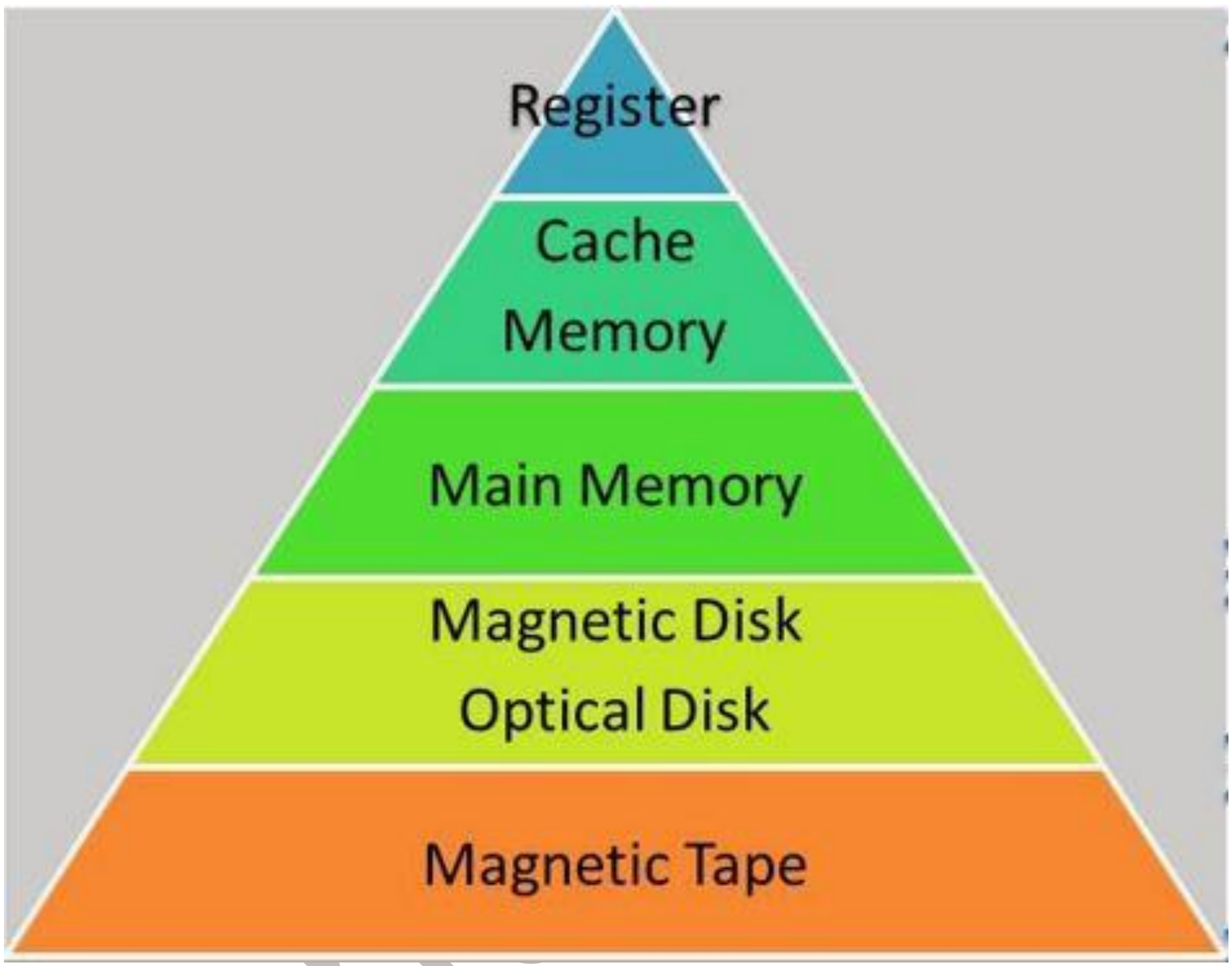
Basics of Memory management

Let's understand how basic architecture of the memory hierarchy works.

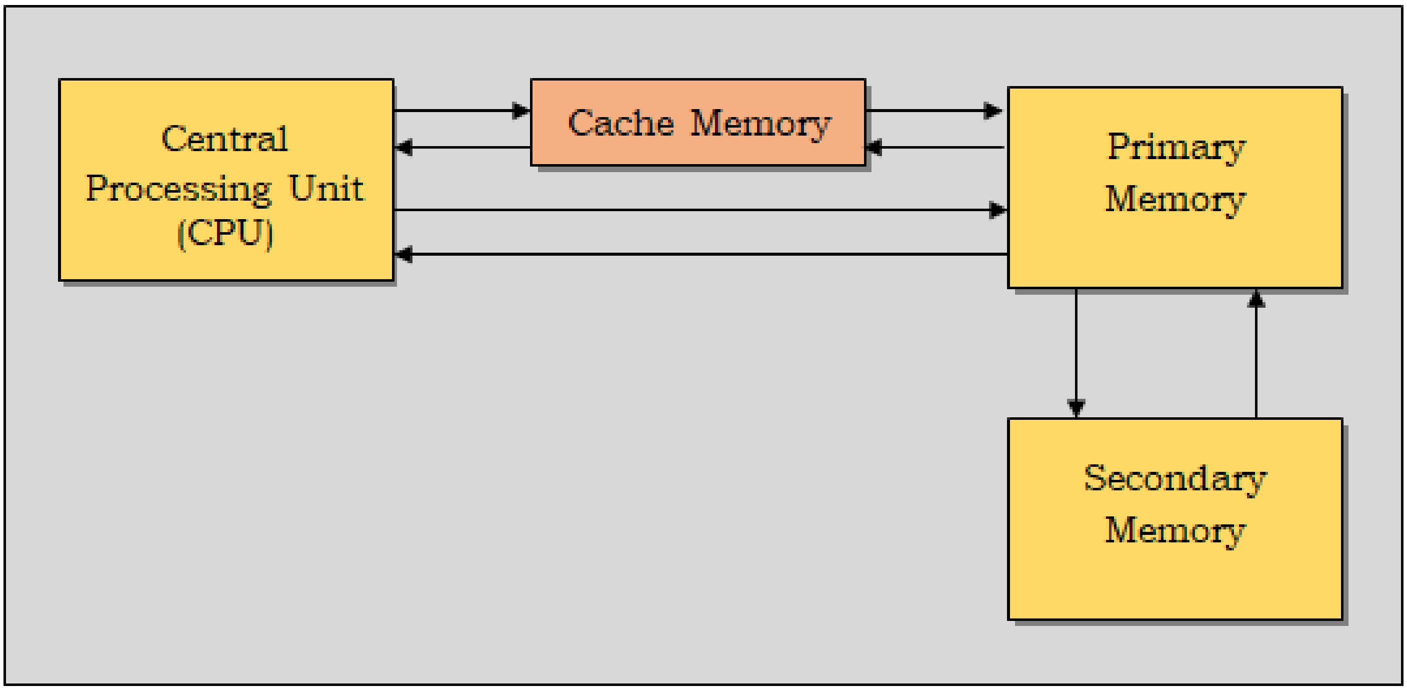


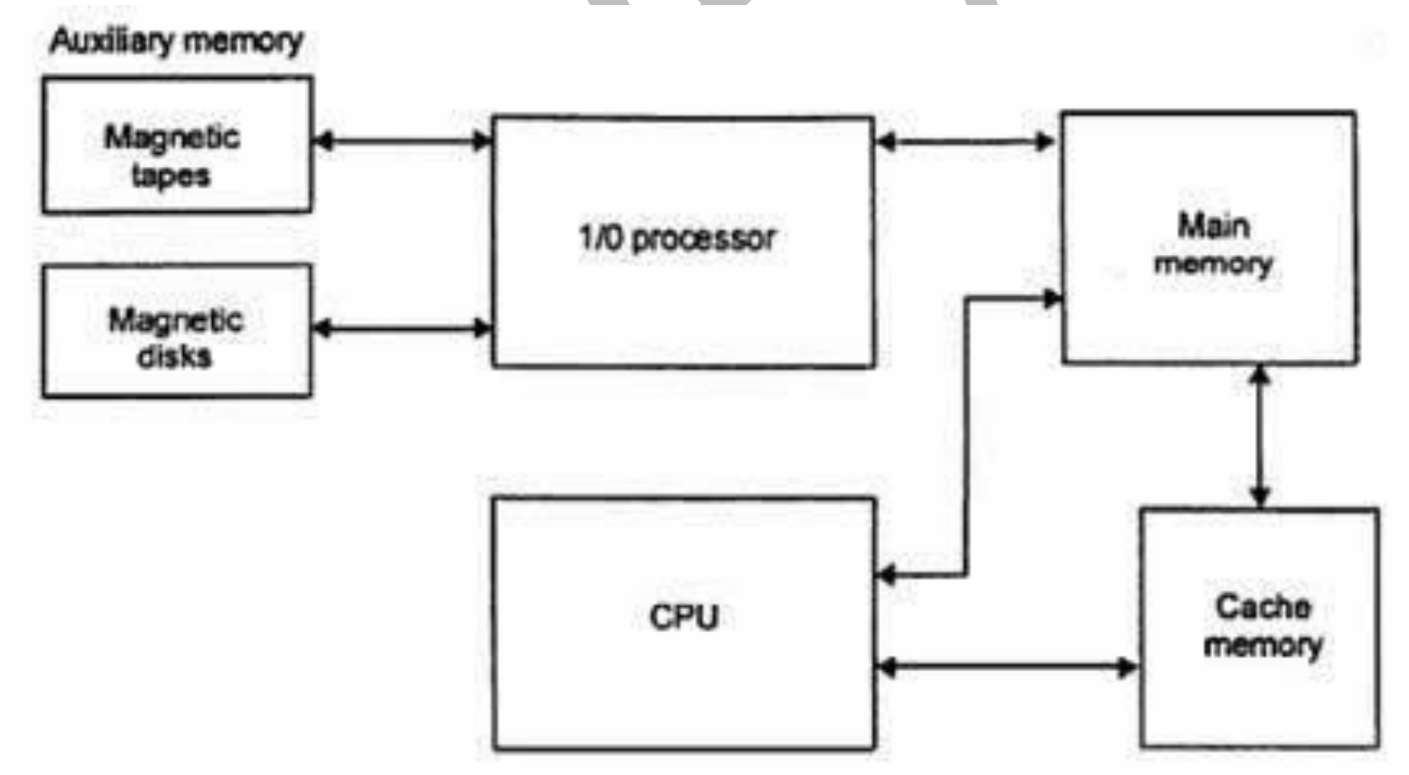
Reason to have memory hierarchy instead of a single memory is

- Average access time (Should be less)
- Capacity (Should be high)
- Cost Per unit (Should be less)

All these three criteria are not achievable with single memory, so we use a memory hierarchy to achieve these goals.

CPU generate a logical address which can be used to access secondary memory but practically CPU always access physical or main memory.





<u>Goal of Operating System:</u> -

- The main goal of memory management is efficient memory utilization.
- Protection of Memory Space protect the operating system from access by user processes and, in addition, to protect user processes from one another. We need to make sure that each process has a separate memory space.

<u>Duty of Operating System: -</u>Operating system is responsible for the following activities in connection with memory management:

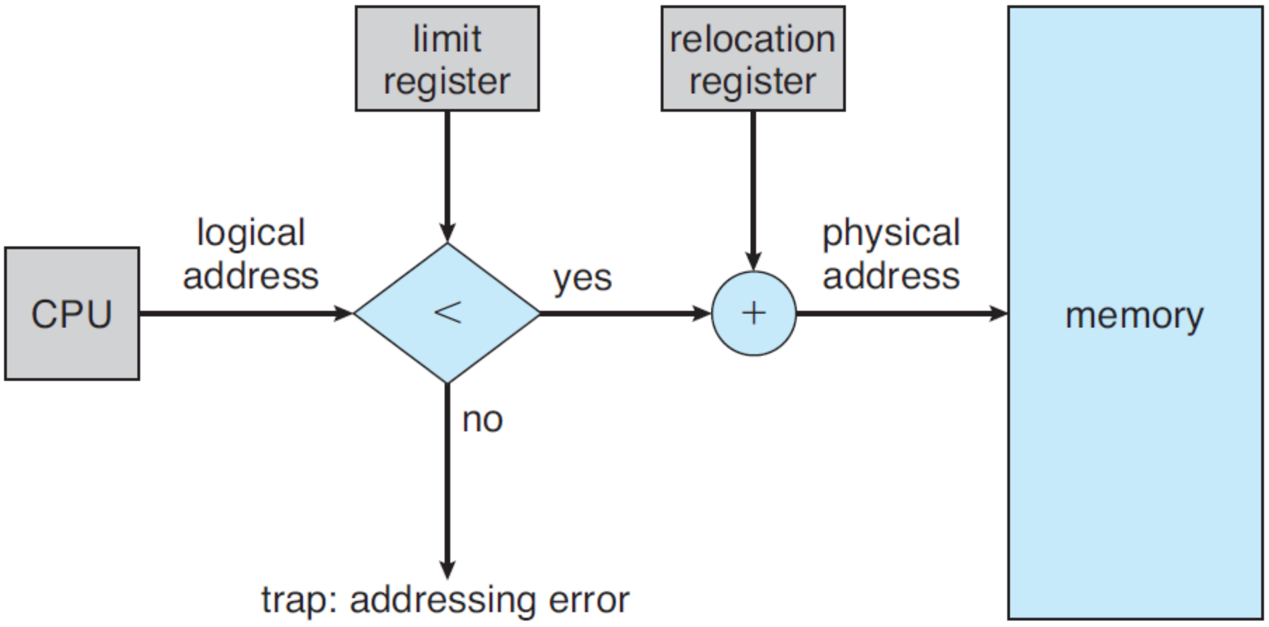
- Address translation from Logical address (Secondary Memory) to Physical address
- (Main Memory).
- Deciding which processes (or parts of processes) and data to move into and out of main memory.
- Allocating and deallocating memory space as needed.
- Keeping track of which parts of memory are currently being used and who is using them.

- There can be two approaches managing main memory in general.
    - **Contiguous allocation policy:** - Where process is loaded entirely in main memory and must be stored in a contiguous fashion without partition.
    - **Non-contiguous allocation policy:** - here process can be partitioned and each partition can be stored into different locations and all partitions are not required to be stored into main memory.

Contiguous allocation policy (Address translation)

- We know that when a process is required to be executed it must be loaded to main memory, by policy has two implications.
  - It must be loaded to main memory completely for execution.
  - Must be stored in main memory in contiguous fashion.
- Here we use a memory management unit whose duty is to take logical address and translate it into physical address.
- MMU contains a relocation register, which contains the base address of the process in the main memory and it is added in the logical address every time.

Solution: -



- In order to check whether address generated to CPU is valid (within range) or invalid, we compare it with the value of limit register, which specifies the size of the range. So, if the value of logical address is less than limit, then it means it's a valid request and we can continue with translation otherwise, if it is an illegal request which is immediately trapped by OS.
- Protection of memory space is accomplished by having the CPU hardware compare *every* address generated in user mode with the registers.
- This scheme prevents a user program from (accidentally or deliberately) modifying the code or data structures of either the operating system or other users.
- The base and limit registers can be loaded only by the operating system.

Q Consider the following tables of relocation and limit registers and find the illegal attempts and if valid find the physical address?

	Limit Register	Relocation Register
P_0	500	1200
P_1	275	550
P_2	212	880
P_3	420	1400
P_4	118	200

Following are the request by process
P_0—450
P_1—300
P_2—210
P_3—450
P_4---80

Q What is the most appropriate function of Memory Management Unit (MMU)? **(NET-JUNE-2015)**
a) It is an associative memory to store TLB
b) It is a technique of supporting multiprogramming by creating dynamic partitions
c) It is a chip to map virtual address to physical address
d) It is an algorithm to allocate and deallocate main memory to a process
Answer: (c)

Q Function of memory management unit is: **(NET-DEC-2015)**
(A) Address translation
(B) Memory allocation
(C) Cache management
(D) All of the above
Answer: (A)

- **Advantage**: -
    - easy strategy, simple to use and understand.
    - Translation is very fast, only single main memory access is required.
- **Disadvantage**: -
    - This translation scheme cannot be used in non- contiguous policy

Contiguous Memory Allocation (Space management)

- There are two idea of managing space in contiguous memory allocation scheme:
    - Fixed Partition Scheme
    - Variable Partition Scheme
- In general, Contiguous memory allocation suffers from External Fragmentation, means that the memory is available to be allocated to a process but because of contiguous allocation constraints (memory not available in contiguous fashion) it cannot be allocated to the process.
- **i.e.** when there is enough total memory space to satisfy a request but the available spaces are not contiguous; storage is fragmented into a large number of small holes.

Fixed Partition Scheme

- We divide memory into partitions, the number and size of partitions in which a memory is divided is fixed but their size of partitions may differ
- A partition can hold only one process at a time, and no process is allowed to have more than one partitions. (1-to-1 map between process and partition).
- The degree of multiprogramming is restricted by the number of partitions in the memory.
- The size of the process is limited by the size of largest partition in the system.
- A process cannot be stored if the size of the process id greater than the size of the partition.
- if a process request for some space, and the size of partition is more than the size of process, then the remaining space will be waisted internally (internal Fragmentation).
- Advantage: -
    - Management and book keeping for OS is relatively easy
    - External Fragmentation (space requested by the process is available but cannot be allocation as it is not contiguous) will be relatively less
- Disadvantage: -
    - Suffers from internal fragmentation as we have to allocate the entire partition to the process even if required is less than the partition size. Unused memory that is internal to the partition.
    - i.e. Internal fragmentation is a function of fixed size partition which means, when a partition is allocated to a process. Which is either the same size or larger than the request then, the unused space by the process in the partition Is called as internal fragmentation

Variable Size Partitioning

- Here Initially, all memory is available for user processes and is considered one large block of available memory called a **hole.**
- In starting, we treat the memory as a whole or a single chunk & whenever a process request for some space, exactly same space is allocated if possible and the remaining space can be reused again.
- As processes are loaded and removed from memory, the free memory space is broken into little pieces. In this policy, no process is allowed to have more than one partitions, even if they are available in contiguous fashion. (1-to-1 map between process and partition).

Space Allocation

- There are in general three algorithms used to:
  - **First Fit Algorithm:** - As the name implies, searching the memory from the base or binging and will allocate first partition which is capable enough. We can stop searching as soon as we find a free hole that is large enough.
    - **Advantage:** - Simple, easy to use and understand fast in starting
    - **Disadvantage**: -Poor performance, both in terms of time and space

- **Best Fit Algorithm:** - Here, we search the entire memory and will allocate the smallest partition which is capable enough.
  - **Advantage**: - Perform best in fix size partitioning scheme. This strategy produces the smallest leftover hole (less internal fragmentation).
  - **Disadvantage**: - Relatively slow, perform worst in variable size partitioning as the remaining spaces will be very small in size and it is very less probable that they are used again.

- **Worst Fit Algorithm:** - We must search the entire list, and allocate the largest partition possible.
  - **Advantage:** - Perform best in variable size partitioning, produces the largest leftover hole, which may be more useful than the smaller leftover hole from a best-fit approach.
  - **Disadvantage:** - Perform worst in fix size partitioning, resulting into large internal fragmentation.

  - **Next fit policy:** - It is the modification in the best fit where, after satisfying a request, we start satisfying next request from the current position.

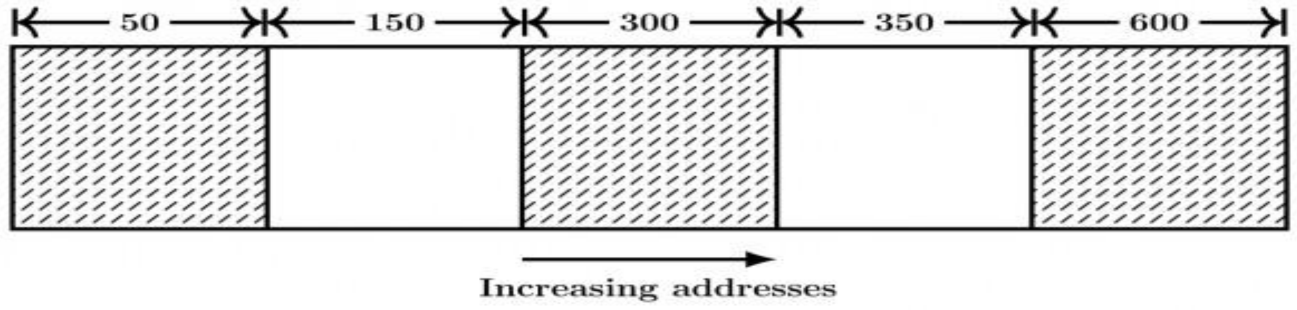
Q Consider six memory partitions of size 200 KB, 400 KB, 600 KB, 500 KB, 300 KB, and 250 KB, where KB refers to kilobyte. These partitions need to be allotted to four processes of sizes 357 KB, 210 KB, 468 KB and 491 KB in that order. If the best fit algorithm is used, which partitions are NOT allotted to any process? **(GATE-2015) (2 Marks)**

(A) 200 KB and 300 KB **(B)** 200 KB and 250 KB

(C) 250 KB and 300 KB **(D)** 300 KB and 400 KB

Answer: (A)

Q Consider the following heap (figure) in which blank regions are not in use and hatched region are in use. **(GATE-1994) (2 Marks)**



The sequence of requests for blocks of size 300, 25, 125, 50 can be satisfied if we use

(a) either first fit or best fit policy (any one **(b)** first fit but not best fit policy

(c) best fit but first fit policy **(d)** None of the above

Answer: (B)

Q Given memory partitions of 100 K, 500 K, 200 K, 300 K and 600 K (in order) and processes of 212 K, 417 K, 112 K, and 426 K (in order), using the first-fit algorithm, in which partition would the process requiring 426 K be placed? **(NET-DEC-2012)**

(A) 500 K **(B)** 200 K **(C)** 300 K **(D)** 600 K

Answer: (all options are wrong)

Q A program is located in the smallest available hole in the memory is ……………**(NET-DEC-2009)**

(A) best – fit **(B)** first – bit **(C)** worst – fit **(D)** buddy

Answer: (A)

Q A 1000 Kbyte memory is managed using variable partitions but to compaction. It currently has two partitions of sizes 200 Kbytes and 260 Kbytes respectively. The smallest allocation request in Kbytes that could be denied is for? **(GATE-1996) (1 Marks)**

(a) 151 **(b)** 181 **(c)** 231 **(d)** 541

Answer: (B)

In general, it can be understood that external fragmentation is more severe issue than internal fragmentation

Solutions to External Fragmentation

There are two solutions that are proposed to deal with external fragmentation.

Compaction

- Solution to external fragmentation is Compaction. We can also swap processes in the main memory after fixed intervals of time & they can be swapped in one part of the memory and the other part become empty. **The goal is to shuffle the memory contents so as to place all free memory together in one large block.** This solution is very costly in respect to time as it will take a lot of time to swap process when system is in running state.
- Either we should go for non-contiguous allocation, which means process can be divided into parts and different parts can be allocated in different areas.

Q Variable partition memory management technique with compaction results in: **(NET-JUNE-2009)**
(A) Reduction of fragmentation **(B)** Minimal wastage
(C) Segment sharing **(D)** None of the above
Answer: (A)

Q Which of the following statements are true? **(NET-JULY-2018)**
(a) External Fragmentation exists when there is enough total memory space to satisfy a request but the available space is contiguous.
(b) Memory Fragmentation can be internal as well as external.
(c) One solution to external Fragmentation is compaction.
Code:
(a) (a) and (b) only **(b)** (a) and (c) only
(c) (b) and (c) only **(d)** (a), (b) and (c)
Answer: (c)

Q In which of the following storage replacement strategies, is a program placed in the largest available hole in the memory? **(NET-DEC-2004)**
(A) Best fit **(B)** First fit **(C)** Worst fit **(D)** Buddy
Answer: (C)

Non-contiguous allocation (Paging)

- **Logical Address:** An address generated by the CPU is commonly referred to as a **Logical Address/virtual address.** The set of all logical addresses generated by a program is a **logical address space**
- **Physical Address: the address used to refer any location in main memory.** The set of all physical addresses corresponding to these logical addresses is a **physical address space**.
- The logical addresses must be mapped to physical addresses before they are used.

Paging

- Paging is a memory-management scheme that permits the physical address space a process to be non-contiguous. Paging avoids external fragmentation and then no need for compaction.
- Secondary memory is divides into fixed size partition all of them of same size called pages.
- Main memory is divided into fix size partitions, each of them having same size called frames.
- Size of frame = size of page
- In general number of pages are much more than number of frames (approx. 250 time)
- A process will have minimum one page, and theoretically may have all the pages in the secondary memory.
- Page cannot have two or more process.

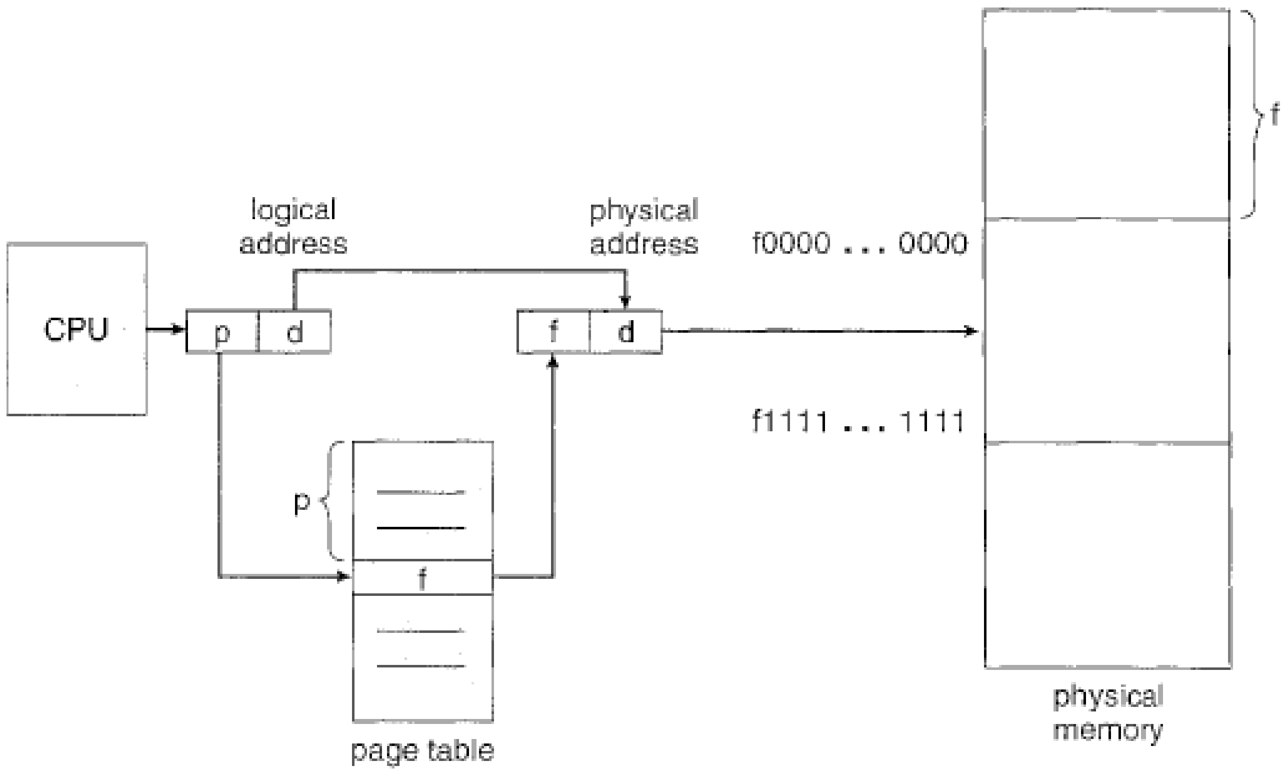
10^3	1 Thousand
10^6	1 Million
10^9	1 Billion
10^{12}	1 Trillion

10^3	1 kilo
10^6	1 Mega
10^9	1 Giga
10^{12}	1 Tera
10^{15}	1 Peta

2^{10}	1 kilo
2^{20}	1 Mega
2^{30}	1 Giga
2^{40}	1 Tera
2^{50}	1 Peta

Translation process: -

- CPU generate a logical address which contains 2 parts- P and d, where p is page no and d stands for instruction offset.
- P is used to find the correct page number, then d is used to find the line number inside a page.
- Page table base register (PTBR) provides the base of the page table Changing page tables requires changing only this one register, substantially reducing context-switch time. and using p then the corresponding page no(index) is accessed.
- Here we will find the corresponding frame no (the base address of that frame in main memory in which the page is stored)
- Combine corresponding frame no with the instruction offset and get the physical address. Which is used to access main memory.



Page Table

- Page table is a data structure not hardware.
- Every process has a separate page table.
- Number of entries a process has in the page table is the number of pages a process has in the secondary memory.
- Size of each entry in the page table is same it is corresponding frame number.
- Page table is a data structure which is itself stored in main memory.
- The page table contains the base address of each page in physical memory.
- Page table is kept in main memory, and a PTBR (Page Table Base Register) points to the page table.

Advantage: -

- Removal of External Fragmentation

Disadvantage: -

- Translation process is slow as page table is accessed two times (one for page table and other for actual access).
- A considerable amount of space a waisted in storing page table (meta data).
- System suffers from internal fragmentation (as paging is an example of fixed size partition).
- Translation process is difficult and complex to understand and implement.

Q Page information in memory is also called as Page Table. The essential contents in each entry of a page table is/are __________. **(NET-JULY-2018)**
(a) Page Access information
(b) Virtual Page number
(c) Page Frame number
(d) Both virtual page number and Page Frame Number
Answer: (c)

Q The essential content(s) in each entry of a page table is / are? **(GATE-2009) (1 Marks)**
(A) Virtual page number **(B)** Page frame number
(C) Both virtual page number and page frame number **(D)** Access right information
Answer: (B)

Address Length in bits	No of Locations
n	**2^n**

No of Locations	Address Length in bits
n	**Upper Bound ($\log_2 n$)**

Memory Size = Number of Location * Size of each Location

S No	SM	LA	MM	PA	p	f	d	addressable	Page Size
1	32 GB		128 MB					1B	1KB
2		42		33			11	1B	
3	512GB			31				1B	512B
4	128*MM		32GB		30			1B	
5					28	14			4096B

S No	SM	LA	MM	PA	p	f	d	addressable	Process Size	PT Size
1	16 GB			26			10	1B		2MB
2		44		36			12	1B	1GB	
3		39		28			8	1B	512MB	

Q A memory management system has 64 pages with 512 bytes page size. Physical memory consists of 32-page frames. Number of bits required in logical and physical address are respectively (NET-DEC-2016)

a) 14 and 15 **b)** 14 and 29 **c)** 15 and 14 **d)** 16 and 32

Answer C

Q Consider a system with byte-addressable memory, 32-bit logical addresses, 4 kilobyte page size and page table entries of 4 bytes each. The size of the page table in the system in megabytes is ___________**(GATE-2015) (2 Marks)**

Answer: 4

Q A Computer system implements 8 kilobyte pages and a 32-bit physical address space. Each page table entry contains a valid bit, a dirty bit three permission bits, and the translation. If the maximum size of the page table of a process is 24 megabytes, the length of the virtual address supported by the system is _______________ bits? **(GATE-2015) (2 Marks)**

Answer: 36

Q A computer system supports 32-bit virtual address as well as 32-bit physical addresses. Since the virtual address space is of same size as that of physical address space, if we want to get rid of virtual memory, which one of the following is true? **(NET-JUNE-2012)**

(A) Efficient implementation of multiuser support is no longer possible.

(B) The processor cache can be made more efficient.

(C) Hardware support for memory management is not needed.

(D) CPU scheduling can be made more efficient.

Answer: (C)

Q Consider a logical address space of 8 pages of 1024 words mapped with memory of 32 frames. How many bits are there in the physical address? **(NET-DEC-2011)**

(A) 9 bits **(B)** 11 bits **(C)** 13 bits **(D)** 15 bits

Answer: (D)

Q Consider a machine with 64 MB physical memory and a 32-bit virtual address space. If the page size is 4KB, what is the approximate size of the page table? **(GATE-2001) (2 Mark)**

(a) 16 MB **(b)** 8 MB **(c)** 2 MB **(d)** 24 MB

Answer: (C)

Translation Look Aside Buffer (TLB)

- A serious problem with page is, translation process is slow as page table is accessed two times (one for page table and other for actual access).
- To solve the problems in paging we take the help of TLB. The TLB is associative, high-speed memory.
- Each entry in the TLB consists of two parts: a key (Page no) and a value (frame no). When the associative memory is search for page no, the page no is compared with all page no simultaneously. If the item is found, the corresponding frame no field is returned.
- The search is fast; **the hardware, however, is expensive, TLB** Contains the frequently referred page numbers and corresponding frame number.
- The TLB is used with page tables in the following way. The TLB contains only a few of the page-table entries. When a logical address is generated by the CPU, its page number is presented to the TLB. If the page number is found, its frame number is immediately available and is used to access memory.

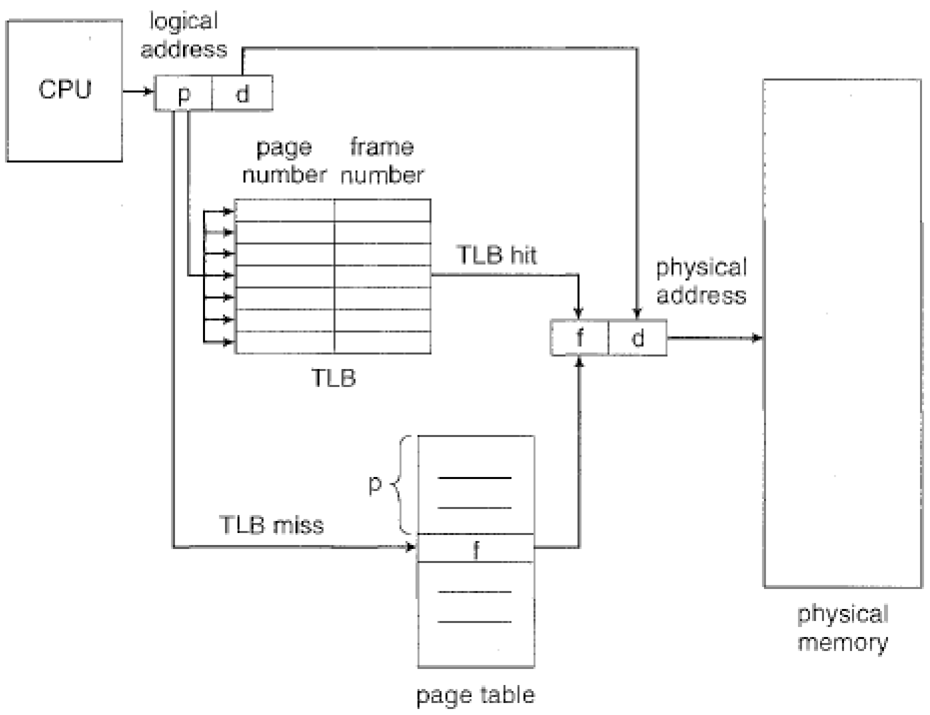


Figure: TLB

- If the page number is not in the TLB (known as a **TLB Miss**), then a memory reference to the page table must be made.
- Also we add the page number and frame number to the TLB, so that they will be found quickly on the next reference.
- If the TLB is already full of entries, the operating system must select one for replacement i.e. Page replacement policies.

- The percentage of times that a particular page number is found in the TLB is called the **Hit Ratio**.
- **Effective Memory Access Time:** Hit [TLB + Main Memory] + 1-Hit [TLB + 2 Main Memory]
- **TLB** removes the problem of slow access.

Disadvantage of TLB:

- TLB can hold the data of one process at a time and in case of multiple context switches TLB will be required to flush frequently.

Solution:

- Use multiple TLB's but it will be costly.
- Some TLBs allow certain entries to be **wired down**, meaning that they cannot be removed from the TLB. Typically, TLB entries for kernel code are wired down.

Q Assume that in a certain computer, the virtual addresses are 64 bits long and the physical addresses are 48 bits long. The memory is word addressable. The page size is 8k Band the word size is 4 bytes. The Translation Look-aside Buffer (TLB) in the address translation path has 128 valid entries. At most how many distinct virtual addresses can be translated without any TLB miss? (GATE-2019) (2 Marks)

(A) 16×2^{10} **(B)** 8×2^{20} **(C)** 4×2^{20} **(D)** 256×2^{10}

Answer: (D)

Q In a paging system, it takes 30 ns to search translation Look-a-side Buffer (TLB) and 90 ns to access the main memory. If the TLB hit ratio is 70%, the effective memory access time is : **(NET-JAN-2017)**

a) 48ns **b)** 147ns **c)** 120ns **d)** 84ns

Answer: B

Q A computer system implements a 40-bit virtual address, page size of 8 kilobytes, and a 128-entry translation look-aside buffer (TLB) organized into 32 sets each having four ways. Assume that the TLB tag does not store any process id. The minimum length of the TLB tag in bits is __________**(GATE-2015) (2 Marks)**

(A) 20 **(B)** 10 **(C)** 11 **(D)** 22

Answer: 22

Q Consider a paging hardware with a TLB. Assume that the entire page table and all the pages are in the physical memory. It takes 10 milliseconds to search the TLB and 80 milliseconds to access the physical memory. If the TLB hit ratio is 0.6, the effective memory

access time (in milliseconds) is __________. **(CS-2014) (2 Marks)**
Answer: 122

Q Consider a paging hardware with a TLB. Assume that the entire page table and all the pages are in the physical memory. It takes 10 milliseconds to search the TLB and 80 milliseconds to access the physical memory. If the TLB hit ratio is 0.6, the effective memory access time (in milliseconds) is __________. **(NET-JUNE-2014)**

(A) 120 **(B)** 122 **(C)** 124 **(D)** 118

Ans: b

Q The hit ratio of a Translation Look Aside Buffer (TLAB) is 80%. It takes 20 nanoseconds (ns) to search TLAB and 100 ns to access main memory. The effective memory access time is _______. **(NET-SEP-2013)**

(A) 36 ns **(B)** 140 ns (C) 122 ns **(D)** 40 ns

Answer: (b)

Q Translation Look-aside Buffer (TLB) is **(NET-DEC-2013)**

(A) a cache-memory in which item to be searched is compared one-by-one with the keys.
(B) a cache-memory in which item to be searched is compared with all the keys simultaneously.
(C) an associative memory in which item to be searched is compared one-by-one with the keys.
(D) an associative memory in which item to be searched is compared with all the keys simultaneously.

Answer: (D)

Q The virtual address generated by a CPU is 32 bits. The Translation Look-aside Buffer (TLB) can hold total 64-page table entries and a 4-way set associative (i.e. with 4- cache lines in the set). The page size is 4 KB. The minimum size of TLB tag is **(NET-DEC-2013)**

(A) 12 bits **(B)** 15 bits **(C)** 16 bits **(D)** 20 bits

Q A paging system tlb takes 10ns main memory takes 50ns what is effective memory access time if tlb hit ratio is 90%? **(GATE-2008) (1 Marks)**

a) 54 **b)** 60 **c)** 65 **d)** 75

Answer: (C)

Q A CPU generates 32-bit virtual addresses. The page size is 4 KB. The processor has a translation look-aside buffer (TLB) which can hold a total of 128-page table entries and is 4-way set associative. The minimum size of the TLB tag is? **(GATE-2006) (2 Marks)**

(A) 11 bits **(B)** 13 bits **(C)** 15 bits **(D)** 20 bits

Answer: (C)

Q Which of the following is not a form of memory? **(GATE-2002) (1 Marks)**

(A) instruction cache **(B)** instruction register

(C) instruction opcode **(D)** translation lookaside buffer

Answer: (C)

Size of Page

- If we increase the size of page table then internal fragmentation increase but size of page table decreases.
- If we decrease the size of page table then internal fragmentation decrease but size of page table increases.
- So, we have to find what should be the size of the page, where both costs are minimal.

Q For the implementation of a paging scheme, suppose the average process size be x bytes, the page size be y bytes, and each page entry requires z bytes. The optimum page size that minimizes the total overhead due to the page table and the internal fragmentation loss is given by **(NET-DEC-2014)**

A) x/2 **b)** xz/2 **c)** root(2xz) **d)** root(xz)/2

Answer: (C)

Q Average process size=s bytes. Each page entry requires e bytes. The optimum page size is given by: **(NET-DEC-2007)**

(A) √(se) **(B)** √(2se) **(C)** s **(D)** e

Answer: B

Multilevel Paging / Hierarchical Paging

- Modern systems support a large logical address space (2^32 to 2^64).
- In such cases, the page table itself becomes excessively large and can contain millions of entries and can take a lot of space in memory, so cannot be accommodated into a single frame.
- A simple solution to this is to divide page table into smaller pieces.
- One way is to use a **two-level paging algorithm**, in which the page table itself is also paged.

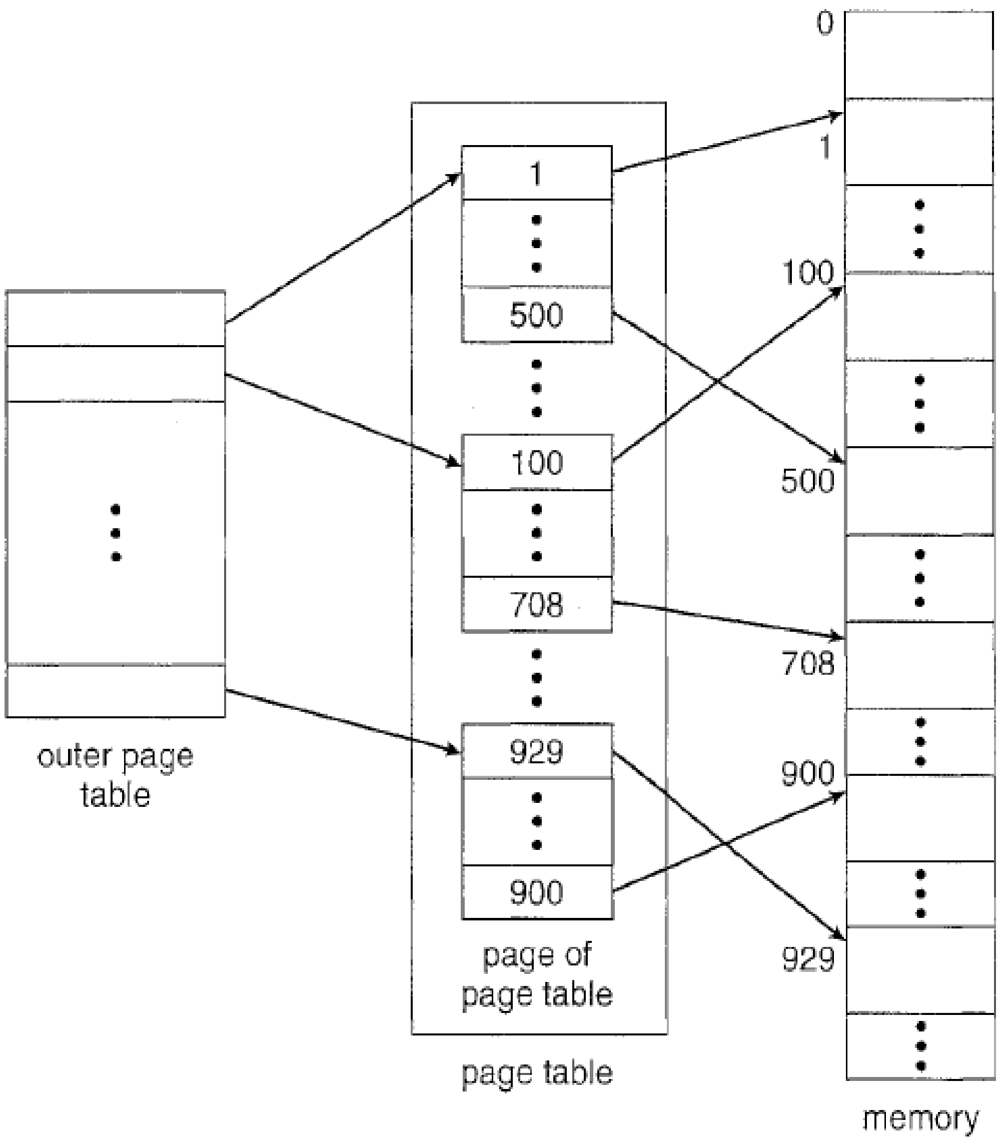
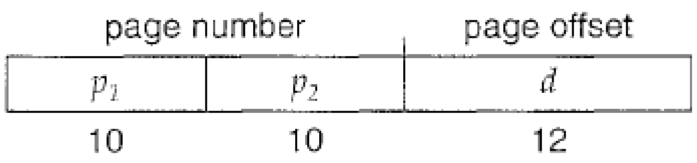


Figure: Two-level Paging

- Consider an example of a 32-bit logical address space and a page size of 4 KB. A logical address is divided into a page number consisting of 20 bits and a page offset consisting of 12 bits. Because we page the page table, the page number is further divided into a 10-bit page number and a 10-bit page offset. Thus, a logical address is as follows:



- p1 is an index into the outer page table and p2 is the displacement within the page of the outer page table.

- The address translation works from the outer page table inward; this scheme is also known as a **Forward Mapped Page Table.**

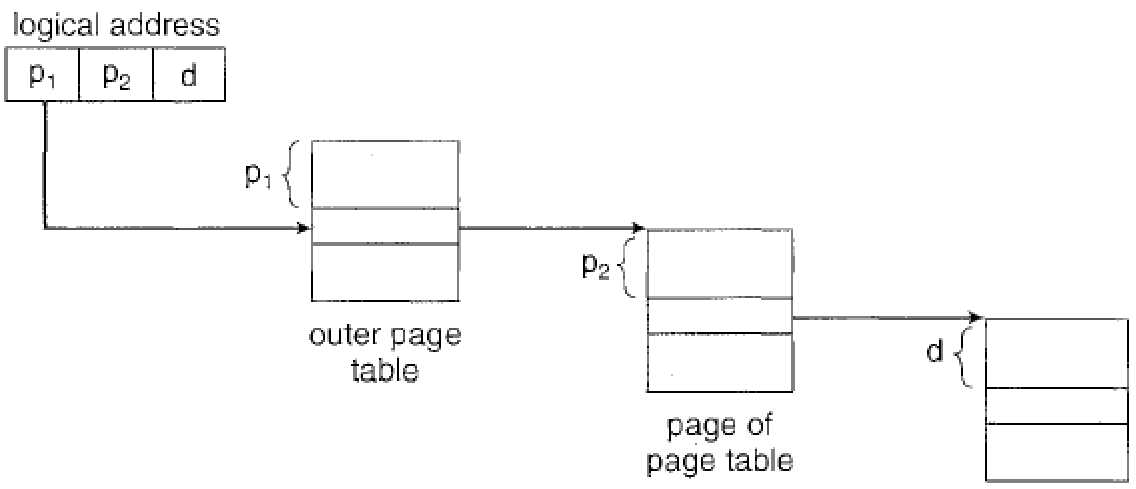


Figure: Address Translation

- In case of the page table kept in main memory without TLB, the memory accesses will be 3.
- Using TLB to improve access time: Effective Memory Access Time: Hit [TLB + Main Memory] + 1-Hit [TLB + 3 Main Memory]

Q In a system with 32 bit virtual addresses and 1 KB page size, use of one-level page tables for virtual to physical address translation is not practical because of **(GATE-2003) (1 Marks)**
(A) the large amount of internal fragmentation
(B) the large amount of external fragmentation
(C) the large memory overhead in maintaining page tables
(D) the large computation overhead in the translation process
Answer: (C)

Q A multilevel page table is preferred in comparison to a single level page table for translating virtual address to physical address because(GATE-2009) (1 Marks)

(A) It reduces the memory access time to read or write a memory location.

(B) It helps to reduce the size of page table needed to implement the virtual address space of a process.

(C) It is required by the translation lookaside buffer.

(D) It helps to reduce the number of page faults in page replacement algorithms.

Answer: (B)

Q A processor uses 2-level page tables for virtual to physical address translation. Page tables for both levels are stored in the main memory. Virtual and physical addresses are both 32 bits wide. The memory is byte addressable. For virtual to physical address translation, the 10 most significant bits of the virtual address are used as index into the first level page table while the next 10 bits are used as index into the second level page table. The 12 least significant bits of the virtual address are used as offset within the page. Assume that the page table entries in both levels of page tables are 4 bytes wide. Further, the processor has a translation look-aside buffer (TLB), with a hit rate of 96%. The TLB caches recently used virtual page numbers and the corresponding physical page numbers. The processor also has a physically addressed cache with a hit rate of 90%. Main memory access time is 10 ns, cache access time is 1 ns, and TLB access time is also 1 ns. Assuming that no page faults occur, the average time taken to access a virtual address is approximately (to the nearest 0.5 ns) **(GATE-2003) (2 Marks)**

(A) 1.5 ns **(B)** 2 ns **(C)** 3 ns **(D)** 4 ns

Answer: (D)

Q A processor uses 36-bit physical addresses and 32-bit virtual addresses, with a page frame size of 4 Kbytes. Each page table entry is of size 4 bytes. A three level page table is used for virtual to physical address translation, where the virtual address is used as follows • Bits 30-31 are used to index into the first level page table • Bits 21-29 are used to index into the second level page table • Bits 12-20 are used to index into the third level page table, and • Bits 0-11 are used as offset within the page The number of bits required for addressing the next level page table (or page frame) in the page table entry of the first, second and third level page tables are respectively **(GATE-2008) (2 Marks)**

(A) 20, 20 and 20 **(B)** 24, 24 and 24 **(C)** 24, 24 and 20 **(D)** 25, 25 and 24

Answer: (D)

Q A computer uses 46–bit virtual address, 32–bit physical address, and a three–level paged page table organization. The page table base register stores the base address of the first–level table (T1), which occupies exactly one page. Each entry of T1 stores the base address

of a page of the second–level table (T2). Each entry of T2 stores the base address of a page of the third–level table (T3). Each entry of T3 stores a page table entry (PTE). The PTE is 32 bits in size. The processor used in the computer has a 1 MB 16 way set associative virtually indexed physically tagged cache. The cache block size is 64 bytes.
What is the size of a page in KB in this computer? **(GATE-2013) (2 Marks)**
(A) 2 **(B)** 4 **(C)** 8 **(D)** 16
Answer: (C)

Q Consider a computer system with 40-bit virtual addressing and page size of sixteen kilobytes. If the computer system has a one-level page table per process and each page table entry requires 48 bits, then the size of the per-process page table is __________ megabytes. **(GATE-2016) (2 Marks)**
Answer: 384

Segmentation

- Paging is unable to separate the user's view of memory from the actual physical memory.
- Segmentation is a memory-management scheme that supports this user view of memory.

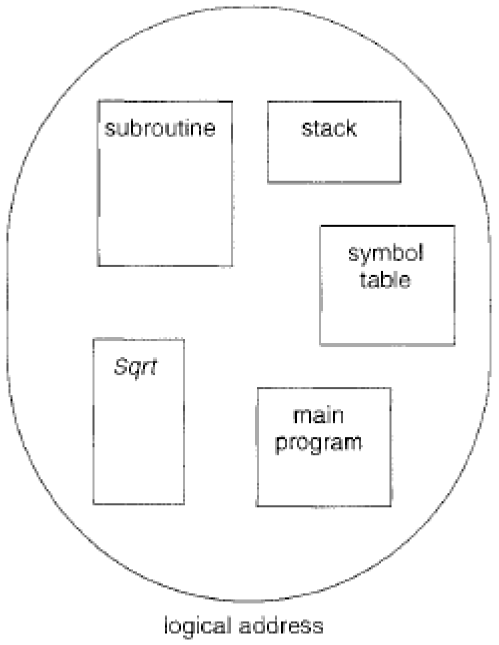


Figure: User View of the Program

- A logical address space is a collection of segments.
- Each segment has a name and a length. The addresses specify both the segment name and the offset within the segment. The user therefore specifies each address by two quantities: **a segment name and an offset.**
- Thus, a logical address consists of a two tuple:

  <segment-number, offset>.
- Segments can be of variable lengths unlike pages and are stored in main memory.

Segmentation Hardware

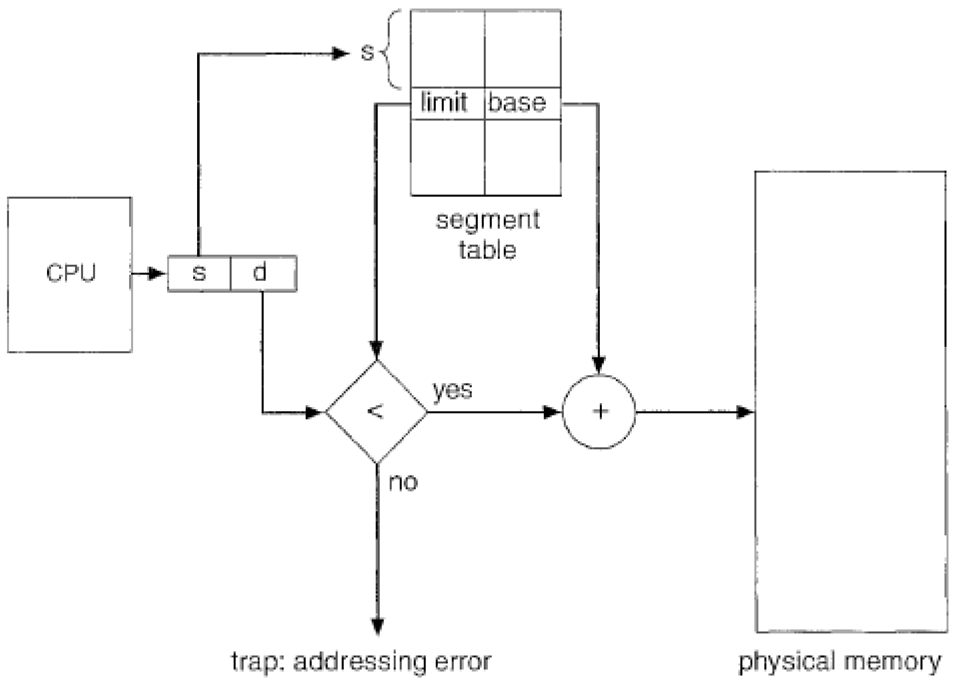
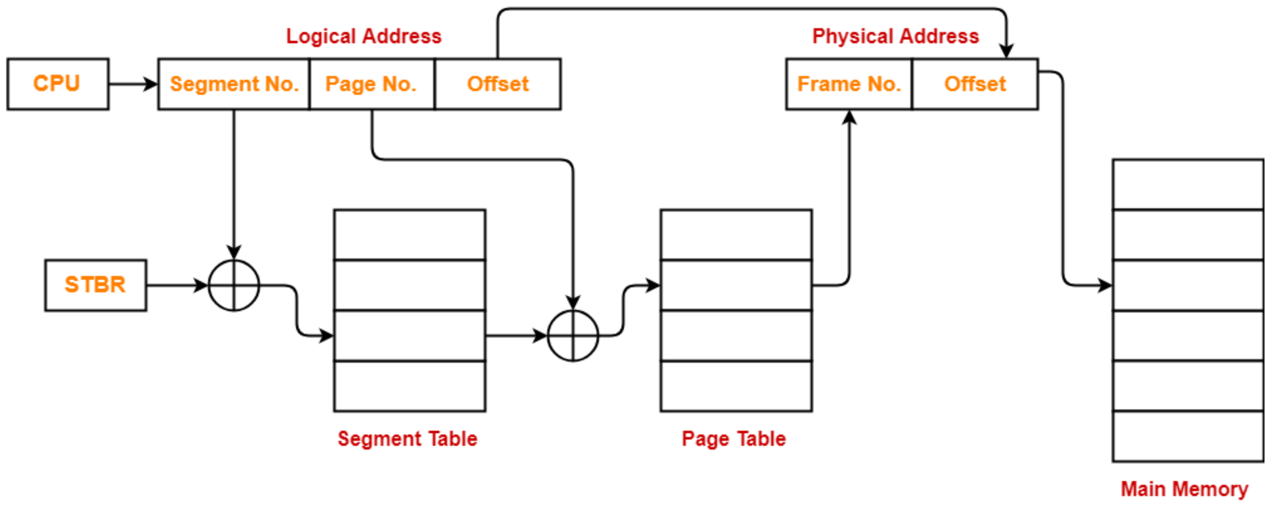


Figure: Segmentation Hardware

- **Segment Table:** Each entry in the segment table has a **segment base** and a **segment limit**. The segment base contains the starting physical address where the segment resides in memory, and the segment limit specifies the length of the segment.
- The segment number is used as an index to the segment table.
- The offset d of the logical address must be between 0 and the segment limit.If it is not, we trap to the operating system.
- When an offset is legal, it is added to the segment base to produce the address in physical memory of the desired byte.
- Segmentation suffers from **External Fragmentation.**

Segmentation with Paging

- Since segmentation also suffers from external fragmentation, **it is better to divide the segments into pages as the segments size increases.**
- In segmentation with paging scheme a process is divided into segments and further the segments are divided into pages.
- One can argue it is segmentation with paging is quite similar to multilevel paging, but actually it is better, because here when page table is divided the size of partition can be different (as actually the size of different chapters can be different).



Translating Logical Address into Physical Address

Inverted Page Tables

- The drawback of previous methods is that each page table may consist of millions of entries. These tables may consume large amounts of physical memory just to keep track of how other physical memory is being used.

- To solve this problem, we can use an **Inverted Page Table.**

- An inverted page table has one entry for each real page (or frame) of memory.

- Each entry consists of the virtual address of the page stored in that real memory location, with information about the process that owns the page. Thus, only one page table is in the system, and it has only one entry for each page of physical memory.

- Thus number of entries in the page table is equal to the number of frames in the physical memory.

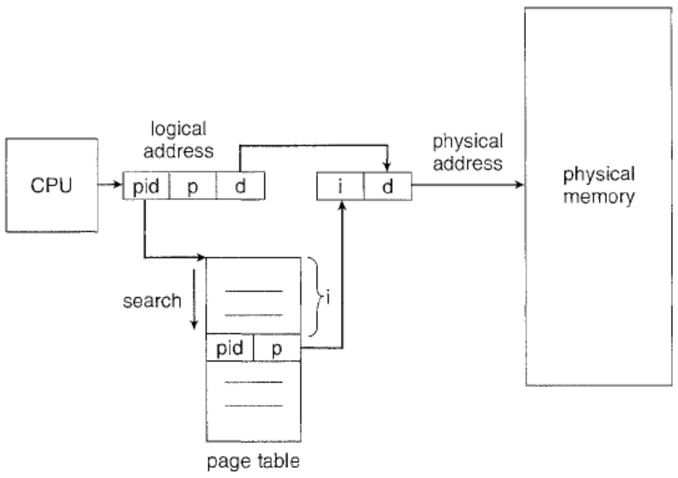


Figure: Inverted Page Table

- Each inverted page-table entry is a pair <process-id, page-number> where the process-id assumes the role of the **address-space identifier (ASID)**.

- **An ASID uniquely identifies each process and is used to provide address-space protection for that process.**

- When a memory reference occurs, part of the virtual address, consisting of <process-id, page number>, is presented to the memory subsystem. The inverted page table is then searched for a match. If a match is found-say, at entry i-then the physical address <i, offset> is generated. If no match is found, then an illegal address access has been attempted.

- **This scheme decreases the amount of memory needed to store each page table, but it increases the amount of time needed to search the table when a page reference occurs. Because the inverted page table is sorted by physical address, but lookups occur on virtual addresses, the whole table might need to be searched for a match.**

Q A computer system supports 32 bit virtual address as well as 32 bit physical addresses. Since the virtual address space is of same size as that of physical address space, if we want to get rid of virtual memory, which one of the following is true? **(NET-JUNE-2012)**

(A) Efficient implementation of multiuser support is no longer possible.

(B) The processor cache can be made more efficient.

(C) Hardware support for memory management is not needed.

(D) CPU scheduling can be made more efficient.
Ans: a

Q A computer system supports 32-bit virtual addresses as well as 32-bit physical addresses. Since the virtual address space is of the same size as the physical address space, the operating system designers decide to get rid of the virtual memory entirely. Which one of the following is true? **(GATE-2006) (1 Marks)**
(A) Efficient implementation of multi-user support is no longer possible
(B) The processor cache organization can be made more efficient now
(C) Hardware support for memory management is no longer needed
(D) CPU scheduling can be made more efficient now
Answer: (C)

Q The memory access time is 1 nanosecond for a read operation with a hit in cache, 5 nanoseconds for a read operation with a miss in cache, 2 nanoseconds for a write operation with a hit in cache and 10 nanoseconds for a write operation with a miss in cache. Execution of a sequence of instructions involves 100 instruction fetch operations, 60 memory operand read operations and 40 memory operands write operations. The cache hit-ratio is 0.9. The average memory access time (in nanoseconds) in executing the sequence of instructions is __________. **(GATE-2014) (2 Marks)**
Answer: 1.68