Summary Note: Non-Contiguous Memory Allocation and Virtual Memory

Storage Allocation and Management Techniques

The Storage allocation can be of two types:

(i) Contiguous storage allocation.

(ii) Non-contiguous storage allocation.

In this lecture, we will discuss mainly non-contiguous storage allocation techniques. But some concepts regarding contiguous storage allocation are given below.

Contiguous Storage Allocation

When a program's data and instructions are assumed to occupy a single contiguous memory area, it is known as Contiguous storage allocation. It is further subdivided into fixed-partition storage allocation strategy and dynamic or variable-partition storage allocation strategy. Various Partitioning Algorithms that are used for contiguous memory allocation are shown below:

1. **Best Fit Algorithm:**
   The Best Fit algorithm searches the whole set of memory blocks and tries to find out the smallest memory block possible in the list that can accommodate the size requirement of the process. The best-fit algorithm results in exhaustive searches of the entire memory space. This allocation process is slower because it scans memory blocks every time before allocation. The Best Fit Algorithm results in minimum hole size.

2. **Worst Fit Algorithm:**
   The Worst Fit algorithm scans the entire memory block every time and tries to find out the biggest block in the list, which can fulfil the requirement of the process. The worst-fit algorithm results in exhaustive searches of the entire memory space.

3. **First Fit Algorithm:**

   The First Fit algorithm scans the memory blocks, and whenever it finds the first big enough memory block to store a process, it stops scanning and loads the job into that block. This algorithm avoids exhaustive searches on entire free space.

4. **Next Fit Algorithm:**
   The Next Fit algorithm works the same as the First Fit algorithm, except that Next fit scans the memory blocks from the block where it previously allocated a job. Like the first-fit algorithm, the next fit algorithm avoids exhaustive searches on the entire free space. This algorithm is not used in most of the cases because it is not better than the first algorithm

Segregated List: Segregated lists are used to improve memory allocation when a particular application has one (or a few) popular-sized requests. We will keep a separate list to manage objects of that size. All other requests are forwarded to a more general memory allocator.

Non-contiguous Storage Allocation

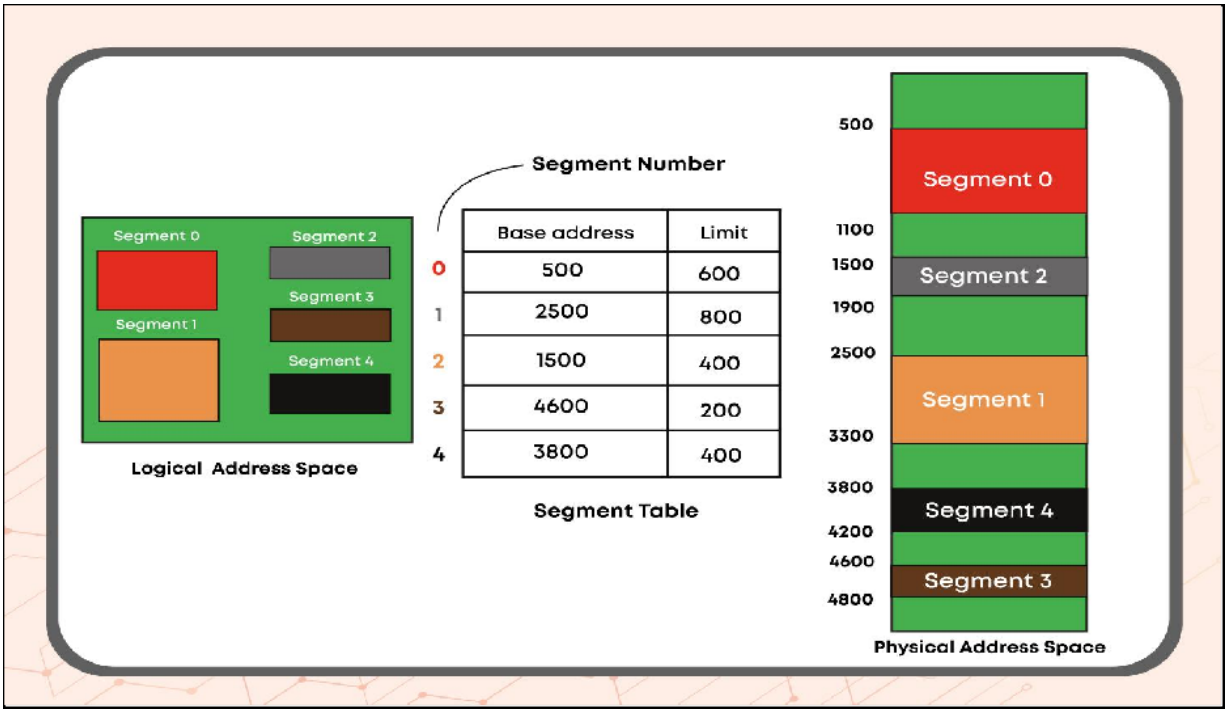
To resolve the problem of external fragmentation and to enhance the degree of multiprogramming to a greater extent, it was decided to sacrifice the simplicity of allocating contiguous memory to every process. It was decided to have a non-contiguous physical address space of a process so that a process could be allocated memory wherever it was available.

There are 2 techniques for non-contiguous allocation:

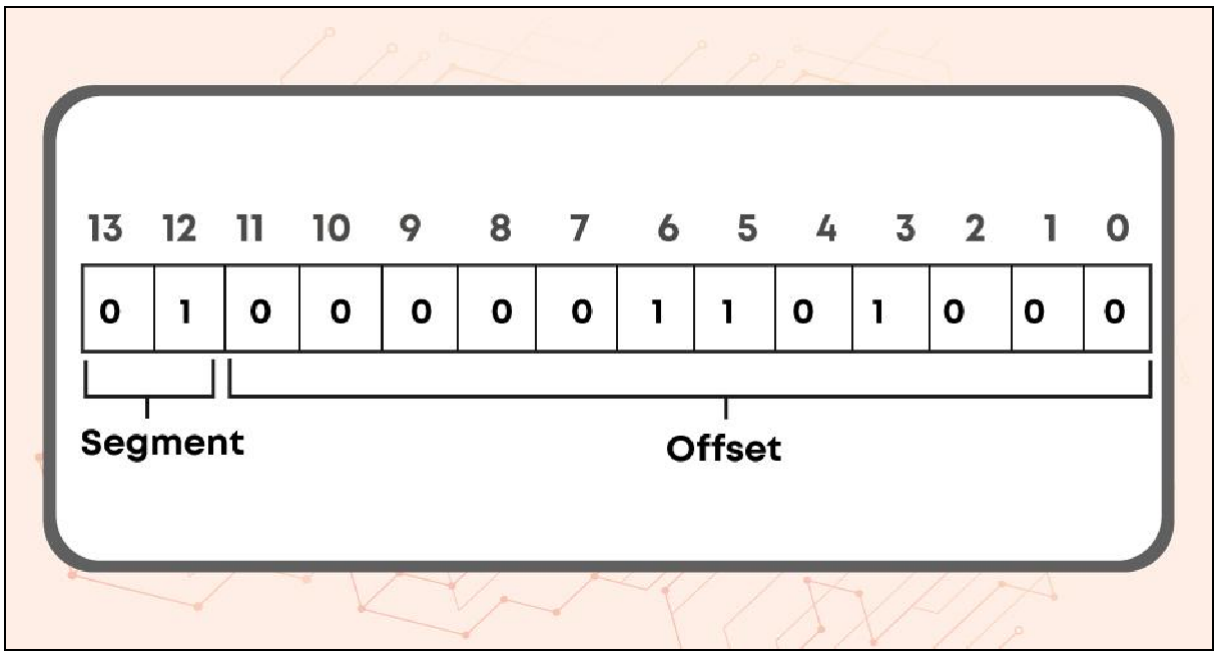
1. Segmentation
2. Paging

Segmentation

Segmentation is a technique for non-contiguous storage allocation. For a programmer, it might be more relevant to divide the logical address space of his program into variable sized segments (with respect to his view of the main program, subroutines, data, etc.) than to divide it into fixed-size pages. Such variable sized segments, which are a collection of logically related information, are the basis of the segmentation technique.



For each segment, the table stores the starting address of the segment and the length of the segment. A reference to a memory location includes a value that identifies a segment and an offset.



Advantages and disadvantages of Segmentation

Advantages :

1. No internal fragmentation.
2. Segment tables consume less memory.
3. Lends itself to sharing data among processes.
4. Lends itself to protection.

Disadvantages:

1. Costly memory management algorithm.

Protection Bits: As we have seen in the segmentation above, we keep the different base and bound segment that helps to convert the virtual address into a physical address and helps to solve the issue of internal fragmentation. However, even after using segmentation, we are still wasting some memory. To save the wastage of that memory, we use extra bits that are known as protection bits. Protection bits are set on each segment which tells whether the process can read or write to a segment or not as well as execute the code in the segment. By setting this bit to read-only, multiple processes can read the code in the segment without worrying about the isolation of the segment. To the process, it still looks like it is accessing its own private memory. However, in the back, The OS is sharing the memory with multiple processes which cannot be modified by any process.

Segment	Base	Size	Grows Positive	Protection
Code	32k	2k	1	Read - Execute
Heap	34k	3k	1	Read - Write
Stack	28k	2k	0	Read - Write

Let us consider our old example. In this, the protection bits for code are set to read and execute This means that the code segment can be shared by another process as well and since the write bit is not set, no process will be able to modify this code

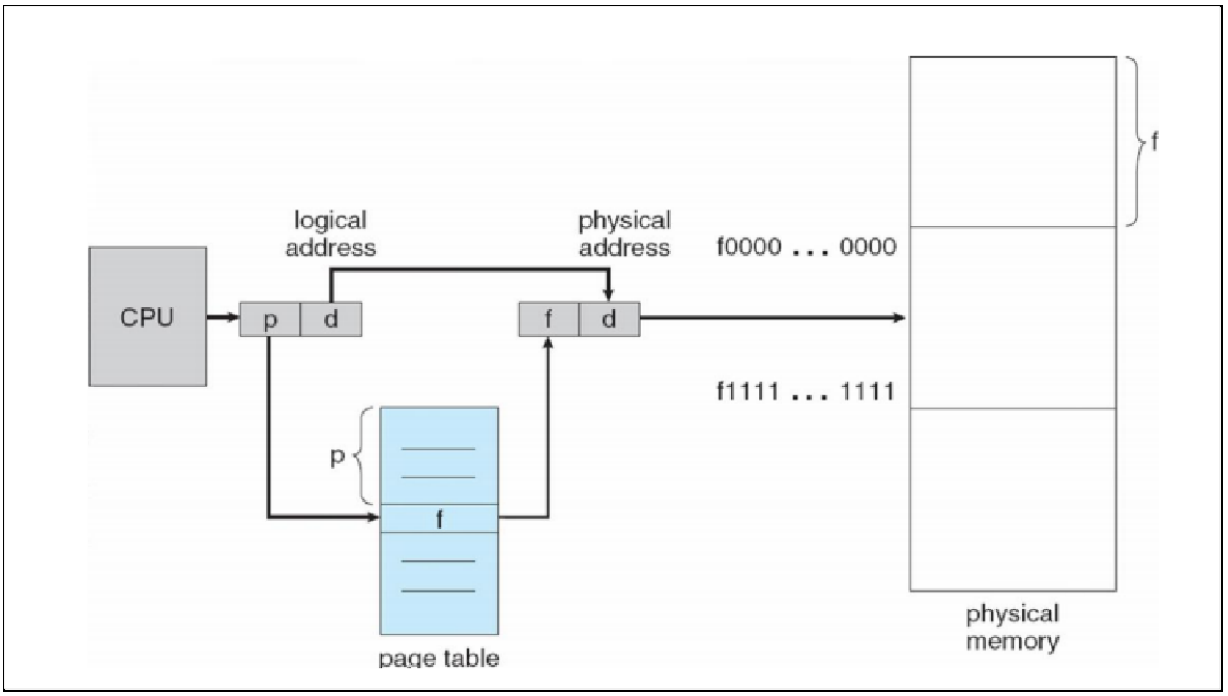
Paging

In paging, physical memory is divided into fixed-size blocks called frames and virtual memory is divided into fixed-sized blocks called pages.

The size of a page is the same as that of a frame. The key idea of this method is to place the pages of a process into the available frames of memory, whenever, this process is to be executed. The address mapping is done by Page table.

Physical memory is divided into fixed-size - blocks called FRAMES. (size is the power of 2, for example, 512 bytes)

Virtual/ Logical memory is divided into blocks of the same size called PAGE



Example: Suppose, if the main memory size is 16 KB and the frame size is 1 KB. Here, the main memory will be divided into a collection of 16 frames of 1 KB each.

Advantages and Disadvantages of paging

Advantages:

1. No external fragmentation.
2. Simple memory management algorithm.
3. Swapping is easy(equal-sized pages and page frames).

Disadvantages:

1. Internal fragmentation.
2. Page tables may consume more memory.

Since, owing to the huge size of modern-day processes, the entire process cannot be added to the main memory because of its limited size, hence, the concept of Paging was extended and support was taken from secondary memory.

Translation Lookaside Buffer (TLB):

To speed the address translation, we need an extra piece of hardware known as translation lookaside buffer or TLB. TLB is basically a hardware cache that is used for translation.Whenever any memory reference is made, the reference is first checked in the TLB. If the reference is present in TLB, it is quickly translated to the corresponding physical address without consulting the page table.

We know that the virtual page number will be present in the virtual address. The OS has to first extract the page number from the virtual address using the bit manipulation operators. Once it has the virtual page number(VPN), the OS then checks the corresponding physical frame number(PFN) in the TLB. If the entry is present in the TLB then it is a **TLB hit**. The OS will then concatenate the offset with the physical frame number(PFN) and generate the physical address. However, if the entry is not present in the TLB then it is a **TLB miss**. The OS will then check the page table to get the corresponding physical frame number(PFN). However, since it has to perform the memory reference, this process is slow. The OS will update the TLB with the details so that in the next run, it is a TLB hit and the translation is quick.

The hit rate of TLB is equal to (Total number of TLB hits) / (Total number of TLB hits + total number of TLB miss).

Address Space Identifiers (ASIDs): ASID is a unique identifier which is present as an extra bit for each entry in TLB to check if the process that is accessing that entry belongs to the process.ASID is represented using 8 bits.

Virtual memory

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of a hard drive that's set up to emulate the computer's RAM. The paging technique plays an important role in implementing virtual memory.

Page faults

Page fault dominates like an error. If any program tries to access a piece of memory or memory page but it does not exist in the main memory, then page fault will occur. The fault specifies to the O/S that it must trace the location of this memory page, and after that move it from secondary memory to the primary memory of the system. This swapping of memory pages is done by Page replacement algorithms.

Page Replacement Algorithm

When memory located in secondary memory is needed, it can be retrieved back to the main memory.

Process of storing data from main memory to secondary memory ->swapping out

Retrieving data back to main memory ->swapping in

Why do we need a page replacement algorithm?

The main goal of page replacement algorithms is to provide the lowest page fault rate.

Algorithms

- **First In First Out (FIFO):**

  First in First Out is the simplest page replacement algorithm. In this algorithm, All the pages in the memory are tracked by the Operating system. All the pages are present in a queue with the front page of the queue containing the oldest page. When a page needs to be replaced by another page during the time of page miss, the page in the front of the queue i.e. The oldest page is selected for removal.

- **Least Recently Used (LRU):**

  Least Recently Used algorithm is a complex algorithm as compared to the FIFO algorithm. We keep track of the number of times a page is accessed and replace the page least accessed among them. Doing this will keep the most accessed page in the memory and prevent paging from slowing down.