Operating System
KCS – 401

Memory

Dr. Pankaj Kumar
Associate Professor – CSE
SRMGPC Lucknow

Outline of the Lecture

Bare Machine

Resident Monitor

Multiprogramming with fixed Partition

Multiprogramming with Variable Partition

Fragmentation

Bare Machine

Bare machine is logical hardware which is used to execute the program in the processor without using the operating system. as of now, we have studied that we can't execute any process without the Operating system. But yes with the help of the Bare machine we can do that.

Initially, when the operating systems are not developed, the execution of an instruction is done by directly on hardware without using any interfering hardware, at that time the only drawback was that the Bare machines accepting the instruction in only machine language, due to this those person who has sufficient knowledge about Computer field are able to operate a computer. so after the development of the operating system Bare machine is referred to as inefficient.

Resident Monitor

The **Resident Monitor** is a code which runs on **Bare Machine**. Its acts like an operating system which controls everything inside a processor and performs all the functions. The Resident Monitor is thus also known as the Job Sequencer because like the Operating system, it also sequences the jobs and sends it to the processor for execution. After the jobs are scheduled, the Resident Monitor loads the Programs one by one into the main memory according to their sequence.

The advantage of using a Resident Monitor over an Operating System is that there is no gap or lag between the program executions. So, the processing is faster in the **Resident Monitors**.

Resident Monitor

The Resident Monitors are divided into 3 parts:

1.Control Language Interpreter

The job of the Control Language Interpreter is to read and carry out the instructions line by line to the next level.

2.Loader

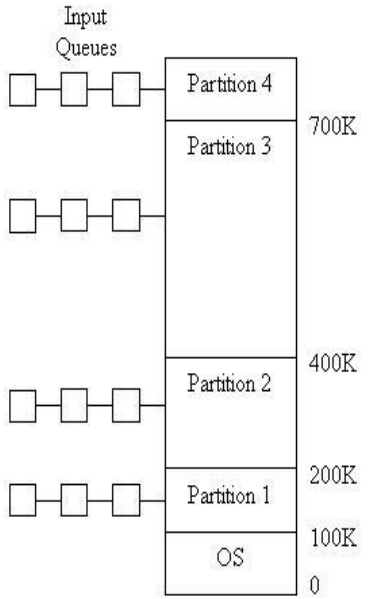
The Loader is the main part of the Resident Monitor. As the name suggests, it Loads all the required system and application programs into the main memory.

3.Device Driver

The Device Driver Takes care of all the Input-Output devices connected with the system. So, all the communication that takes place between the user and the system is handled by the Device Driver. It simply acts as an intermediate between the requests and the response, requests that are made by the user to the system, and they respond that the system produces to fulfill these requests.

Multiprogramming with fixed Partition

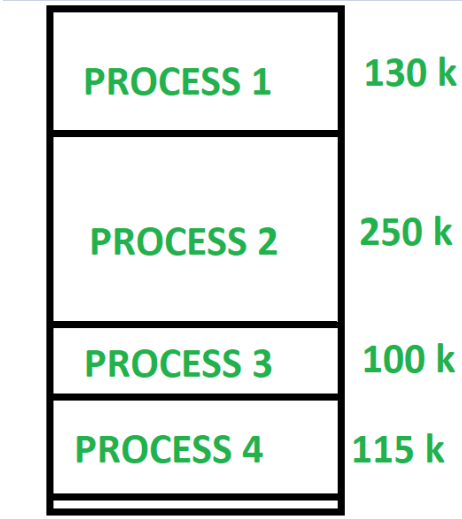
If we accept that multiprogramming is a good idea, we next need to decide how to organize the available memory in order to make effective use of the resource. One method is to divide the memory into fixed sized partitions. These partitions can be of different sizes but once a partition has taken on a certain size then it remains at that size. There is no provision for changing its size.



- Several users simultaneously compete for system resources
- switch between I/O jobs and calculation jobs for instance
- To take advantage of this sharing of CPU, important for many jobs to be present in main memory
- Allowing **Relocation** and **Transfers** between partitions
- **Protection** implemented by the use of several *boundary registers* : *low and high* boundary registers, or *base* register with *length*
- **Fragmentation** occurs if user programs cannot completely fill a partition - wasteful.

Multiprogramming with variable Partition

Multi-programming with variable partitioning is a contiguous memory management technique in which the main memory is not divided into partitions and the process is allocated a chunk of free memory that is big enough for it to fit. The space which is left is considered as the free space which can be further used by other processes.



Advantages

No Internal Fragmentation

No Limitation on the size of the process

Degree of multiprogramming is dynamic

Disadvantages

External Fragmentation

Complex Memory Allocation

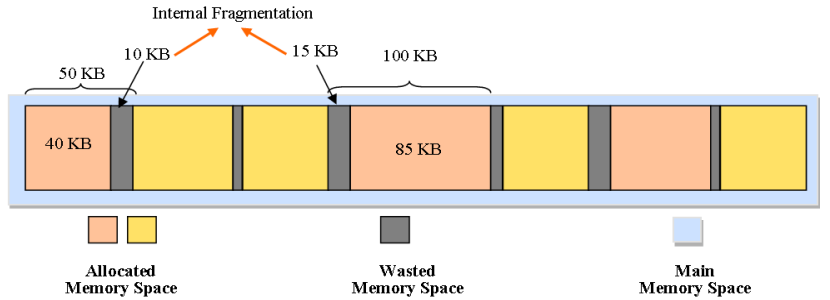
Fragmentation

As processes are loaded and removed from memory, the free memory space is broken into little pieces. It happens after sometimes that processes cannot be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is known as Fragmentation.

Fragmentation is of two types −

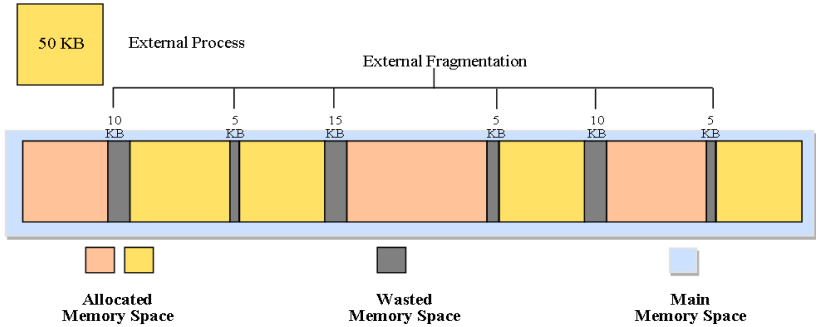
External fragmentation

Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous, so it cannot be used.



Internal fragmentation

Memory block assigned to process is bigger. Some portion of memory is left unused, as it cannot be used by another process.



Fragmentation

Internal fragmentation	External fragmentation
In internal fragmentation fixed-sized memory, blocks square measure appointed to process.	In external fragmentation, variable-sized memory blocks square measure appointed to method.
Internal fragmentation happens when the method or process is larger than the memory.	External fragmentation happens when the method or process is removed.
The solution of internal fragmentation is best-fit block.	Solution of external fragmentation is compaction, paging and segmentation.
Internal fragmentation occurs when memory is divided into fixed sized partitions.	External fragmentation occurs when memory is divided into variable size partitions based on the size of processes.
The difference between memory allocated and required space or memory is called Internal fragmentation.	The unused spaces formed between non-contiguous memory fragments are too small to serve a new process, is called External fragmentation .

In compaction the spaces that are free and the spaces which are not allocated to the process are combined and single large memory space is made.

Protection

Protection refers to a mechanism which controls the access of programs, processes, or users to the resources defined by a computer system. We can take protection as a helper to multi programming operating system, so that many users might safely share a common logical name space such as directory or files.

Need of Protection:

•To prevent the access of unauthorized users and

•To ensure that each active programs or processes in the system uses resources only as the stated policy,

•To improve reliability by detecting latent errors.

Outline of the Lecture

Memory Hierarchy Design

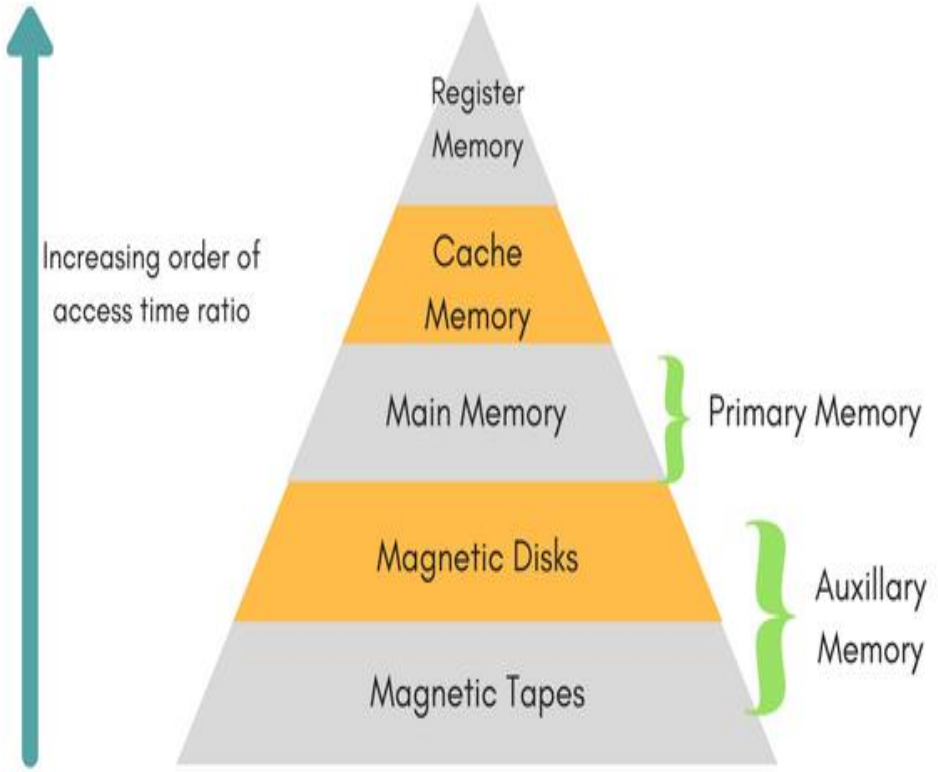
Cache Memory

Address and Address Space

Virtual Memory

Memory Hierarchy Design

Memory Hierarchy is an enhancement to organize the memory such that it can minimize the access time. The Memory Hierarchy was developed based on a program behavior known as locality of references.



Memory Hierarchy Design is divided into 2 main types:

1. **External Memory or Secondary Memory** – Comprising of Magnetic Disk, Optical Disk, Magnetic Tape i.e. peripheral storage devices which are accessible by the processor via I/O Module.
2. **Internal Memory or Primary Memory** – Comprising of Main Memory, Cache Memory & CPU registers. This is directly accessible by the processor.

Memory Hierarchy Design

Auxiliary Memory

Auxiliary memory is known as the lowest-cost, highest-capacity and slowest-access storage in a computer system. Auxiliary memory provides storage for programs and data that are kept for long-term storage or when not in immediate use. The most common examples of auxiliary memories are magnetic tapes and magnetic disks.

A magnetic disk is a digital computer memory that uses a magnetization process to write, rewrite and access data. For example, hard drives, zip disks, and floppy disks.

Magnetic tape is a storage medium that allows for data archiving, collection, and backup for different kinds of data.

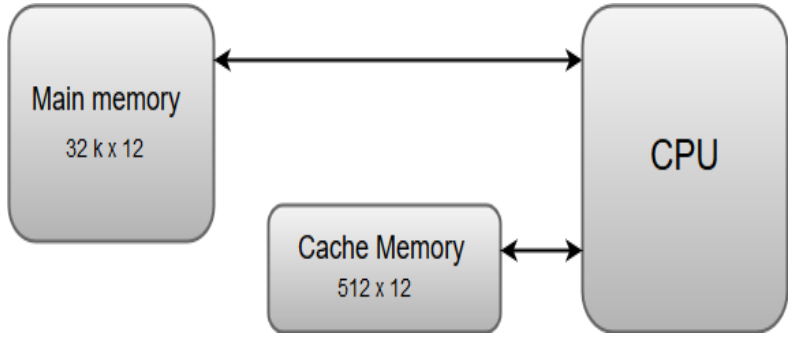
Main Memory

The main memory in a computer system is often referred to as **Random Access Memory (RAM)**. This memory unit communicates directly with the CPU and with auxiliary memory devices through an I/O processor.

The programs that are not currently required in the main memory are transferred into auxiliary memory to provide space for currently used programs and data.

Cache Memory

Cache Memory is a special very high-speed memory. It is used to speed up and synchronizing with high-speed CPU. Cache memory is costlier than main memory or disk memory but economical than CPU registers. Cache memory is an extremely fast memory type that acts as a buffer between RAM and the CPU.



Purpose-

It is used for bridging the speed mismatch between the fastest CPU and the main memory.

•It does not let the CPU performance suffer due to the slower speed of the main memory.

Execution Of Program-

Whenever any program has to be executed, it is first loaded in the main memory.

•The portion of the program that is mostly probably going to be executed in the near future is kept in the cache memory.

•This allows CPU to access the most probable portion at a faster speed.

Cache Memory Locality of reference

Locality of reference –

Since size of cache memory is less as compared to main memory. So to check which part of main memory should be given priority and loaded in cache is decided based on locality of reference.

Types of Locality of reference

Spatial Locality of reference

This says that there is a chance that element will be present in the close nearness to the reference point and next time if again searched then more close proximity to the point of reference.

Temporal Locality of reference

In this Least recently used algorithm will be used. Whenever there is page fault occurs within a word will not only load word in main memory but complete page fault will be loaded because spatial locality of reference rule says that if you are referring any word next word will be referred in its register that's why we load complete page table so the complete block will be loaded.

Address and Address Space in OS

Logical Address is generated by CPU while a program is running. The logical address is virtual address as it does not exist physically, therefore, it is also known as Virtual Address. This address is used as a reference to access the physical memory location by CPU. The term Logical Address Space is used for the set of all logical addresses generated by a program's perspective.

The hardware device called Memory-Management Unit is used for mapping logical address to its corresponding physical address.

Physical Address identifies a physical location of required data in a memory. The user never directly deals with the physical address but can access by its corresponding logical address. 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, therefore, the logical address must be mapped to the physical address by MMU before they are used. The term Physical Address Space is used for all physical addresses corresponding to the logical addresses in a Logical address space.

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 disk that's set up to emulate the computer's RAM.

Virtual Memory is a storage scheme that provides user an illusion of having a very big main memory. This is done by treating a part of secondary memory as the main memory.

In this scheme, User can load the bigger size processes than the available main memory by having the illusion that the memory is available to load the process.

Instead of loading one big process in the main memory, the Operating System loads the different parts of more than one process in the main memory.

By doing this, the degree of multiprogramming will be increased and therefore, the CPU utilization will also be increased.

Operating System
KCS – 401

Page Replacement Policy

Dr. Pankaj Kumar
Associate Professor – CSE
SRMGPC Lucknow

Outline of the Lecture

Introduction

FIFIO

OTIMAKL

LRU

Page Replacement Policy

In Demand Paging, only certain pages of a process are loaded initially into the memory. This allows us to get more processes into memory at the same time. but what happens when a process requests for more pages and no free memory is available to bring them in. Following steps can be taken to deal with this problem :

1. Put the process in the wait queue, until any other process finishes its execution thereby freeing frames.
2. Or, remove some other process completely from the memory to free frames.
3. Or, find some pages that are not being used right now, move them to the disk to get free frames.

This technique is called **Page replacement** and is most commonly used.

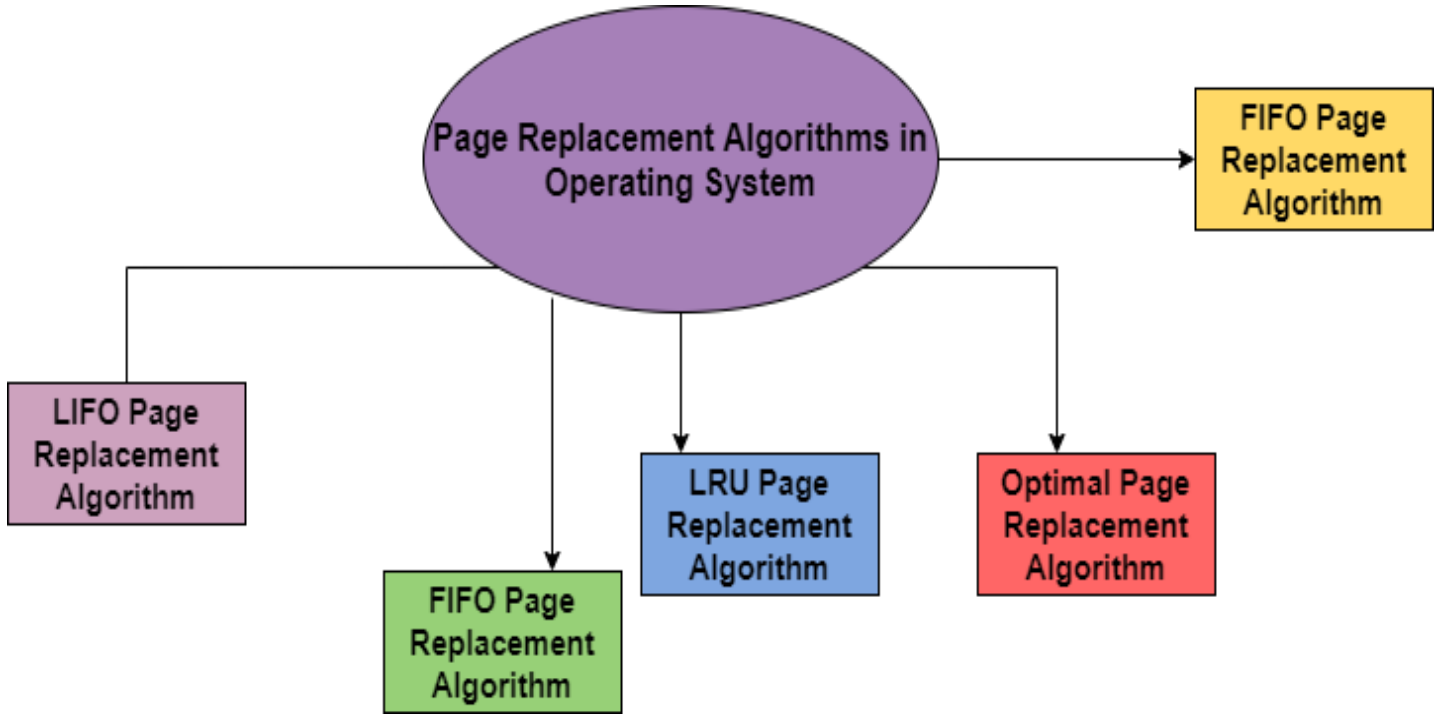
In this case, if a process requests a new page and supposes there are no free frames, then the Operating system needs to decide which page to replace. The operating system must use any page replacement algorithm in order to select the victim frame. The Operating system must then write the victim frame to the disk then read the desired page into the frame and then update the page tables.

Page Replacement Policy

Page Replacement Algorithms

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.

This algorithm helps to decide which pages must be swapped out from the main memory in order to create a room for the incoming page. This Algorithm wants the lowest page-fault rate.



Page Replacement Policy

Page Replacement Algorithms

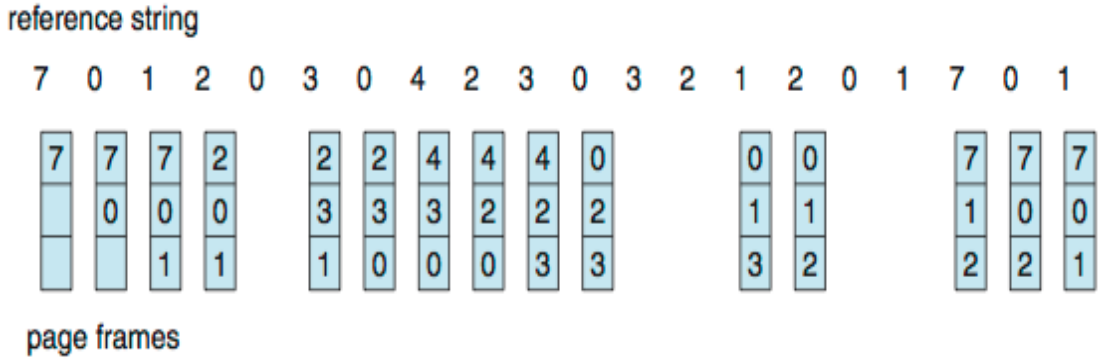
First In First Out (FIFO) –

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.

Example-1 Consider page reference string 7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1

3 frames (3 pages can be in memory at a time per process)

Find number of page faults.



15 page faults

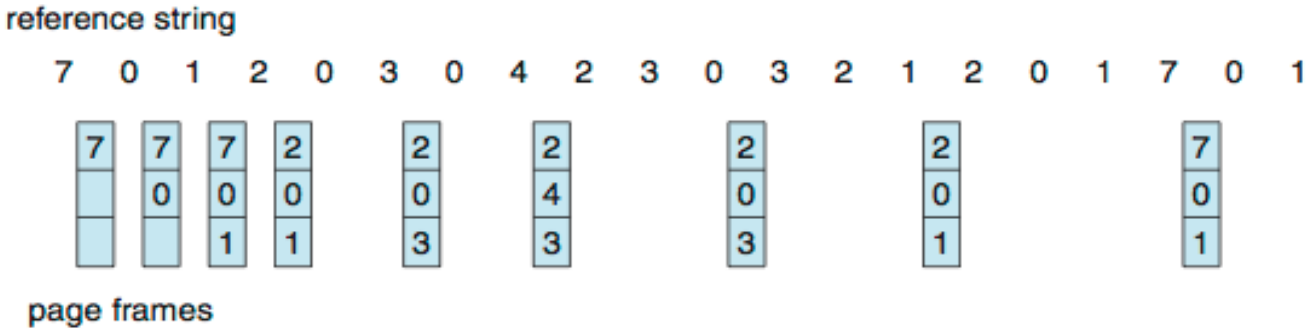
Page Replacement Policy

Page Replacement Algorithms

Optimal Page replacement –

In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.

Example-1 Consider page reference string 7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1
3 frames (3 pages can be in memory at a time per process)
Find number of page faults.



9 page faults

Page Replacement Policy

Page Replacement Algorithms

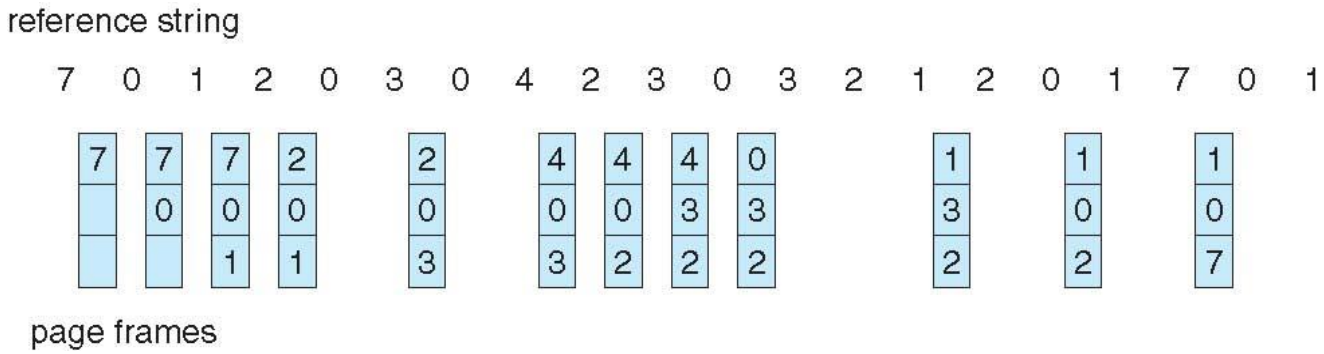
Least Recently Used –

In this algorithm page will be replaced which is least recently used.

Example-1 Consider page reference string 7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1

3 frames (3 pages can be in memory at a time per process)

Find number of page faults.



12 page faults

Operating System
KCS – 401

Paging

Dr. Pankaj Kumar
Associate Professor – CSE
SRMGPC Lucknow

Outline of the Lecture

Paging

Segmentation

Demand Paging

Paging

Paging is a storage mechanism that allows OS to retrieve processes from the secondary storage into the main memory in the form of pages. In the Paging method, the main memory is divided into small fixed-size blocks of physical memory, which is called frames. The size of a frame should be kept the same as that of a page to have maximum utilization of the main memory and to avoid external fragmentation. Paging is used for faster access to data, and it is a logical concept.

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.

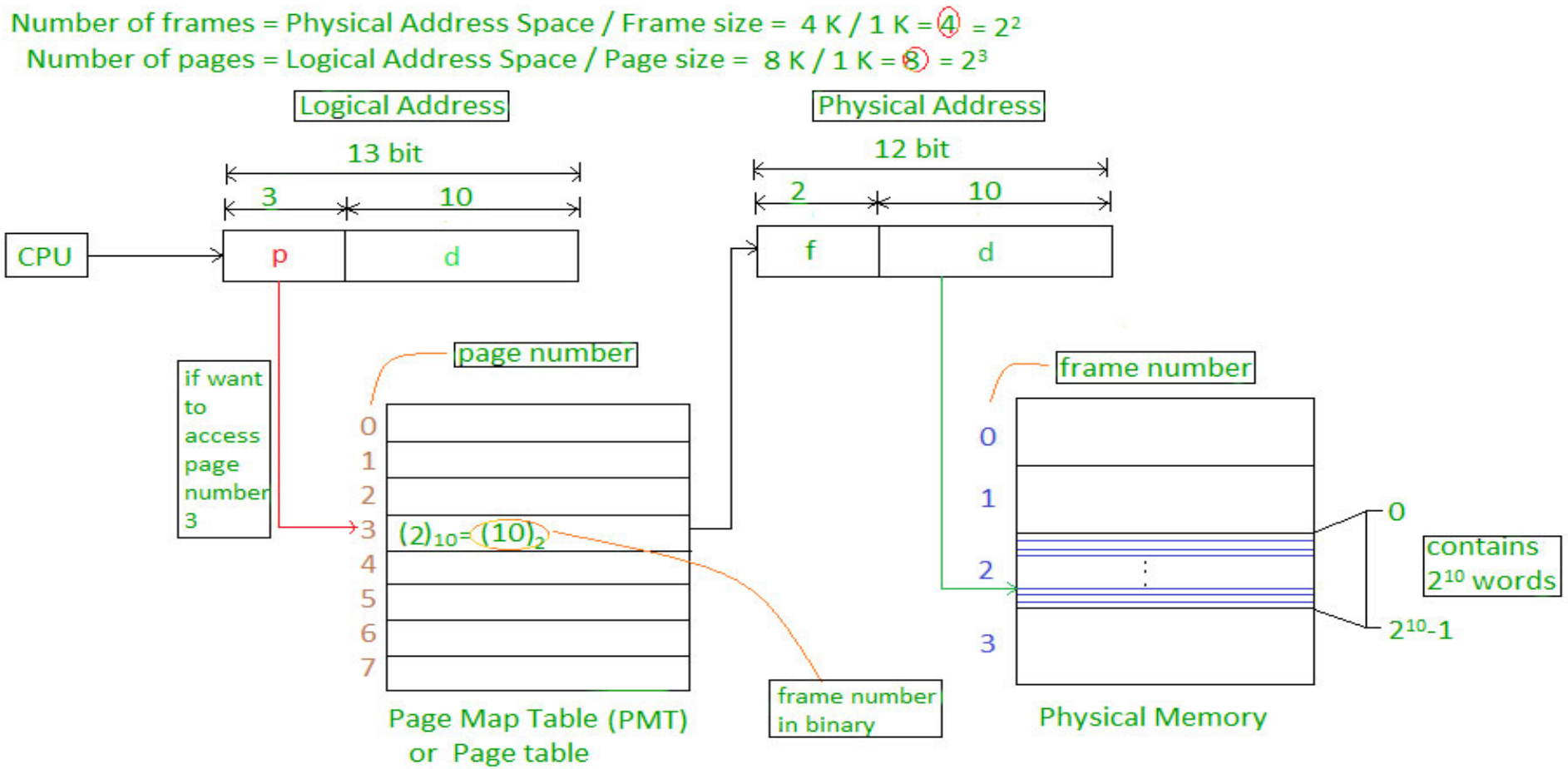
Paging

Example:

- If Logical Address = 31 bit, then Logical Address Space = 2^{31} words = 2 G words (1 G = 2^{30})
- If Logical Address Space = 128 M words = $2^7 * 2^{20}$ words, then Logical Address = $\log_2 2^{27}$ = 27 bits
- If Physical Address = 22 bit, then Physical Address Space = 2^{22} words = 4 M words (1 M = 2^{20})
- If Physical Address Space = 16 M words = $2^4 * 2^{20}$ words, then Physical Address = $\log_2 2^{24}$ = 24 bits
- The mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device and this mapping is known as paging technique.
- 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

Paging

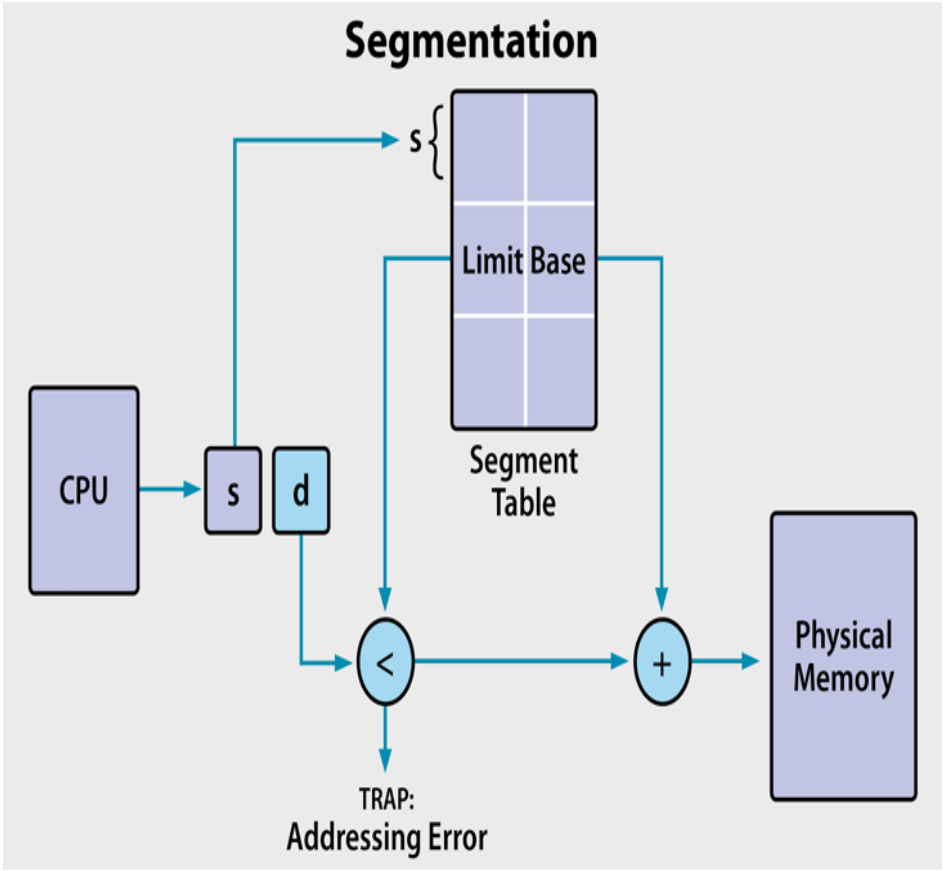
- Physical Address = 12 bits, then Physical Address Space = 4 K words
- Logical Address = 13 bits, then Logical Address Space = 8 K words
- Page size = frame size = 1 K words (assumption)



Segmentation

Segmentation is another technique of memory management. This technique is similar to Paging except for the fact that in segmentation the segments of a process are of variable length but in Paging the pages are of fixed size.

The memory is split into variable-length parts in segmentation. Each part is commonly known as segments. Information related to each segment is stored in a table and this table is commonly known as **the segment table.** The **segment table** generally occupies less space as compared to the **paging table.**

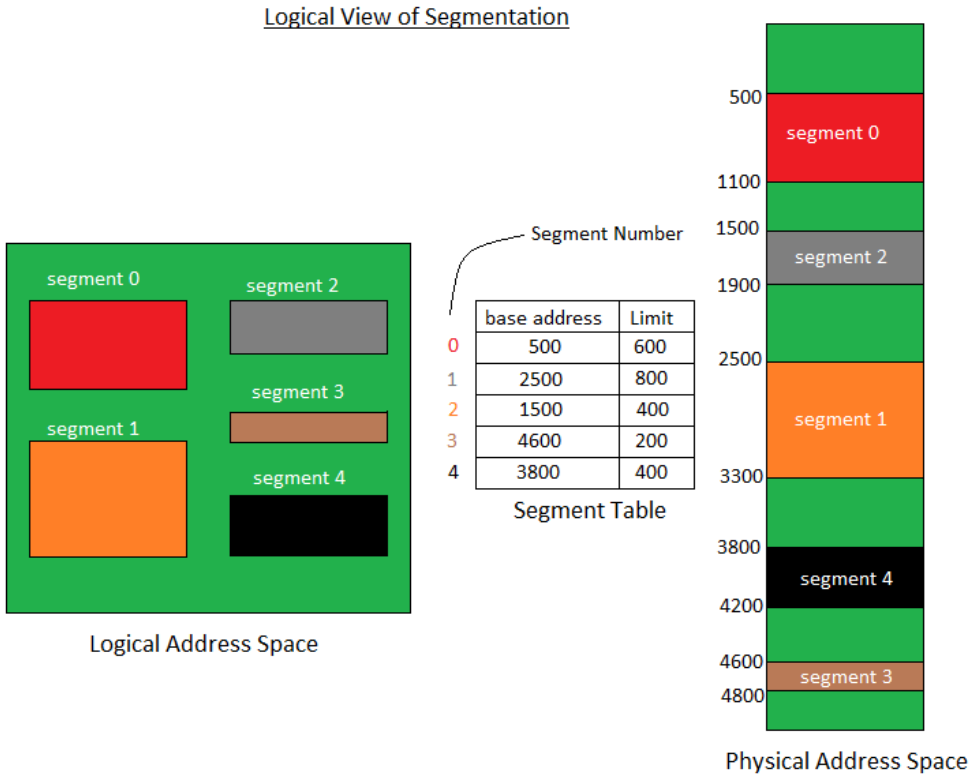


Segmentation

There is no simple relationship between logical addresses and physical addresses in segmentation. A table stores the information about all such segments and is called Segment Table.

Segment Table – It maps two-dimensional Logical address into one-dimensional Physical address. It's each table entry has:

- **Base Address:** It contains the starting physical address where the segments reside in memory.
- **Limit:** It specifies the length of the segment.

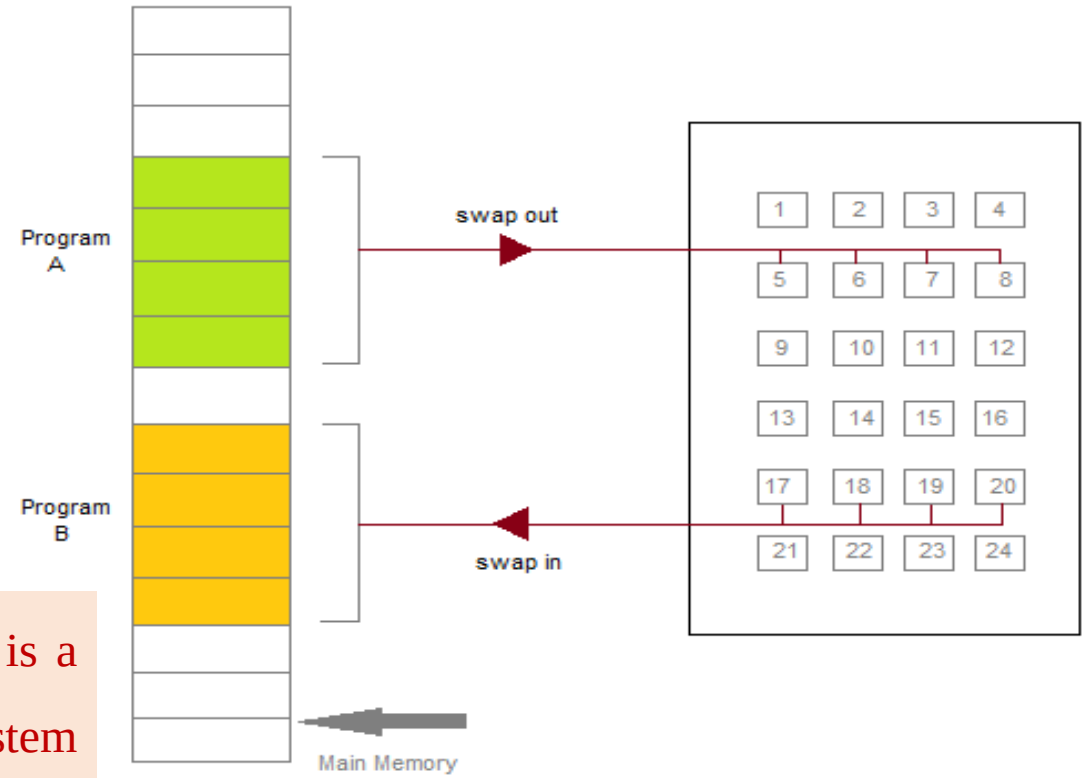


Segmentation

Sr. No.	Key	Paging	Segmentation
1	Memory Size	In Paging, a process address space is broken into fixed sized blocks called pages.	In Segmentation, a process address space is broken in varying sized blocks called sections.
2	Accountability	Operating System divides the memory into pages.	Compiler is responsible to calculate the segment size, the virtual address and actual address.
3	Size	Page size is determined by available memory.	Section size is determined by the user.
4	Speed	Paging technique is faster in terms of memory access.	Segmentation is slower than paging.
5	Fragmentation	Paging can cause internal fragmentation as some pages may go underutilized.	Segmentation can cause external fragmentation as some memory block may not be used at all.
6	Logical Address	During paging, a logical address is divided into page number and page offset.	During segmentation, a logical address is divided into section number and section offset.
7	Table	During paging, a logical address is divided into page number and page offset.	During segmentation, a logical address is divided into section number and section offset.
8	Data Storage	Page table stores the page data.	Segmentation table stores the segmentation data.

Demand Paging

Demand Paging is a technique in which a page is usually brought into the main memory only when it is needed or demanded by the CPU. Initially, only those pages are loaded that are required by the process immediately. Those pages that are never accessed are thus never loaded into the physical memory.



In computer operating systems, demand 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.

Operating System
KCS – 401

Virtual Memory

Dr. Pankaj Kumar
Associate Professor – CSE
SRMGPC Lucknow

Background

Code needs to be in memory to execute, but entire program rarely used

- Error code, unusual routines, large data structures

Entire program code not needed at same time

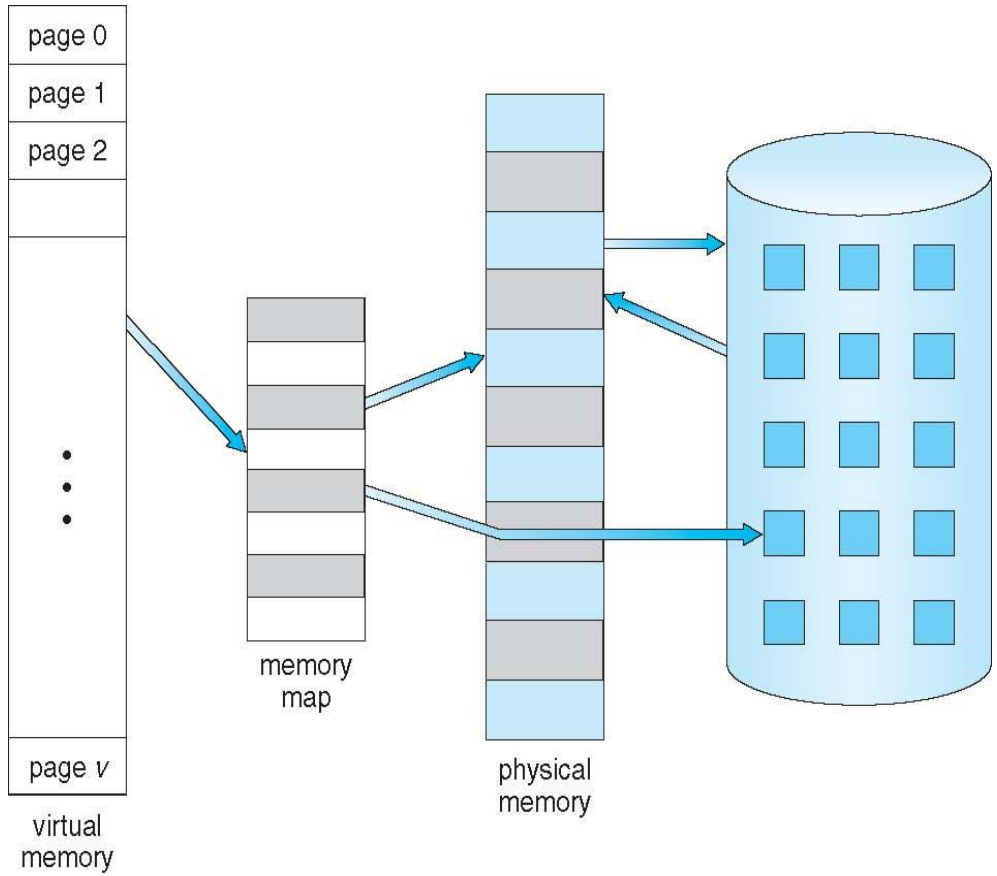
Consider ability to execute partially-loaded program

- Program no longer constrained by limits of physical memory
- Each program takes less memory while running -> more programs run at the same time
  - Increased CPU utilization and throughput with no increase in response time or turnaround time
- Less I/O needed to load or swap programs into memory -> each user program runs faster

Virtual Memory

Virtual memory – separation of user logical memory from physical memory

- Only part of the program needs to be in memory for execution
- Logical address space can therefore be much larger than physical address space
- Allows address spaces to be shared by several processes
- Allows for more efficient process creation
- More programs running concurrently
- Less I/O needed to load or swap processes



Virtual Memory

Virtual address space – logical view of how process is stored in memory

- Usually start at address 0, contiguous addresses until end of space
- Meanwhile, physical memory organized in page frames
- MMU must map logical to physical

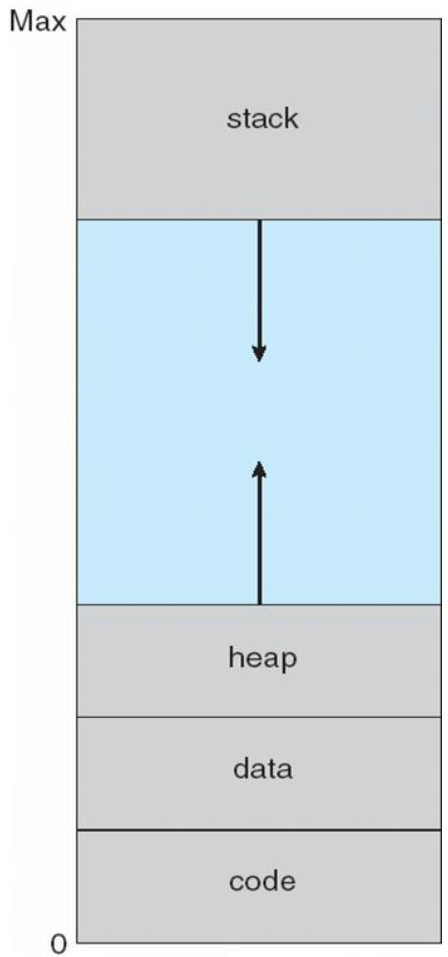
Virtual memory can be implemented via:

- Demand paging
- Demand segmentation

Usually design logical address space for stack to start at Max logical address and grow "down" while heap grows "up"

- Maximizes address space use
- Unused address space between the two is hole
  - No physical memory needed until heap or stack grows to a given new page

Enables **sparse** address spaces with holes left for growth, dynamically linked libraries, etc



Virtual Memory

Virtual address space – logical view of how process is stored in memory

- Usually start at address 0, contiguous addresses until end of space
- Meanwhile, physical memory organized in page frames
- MMU must map logical to physical

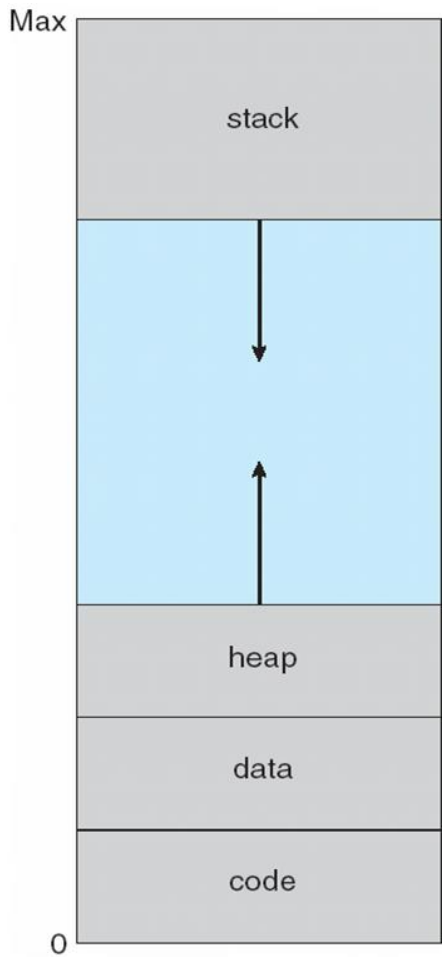
Virtual memory can be implemented via:

- Demand paging
- Demand segmentation

Usually design logical address space for stack to start at Max logical address and grow "down" while heap grows "up"

- Maximizes address space use
- Unused address space between the two is hole
  - No physical memory needed until heap or stack grows to a given new page

Enables **sparse** address spaces with holes left for growth, dynamically linked libraries, etc



Page Fault

If there is a reference to a page, first reference to that page will trap to operating system:

page fault

1.Operating system looks at another table to decide:

Invalid reference ⇒ abort

Just not in memory

2.Find free frame

3.Swap page into frame via scheduled disk operation

4.Reset tables to indicate page now in memory

Set validation bit = **v**

5.Restart the instruction that caused the page fault