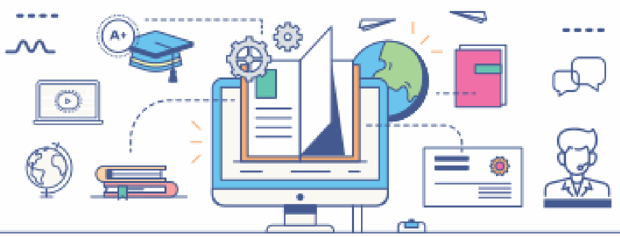
Operating System

Prof. Umang Panchal , Prof. Nirali Bhaliya, Prof. Riddhi Mehta Assistant Professor
Computer & Science Engineering



CHAPTER-5

Memory Management

Topics

1. Memory Management: Basic Concept.
2. Logical and Physical address map.
3. Memory Allocation: Contiguous Memory Allocation.
4. Fixed and Variable Partition.
5. Internal and External Fragmentation and Compaction.
6. Paging: Principle of Operation.
7. Page Allocation.
8. Hardware support for paging

What is Memory Management?

- Memory Management is an important characteristic of the Operating System. This comes beneath one of the two main capabilities of an Operating System, useful resource management. **Main memory (RAM)** is where maximum of the programs run. It is one of the most important matters people search for whilst buying a new cellphone or a new laptop.

What is Memory hierarchy ?

- In computer system architecture memory Hierarchy is part to arrange the memory in a such a way it can minimize access time.

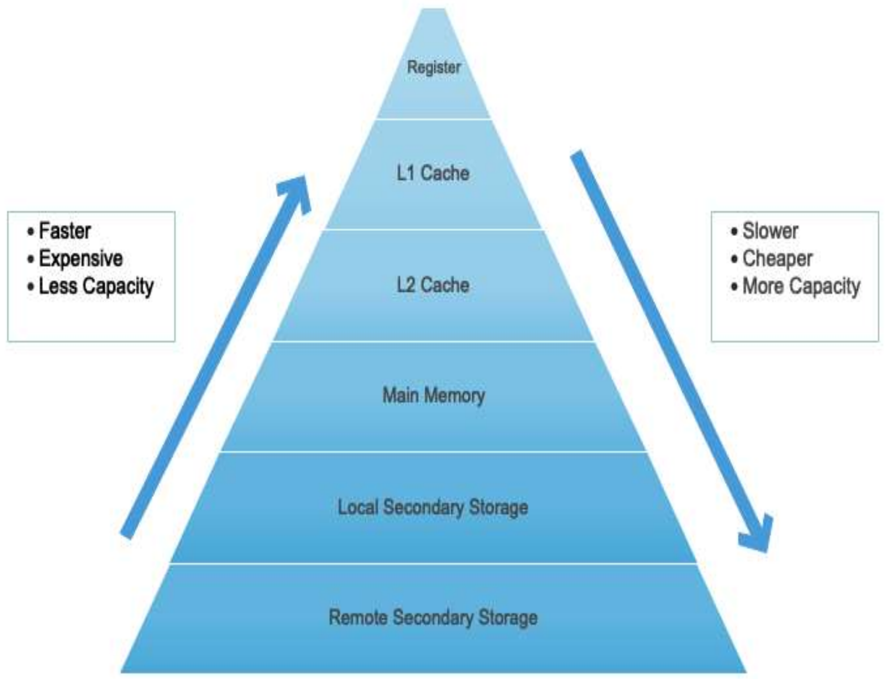


Figure - 1 Memory Hierarchy [1]

Logical and physical address

- **Logical address** is CPU generated address and it generate while program is running.
- It considers as a virtual address and it not exist physically.
- It use as a reference for access Physical memory point by CPU.
- In a memory physical location identifies by **Physical address.**

Logical and physical address

- Physical address not directly access by the user but it can be accessible by its corresponding logical address.
- The term Physical Address Space is utilized for every physical location relating to the logical addresses in a Logical location space.
- MMU computes **Physical address.**

What is Memory hierarchy ?

- It is a hardware device which uses to map virtual address to physical address.

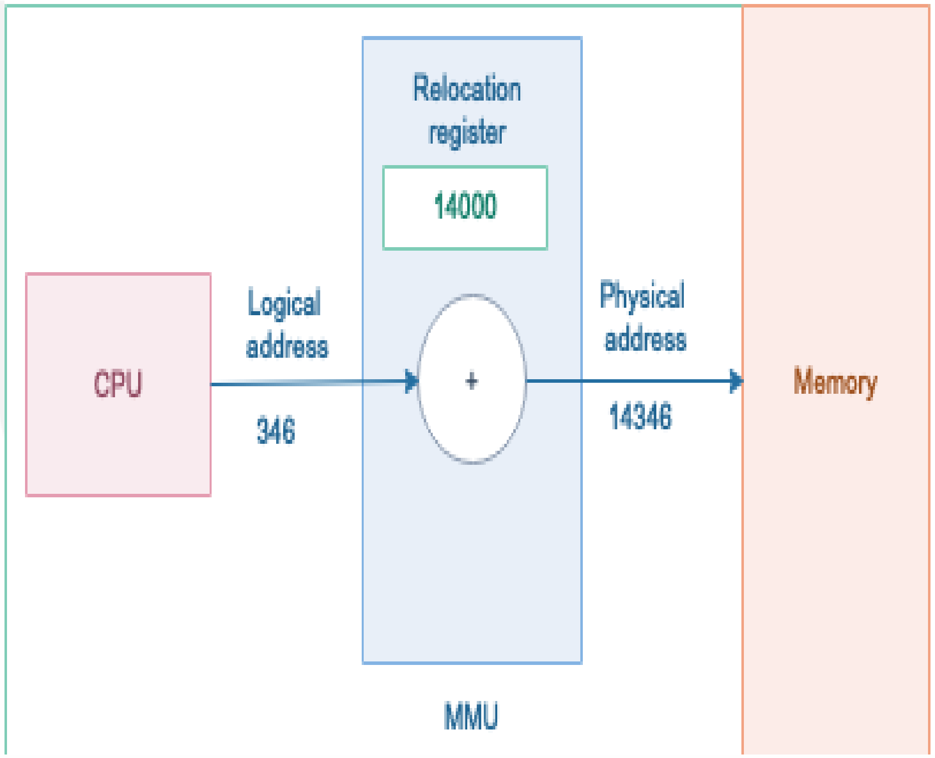


Figure - 2 MMU [2]

Binding of Instructions and Data to Memory

- **Compile time**: Helps to convert symbolic addresses to absolute addresses. If you know at compile time where the process will reside in memory, then absolute code can be generated (Static).

- **Load time**: The compiler translates symbolic addresses to relative addresses. The loader translates these to absolute addresses. If it is not known at compile time where the process will reside in memory, then the compiler must generate relocatable code (Static).

Binding of Instructions and Data to Memory

- **Execution time:** If the process can be moved during its execution from one memory segment to another, then binding must be delayed until run time. The absolute addresses are generated by hardware. Most general-purpose OSs use this method (Dynamic).

- **Absolute code:** It is a code and data which will placed where you insist assembler to be placed.

- **Reloadable code:** Can load anywhere in memory it generally divide into control section and all memory address are expressed relative to the start of a control section.

Multistep processing of user program

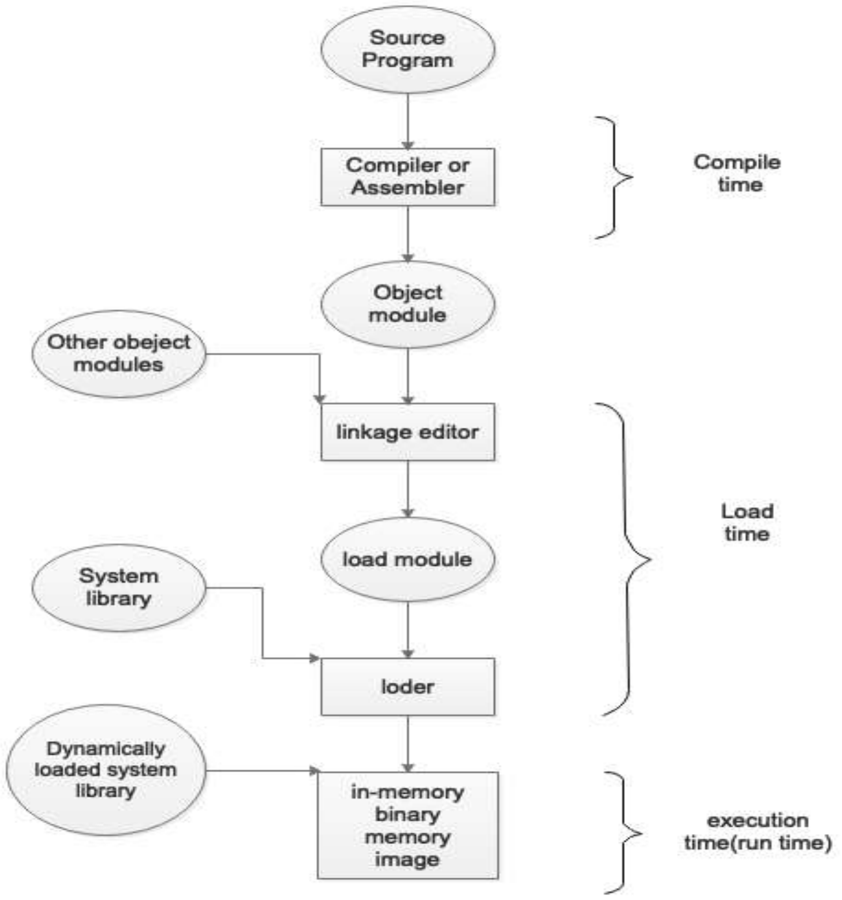


Figure – 3 Diagram of Multistep Processing of User program [3]

Memory Management Requirements

Relocation : when program get execute programmer does not know where the program will be placed in memory.

- It may be swapped to disk and get return into main memory at different location while the program is executing.
- Memory references must be translated in the code to actual physical memory address.

Memory Management Requirements

Protection: process need permission to reference memory locations into another process.

- At compile time It is impossible to check absolute addresses

Checked at run time.

Base and Limit register: Memory Protection

- Set of address that a process can use to address memory called an address space. It is range od valid addresses in a memory which are available for a program or process.
- Start address of program **is base register.**
- **Length of program is limit register.**
- **Base and limit register only modify by OS**

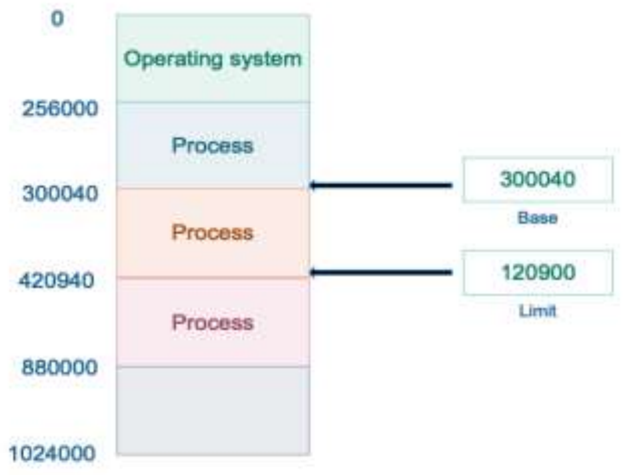


Figure – 4 a based and a limit register define a logical address space [4]

Hardware Address protection using base and limit registers: Dynamic relocation

- Assurance of memory space is practiced by having CPU equipment think about each address produced in client mode with the registers.
- Any endeavor by client program running in client mode to get to the memory space of the working framework or some other client program, brings about a trap.
- This plan restricts a client program from adjusting the code or information structure of the working framework or other client programs.

Parul® University

DIGITAL LEARNING CONTENT

Hardware Address protection using base and limit registers: Dynamic relocation

- Steps in dynamic relocation .
- Equipment includes migration register (base) to the virtual location to get a physical location.
- Equipment contrasts address and breaking point register; the location must be not exactly or equivalent cutoff.
- In the event that the test falls flat, the processor takes a location trap and overlooks the physical location.

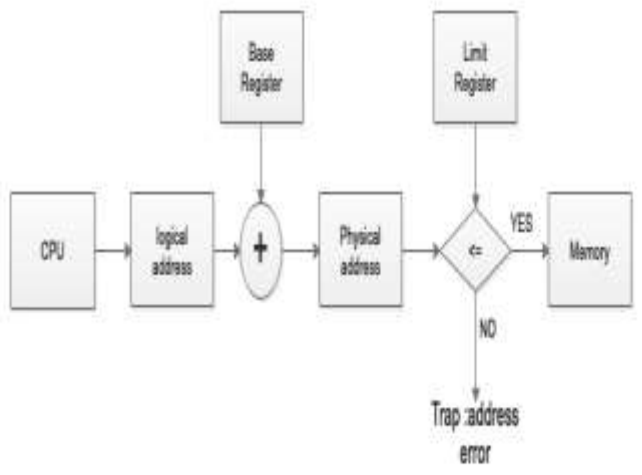


Figure – 5 Hardware address protection [5]

Static vs Dynamic Loading

Static	Dynamic
1.Entire program load into main memory before program start execution.	1.On demand it load program into memory
2.Memory utilization is inefficient because it require or not it load whole program into main memory.	2. Memory utilization is efficient because it load program on demand.
3. Fast execution of program	3. Slower execution of program
4. Every time it takes load time when program load into main memory	4. Perform at a run time by operating system
5. Static linking is applied if the static loading used accordingly	5. dynamic linking is applied if the dynamic loading used accordingly
6.To start execution absolute data and program load into memory	6. Bit by bit loading of data and information in run time

Swapping

- For temporarily any process can be swapped out of memory to backing store and take back into memory for further execution.

- **Backing store** : disk large enough so that can accommodate copies of all memory images for all users.

- **Roll out roll in:** for priority based scheduling algorithm swapping variant used; to load high-priority process and to get execute lower priority process need to swapped out.

- Ready queue maintain by the system for ready -to-run processes which have memory images on disk.

Schematic View of Swapping

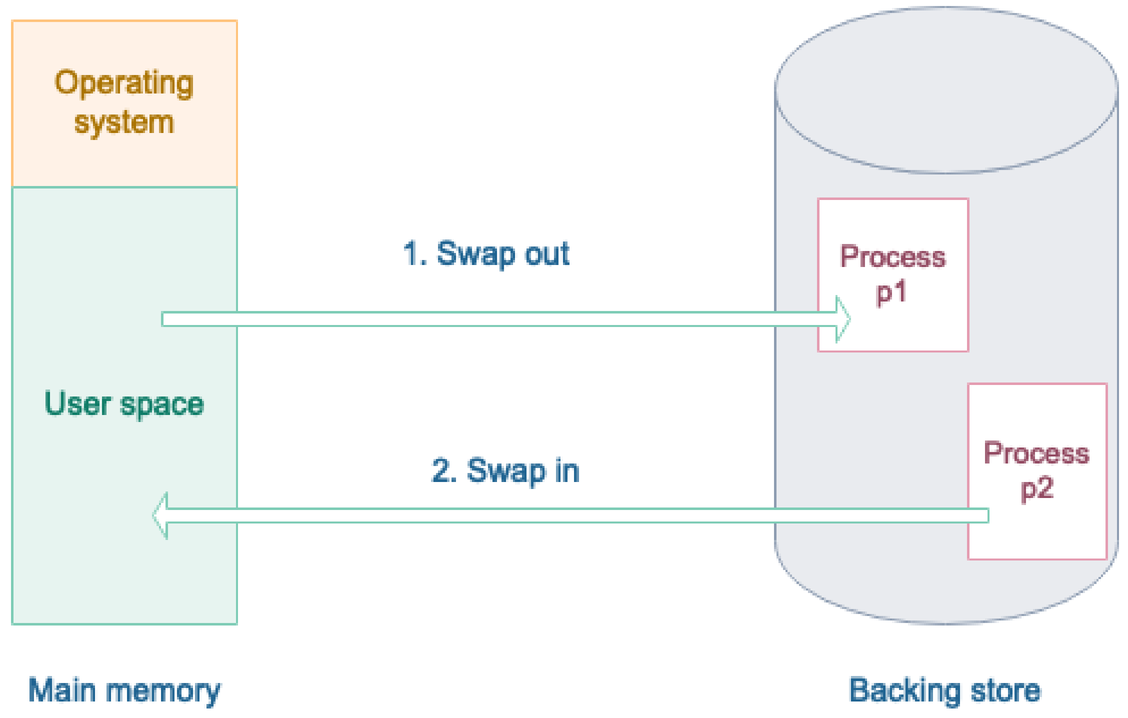


Figure – 6 Schematic view Swapping [6]

Memory Management Techniques (MMT)

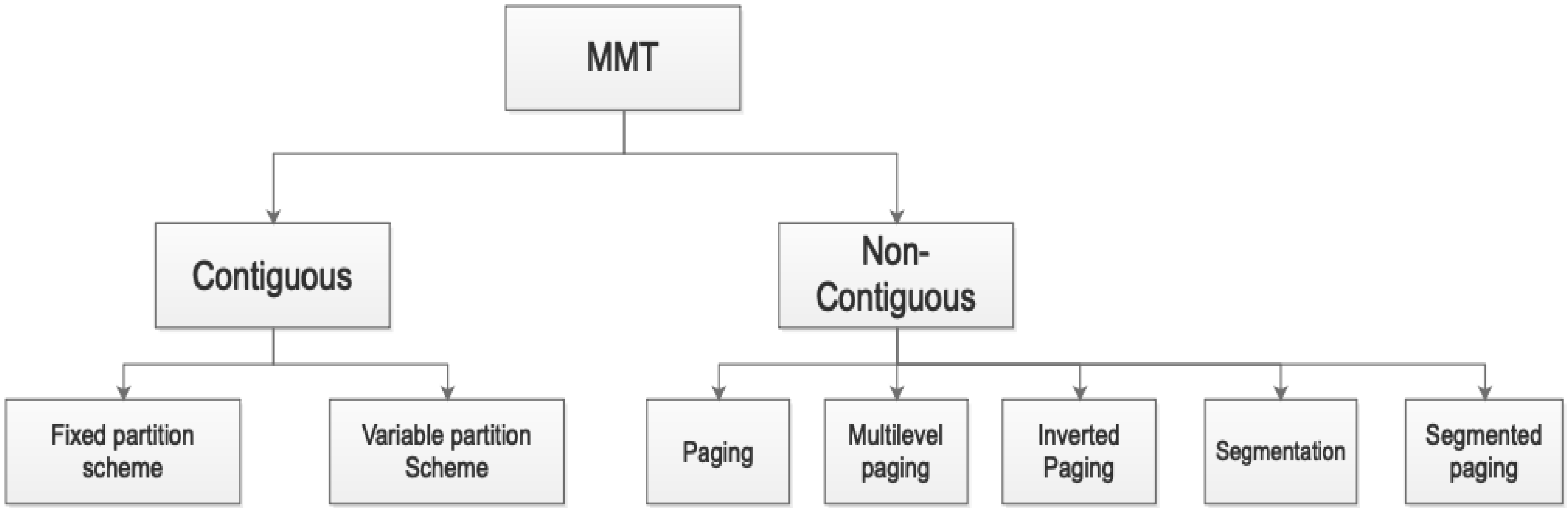


Figure – 7 MMT [7]

Memory Allocation: Contiguous Memory Allocation

• Memory allocation is a mechanism by which computer programs and services are assigned with physical or virtual memory space.

• Memory allocation is the process of reserving a partially or entirely portion of computer memory.

• There are 2 types of memory allocation:

1. Contiguous memory allocation

2. Non – Contiguous memory allocation

Contiguous Memory Allocation

- Contiguous memory allocation is known as a memory allocation model in which a single contiguous adjacent area is allocated in the memory for each program.
- When a process needs to execute, memory is requested by the process.
- The size of the process is compared with the amount of contiguous main memory available to execute the process.

Contiguous Memory Allocation

- If enough contiguous memory is found, the process is allocated memory to start its execution.

- Otherwise, waiting processes are added to a queue until there is sufficient free contiguous memory available.

Contiguous Memory Allocation

- In operating systems, the contiguous memory allocation scheme can be implemented with the aid of two registers, known as base and limit registers.
  - Main memory usually has two partitions :

Low Memory : Operating system resides in this memory.

High Memory : User processes are held in high memory.

Contiguous Memory Allocation

- Partitioning of memory can be done in the following ways:

  1. **Single Process Monitor**
  2. **Multiprogramming with Fixed Partition**
  3. **Multiprogramming with Variable Partition**

Single Process Monitor

- Main memory is dividing into the two parts. One is part of O.S. and second for user process area.
- For example: Here main memory 0-400MB space is reserve for the Operating System.
- Second part is part for user process in which user's process will be loaded into memory.

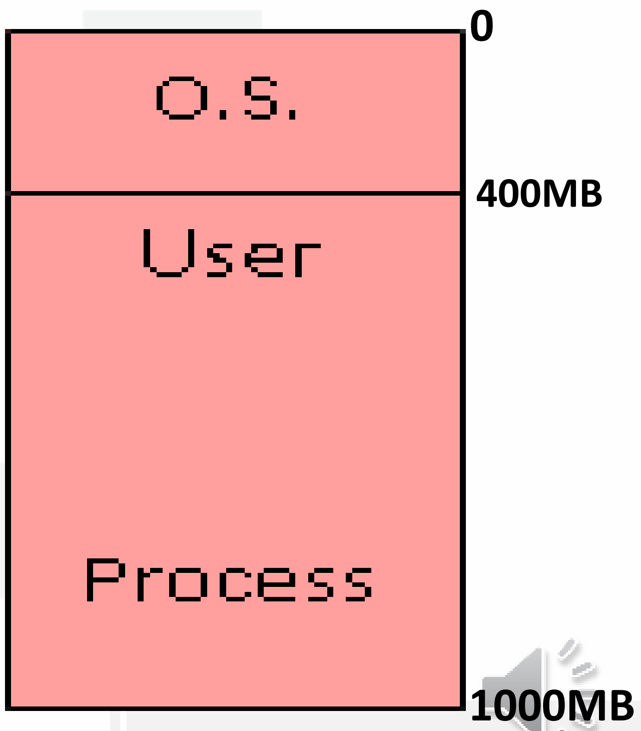


Figure – 8 Single Process Monitor[9]

Single Process Monitor

- Only one process can load to memory in Single Process Monitor. The next process will be loaded into memory when one process is complete.

- Single process monitor does not support multiprogramming but only supports uniprogramming.

Multiprogramming with Fixed Partition (MFT)

- The main memory is divided into multiple partitions to support the multiprogramming.
- The number of partitions are fixed but the size of each partition can be different.
- Each process will store into these partitions contiguously.

Multiprogramming with Fixed Partition (MFT)

- Here only one process is allowed into one partition.

- The degree of multi-programming is equivalent to the number of partitions as there can be many processes in the main memory at one time

Parul® University

DIGITAL LEARNING CONTENT

Multiprogramming with Fixed Partition (MFT)

- In fig.2 memory is divided into 5 equal size partitions. Each partition of 100MB.

Advantages of Fixed Partition:

1. **Easy to implement.**
2. **Little OS overhead.**

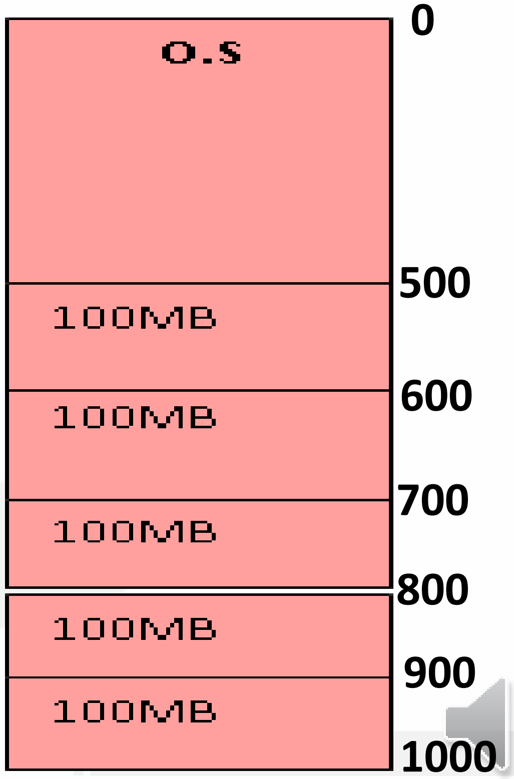


Figure - 9 Multiprogramming with Fixed Partition[9]

Multiprogramming with Fixed Partition (MFT)

Disadvantages of Fixed Partition:

1. Internal Fragmentation
2. External Fragmentation
3. Limit process size
4. Limitation on Degree of Multiprogramming

Problems with Fixed Partitioning

- Main problem with this arrangement is fragmentation.
- Fragmentation is the presence of unusable areas in the computer memory
- There are two types of Fragmentation

  1. Internal Fragmentation

  2. External fragmentation

Multiprogramming with Variable Partition(MVT)

- Partitions are created dynamically to avoid the difficulties of the fixed partitions.

- No partition is fixed in memory in dynamic partitions, but the partition is only created when the process is loaded into memory.

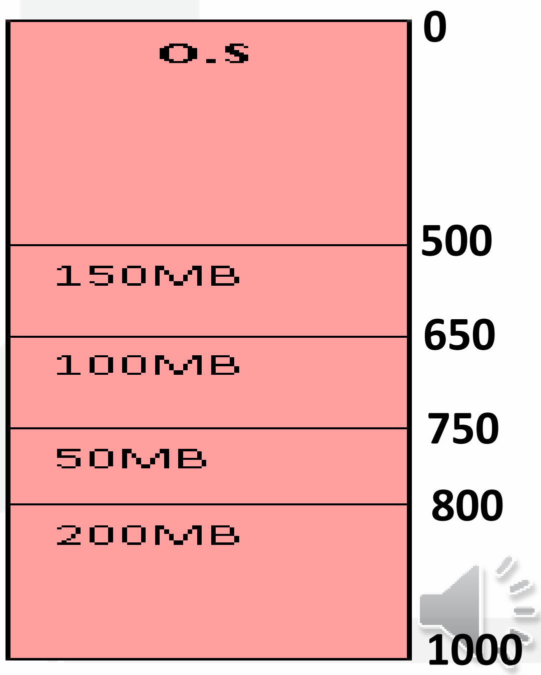


Figure - 10 Multiprogramming with Variable Partition[9]

Multiprogramming with Variable Partition(MVT)

- In figure above memory is divide into 4 unequal size partitions.

Partition size is not equal.

- Here the not problem of internal fragmentation. But here the

problem of external fragmentation.

Multiprogramming with Variable Partition(MVT)

Advantages of Variable Partition:

1. No internal fragmentation.
2. Efficient use of memory.

Disadvantages of Variable Partition:

1. Problem of external fragmentation.
2. Not easy to implement.

Parul® University

DIGITAL LEARNING CONTENT

Internal Fragmentation

- Internal fragmentation occurs when the memory is divided into fixed sized blocks.

- Whenever a process request for the memory, the fixed sized block is allocated to the process.

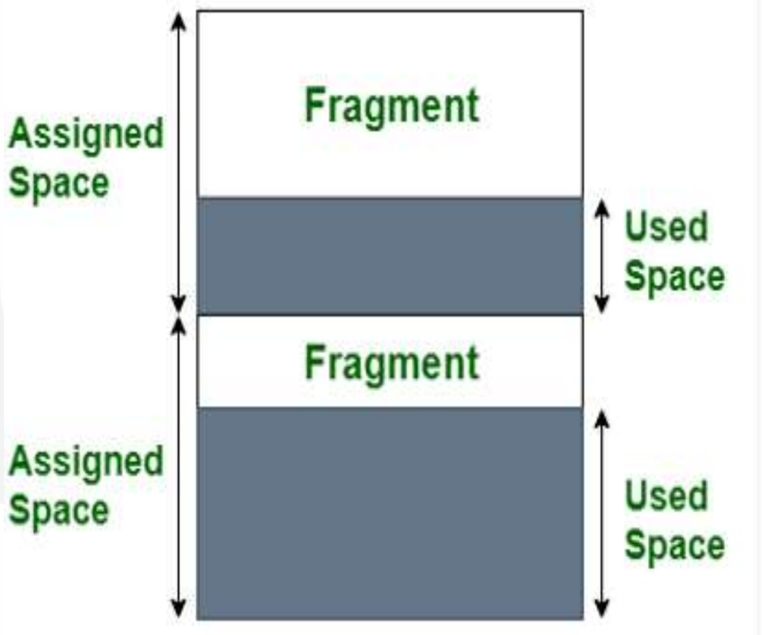


Figure - 11 Multiprogramming with Variable Partition[13]

Internal Fragmentation

- In case the memory assigned to the process is somewhat larger than the memory requested, then the difference between assigned and requested memory is the **Internal Fragmentation**.

- Internal fragmentation occurs when fixed sized memory blocks are allocated to the process without concerning about the size of the process.

Internal Fragmentation

- The diagram clearly shows the internal fragmentation because the difference between memory allocated and required space or memory is called Internal fragmentation.

Example of Internal Fragmentation

- Here p1, p2,p3,p4 is loaded into memory.
- Process P1 is loaded into first partition whose size is 150MB and size of process P1 is 110MB ,so in partition 40MB space is waste.
- This is called **Internal Fragmentation.**

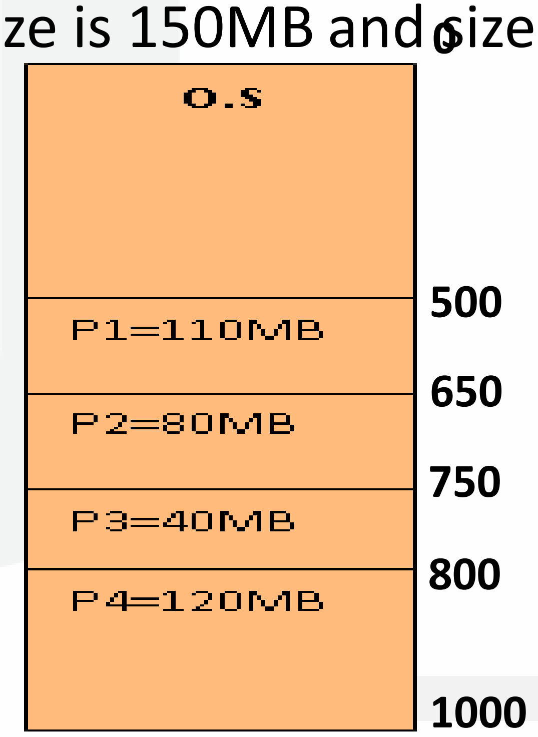


Figure - 12 Example of Internal Fragmentation [13]

External Fragmentation

- When the process is loaded or removed from the memory, a free space is created. This free space creates an empty space in the memory which is called external fragmentation.

- When the memory assigned to the process dynamically based on process request.

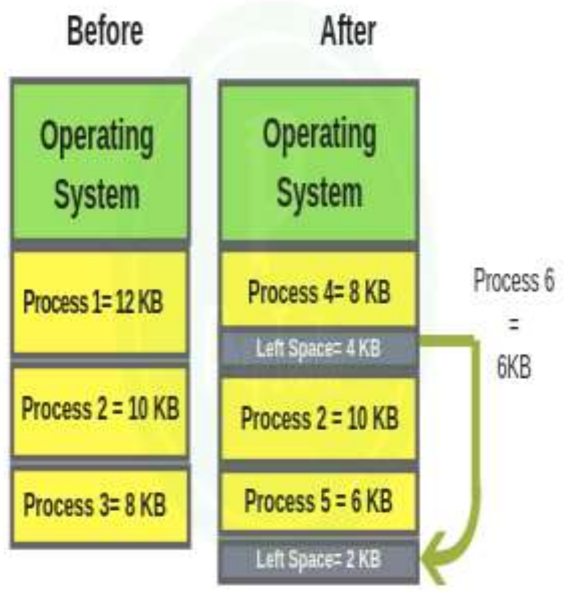


Figure - 13 Example of Internal Fragmentation [8]

External Fragmentation

- Memory is partitioned into variable size blocks.

- External fragmentation can be eliminated by compaction, paging, and segmentation so that memory can be allocated in a non-contiguous manner to a process.

Parul® University

DIGITAL LEARNING CONTENT

Compaction

- Compaction is **one of the technique to avoid External Fragmentation**.

- Move all the processes towards the top or towards the bottom to make free available space in a single contiguous block is known as **compaction.**

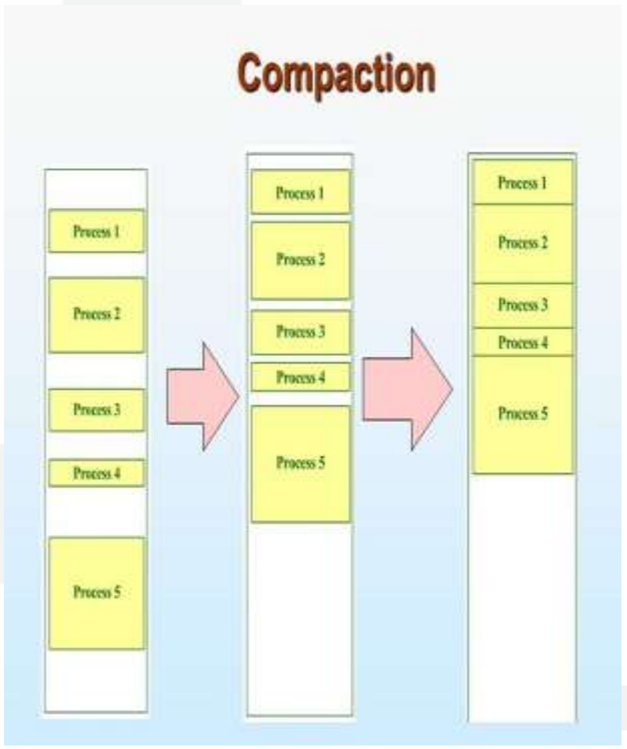


Figure - 14 Example of Internal Fragmentation [13]

Compaction

- Compaction is undesirable to implement because it interrupts all the running processes in the main Memory.

- The other technique to avoid external fragmentation is to implement non-contiguous memory allocation techniques.

Paging: Principle of Operation

- In Operating Systems, Paging is a storage mechanism used to retrieve processes from the secondary storage into the main memory in the form of pages.

- The main idea behind the paging is to divide each process in the form of pages.

- The main memory will also be divided in the form of frames.

Paging: Principle of Operation

- One page of the process is to be stored in one of the frames of the memory.

- The pages can be stored at the different locations of the memory, but the priority is always to find the contiguous frames or holes. Different operating system defines different frame sizes.

Paging: Principle of Operation

- The sizes of each frame must be equal. Considering the fact that the pages are mapped to the frames in Paging, page size needs to be as same as frame size.

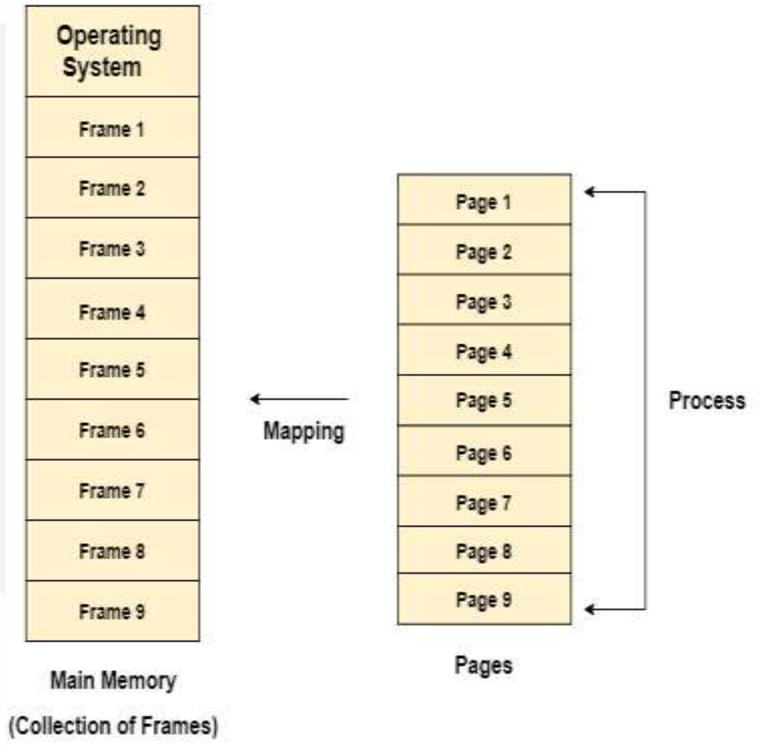


Figure - 15 Paging [11]

Example of Paging

- Let us consider the main memory size 16 Kb and Frame size is 1 KB therefore the main memory will be divided into the collection of 16 frames of 1 KB each.

- There are 4 processes In the system that is P1, P2, P3 and P4 of 4 KB each.

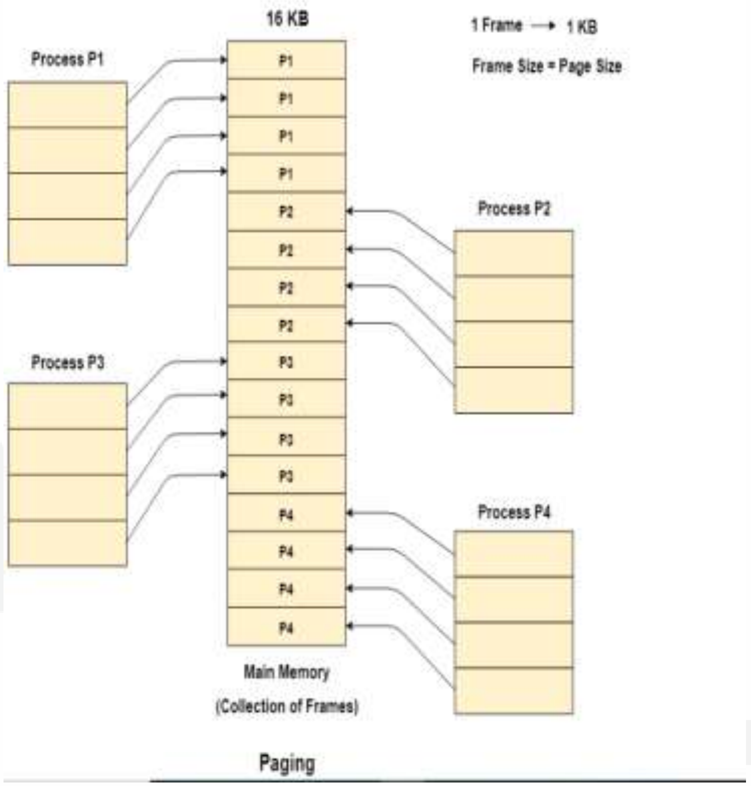


Figure – 16 Example of Paging[11]

Example of Paging

- Each process is divided into pages of 1 KB each so that one page can be stored in one frame.

- Initially, all the frames are empty therefore pages of the processes will get stored in the contiguous way.

Example of Paging

- Let us consider that, P2 and P4 are moved to waiting state after some time.

- Now, 8 frames become empty and therefore other pages can be loaded in that empty place.

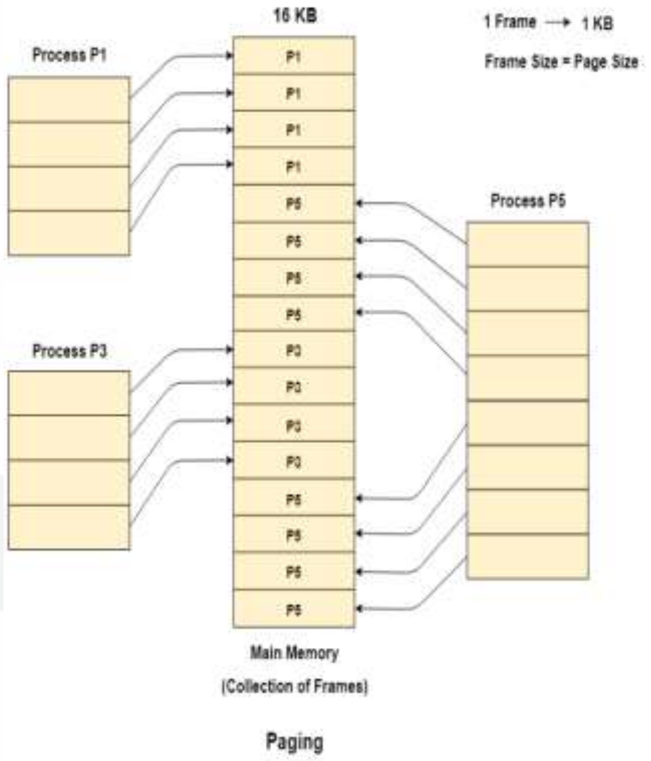


Figure - 17 Example of Paging[11]

Example of Paging

- The process P5 of size 8 KB (8 pages) is waiting inside the ready queue.

- Given the fact that, we have 8 non contiguous frames available in the memory and paging provides the flexibility of storing the process at the different places. Therefore, we can load the pages of process P5 in the place of P2 and P4.

Memory Management Unit

- The purpose of Memory Management Unit (MMU) is to convert the logical address into the physical address.

- The logical address is the address generated by the CPU for every page while the physical address is the actual address of the frame where each page will be stored.

Memory Management Unit

- The logical address has two parts:

  1. **Page Number**
  2. **Offset**

- Memory management unit of OS needs to convert the page number to the frame number.

.

Physical Address Space

- Physical address space in a system can be defined as the size of the main memory.
- It is really important to compare the process size with the physical address space. The process size must be less than the physical address space.
- **Syntax of Physical Address Space is:**

Physical Address Space = Size of the Main Memory

Example of Physical Address Space

If, physical address space = 64 KB = 2 ^ 6 KB = 2 ^ 6 X 2 ^ 10 Bytes = 2 ^ 16 bytes

Let us consider,

word size = 8 Bytes = 2 ^ 3 Bytes

Hence,

Physical address space (in words) = (2 ^ 16) / (2 ^ 3) = 2 ^ 13 Words

Therefore,

Physical Address = 13 bits

Logical Address Space

- Logical address space can be defined as the size of the process. The size of the process should be less enough so that it can reside in the main memory.

For Example of Logical Address Space is:

Logical Address Space = 128 MB = (2 ^ 7 X 2 ^ 20) Bytes = 2 ^ 27 Bytes

Word size = 4 Bytes = 2 ^ 2 Bytes

Logical Address Space (in words) = (2 ^ 27) / (2 ^ 2) = 2 ^ 25 Words

Logical Address = 25 Bits

Logical Address Space

- What is a Word?

The Word is the smallest unit of the memory. It is the collection of bytes. Every operating system defines different word sizes after analyzing the n-bit address that is inputted to the decoder and the 2 ^ n memory locations that are produced from the decoder.

.

Page Table

- Page Table is a data structure used by the virtual memory system to store the mapping between logical addresses and physical addresses.
- Logical addresses are generated by the CPU for the pages of the processes therefore they are generally used by the processes.
- Physical addresses are the actual frame address of the memory.
- They are generally used by the hardware or more specifically by RAM subsystems.

Parul® University

DIGITAL LEARNING CONTENT

Representation of Physical and Logical Address

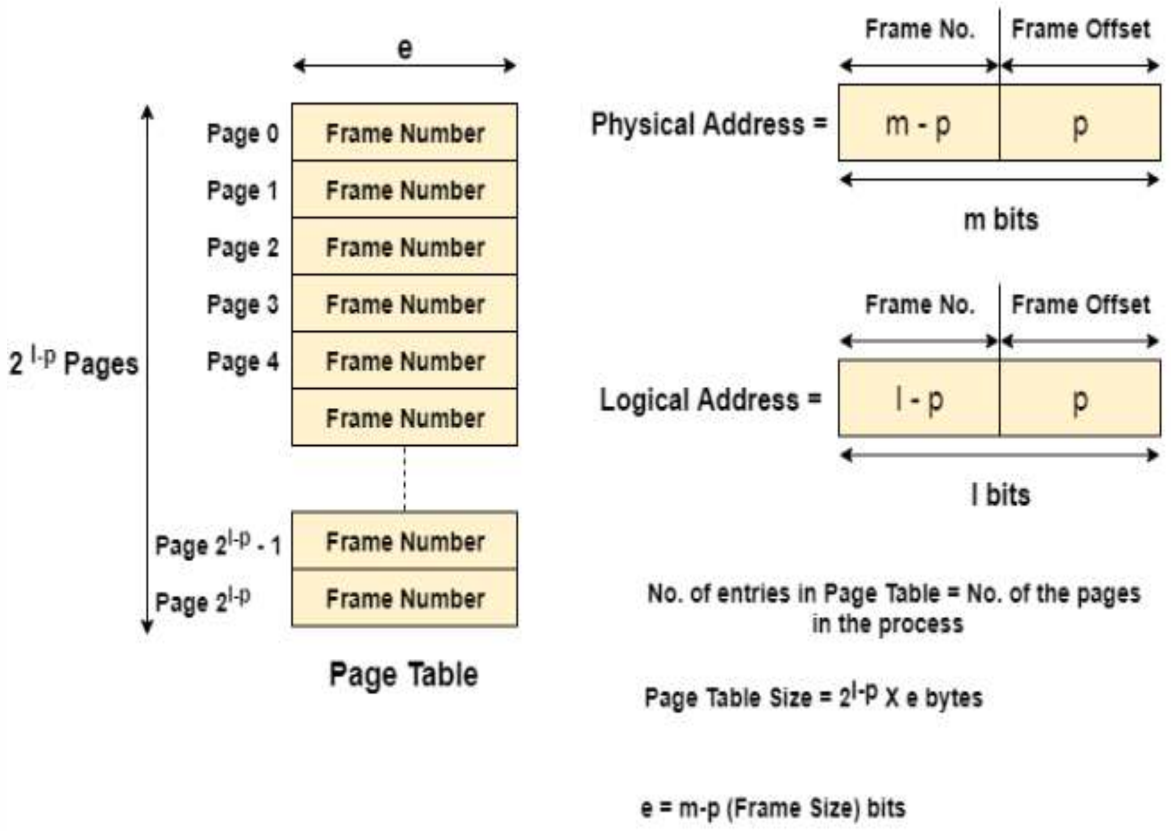


Figure - 18 Representation of Physical and Logical Address [10]

Parul® University

DIGITAL LEARNING CONTENT

Example of Page Table

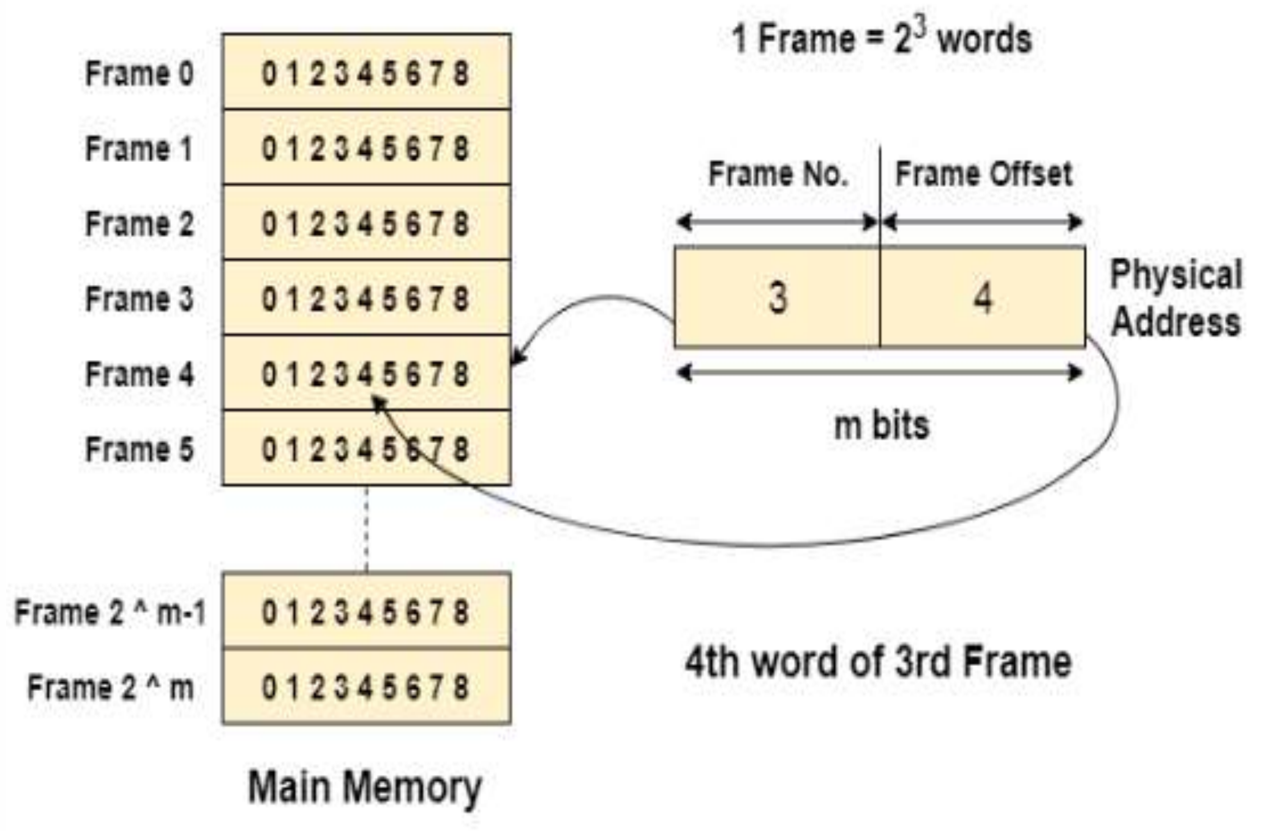


Figure - 19 Example of Page Table[10]

Mapping From Page Table To Main Memory

- In operating systems, there is always a requirement of mapping from logical address to the physical address.

- However, this process involves various steps which are defined as follows.

Step 1: Generation of logical address

Mapping From Page Table To Main Memory

Step 2: To determine the actual page number of the process, CPU stores the page table base in a special register. Each time the address is generated, the value of the page table base is added to the page number to get the actual location of the page entry in the table. This process is called scaling.

Parul® University

DIGITAL LEARNING CONTENT

Mapping From Page Table To Main Memory

Step 3:Generation of physical Address.

Step 4: Getting Actual Frame Number.

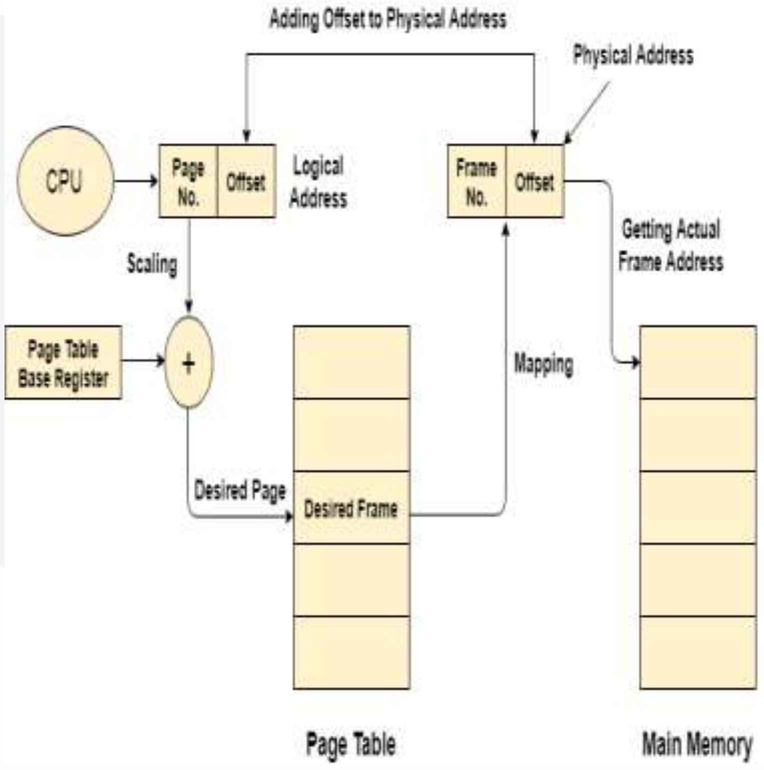


Figure – 20 Mapping From Page Table to Main Memory [10]

Page Allocation

- Paged allocation in memory management divides computer physical memory (Random Access Memory – RAM) into fixed size of memory units called page frames (memory block size).

- Also divides virtual address space to size of the pages.

Parul® University

DIGITAL LEARNING CONTENT

Page Allocation

- The main advantage of using paged allocation is, an empty page frame can be used by any job however, need to keep track of number of pages and where the pages of individual job are located in memory.

- In other words memory can be used efficiently and each job runs on its own address space. Frames, pages and the mapping between the two is shown in the .

Example of Page Allocation

- From figure, the size of each page is 100 memory locations.

- The memory utilized by the process1 in the given figure is 350 memory locations where it takes 4 pages to store 350 memory locations.

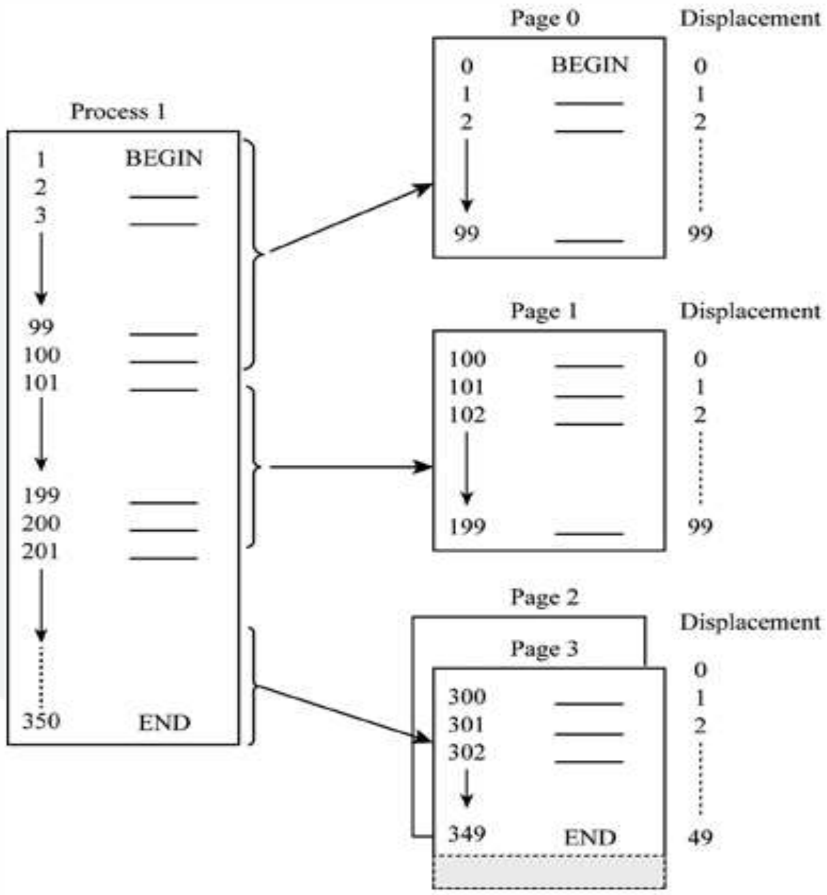


Figure - 21 Example of Page Allocation [12]

Example of Page Allocation

- That is, in memory block 0 or page 0 holds first 100 memory locations of data.

- In second page, memory locations of 100 to 199 are stored and so on.

Steps to determine exact location of the instruction or code line

- Find the page number and displacement.
- Determine memory block size which contains required page.
- Obtain the address of beginning of the first memory block.

Steps to determine exact location of the instruction or code line

- Add the displacement to starting address of memory blocks if the required page is in other than page 0.

- When the system moves required pages from main memory to secondary memory it gives rise to demand paging, in other words, moving only the required page to physical memory.

Hardware Support

- Each OS has its own method for storing page tables.
- Page table for each process is there.
- Pointer to the page table is stored with other register values in PCB.
- The h/w implementation of page table can be done in several ways.

Hardware Support

1) The page table is implemented as a set of dedicated registers.

- The main use of registers for the page table is satisfactory if the page table is reasonably small.

- But current systems have Page table size that is very large, so use of only registers is not feasible.

Hardware Support

2) Using Translation look-aside buffer (TLB).

- The problem of previous approach is the time required to access a memory, with this scheme 2 memory access required like, a P.T. is kept in M.M., and a page- table based register (PTBR) points to the page table.

- The solution of this problem is to use a special, small, fast-lookup hardware, known as translation look-aside buffer (TLB).

TLB (Translation Look-Aside Buffer)

- It is associative, high speed memory.

- **Each entry in TLB consists of 2 parts:**
  1) **a key (tag)**
  2) **a value.**

TLB (Translation Look-Aside Buffer)

- TLB used with P.T. in following way:
  - It contains a few of the page-table entries.
  - When a logical address is generated by the CPU, its page no. is presented to the TLB.
  - **If the page no. is found in TLB**, its frame no. is immediately available and is used to access memory.
  - **If the page no. is not in TLB** (known as TLB miss), a memory reference to the page table must be made.
  - When the frame no. is found then we can use it to access memory.

TLB (Translation Look-Aside Buffer)

Example:-

- A CPU generates 32-bit virtual addresses and the page size is 4 KB. The processor have translation look-aside buffer (TLB) which can hold a total of 128 page table entries and it is 4-way set associative.

- Now calculate the minimum size of the TLB tag.

TLB (Translation Look-Aside Buffer)

Answer:-

Here, size of a page = 4KB = 2^12

Total number of bits required to address a page frame = 32 – 12 = 20

If there are ‘n’ cache lines in a set, the cache placement is called n-way set associative.

TLB is 4 way set associative and can hold total 128 (2^7) page table entries, number of sets in cache = 2^7/4 = 2^5.

So 5 bits are needed to address a set, and 15 (20 – 5) bits are needed for tag.

Paging Hardware with TLB

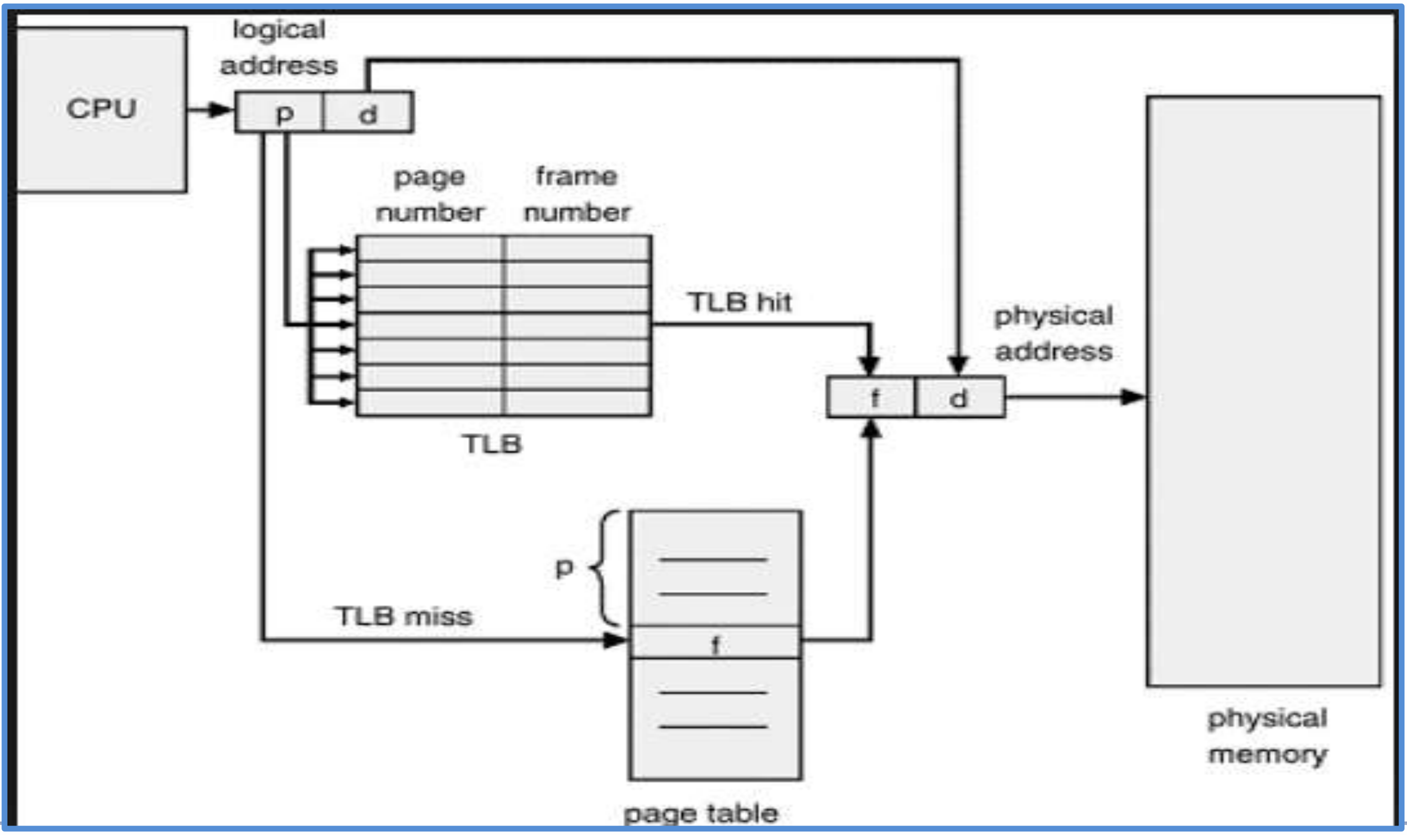


Figure – 22 Paging hardware with TLB [15]

Paging Hardware with TLB

- Now add that page no. and frame no. into TLB so that next time it would be helpful to access a physical memory.
- If the TLB is full of entries, some of them can be removed. This decision has been taken by using LRU algorithm.
- Some TLBs allowed entries to be wired down that means it can’t be removed.
- Some TLBs store address-space identifiers (ASIDs) in each TLB entry.

Paging Hardware with TLB

- ASID –uniquely identify each process and used to provide address space protection to corresponding process.
- It ensures that the ASID for the currently running process must matches with the virtual page that it is now resolving.
- If it doesn't match , it is known as TLB miss.
- An ASIDs allow TLB to contains different process simultaneously.
- If the TLB does not support separate ASIDs, then every time a different page table is selected, and the previous TLB must be erased.

Paging Hardware with TLB

- The percentage of times that a page is found in TLB is known as hit ratio.

For Example:-

- 80 % hit ratio means we find desired page in 80 percent of time.
- It takes 20 nanosecond to search the TLB and 100 nanoseconds to access memory. So mapping requires total 120 nanoseconds total.
- If its not in TLB then, 20 ns for TLB search, 100 ns for memory access for page table, then 100 ns for mapping, so total 220.
- Effective Access time = (0.80 *120) + (0.20 * 220)

= 140 ns

Paging Hardware with TLB

Example:- 2

- If hit ratio increase as 98 %
- Effective Access time = (0.98 *120) + (0.02 * 220)

  = 122 ns

Advantages of Paging

- Memory allocation and de-allocation is very fast.
- It is noncontiguous memory allocation so we do not require contiguous memory space for one process.
- No external fragmentation problem.
- Swap-in and Swap-out operations are performed fast.

Disadvantages of Paging

- This technique is suffering with internal fragmentation.

- **For Example:-**

  If pages are 120 bytes, a process of 400 bytes requires 3 pages + for 40 bytes it requires again 1 page. So total 4 pages will be allocated.

  (120-40) = 80 bytes size will be free and its internal fragmentation.

- It requires more size of memory to store page table. As the page number increases the page table size is increase.

Memory Protection

- Memory protection in paging must be required. That is frame allocated to the one page cannot be reallocated to other page meanwhile one page is allocated to it.

- This memory protection can be performed by using one special bit associated with each frame, and stored in page table, that is Protection Bit.

- Page table has index starting from 0 to maximum allowed page in memory.

Memory Protection

- But we are not using all pages at a time so to find out which page is used and write now present in memory and which is not we are using one special bit that is known as **Valid/Invalid bit** as shown in fig.

- Instead of using valid/invalid bit to determine the use of the page, some system uses **Page Table Length Register (PTLR)** which indicates the actual size of the page table and from it we can know the last valid address of page

Memory Protection

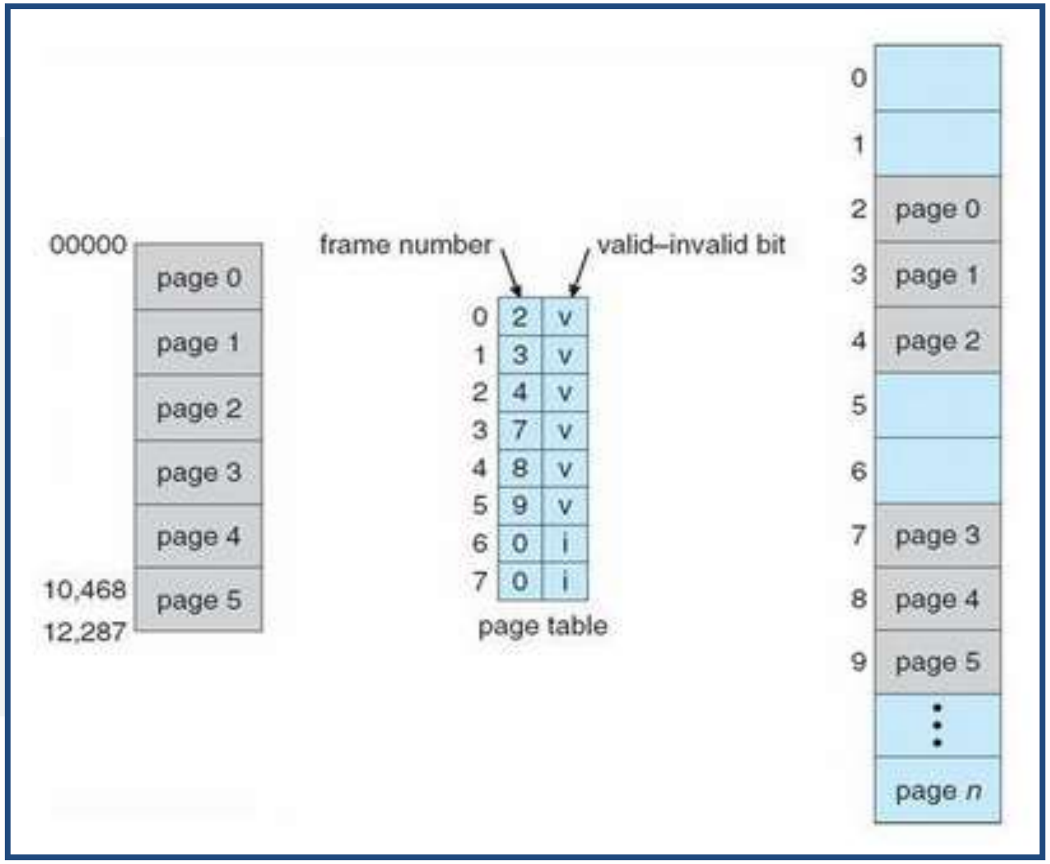


Figure – 23 Valid(v) or invalid(i) bit in a page table [15]

Parul® University

DIGITAL LEARNING CONTENT

Segmentation

- Segmentation is a memory-management scheme that supports user view of memory and contains segment.

- In fig. we can see that memory is divided into multiple segments like, subroutine, stack, code, heap, etc.

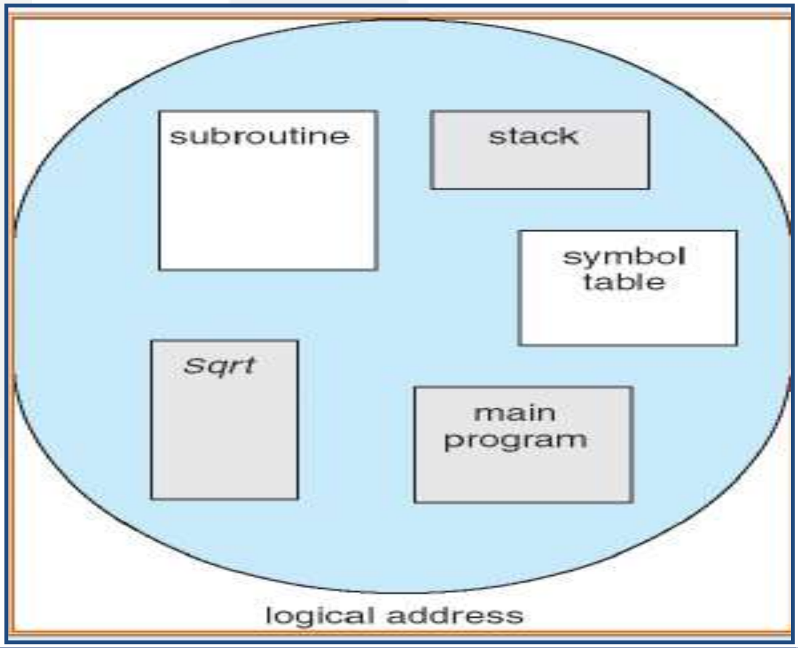


Figure – 24 Memory segmentation [15]

Segmentation

- Users or programmers specify each address by 2 quantities:

  1) a segment name

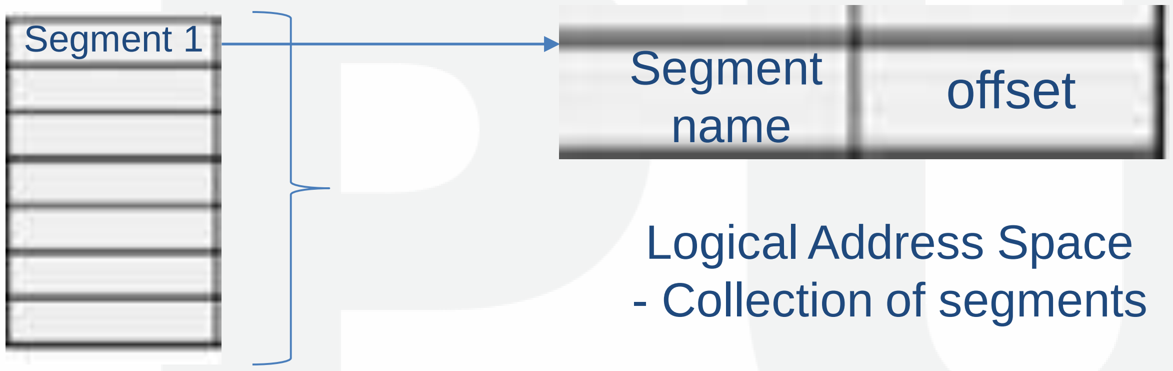
  2) an offset. (contrast to the paging)

- For easiest way of implementation, segments are numbered and referred by segment no. rather than name.

Segmentation

- So logical address consists of 2 tuple:

<segment-number, offset>



- Here, Segment-number is used to identify a segment.offset is an original location within a segment.

Parul® University

DIGITAL LEARNING CONTENT

Segmentation

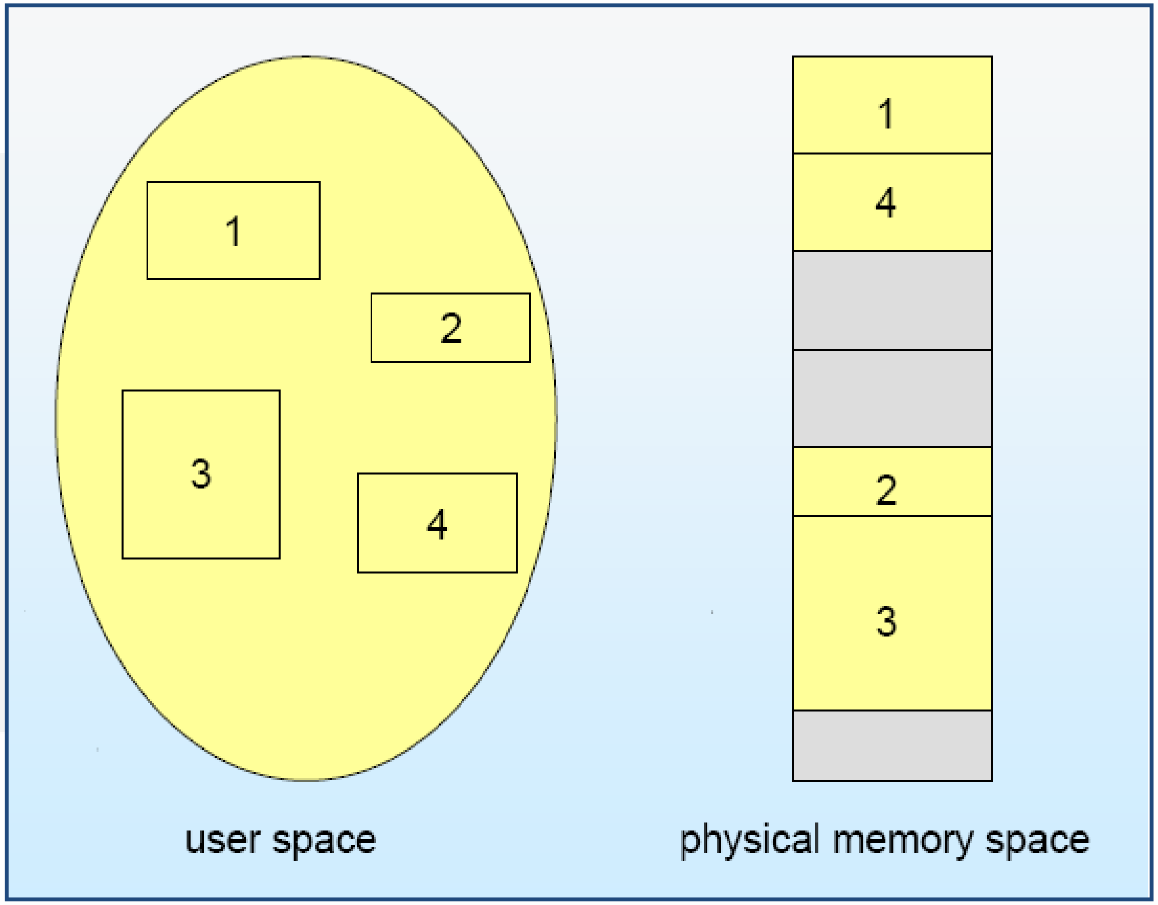
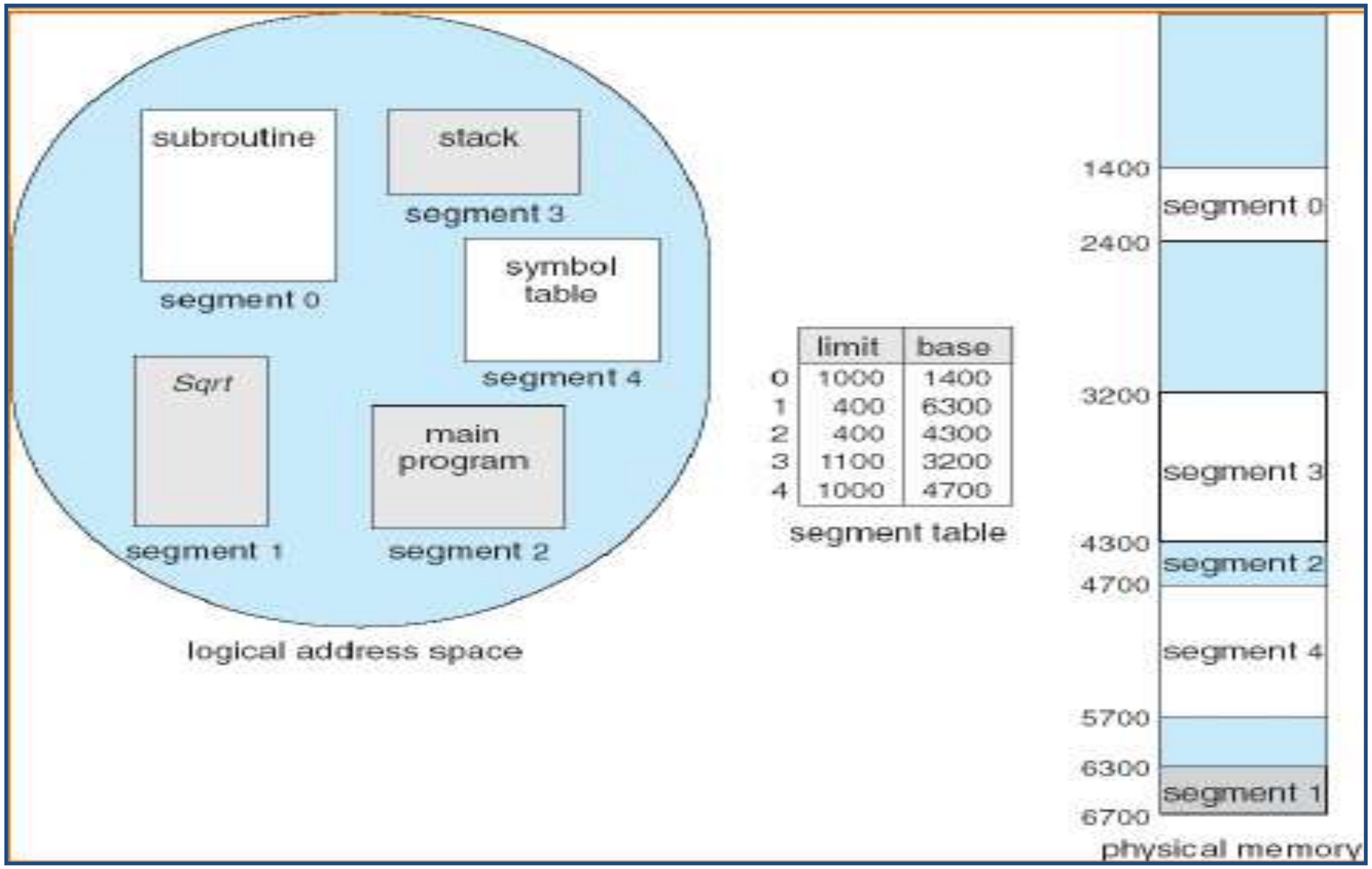


Figure – 25 Memory Segmentation [15]

Example of Segmentation



	limit	base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

Figure – 26 Memory Segmentation Example [15]

Hardware Implementation of Segmentation

- Logical address is 2D
- Physical address is 1D
- Thus, we must define an implementation that maps 2d into 1d.
- This mapping is overdone by a Segment Table.
- Each entry of the segment table has Base and Limit. The segment base contains the starting address of the segment where as limit specifies the length of the segment.

Hardware Implementation of Segmentation

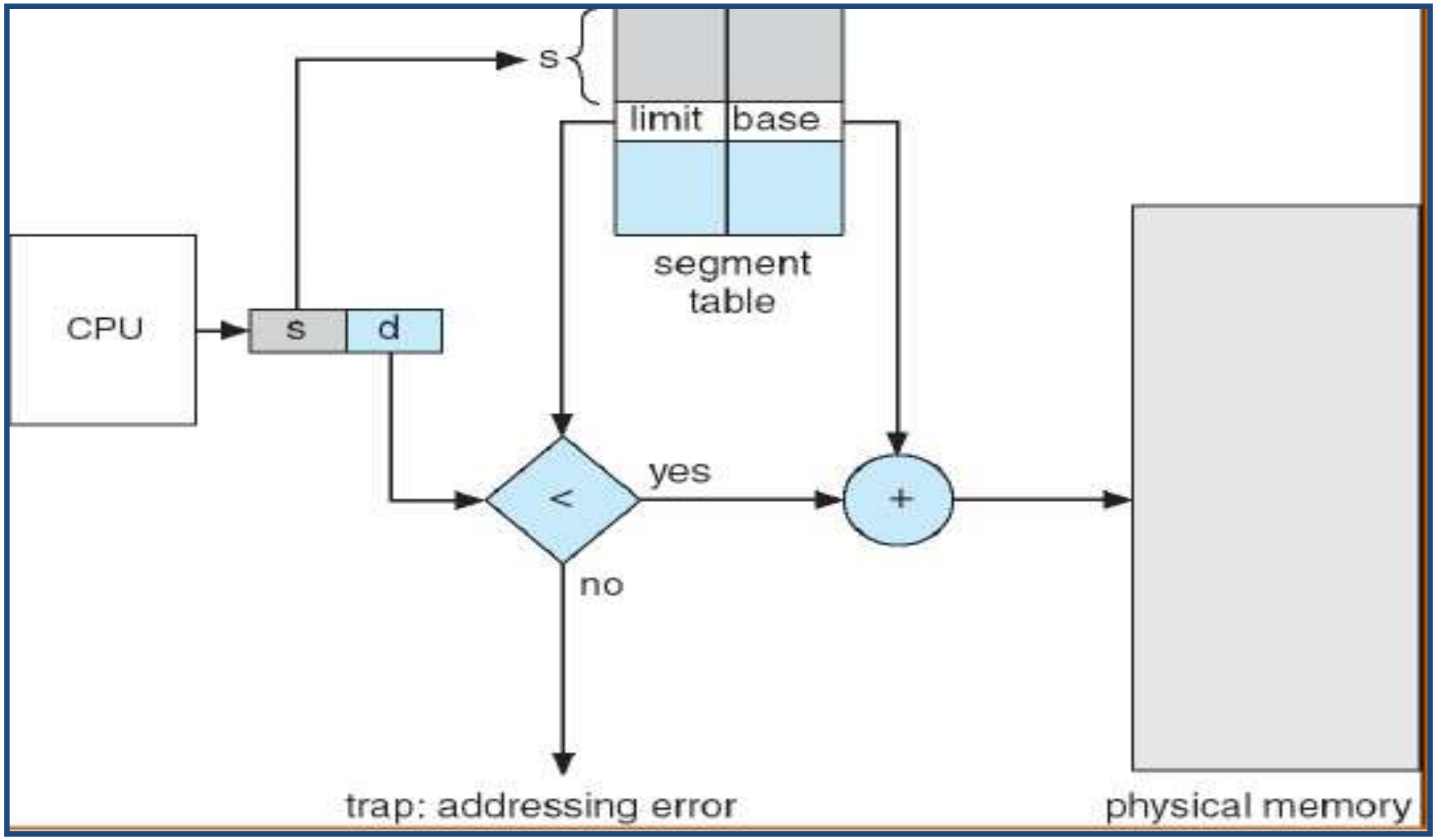


Figure – 27 Address generated by the CPU in segmentation [15]

Hardware Implementation of Segmentation

- A logical address consists of 2 parts: s & d.
- Here, S is used as an index into segment table
  - a value of D must be lies between 0 and segment limit.
  - If its not, it will be trapped to OS.

For Example,

- Segment 2 is 400 bytes long. Begins at 4300.
- It will check 53 < 0 – limit (i.e. 400)
- So that, a reference to byte 53 of segment 2 is mapped as
  4300 + 53 = 4353.
- now check for 1222 byte for segment 0 .
- 1222 < 0 – 1000 , which is false so it will be trapped to OS.

Difference between Segmentation and Paging

	Segmentation	Paging
1	Program is divided into variable size segments.	Program is divided into fix size of pages.
2	Segmentation is slower than paging.	Paging is faster than segmentation.
3	Segmentation is visible to the user.	Paging is invisible to the user.
4	Segmentation eliminates internal fragmentation.	Paging suffers from internal fragmentation.
5	Segmentation suffers from external fragmentation.	There is no external fragmentation.
6	OS maintain a list of free holes in main memory.	OS must maintain a free frame list

Advantages of Segmentation

- Segmentation is useful for memory management.
- Using segmentation users can partition their programs into number of subtask i.e. modules which are operate independently from each another.
- Segments increases data sharing between two processes.
- Segmentation allows to extend the address ability of a processor
- Segmentation provides facility to separate the memory areas for heap, code, stack, data, etc.

Disadvantages of Segmentation

- In segmentation external fragmentation is mostly present.
- Costly algorithm.
- Segmentation finds free memory area big enough.
- When we use paging it generates list of free pages.
- Implementation is very complex.

Structure of Page Table

It can be classified as:

1) **Hierarchical Paging (Multilevel Page Table).**
2) **Inverted Page Table.**
3) **Hashed Page Table.**

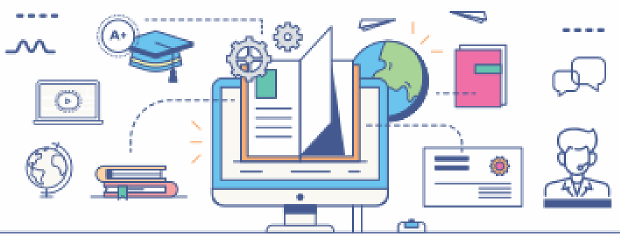
1) Hierarchical Paging (Multilevel Page Table)

- **Multilevel Paging** is a paging scheme which contains two or more than two levels of page tables in a hierarchical manner. It is also known as hierarchical paging.
- The entries of the level 1 page table are points to a level 2 page table, same way entries of the level 2 page tables are pointers to a level 3 page table and so on.
- The entries of the last level P.T. are stores actual frame information.
- Level 1 contain single page table and the address of that table is stored in to the **PTBR (Page Table Base Register).**

Two – Level Paging Example

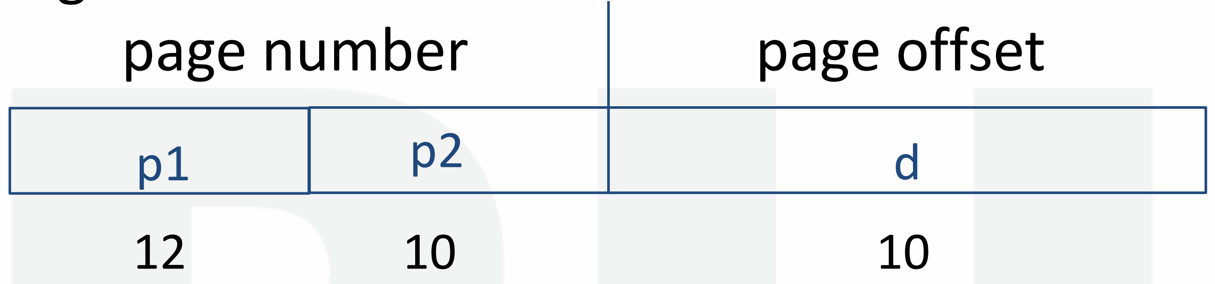
- We can divide a logical address (on 32-bit machine with 1K page size) is divided into following:
  - a page number consisting of 22 bits
  - a page offset consisting of 10 bits

- So that the page table is paged, the page number is again divided into following:
  - a 12-bit page number
  - a 10-bit page offset (p2)



Two – Level Paging Example

- Thus, a logical address is as follows:



- where,
  - the outer page table use p1 as an index number
  - *p2* is the rearrangement within the page of the inner page table

Known as **forward-mapped page table**

Two – Level Page Table Scheme

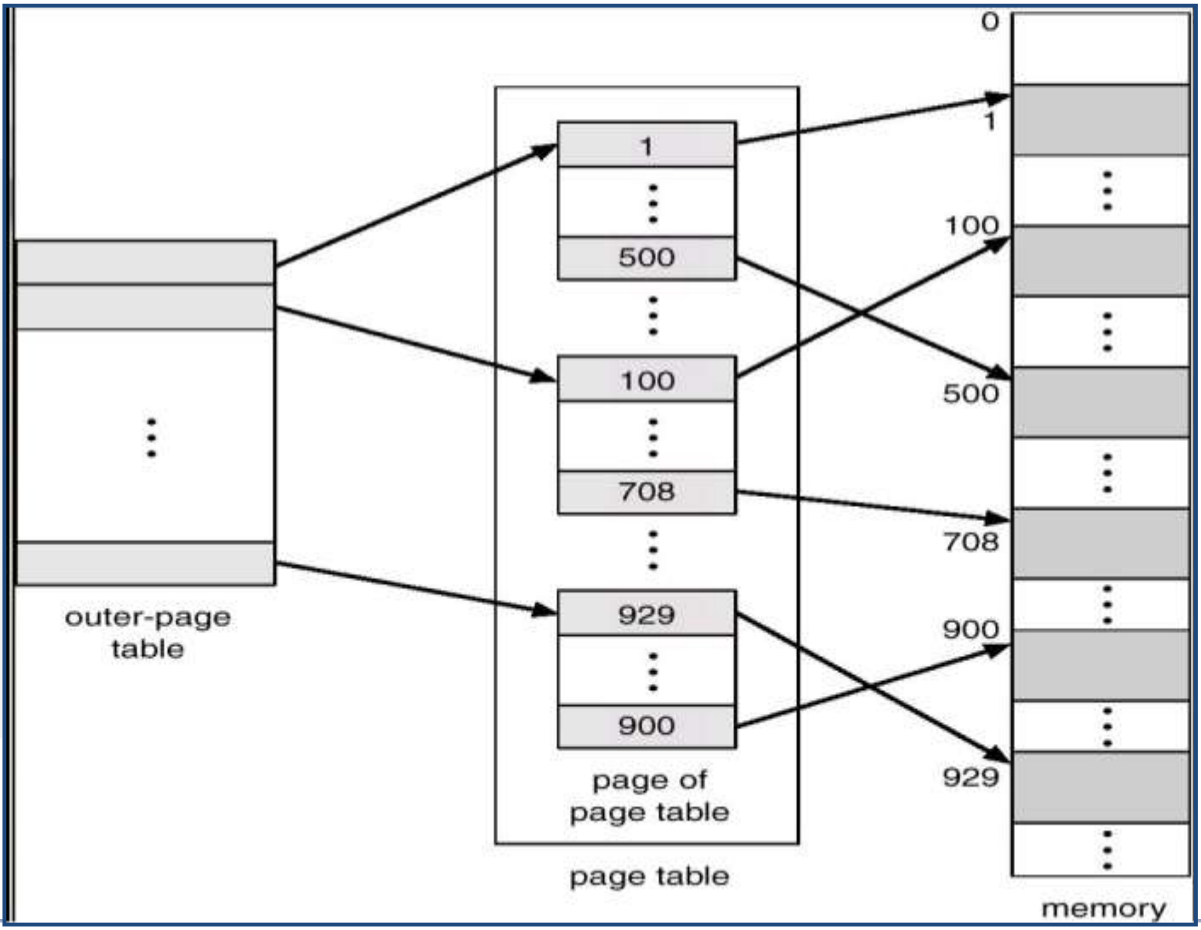


Figure – 28 Two level page table [15]

Address Translation Scheme

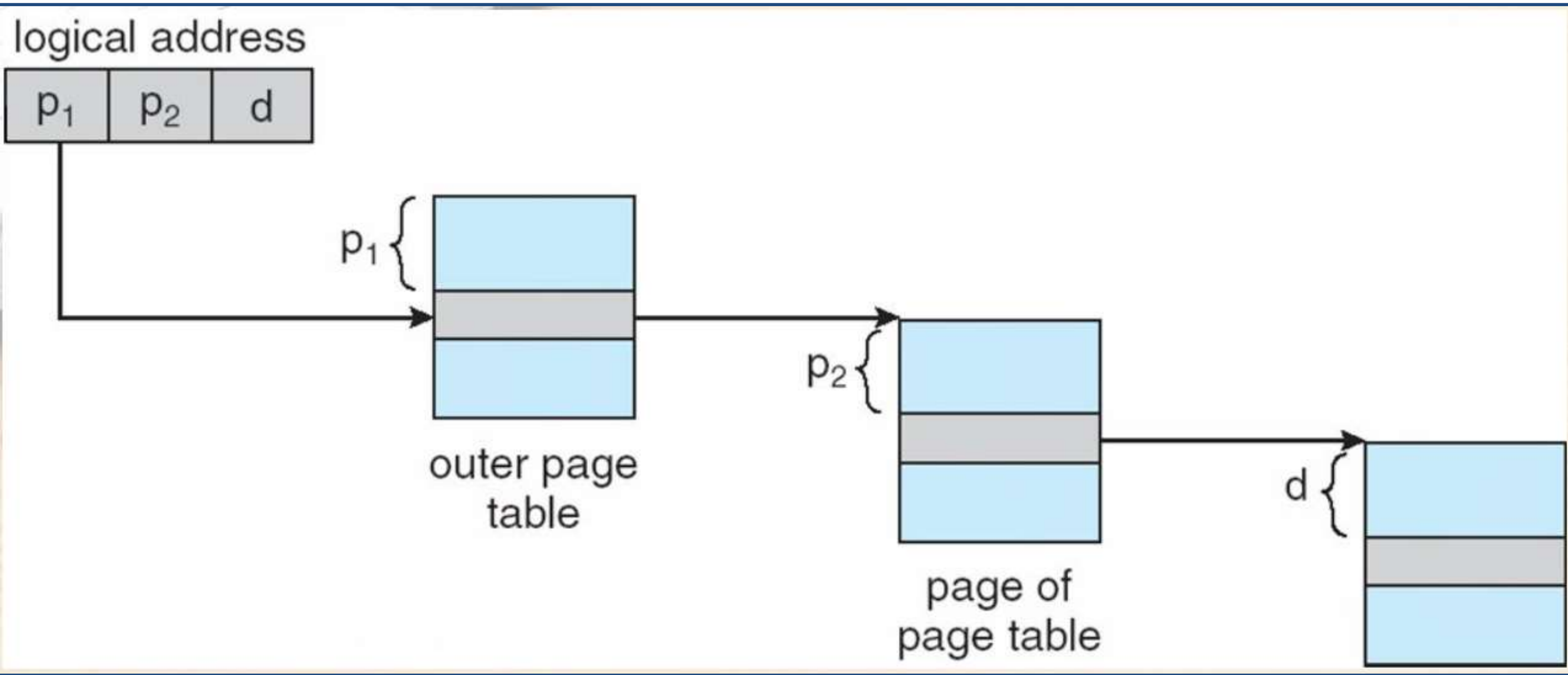


Figure – 29 Address translation scheme [15]

2) Hashed Page Table

- Common in address spaces > 32 bits
- The virtual page no. is hashed into a page table
  - **This page table stores a chain of elements hashing to the same location**
- Each element contains:
  (1) the virtual page number
  (2) the value of the mapped page frame
  (3) a pointer to the next element
- Virtual page numbers are compared with this chain searching for a match

  If a it is found, the related physical frame is extracted

2) Hashed Page Table

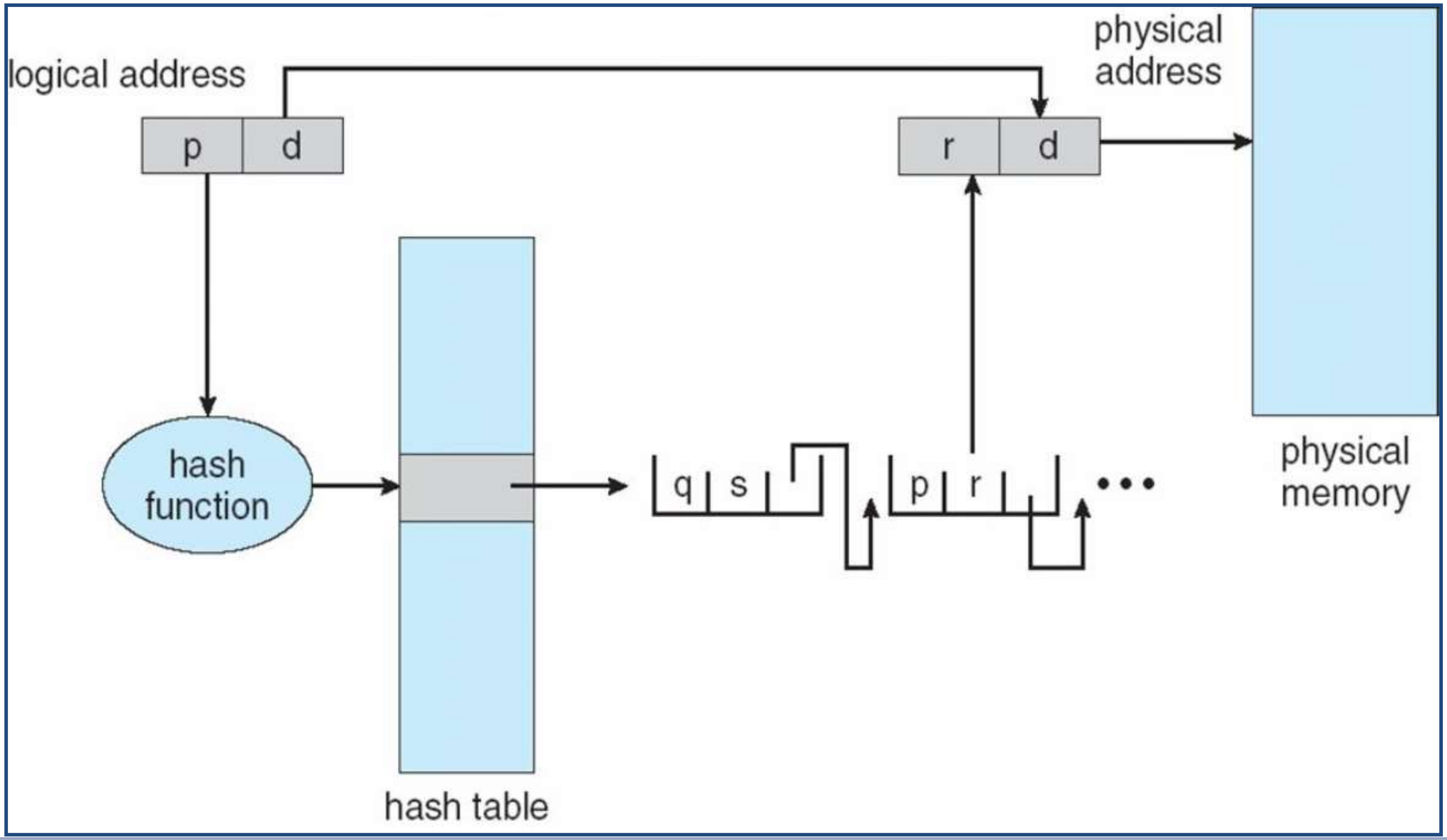


Figure – 30 Working of hashed page table [15]

Hashed Address Translation

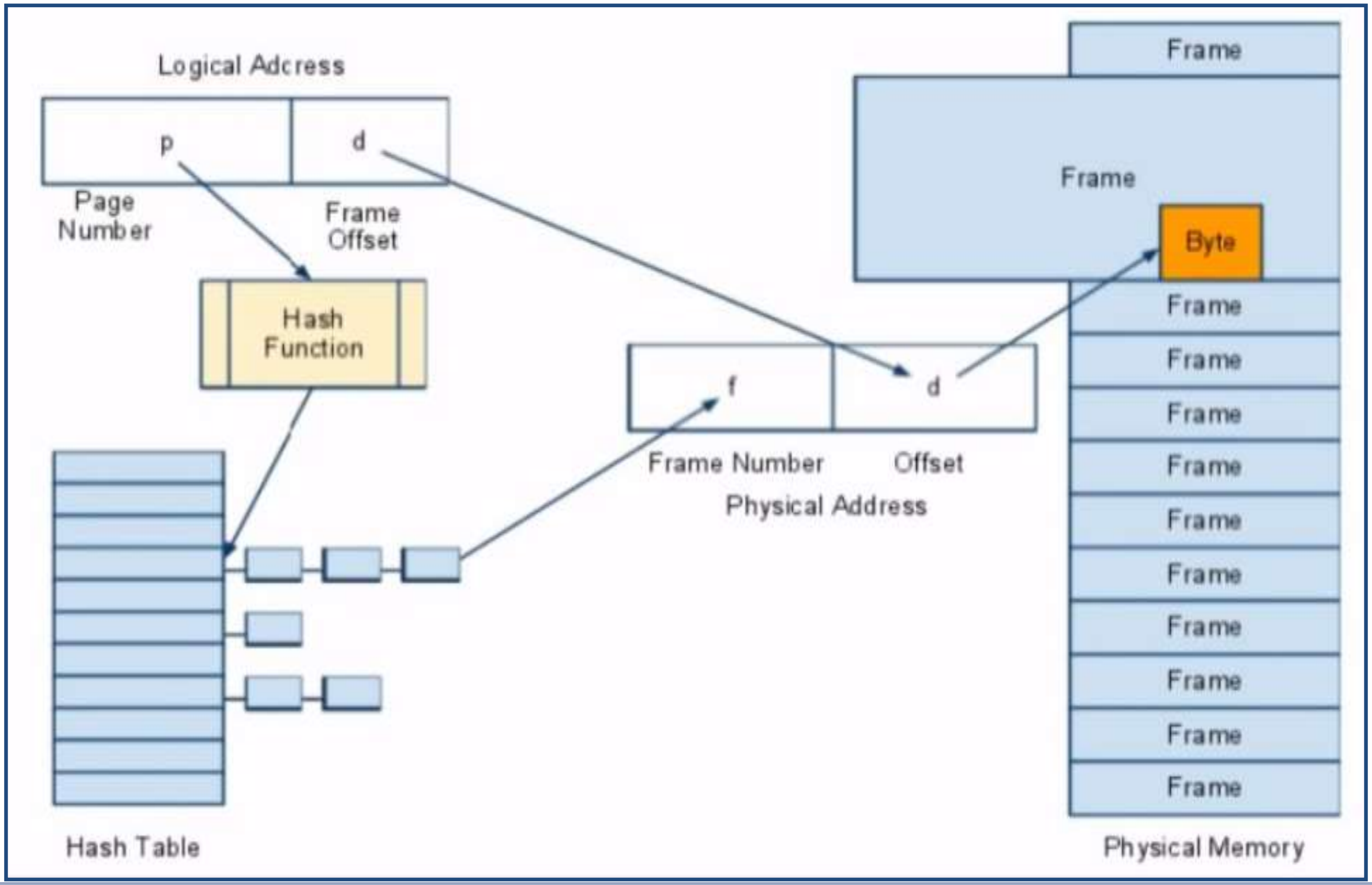


Figure – 31 Hashed address translation [15]

3) Inverted Page Table

- Preferably each process having a page table and keeping track of all possible logical pages, track all physical pages.
- Inverted page table keeps one entry for each real page (or frame) of memory.
- This entry contains the virtual address of the page stored in that real memory location, along with information about the process that owns that page.
- Table stores the corresponding value of pid and page no.
- Decreases memory needed to store each page table, but required more time to search the table when a page reference occurs

3) Inverted Page Table

- Requires the linear search of the entire table to perform the translation from page to frame. But it is inefficient in terms of performance.

- Use hash table to limit the search to one — or at most a few — page-table entries
  TLB can accelerate access

3) Inverted Page Table Architecture

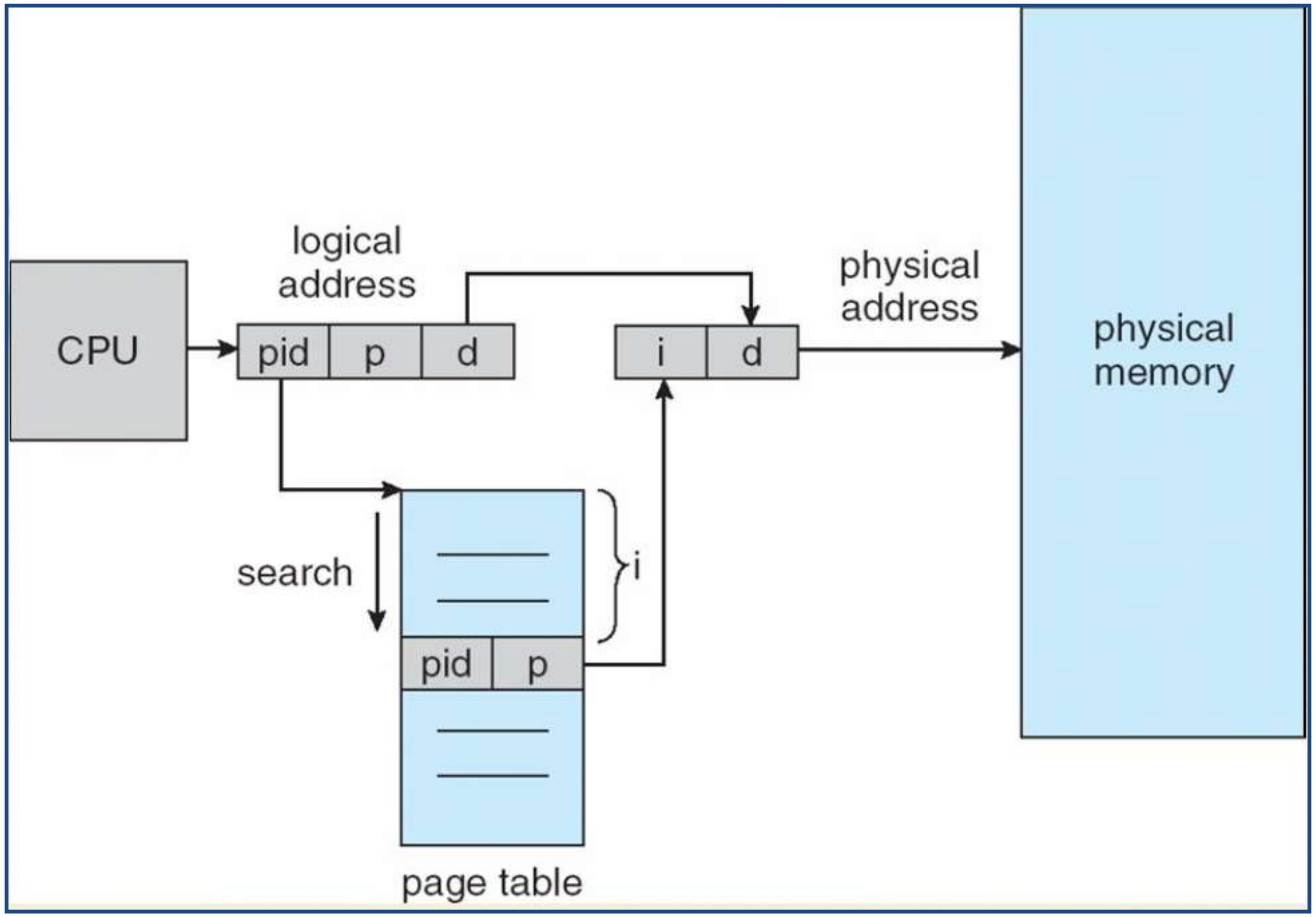


Figure – 32 Architecture of inverted page table [15]

References

[1]https://www.researchgate.net/publication/319529366_Emerging_NVM_A_Survey_on_Architectural_Integration_and_Research_Challenges/figures?lo=1

[2] https://www.techtud.com/short-notes/logical-vs-physical-address-space

[3] https://basicittopic.com/memory-management/

[4] https://professormerwyn.wordpress.com/tag/logical-address-space/

[5]https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/8_MainMemory.html

[6] https://www.inf.ed.ac.uk/teaching/courses/os/slides/09-memory16.pdf

References

[7] https://www.geeksforgeeks.org/implementing-non-contiguous-memory-management-techniques/

[8]Operating Systems: Internals and Design Principles, by William Stallings.

[9]Operating Systems: Design and Implementation, Textbook by Andrew S. Tanenbaum.

[10]Operating System Concepts by Galvin

[11]Modern Operating Systems, Andrew S. Tanenbaum

[12]Operating system concepts by Abraham Silberschatz.Peter B Galvin.Gerg

[13]OPERATING SYSTEMS MEMORY MANAGEMENT by Jerry Breecher

References

[14]Operating System Generations. Tutorialspoint.

https://www.tutorialspoint.com/operating-system-generations

[15]Image source   www.google.com

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in