Difference Between Logical and Physical Address in Operating System

Address uniquely identifies a location in the memory. We have two types of addresses that are logical address and physical address. The logical address is a virtual address and can be viewed by the user. The user can't view the physical address directly. The logical address is used like a reference, to access the physical address. The fundamental difference between logical and physical address is that **logical address** is generated by CPU during a program execution whereas, the **physical address** refers to a location in the memory unit.

There are some other differences between the logical and physical address. Let us discuss them with the help of comparison chart shown below.

Content: Logical and Physical Address

Comparison Chart

BASIS FOR COMPARISON	**LOGICAL ADDRESS**	**PHYSICAL ADDRESS**
Basic	It is the virtual address generated by CPU	The physical address is a location in a memory unit.
Address Space	Set of all logical addresses generated by CPU in reference to a program is referred as Logical Address Space.	Set of all physical addresses mapped to the corresponding logical addresses is referred as Physical Address.
Visibility	The user can view the logical address of a	The user can never view physical address of program

BASIS FOR COMPARISON	LOGICAL ADDRESS	
	program.	
Access	The user uses the logical address to access the physical address.	The user can not directly access physical address.
Generation	The Logical Address is generated by the CPU	Physical Address is Computed by MMU

Definition of Logical Address

Address generated by **CPU** while a program is running is referred as **Logical Address**. The logical address is virtual as it does not exist physically. Hence, it is also called as **Virtual Address**. This address is used as a reference to access the physical memory location. The set of all logical addresses generated by a programs perspective is called **Logical Address Space**.

The logical address is mapped to its corresponding physical address by a hardware device called **Memory-Management Unit**. The address-binding methods used by MMU generates **identical** logical and physical address during **compile time** and **load time**. However, while **run-time** the address-binding methods generate **different** logical and physical address.

Definition of Physical Address

Physical Address identifies a physical location in a memory. MMU (**Memory-Management Unit)** computes the physical address for the corresponding logical address. MMU also uses logical address computing physical address. The user never deals with the physical address. Instead, the physical address is accessed by its corresponding logical address by the user. The user program generates the logical address and thinks that the program is running in this logical address. But the program needs physical memory for its execution. Hence, the logical address must be mapped to the physical address before they are used.

The logical address is mapped to the physical address using a hardware called **Memory-Management Unit**. The set of all physical addresses corresponding to the logical addresses in a Logical address space is called **Physical Address Space**.

Key Differences Between Logical and Physical Address in OS

1. The basic difference between Logical and physical address is that Logical address is generated by CPU in perspective of a program. On the other hand, the physical address is a location that exists in the memory unit.
2. The set of all logical addresses generated by CPU for a program is called Logical Address Space. However, the set of all physical address mapped to corresponding logical addresses is referred as Physical Address Space.
3. The logical address is also called virtual address as the logical address does not exist physically in the memory unit. The physical address is a location in the memory unit that can be accessed physically.
4. Identical logical address and physical address are generated by Compile-time and Load time address binding methods.
5. The logical and physical address generated while run-time address binding method differs from each other.
6. The logical address is generated by the CPU while program is running whereas, the physical addres is computed by the MMU (Memory Management Unit).

Conclusion:

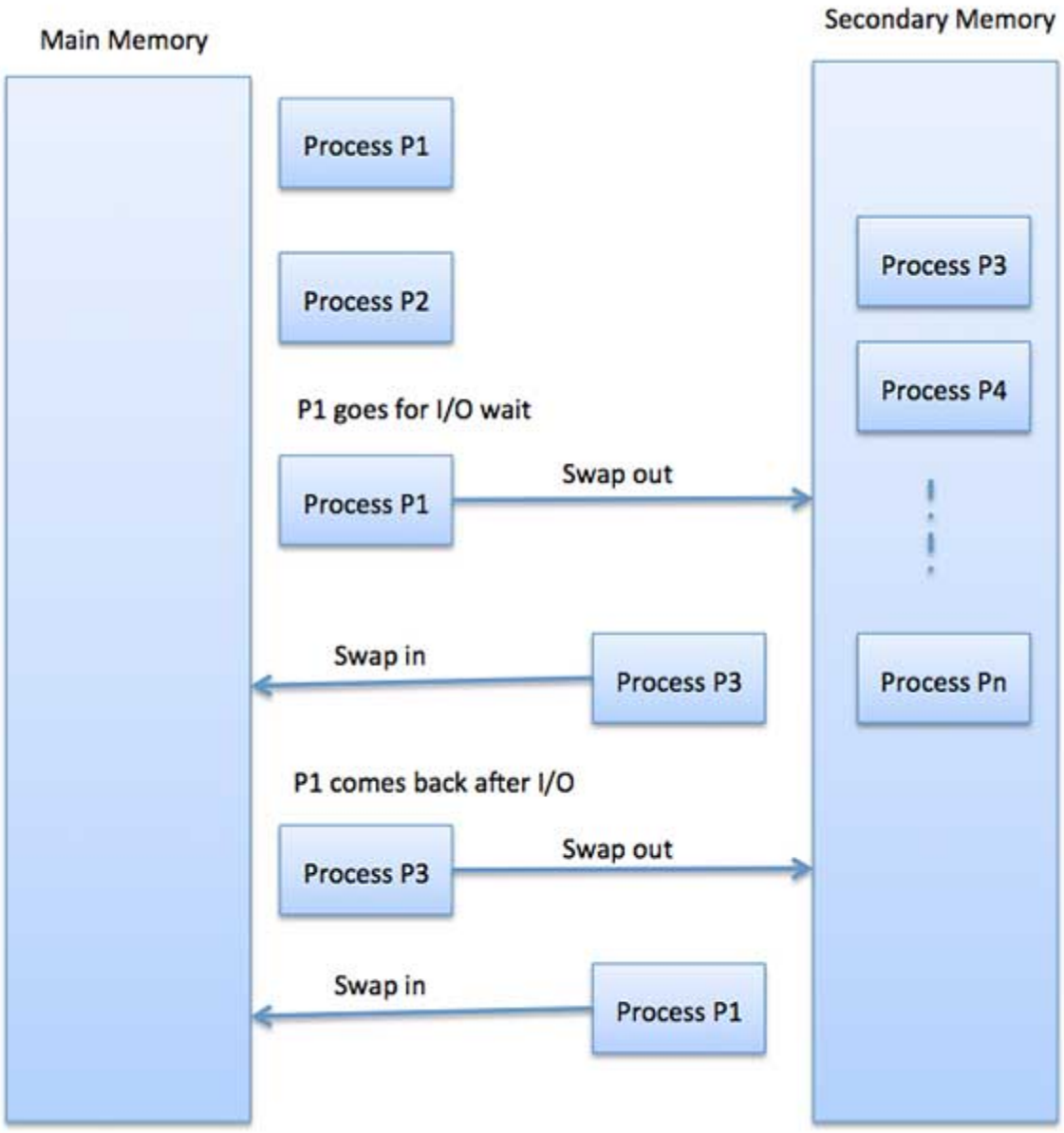
The logical address is a reference used to access physical address. The user can access physical address in the memory unit using this logical address.

Swapping

Swapping is a mechanism in which a process can be **swapped** temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other processes. At some later time, the **system** swaps back the process from the secondary storage to main memory.

Swapping is a mechanism in which a process can be swapped temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other processes. At some later time, the system swaps back the process from the secondary storage to main memory.

Though performance is usually affected by swapping process but it helps in running multiple and big processes in parallel and that's the reason **Swapping is also known as a technique for memory compaction**.

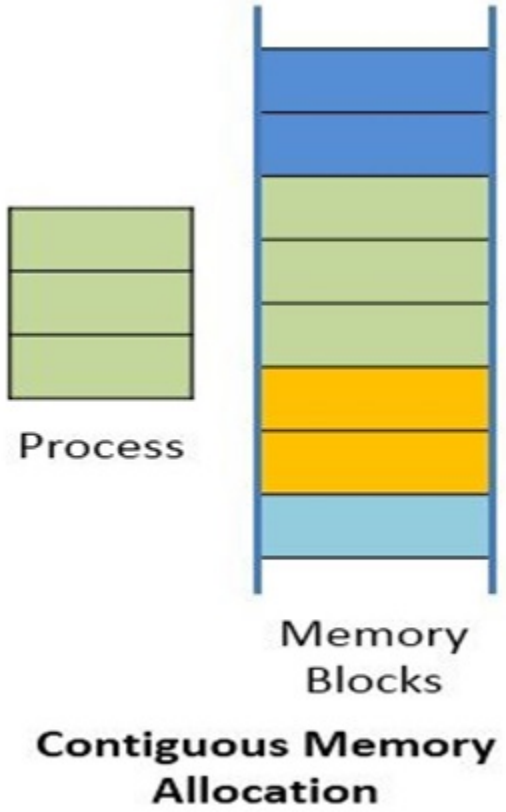


The total time taken by swapping process includes the time it takes to move the entire process to a secondary disk and then to copy the process back to memory, as well as the time the process takes to regain main memory.

Contiguous memory allocation

It means freely available **memory** partitions are not scattered here and there across the whole **memory** space. In the **contiguous memory allocation**, both the **operating system** and the user must reside in the main **memory**. A single process is **allocated** in that fixed sized single partition.

In the **contiguous memory allocation**, both the operating system and the user must reside in the main memory. The main memory is divided into two portions one portion is for the operating and other is for the user program.
In the **contiguous memory allocation** when any user process request for the memory a single section of the contiguous memory block is given to that process according to its need. We can achieve contiguous memory allocation by dividing memory into the fixed-sized partition.
A single process is allocated in that fixed sized single partition. But this will increase the degree of multiprogramming means more than one process in the main memory that bounds the number of fixed partition done in memory. Internal fragmentation increases because of the contiguous memory allocation.



Contiguous Memory Allocation

Fixed sized partition
In the fixed sized partition the system divides memory into fixed size partition (may or may not be of the same size) here entire partition is allowed to a process and if there is some wastage inside the partition is allocated to a process and if there is some wastage inside the partition then it is called internal fragmentation.
Advantage: Management or book keeping is easy.
Disadvantage: Internal fragmentation
Variable size partition

In the variable size partition, the memory is treated as one unit and space allocated to a process is exactly the same as required and the leftover space can be reused again.
Advantage: There is no internal fragmentation.

Paging

In computer **operating systems**, **paging** is a memory management scheme by which a computer stores and retrieves data from secondary storage for use in main memory. In this scheme, the **operating system** retrieves data from secondary storage in same-size blocks called pages.Paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory. This scheme permits the physical address space of a process to be non – contiguous.

- Logical Address or Virtual Address (represented in bits): An address generated by the CPU
- Logical Address Space or Virtual Address Space(represented in words or bytes): The set of all logical addresses generated by a program
- Physical Address (represented in bits): An address actually available on memory unit
- Physical Address Space (represented in words or bytes): The set of all physical addresses corresponding to the logical addresses
- The Physical Address Space is conceptually divided into a number of fixed-size blocks, called **frames**.
- The Logical address Space is also splitted into fixed-size blocks, called **pages**.
- Page Size = Frame Size

Address generated by CPU is divided into

- **Page number(p):** Number of bits required to represent the pages in Logical Address Space or Page number
- **Page offset(d):** Number of bits required to represent particular word in a page or page size of Logical Address Space or word number of a page or page offset.

Physical Address is divided into

- **Frame number(f):** Number of bits required to represent the frame of Physical Address Space or Frame number.
- **Frame offset(d):** Number of bits required to represent particular word in a frame or frame size of Physical Address Space or word number of a frame or frame offset.

The hardware implementation of page table can be done by using dedicated registers. But the usage of register for the page table is satisfactory only if page table is small. If page table contain large number of entries then we can use TLB(translation Look-aside buffer), a special, small, fast look up hardware cache.

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**segment** which can be allocated to a process. The details about each **segment** are stored in a table called as **segment** table.

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 segment which can be allocated to a process.
The details about each segment are stored in a table called as segment table. Segment table is stored in one (or many) of the segments.
Segment table contains mainly two information about segment:

1. Base: It is the base address of the segment
2. Limit: It is the length of the segment.

Why Segmentation is required?

Till now, we were using Paging as our main memory management technique. Paging is more close to Operating system rather than the User. It divides all the process into the form of pages regardless of the fact that a process can have some relative parts of functions which needs to be loaded in the same page.
Operating system doesn't care about the User's view of the process. It may divide the same function into different pages and those pages may or may not be loaded at the same time into the memory. It decreases the efficiency of the system.
It is better to have segmentation which divides the process into the segments. Each segment contain same type of functions such as main function can be included in one segment and the library functions can be included in the other segment,

Translation of Logical address into physical address by segment table

CPU generates a logical address which contains two parts:

1. Segment Number
2. Offset

The Segment number is mapped to the segment table. The limit of the respective segment is compared with the offset. If the offset is less than the limit then the address is valid otherwise it throws an error as the address is invalid.
In the case of valid address, the base address of the segment is added to the offset to get the physical address of actual word in the main memory.

Advantages of Segmentation

1. No internal fragmentation
2. Average Segment Size is larger than the actual page size.
3. Less overhead
4. It is easier to relocate segments than entire address space.
5. The segment table is of lesser size as compare to the page table in paging.

Disadvantages

1. It can have external fragmentation.
2. it is difficult to allocate contiguous memory to variable sized partition.
3. Costly memory management algorithms.

Virtual Memory

Virtual memory is a **memory** management capability of an **operating system** (**OS**) that uses hardware and software to allow a computer to compensate for physical **memory** shortages by temporarily transferring data from random access **memory** (**RAM**) to disk **storage**.

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 corresponding machine addresses.
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 address 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.

Hardware and Control Structures

Memory references are dynamically translated into physical addresses at run time – A process may be swapped in and out of main memory such that it occupies different regions • A process may be broken up into pieces that do not need to be located contiguously in main memory • All pieces of a process do not need to be loaded in main memory during execution

Execution of a Program

Operating system brings into main memory a few pieces of the program
• Resident set - portion of process that is in main memory
• An interrupt is generated when an address is needed that is not in main memory
• Operating system places the process in a blocking state

Piece of process that contains the logical address is brought into main memory – Operating system issues a disk I/O Read request – Another process is dispatched to run while the disk I/O takes place – An interrupt is issued when disk I/O complete which causes the operating system to place the affected process in the Ready state.

Advantages of Breaking up a Process

• More processes may be maintained in main memory – Only load in some of the pieces of each process – With so many processes in main memory, it is very likely a process will be in the Ready state at any particular time
• A process may be larger than all of main memory

Types of Memory

• Real memory – Main memory
• Virtual memory – Memory on disk – Allows for effective multiprogramming and relieves the user of tight constraints of main memory – Programming convenience

Thrashing

• Swapping out a piece of a process just before that piece is needed
• The processor spends most of its time swapping pieces rather than executing user instructions

Support Needed for Virtual Memory

• Hardware must support paging and segmentation
• Operating system must be able to management the movement of pages and/or segments between secondary memory and main memory

Paging

• Each process has its own page table
• Each page table entry contains the frame number of the corresponding page in main memory
• A bit is needed to indicate whether the page is in main memory or not

Modify Bit in Page Table

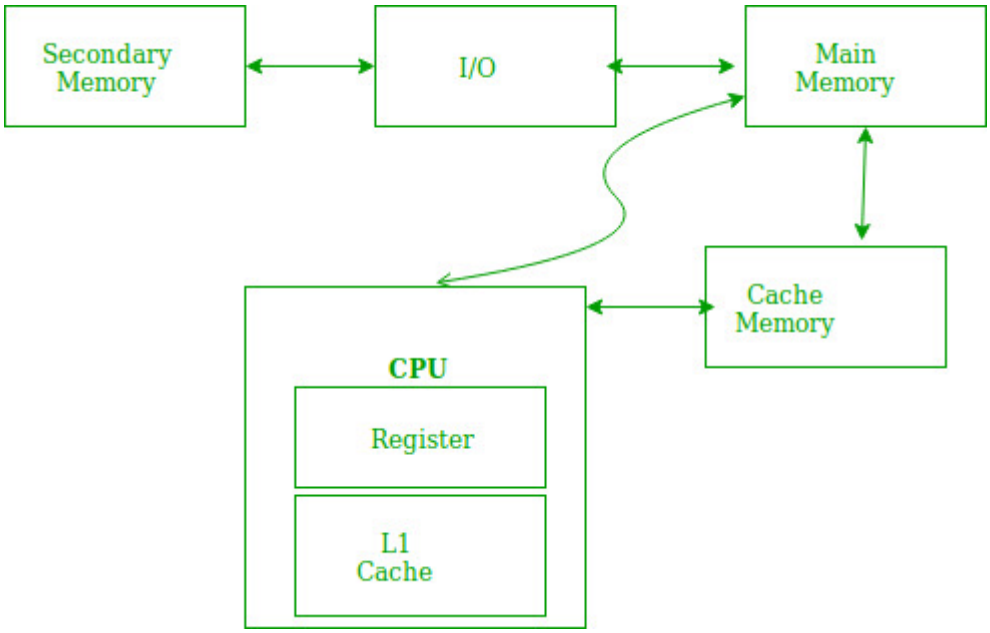
• Modify bit is needed to indicate if the page has been altered since it was last loaded into main memory
• If no change has been made, the page does not have to be written to the disk when it needs to be swapped out

Locality of Reference

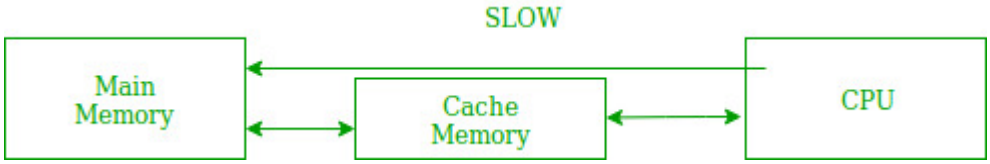
In computer science, **locality of reference**, also known as the principle of **locality**, is the tendency of a processor to access the same set of memory locations repetitively over a short period of time. There are two basic types of **reference locality** – temporal and spatial **locality**.
Locality of **reference** is accessing a value or related storage frequently. And the types are temporal, spatial and sequential. Temporal: It tells us whether memory locations in a program are likely to be accessed again in the near future.

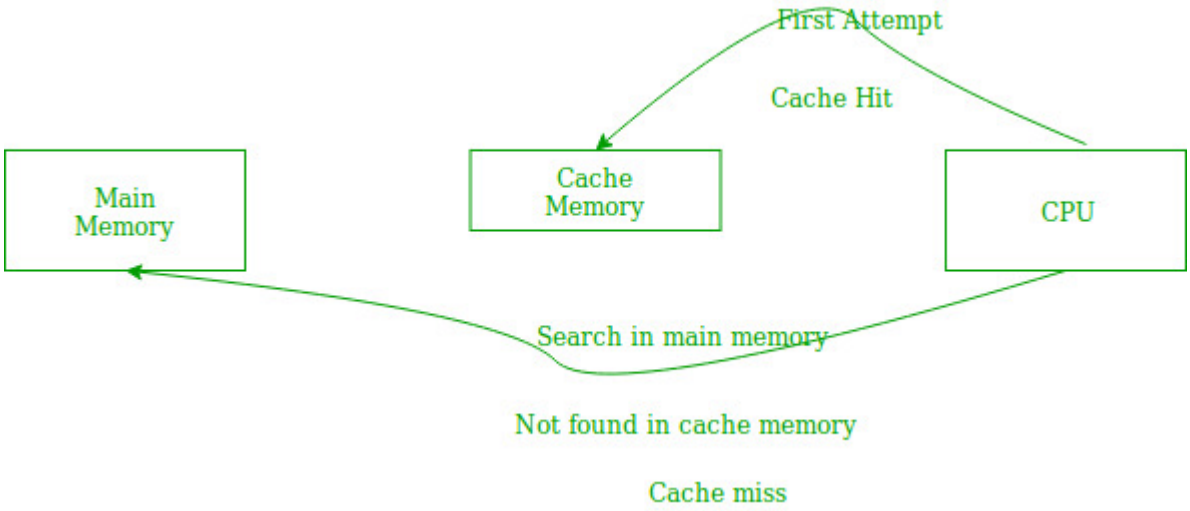
Locality of Reference and Cache Operation in Cache Memory

Locality of reference refers to a phenomenon in which a computer program tends to access same set of memory locations for a particular time period. In other words, **Locality of Reference** refers to the tendency of the computer program to access instructions whose addresses are near one another. The property of locality of reference is mainly shown by loops and subroutine calls in a program.



1. In case of loops in program control processing unit repeatedly refers to the set of instructions that constitute the loop.
2. In case of subroutine calls, everytime the set of instructions are fetched from memory.
3. References to data items also get localized that means same data item is referenced again and again.





In the above figure, you can see that the CPU wants to read or fetch the data or instruction. First, it will access the cache memory as it is near to it and provides very fast access. If the required data or instruction is found, it will be fetched. This situation is known as a cache hit. But if the required data or instruction is not found in the cache memory then this situation is known as a cache miss. Now the main memory will be searched for the required data or instruction that was being searched and if found will go through one of the two ways:

1. First way is that the CPU should fetch the required data or instruction and use it and that's it but what, when the same data or instruction is required again.CPU again has to access the same main memory location for it and we already know that main memory is the slowest to access.
2. The second way is to store the data or instruction in the cache memory so that if it is needed soon again in the near future it could be fetched in a much faster way.

Cache Operation:

It is based on the principle of locality of reference. There are two ways with which data or instruction is fetched from main memory and get stored in cache memory. These two ways are the following:

1. **Temporal Locality –**
   Temporal locality means current data or instruction that is being fetched may be needed soon. So we should store that data or instruction in the cache memory so that we can avoid again searching in main memory for the same data.
   When CPU accesses the current main memory location for reading required data or instruction, it also gets stored in the cache memory which is based on the fact that same data or instruction may be needed in near future. This is known as temporal locality. If some data is referenced, then there is a high probability that it will be referenced again in the near future.
2. **Spatial Locality –**
   Spatial locality means instruction or data near to the current memory location that is being fetched, may be needed soon in the near future. This is slightly different from the temporal locality. Here we are talking about nearly located memory locations while in temporal locality we were talking about the actual memory location that was being fetched.

Cache Performance:

The performance of the cache is measured in terms of hit ratio. When CPU refers to memory and find the data or instruction within the Cache Memory, it is known as cache hit. If the desired data or instruction is not found in the cache memory and CPU refers to the main memory to find that data or instruction, it is known as a cache miss.

Demand Paging

According to the concept of Virtual Memory, in order to execute some process, only a part of the process needs to be present in the main memory which means that only a few pages will only be present in the main memory at any time.
However, deciding, which pages need to be kept in the main memory and which need to be kept in the secondary memory, is going to be difficult because we cannot say in advance that a process will require a particular page at particular time.
Therefore, to overcome this problem, there is a concept called Demand Paging is introduced. It suggests keeping all pages of the frames in the secondary memory until they are required. In other words, it says that do not load any page in the main memory until it is required.
Whenever any page is referred for the first time in the main memory, then that page will be found in the secondary memory.

In computer operating systems, **demand paging** (as opposed to anticipatory paging) is a method of virtual memory management. In a system that uses demand paging, the operating system copies a disk page into physical memory only if an attempt is made to access it and that page is not already in memory (*i.e.*, if a page fault occurs). It follows that a process begins execution with none of its pages in physical memory, and many page faults will occur until most of a process's working set of pages are located in physical memory. This is an example of a lazy loading technique.

Basic concept

Demand paging follows that pages should only be brought into memory if the executing process demands them. This is often referred to as lazy evaluation as only those pages demanded by the process are swapped from secondary storage to main memory. Contrast this to pure swapping, where all memory for a process is swapped from secondary storage to main memory during the process startup.

Commonly, to achieve this process a page table implementation is used. The page table maps logical memory to physical memory. The page table uses a bitwise operator to mark if a page is valid or invalid. A valid page is one that currently resides in main memory. An invalid page is one that currently resides in secondary memory. When a process tries to access a page, the following steps are generally followed:

- Attempt to access page.
- If page is valid (in memory) then continue processing instruction as normal.
- If page is invalid then a **page-fault trap** occurs.
- Check if the memory reference is a valid reference to a location on secondary memory. If not, the process is terminated (**illegal memory access**). Otherwise, we have to **page in** the required page.
- Schedule disk operation to read the desired page into main memory.
- Restart the instruction that was interrupted by the operating system trap.

Advantages

Demand paging, as opposed to loading all pages immediately:

- Only loads pages that are demanded by the executing process.
- As there is more space in main memory, more processes can be loaded, reducing the context switching time, which utilizes large amounts of resources.

- Less loading latency occurs at program startup, as less information is accessed from secondary storage and less information is brought into main memory.
- As main memory is expensive compared to secondary memory, this technique helps significantly reduce the bill of material (BOM) cost in smart phones for example. Symbian OS had this feature.

Disadvantages

- Individual programs face extra latency when they access a page for the first time.
- Low-cost, low-power embedded systems may not have a memory management unit that supports page replacement.
- Memory management with page replacement algorithms becomes slightly more complex.
- Possible security risks, including vulnerability to timing attacks.

Page replacement algorithms are the techniques using which an Operating System decides which memory **pages** to swap out, write to disk when a **page** of memory needs to be allocated.

Page Replacement Algorithms in Operating Systems

In an operating system that uses paging for memory management, a page replacement algorithm is needed to decide which page needs to be replaced when new page comes in.

Page Fault – A page fault happens when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory.
Since actual physical memory is much smaller than virtual memory, page faults happen. In case of page fault, Operating System might have to replace one of the existing pages with the newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce the number of page faults.

Page Replacement Algorithms :

- **First In First Out (FIFO) –**
  This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.
- **Optimal Page replacement –**
  In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.
- **Least Recently Used –**
  In this algorithm page will be replaced which is least recently used.

.