the user.

11. In paging, the
processor needs the
page number, and

In segmentation, the
processor uses segment
number, and offset to

offset to calculate the
absolute address.

calculate the full address.

12. It is hard to allow
sharing of procedures
between processes.

Facilitates sharing of procedures between the processes.

13 In paging, a

programmer cannot efficiently handle data structure.

It can efficiently handle data structures.

14. This protection is hard to apply.

Easy to apply for protection in segmentation.

15. The size of the page
needs always be equal
to the size of frames.

There is no constraint on
the size of segments.

16. A page is referred to
as a physical unit of
information.

A segment is referred to
as a logical unit of
information.

17. Paging results in a less

efficient system.

Segmentation results in a

more efficient system.

3.7 Virtual Memory Management

- Virtual memory uses both hardware and software to enable a computer to compensate for physical memory shortages, temporarily

  transferring data from random access memory (RAM) to disk storage.
- Mapping chunks of memory to disk files enables a computer to treat secondary memory as though it were main memory.
- Virtual memory is a technique to allow a large logical address space to be mapped onto a smaller physical memory.
- Virtual memory allows extremely large process to be run, and also allows the degree of multiprogramming to be raised, increasing CPU utilization.
- Further, it frees application programmers from

worrying about memory availability.

 The basic idea behind virtual memory is that the combined size of the program, data, and stack may exceed the amount of physical memory available for it.

 The operating system keeps those parts of the program currently in use in main memory, and the rest on the disk (swap area called virtual memory).

 For example, a 512-MB program can run on a 256-MB machine by carefully choosing which

256 MB to keep in memory at each instant, with pieces of the program being swapped between disk and memory as needed.

 Virtual storage is not a new concept; this concept was devised by fortherigham in 1961 and used in Atlas computer system.

 But the common use in OS is the recent concept, all microprocessor now support virtual memory.

 Virtual memory can be implemented by two

most common used methods:

 Paging

 Segmentation

 Or mixed strategy of both.