Virtual Memory Part - II

Foundation Course on Operating Systems for GATE 2022

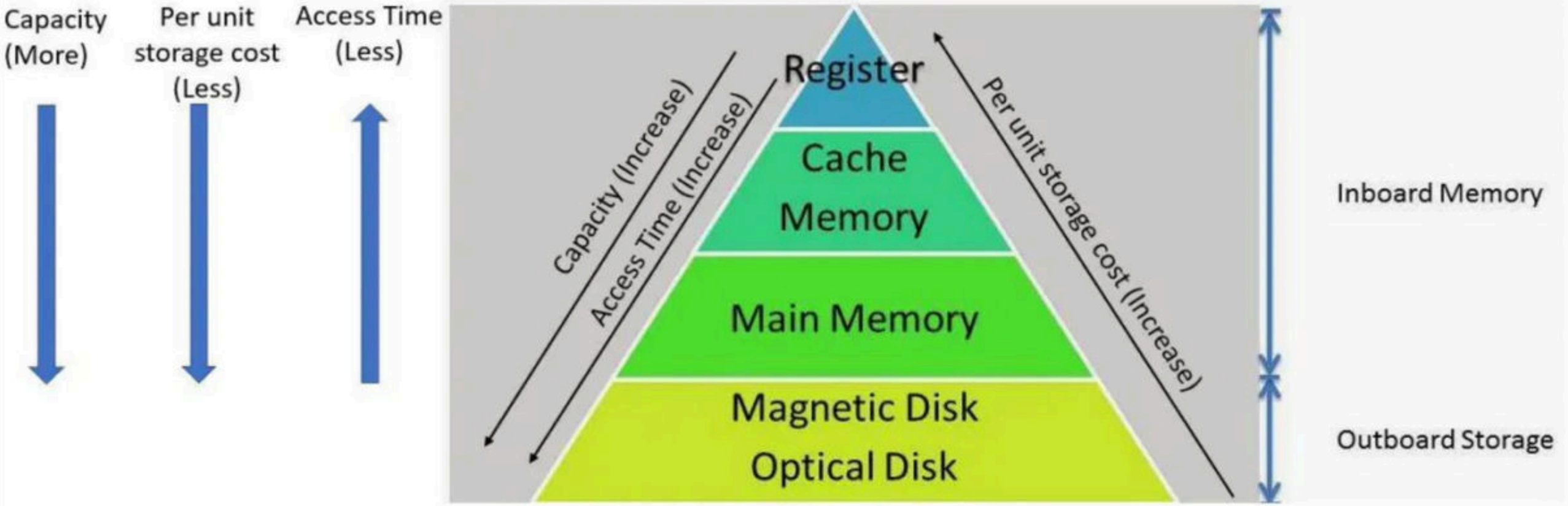
Sanchit Jain • Lesson 25 • Aug 17, 2021

Memory Hierarchy

- Let first understand what we need from a memory
  - Large capacity
  - Less per unit cost
  - Less access time(fast access)

Memory Hierarchy

- The memory hierarchy system consists of all storage devices employed in a computer system from the slow but high-capacity auxiliary memory to a relatively faster main memory, to an even smaller and faster cache memory accessible to the high-speed



Cycle

Car

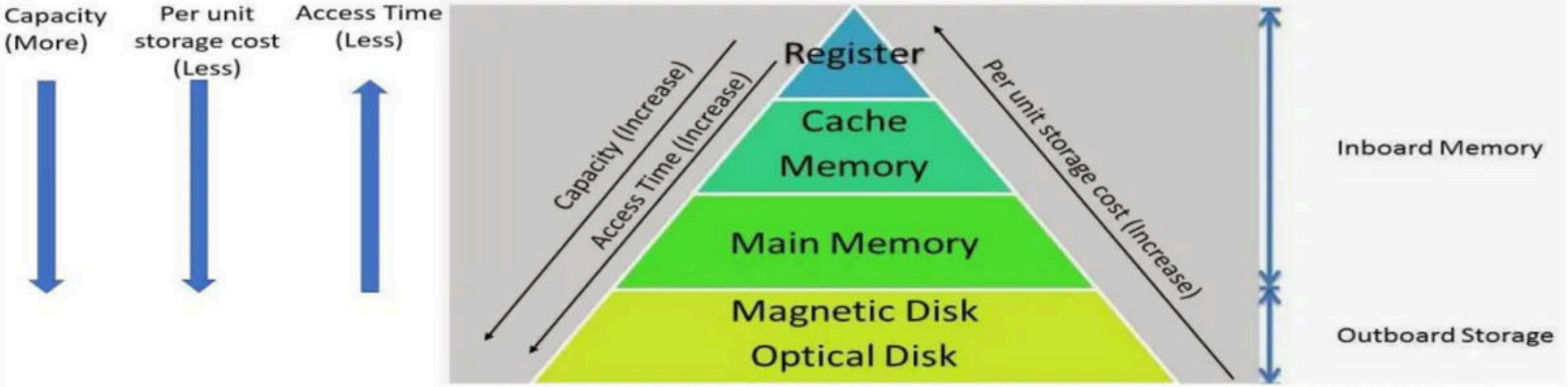
Airbus

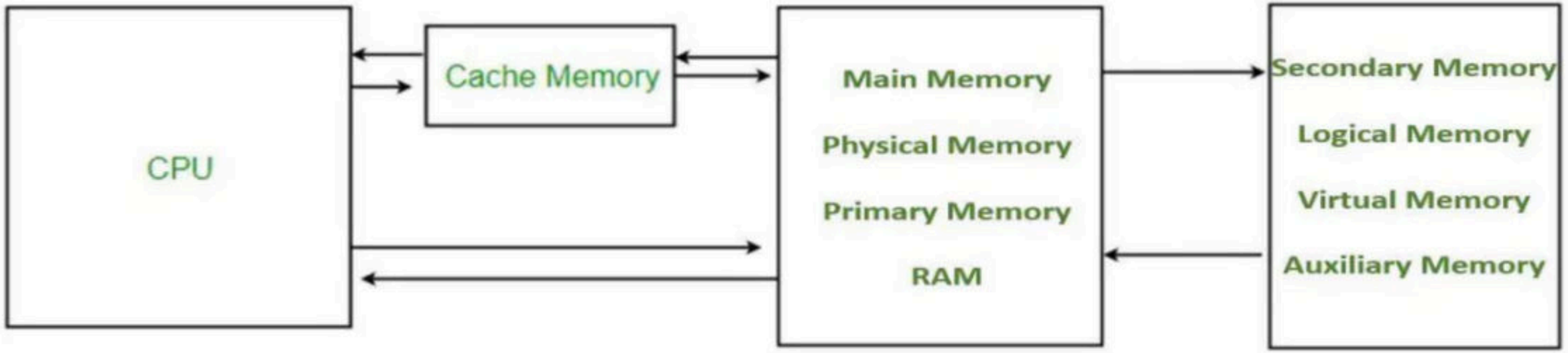
सत्यमेव जयते

Lathi

303

AK-47


Capacity (More)
Per unit storage cost (Less)
Access Time (Less)
Register
Cache Memory
Main Memory
Magnetic Disk
Optical Disk
Capacity (Increase)
Access Time (Increase)
Per unit storage cost (Increase)
Inboard Memory
Outboard Storage


CPU
Cache Memory
Main Memory
Physical Memory
Primary Memory
RAM
Secondary Memory
Logical Memory
Virtual Memory
Auxiliary Memory

Showroom

Go down

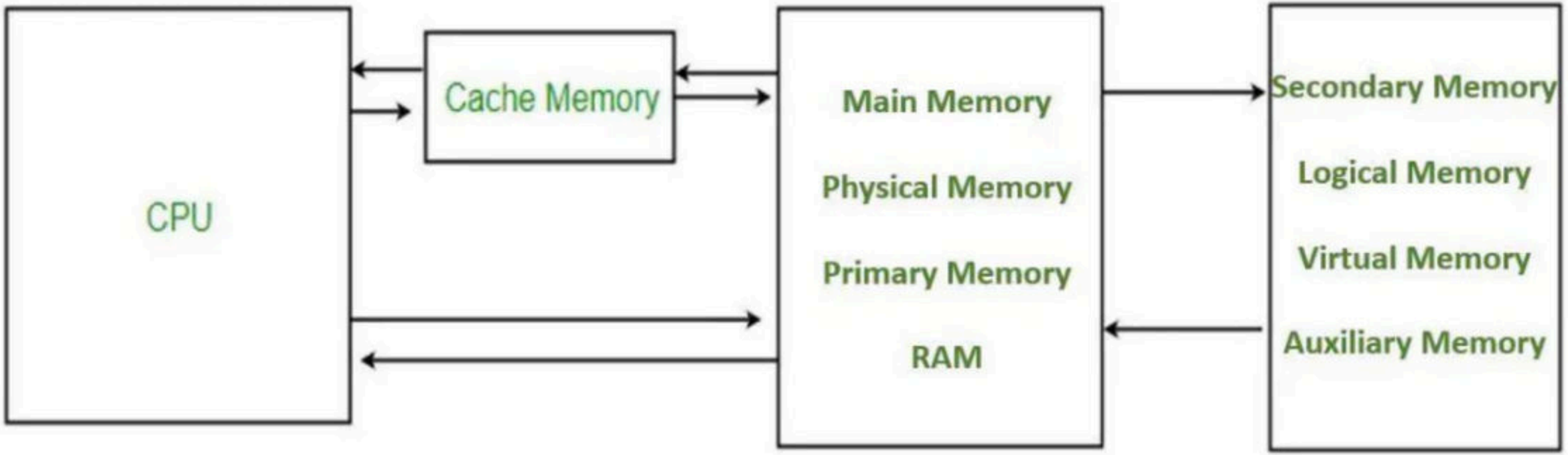
Factory

- If we want this hierarchy to work fine, then most of the time when CPU needs a data it must be present in Cache, if not possible main memory, worst case Secondary memory.

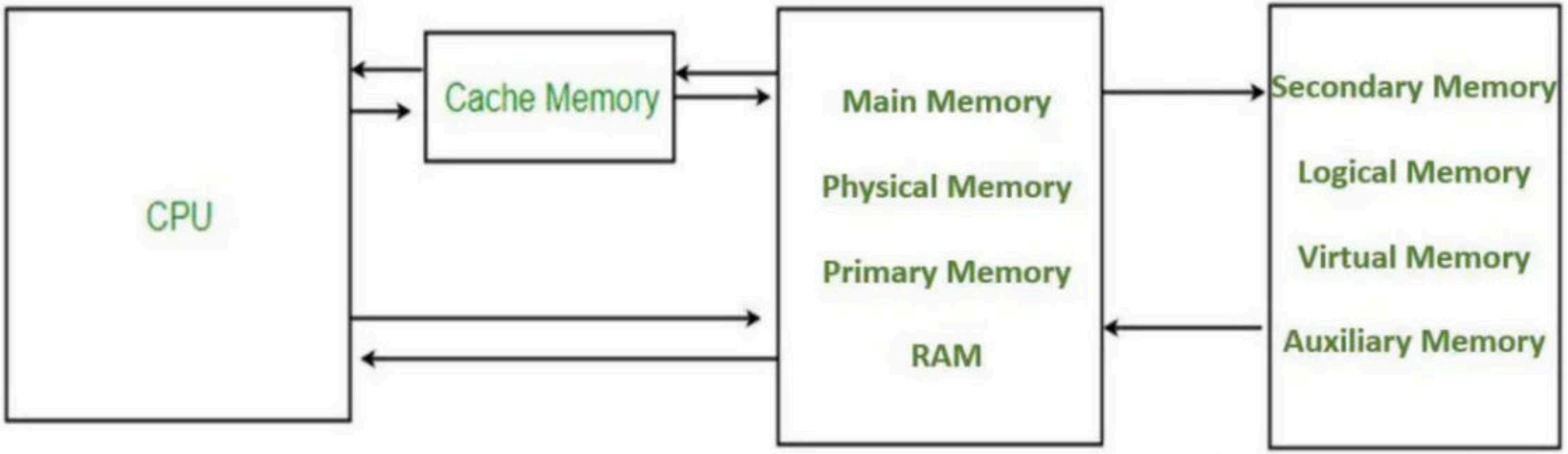
- But this is a difficult task to do as my computer has, 8 TB of Secondary Memory, 32 GB of Main Memory, but only 768KB, 4MB, 16 MB of L_1, L_2, L_3 cache respectively.

- If somehow we can estimate what data CPU will require if future we can prefetch in Cache and Main Memory.

- Locality of reference Helps we to perform this estimation.

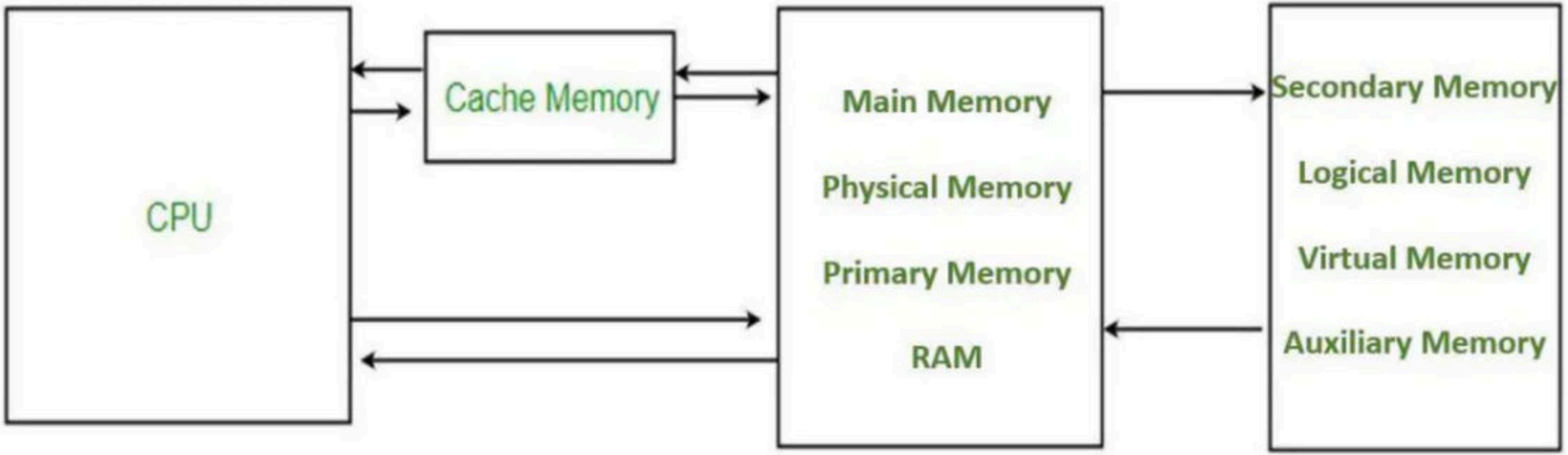


- The main memory occupies a central position by being able to communicate directly with the CPU.
- When programs not residing in main memory are needed by the CPU, they are brought in from auxiliary memory.
- Programs not currently needed in main memory are transferred into auxiliary memory to provide space for currently used programs and data.



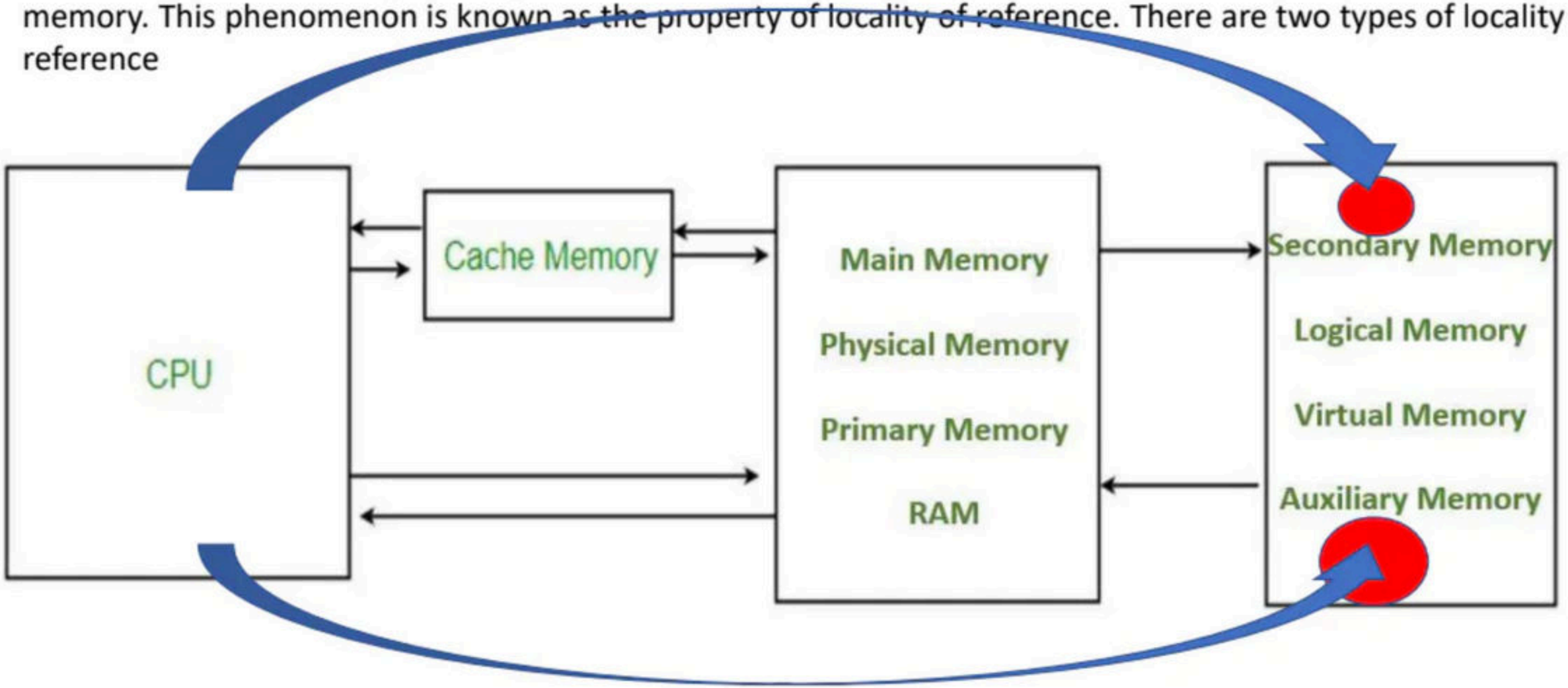
- A special very-high speed memory called a Cache is used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate.

- The cache memory is employed in computer systems to compensate for the speed differential between main memory access time and processor logic.

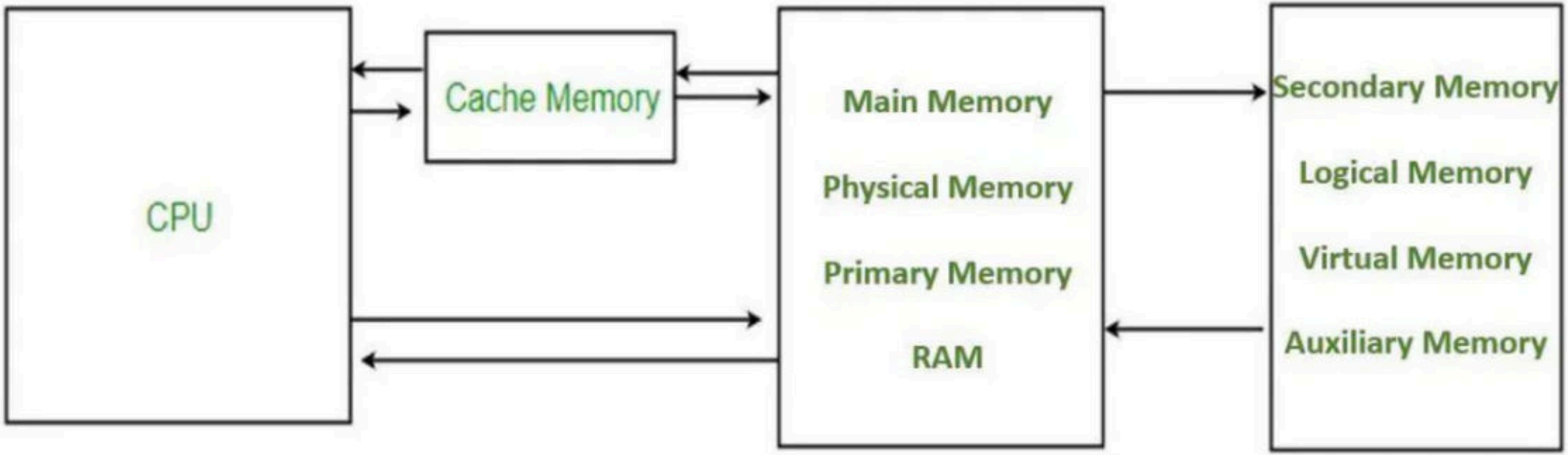


Locality of Reference

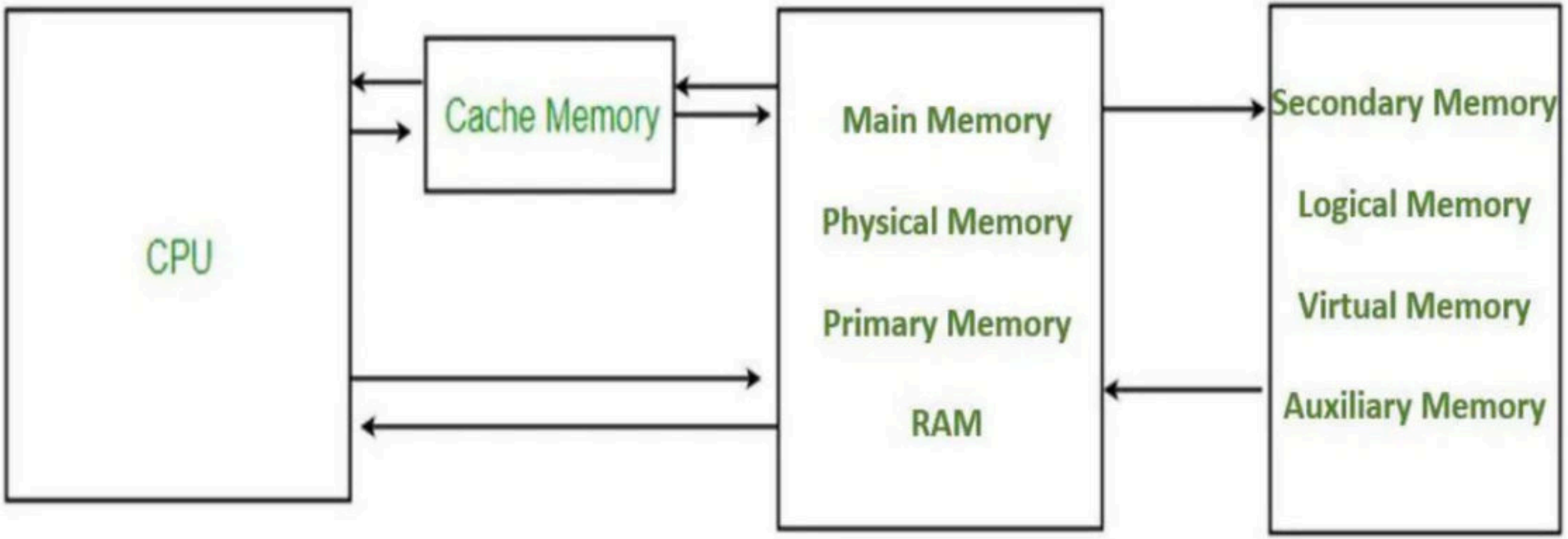
- The references to memory at any given interval of time tend to be confined within a few localized areas in memory. This phenomenon is known as the property of locality of reference. There are two types of locality of reference



- **Spatial Locality:** Spatial locality refers to the use of data elements in the nearby locations.



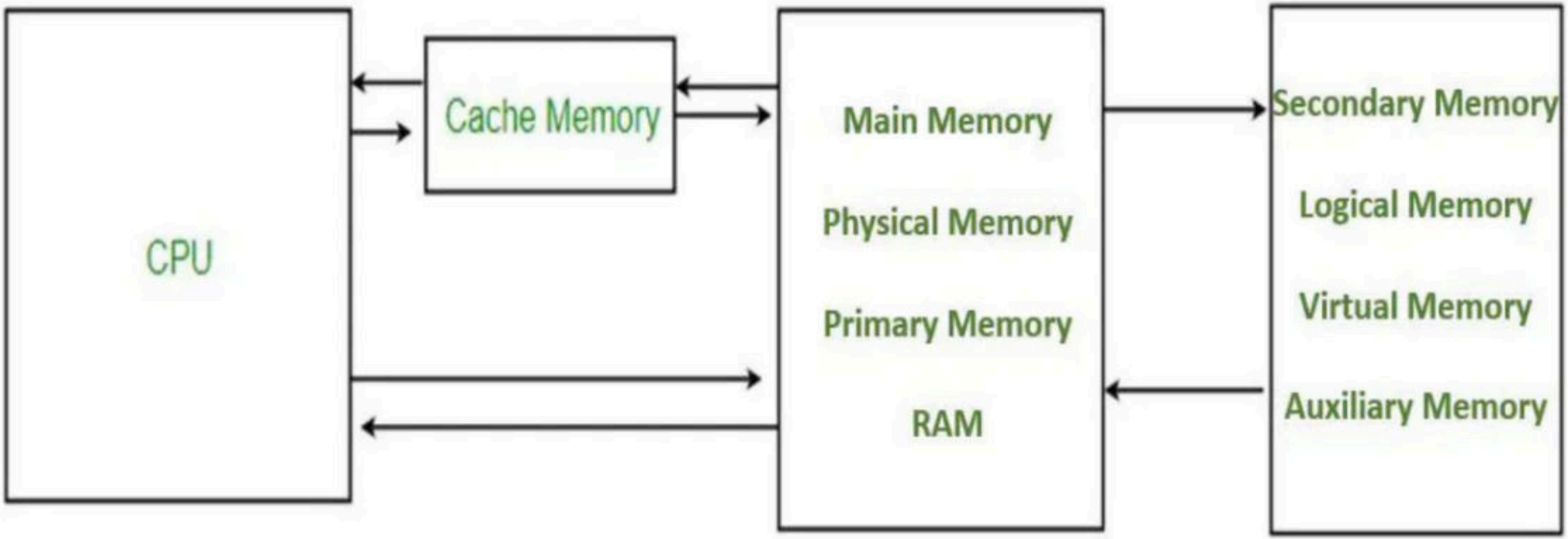
- **Temporal Locality:** Temporal locality refers to the reuse of specific data, and/or resources, within a relatively small-time duration. Or, the most frequently used items will be needed soon. (LRU is used for temporal locality)



Showroom

Go down

Factory


CPU
Cache Memory
Main Memory
Physical Memory
Primary Memory
RAM
Secondary Memory
Logical Memory
Virtual Memory
Auxiliary Memory

- 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)

Break

Duty of Operating System

- Operating system is responsible for the following activities in connection with memory management:
  - Address translation from Logical address to Physical address.
  - Deciding which processes (or parts of processes) and data to move into and out of memory. Allocating and deallocating memory space as needed.
  - Keeping track of which parts of memory are currently being used and who is using them.

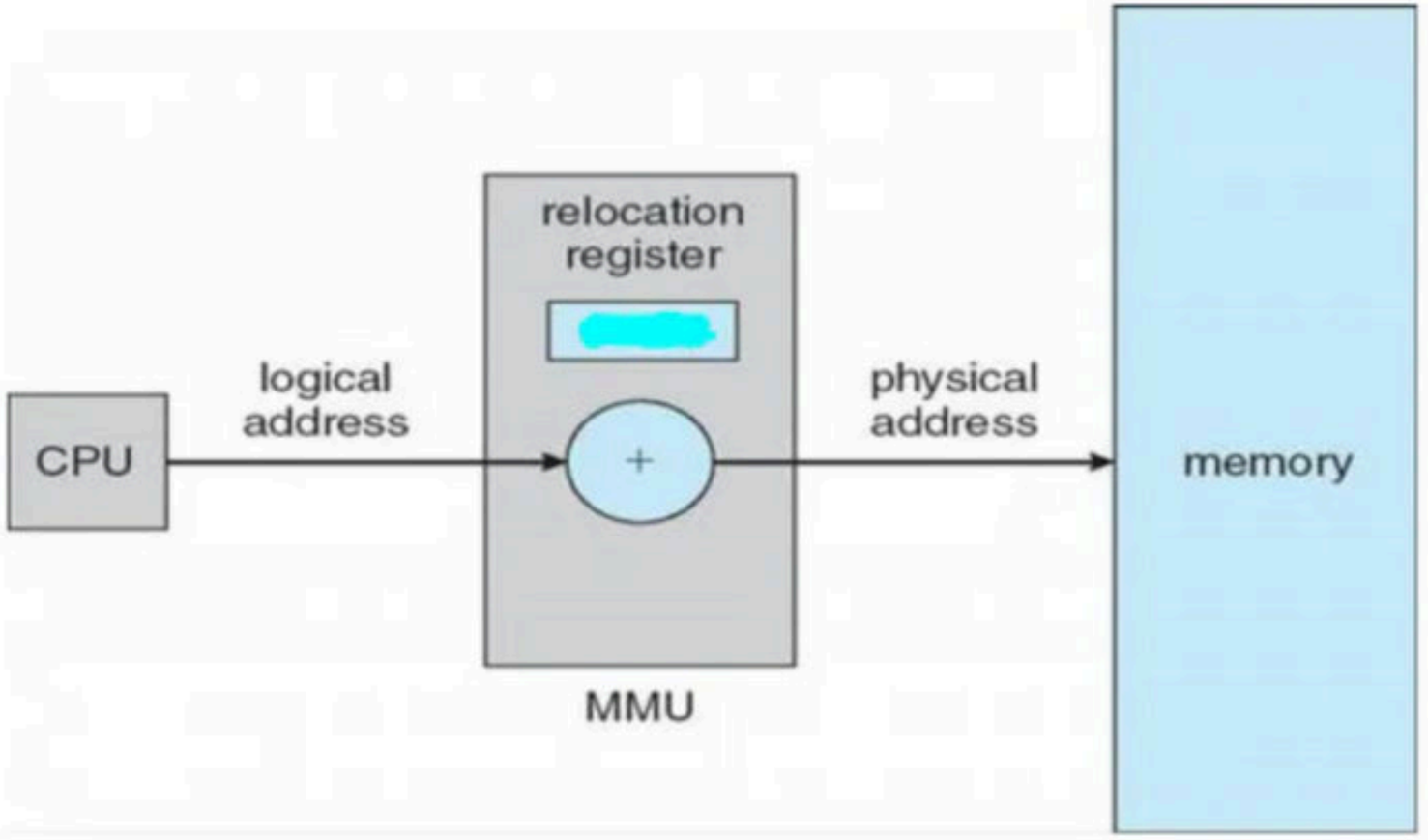
- The first duty of OS is to translate this logical address into physical address. There can be two approaches for storing a process in main memory.
    - Contiguous allocation policy
    - Non-contiguous allocation policy

Contiguous allocation policy

- 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(OS) 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

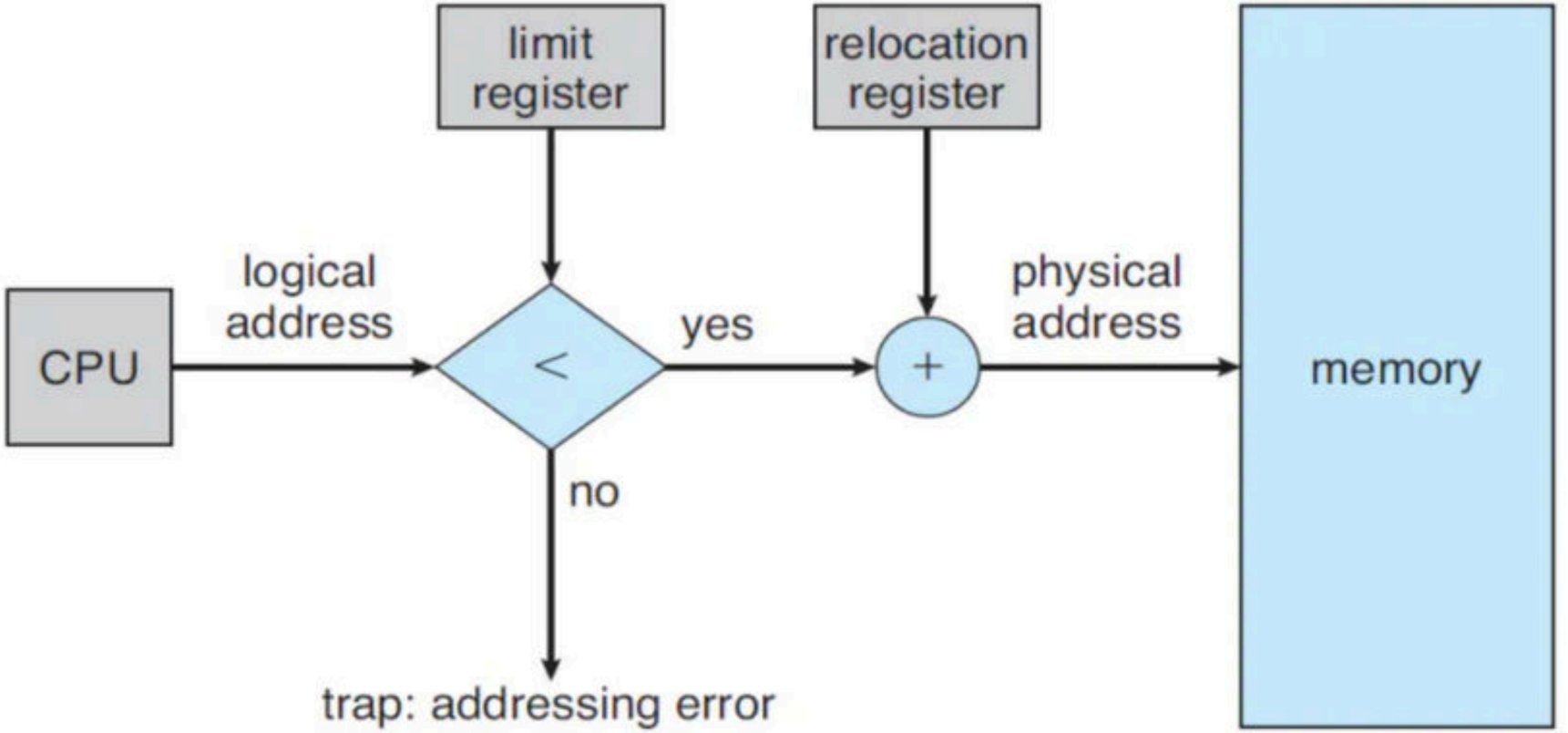


- **Advantage**: - easy strategy, simple to use, simple to understand.

- **Disadvantage**: - if a process generates logical address greater than its maximum limit, then it can easily access the data of some other process.

- In order to check whether address generated to CPU is valid(with in range) or invalid, we compare it with the value of limit register, which contains the max no of instructions in the process.

- 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, it is a illegal request which is immediately trapped by OS.



LA	PA
0	0 + r
Max	Max + r
Min	Min + r

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

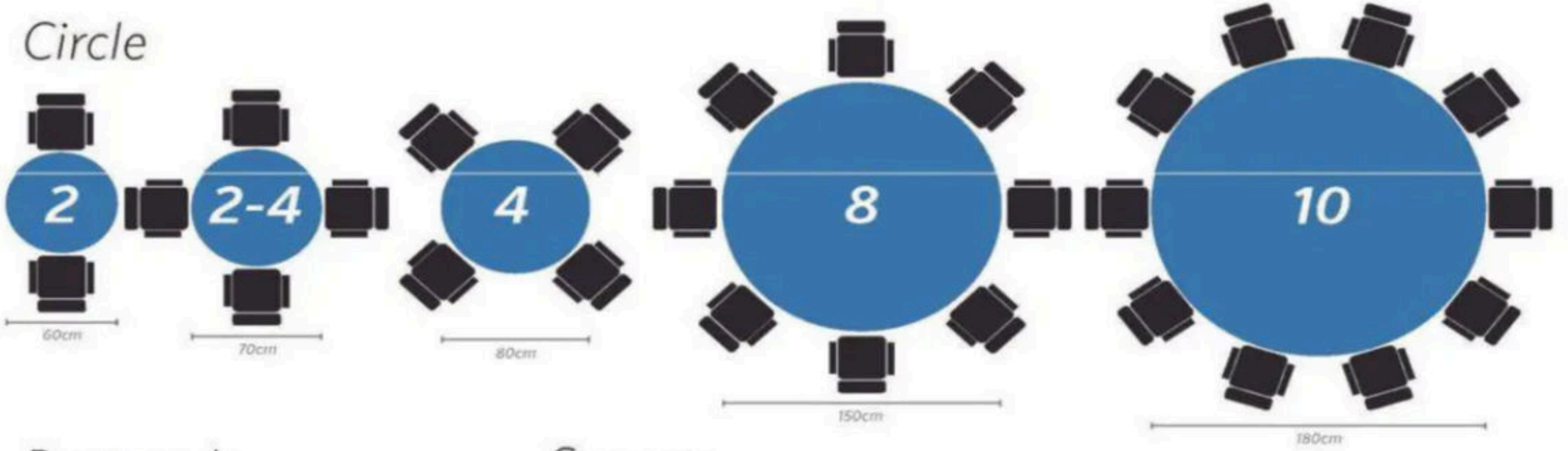
Break

Space Allocation Method in Contiguous Allocation

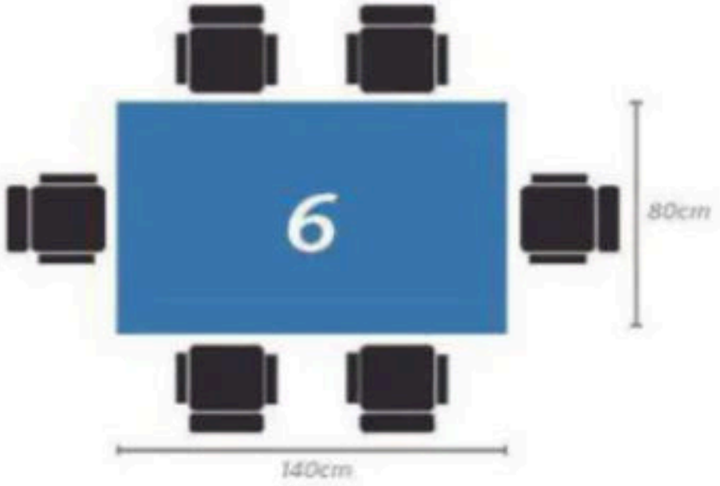
- **Variable size partitioning:** -In this policy, in starting, we treat the memory as a whole or a single chunk & whenever a process request for some space, exactly same space is allocated if possible and the remaining space can be reused again.

- **<u>Fixed size partitioning</u>:** - here, we divide memory into fixed size partitions, which may be of different sizes, but here if a process request for some space, then a partition is allocated entirely if possible, and the remaining space will be waisted internally.

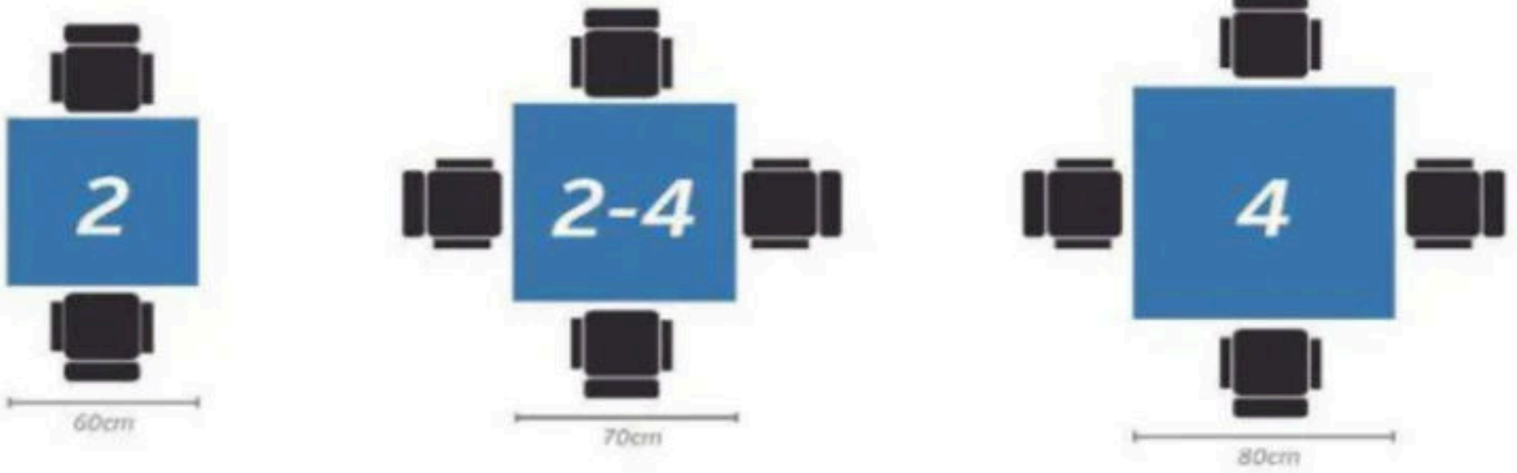
Circle



Rectangle



Square

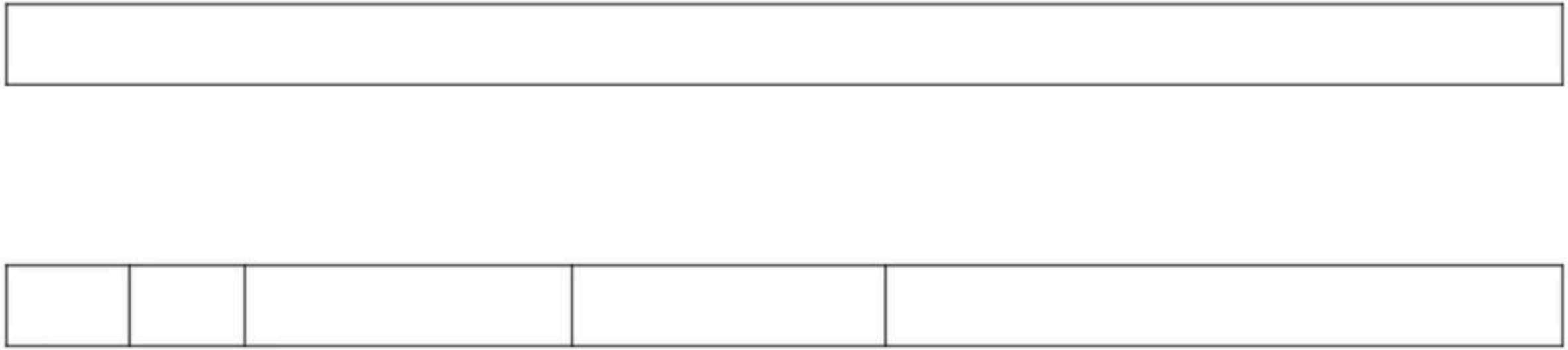


- **First fit policy:** - as the name implies, it states searching the memory from the base and will allocate first partition which is capable enough.
  - **Advantage:** - simple, easy to use, easy to understand
  - **Disadvantage**: -poor performance, both in terms of time and space

- **Best fit policy:** - Here, we search the entire memory and will allocate the smallest partition which is capable enough.
  - **Advantage**: - perform best in fix size partitioning scheme.
  - **Disadvantage**: - difficult to implement, perform worst in variable size partitioning as the remaining spaces which are of very small size.

- **Worst fit policy:** - it also searches the entire memory and allocate the largest partition possible.
  - **Advantage:** - perform best in variable size partitioning
  - **Disadvantage:** - perform worst in fix size partitioning, resulting into large internal fragmentation.

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



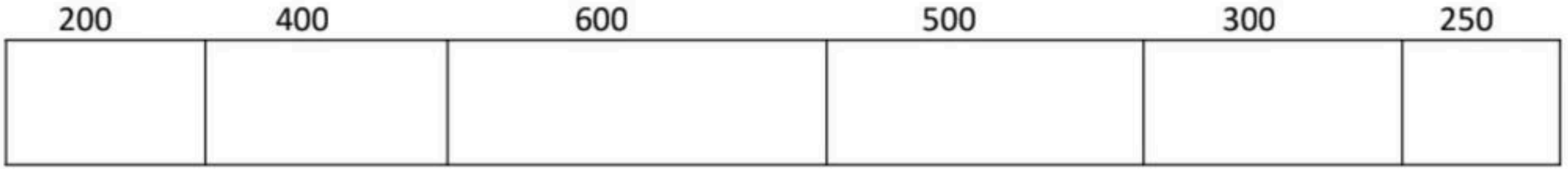
- **External fragmentation:** - External fragmentation is a function of contiguous allocation policy. The space requested by the process is available in memory but, as it is not being contiguous, cannot be allocated this wastage is called external fragmentation.

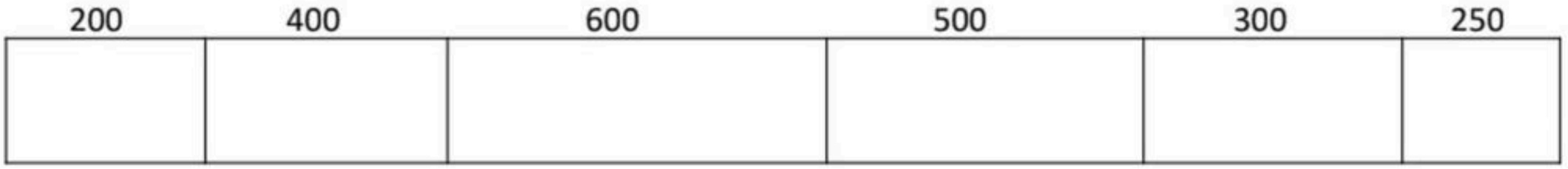

The Big Cason Family Are Expecting Their 17th Child

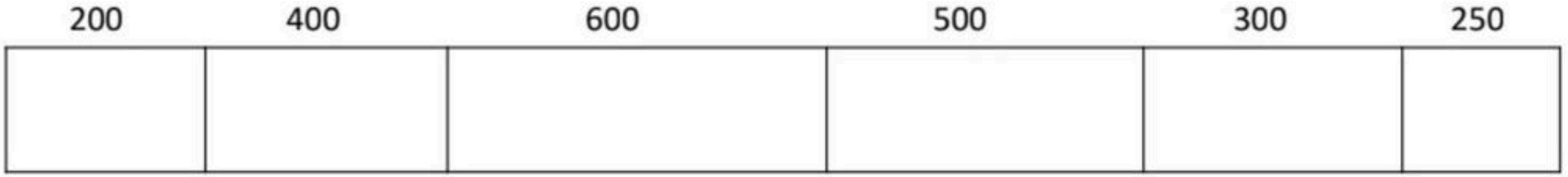
- **Internal fragmentation:** - Internal fragmentation is a function of fixed size partition which means, when a partition is allocated to a process. Which is either the same size or larger than the request then, the unused space by the process in the partition Is called as internal fragmentation

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







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)

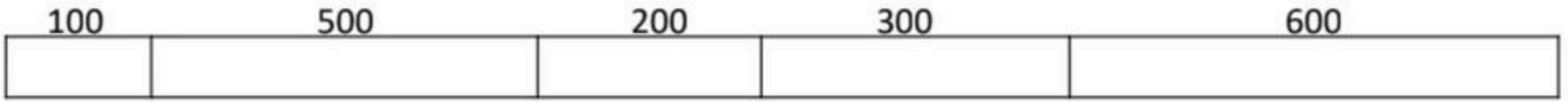
(b) first fit but not best fit policy

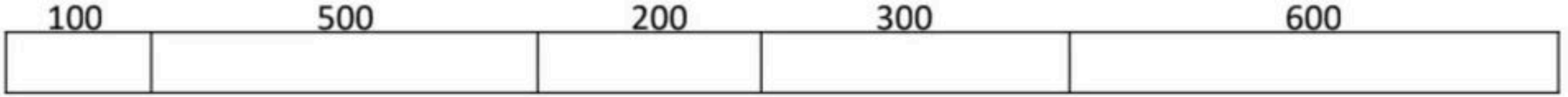
(c) best fit but first fit policy

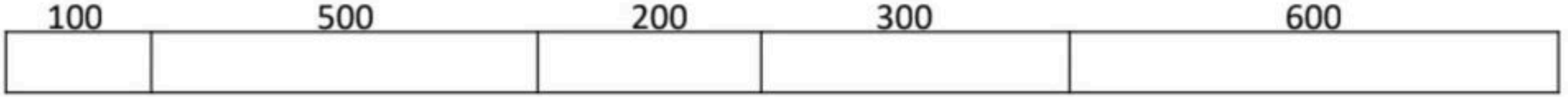
(d) None of the above

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







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

Q A 1000 Kbyte memory is managed using variable partitions but No 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

- How can we solve external fragmentation?
  - 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(Compaction, defragmentation). 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

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)

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

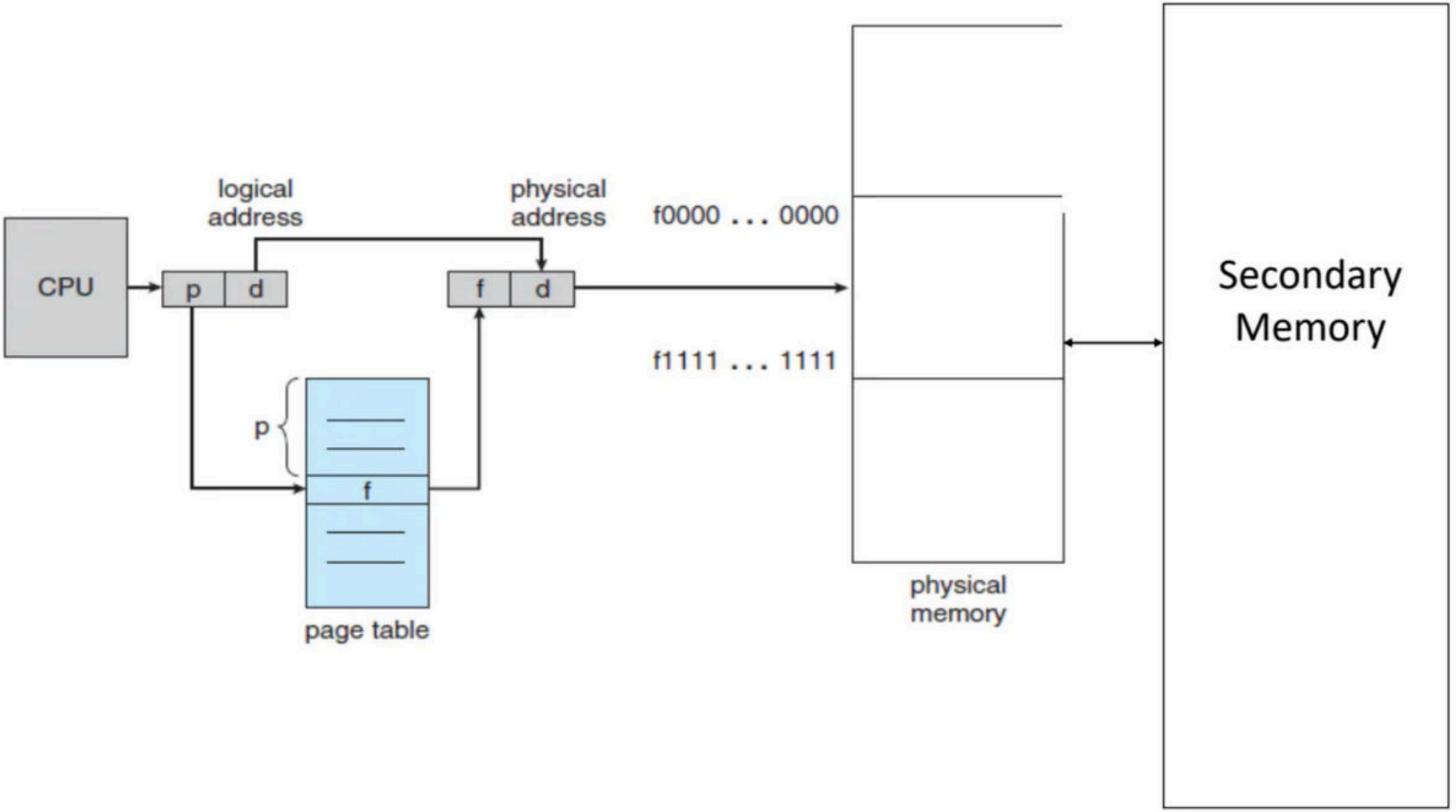
(D) Buddy

Break

Non-Contiguous Memory allocation(Paging)

- Paging is a memory-management scheme that permits the physical address space of a process to be non-contiguous.

- Paging avoids external fragmentation

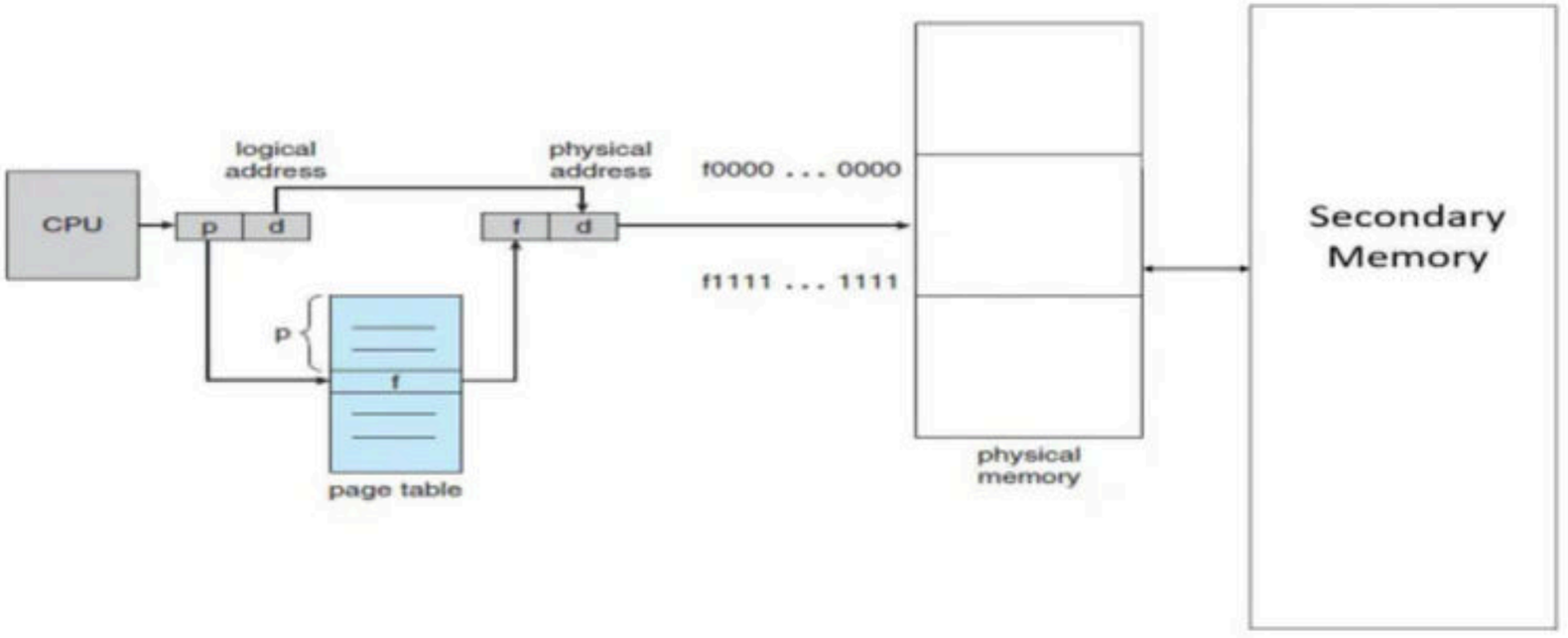

CPU
logical address
p
d
physical address
f
d
f0000 . . . 0000
f1111 . . . 1111
p
f
page table
physical memory
Secondary Memory

- Secondary memory is divides into fixed size partition(because management is easy) all of them of same size called pages(easy swapping and no external fragmentation).

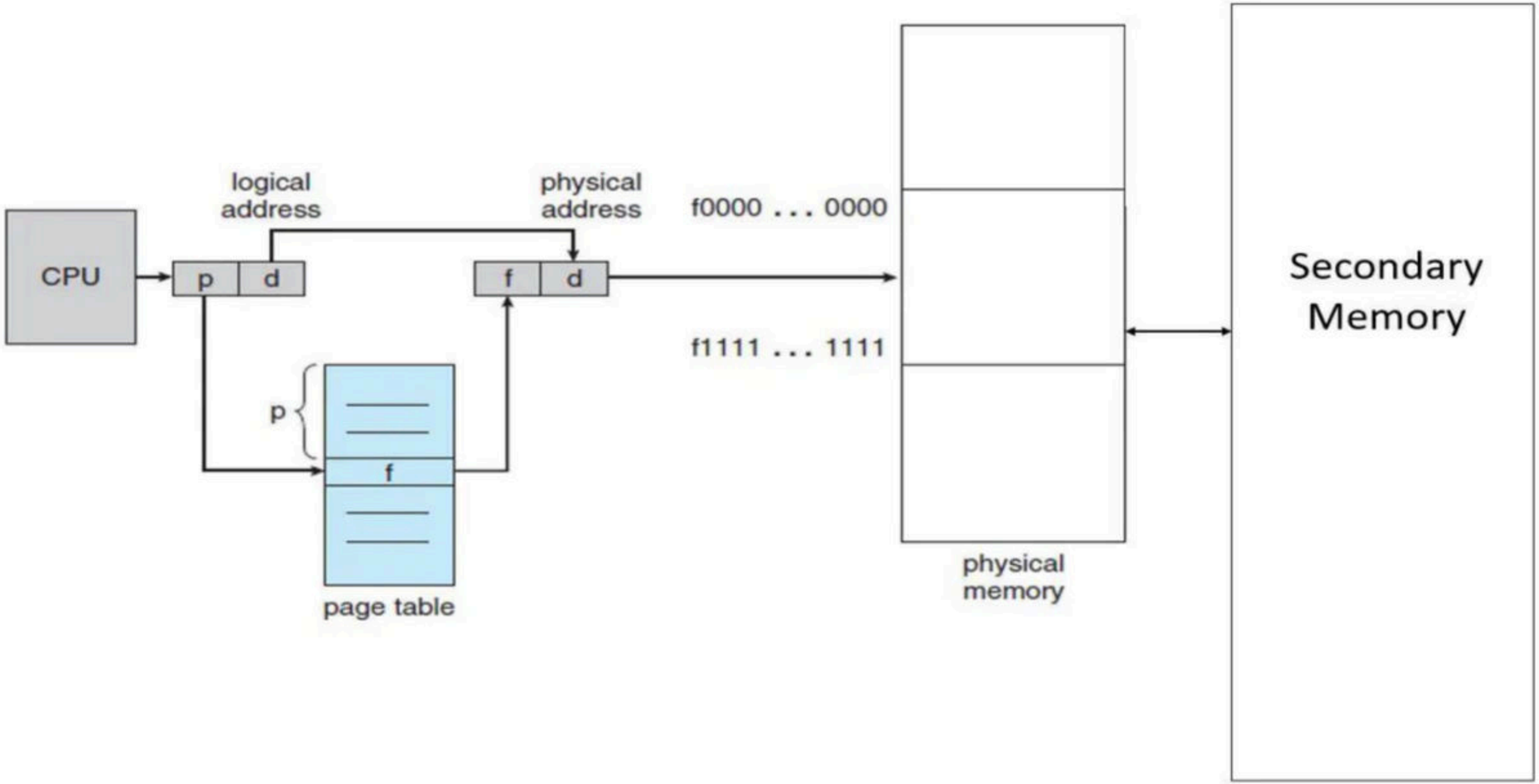
- Main memory is divided into fix size partitions (because management is easy), each of them having same size called frames(easy swapping and no external fragmentation).

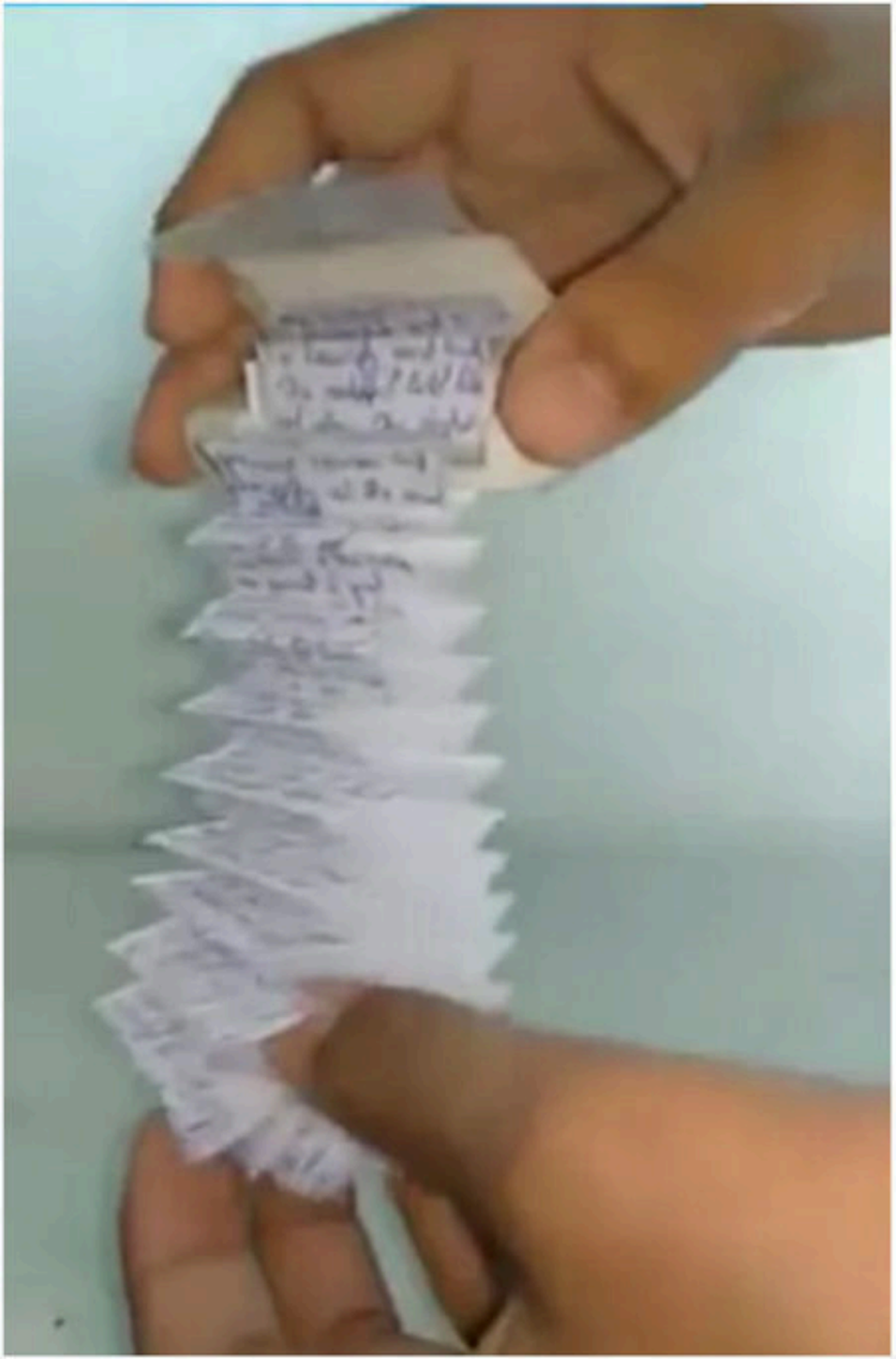
- Size of frame = size of page

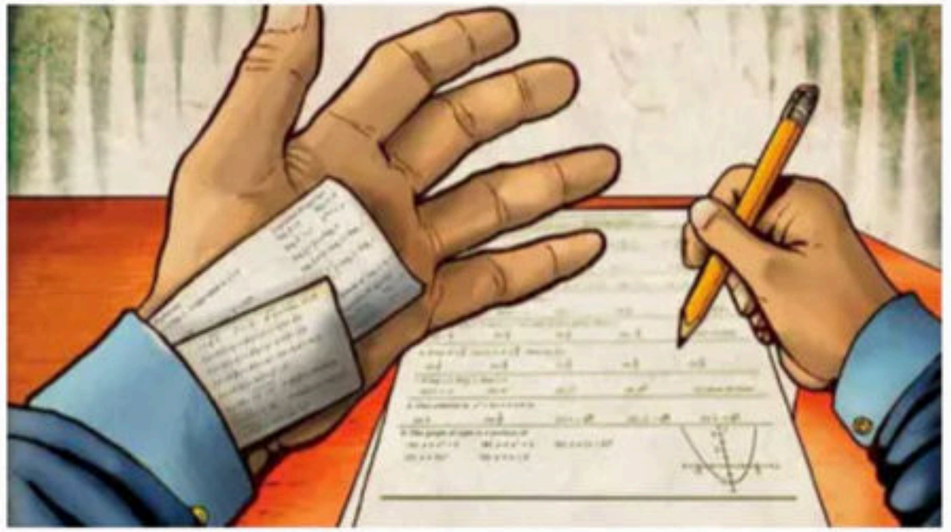
- In general number of pages are much more than number of frames (approx. 128 time)



Break


CPU
logical address
p
d
physical address
f
d
f0000 . . . 0000
f1111 . . . 1111
p
f
page table
physical memory
Secondary Memory





Translation process

1. CPU generate a logical address is divided into two parts - **p** and **d**
    1. where p stands for page no and d stands for instruction offset.

2. The **page number(p)** is used as an index into a **Page table**

3. **Page table base register(PTBR)** provides the base of the page table and then the corresponding page no is accessed using p.

4. Here we will finds the corresponding frame no (the base address of that frame in main memory in which the page is stored)

5. Combine corresponding frame no with the instruction offset and get the physical address. Which is used to access main memory.

Break

Page Table

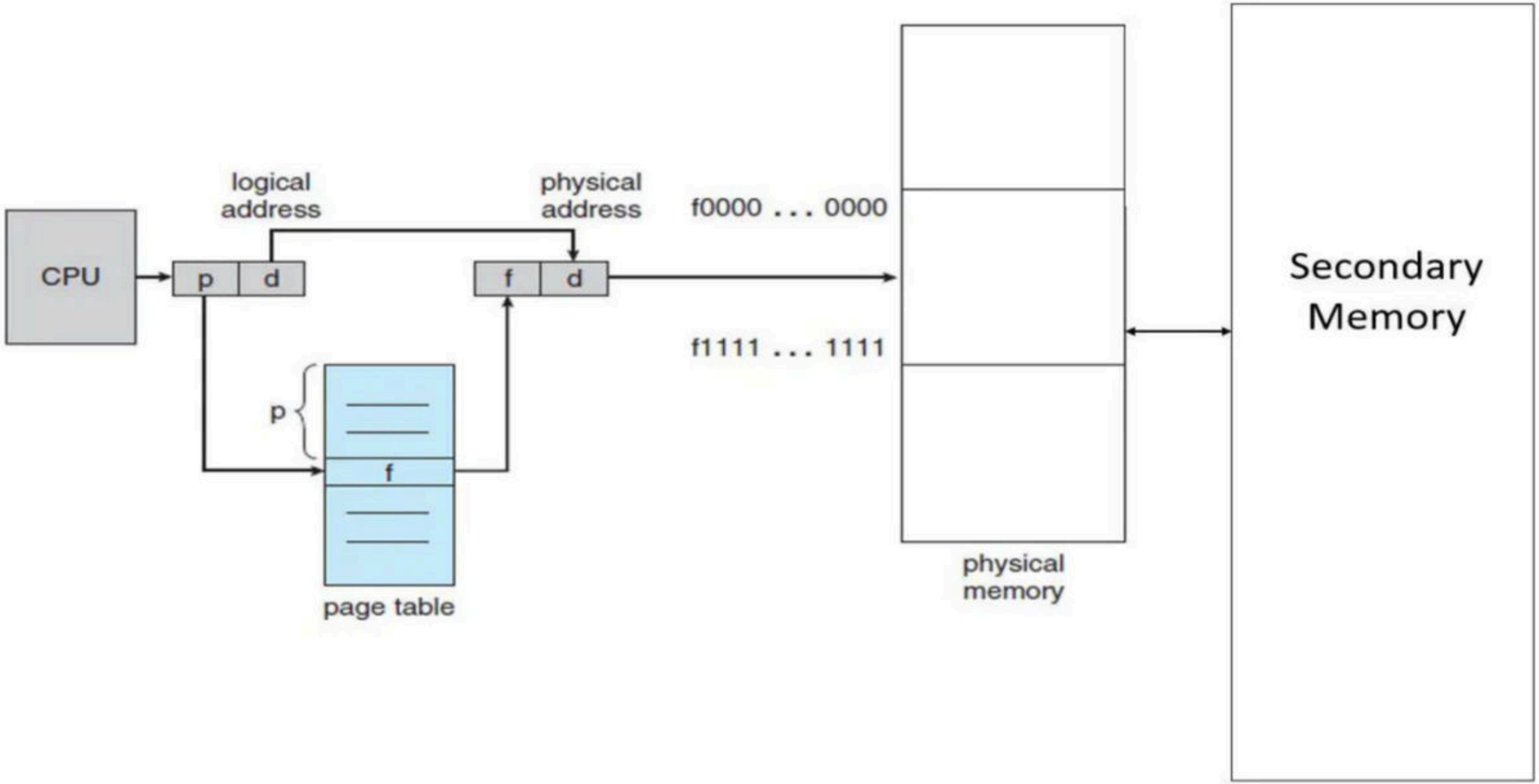
1. Page table is a data structure not hardware.

2. Every process have a separate page table.

3. Number of entries a process have in the page table is the number of pages a process have in the secondary memory.

4. Size of each entry in the page table is same it is corresponding frame number.

5. Page table is a data structure which is it self stored in main memory.


CPU
logical address
p
d
physical address
f
d
f0000 . . . 0000
f1111 . . . 1111
p
f
page table
physical memory
Secondary Memory

Advantage

- Removal of External Fragmentation

Disadvantage

- Translation process is slow as Main Memory 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.

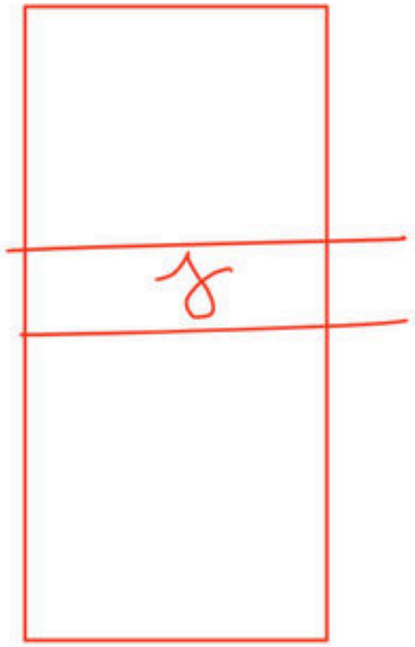
Q Page information in memory is also called as Page Table. The essential contents in each entry of a page table is/are _____. **(NET-JULY-2018)**

(a) Page Access information

(b) Virtual Page number

(c) Page Frame number

(d) Both virtual page number and Page Frame Number



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

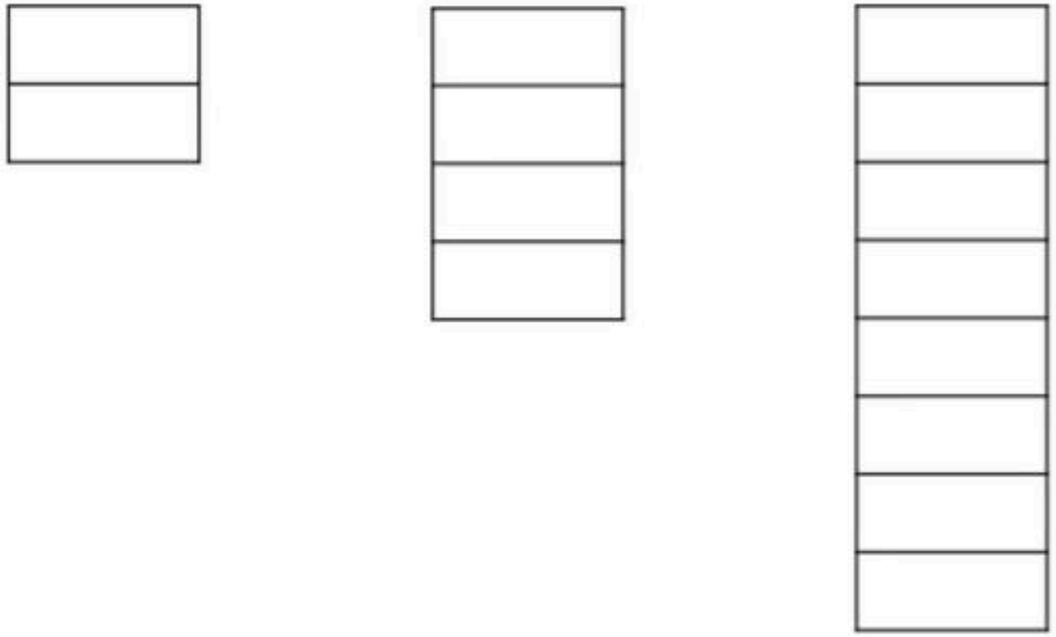
Break

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

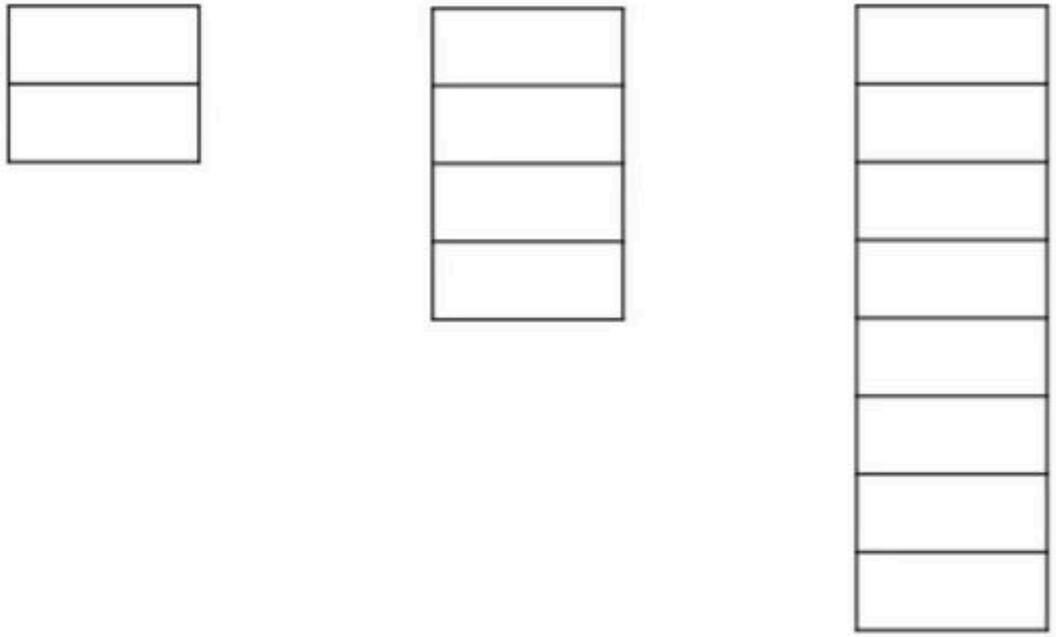
10^3	1 kilo
10^6	1 Mega
10^9	1 Giga
10^{12}	1 Tera
10^{15}	1 Peta
10^{18}	1 Exa
10^{21}	1 Zetta
10^{24}	1 Yotta

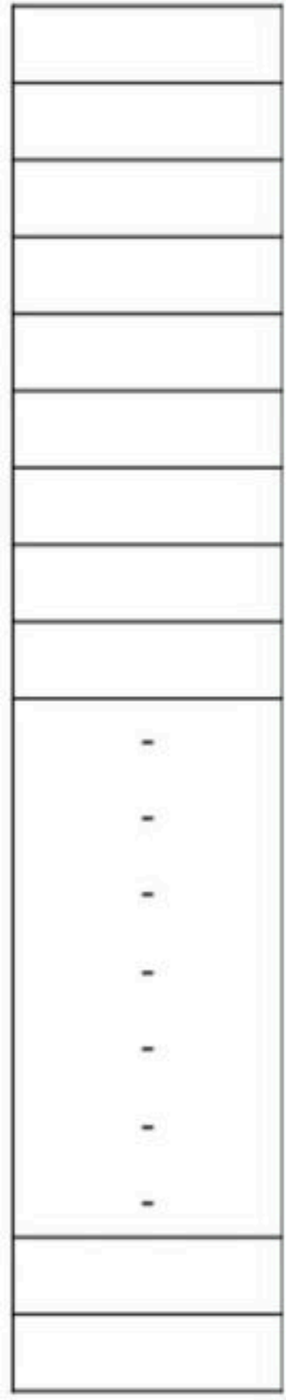
2^{10}	1 kilo
2^{20}	1 Mega
2^{30}	1 Giga
2^{40}	1 Tera
2^{50}	1 Peta
2^{60}	1 Exa
2^{70}	1 Zetta
2^{80}	1 Yotta

Address Length in bits	n
No of Locations	2^n



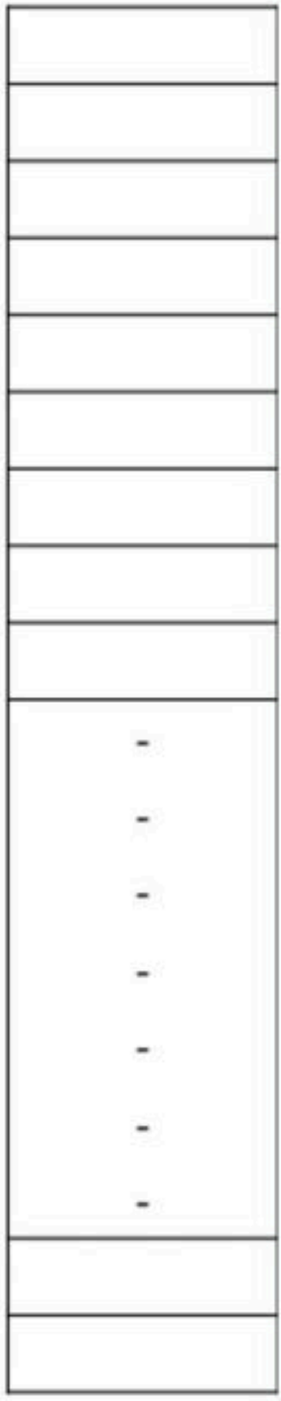
Address Length in bits	n
No of Locations	2^n



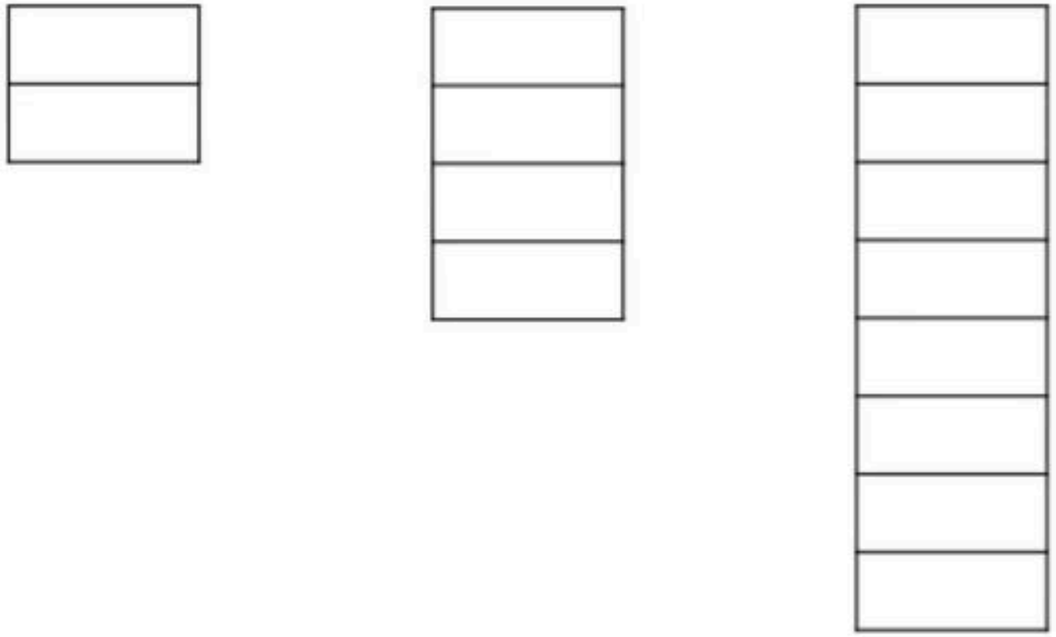


Memory Size = Number of Location * Size of each Location

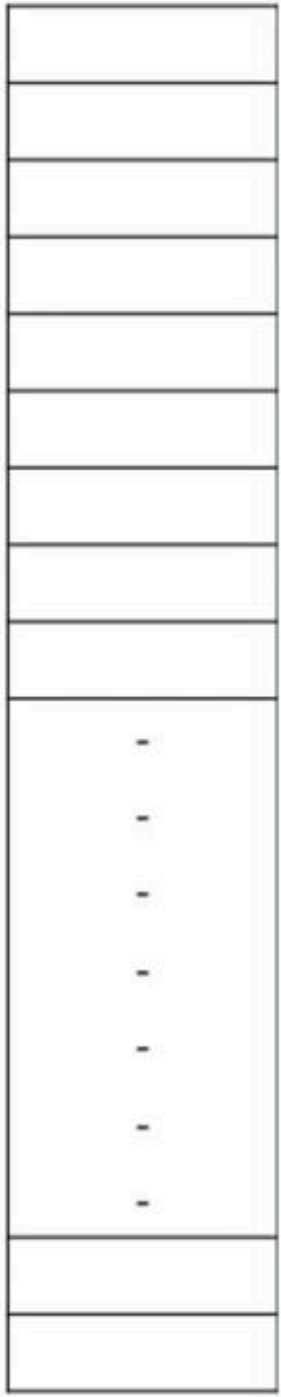
No of Locations	n
Address Length in bits	Upper Bound($Log_2 n$)



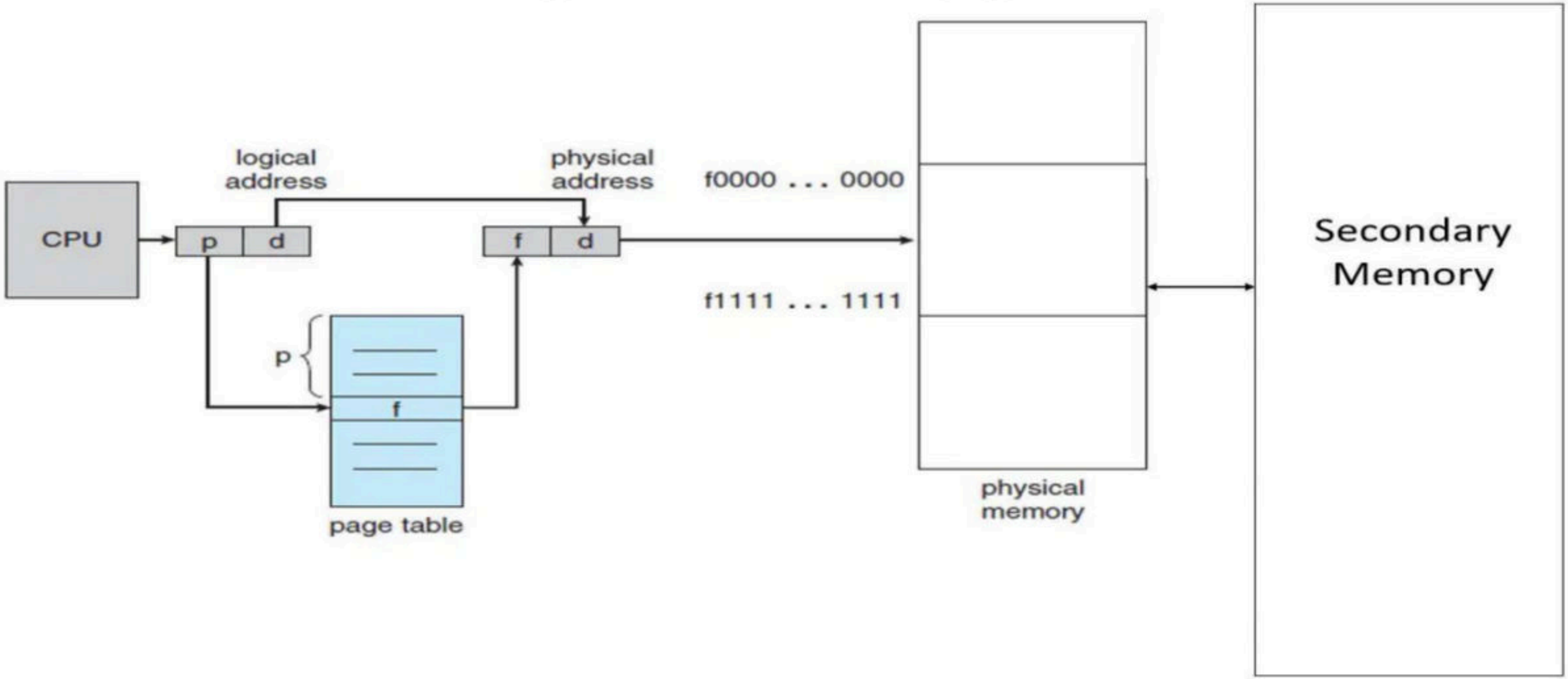
No of Locations	n
Address Length in bits	Upper Bound($Log_2 n$)



Memory Size/ Size of each Location = Number of Location



- Page Table Size = No of entries in Page table * Size of each entry(f)
- Process Size = No of Pages * Size of each page



Logical Address in bits

Secondary Memory

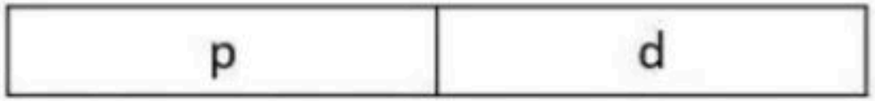
Physical Address in bits

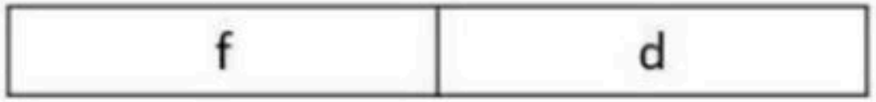
Main Memory

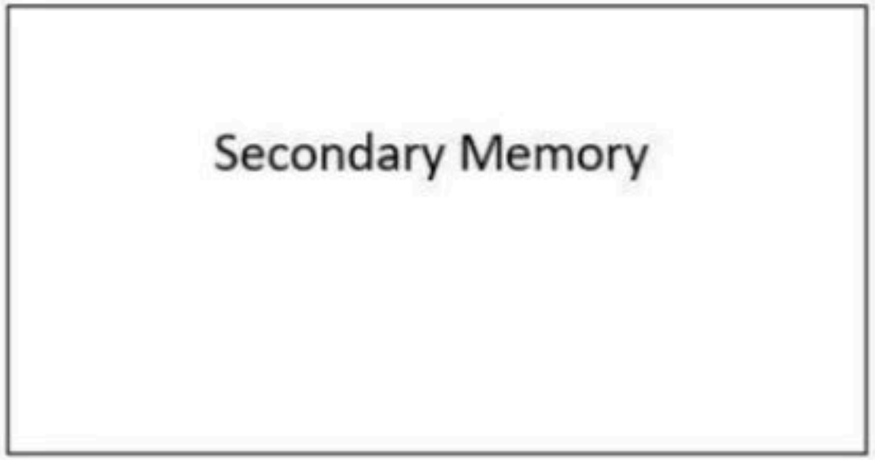
p

f

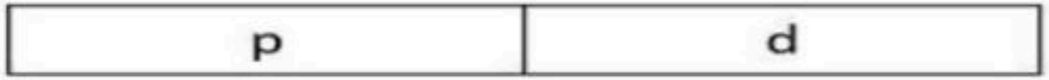
d

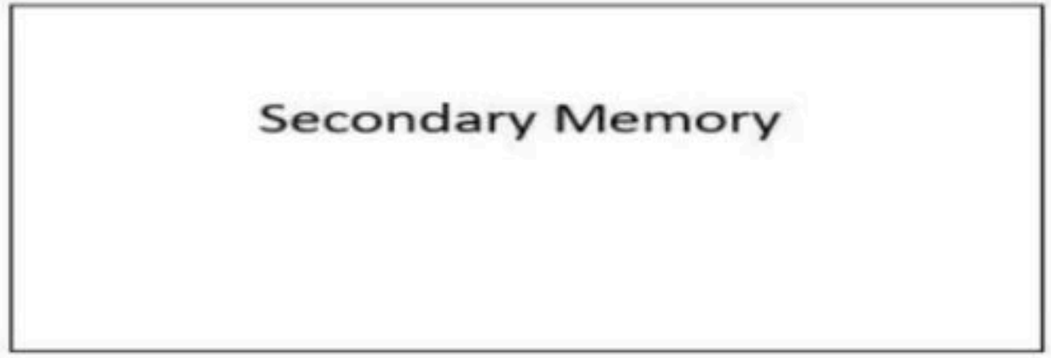


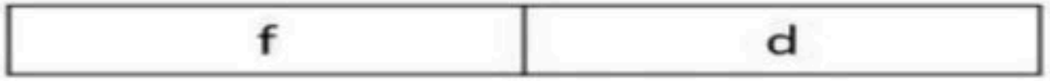


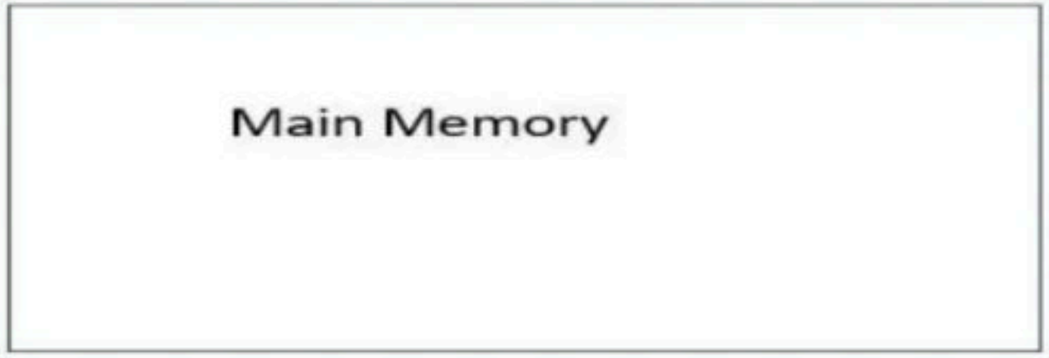


S No	SM	LA	MM	PA	p	f	d	addressable	Page Size
1	32 GB		128 MB					1B	1KB
2		42		33			11	1B	
3	512GB			31				1B	512B
4	128GB		32GB		30			1B	
5					28	14			4096B









Break

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

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)**

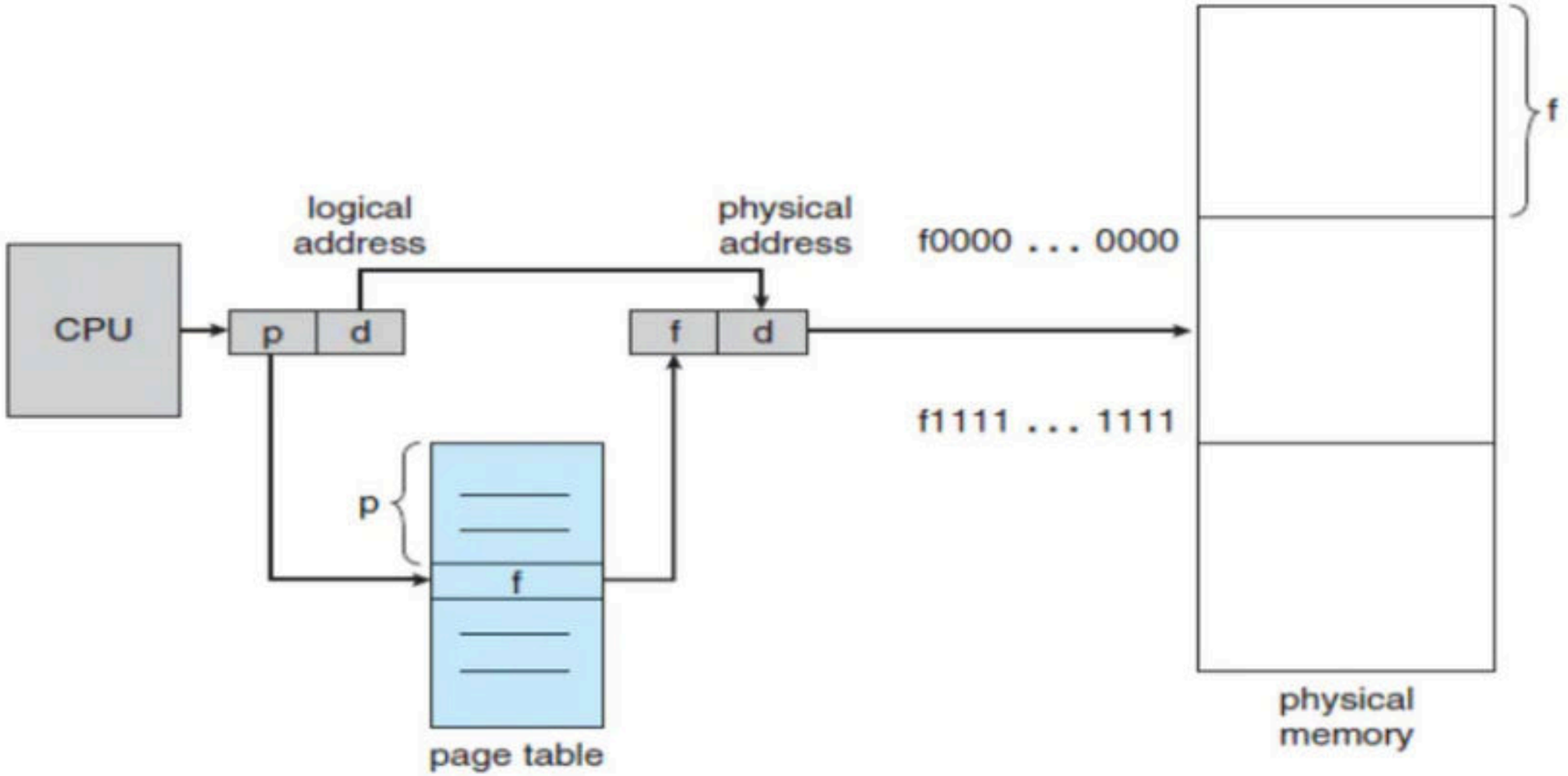


Figure 8.10 Paging hardware.

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)**

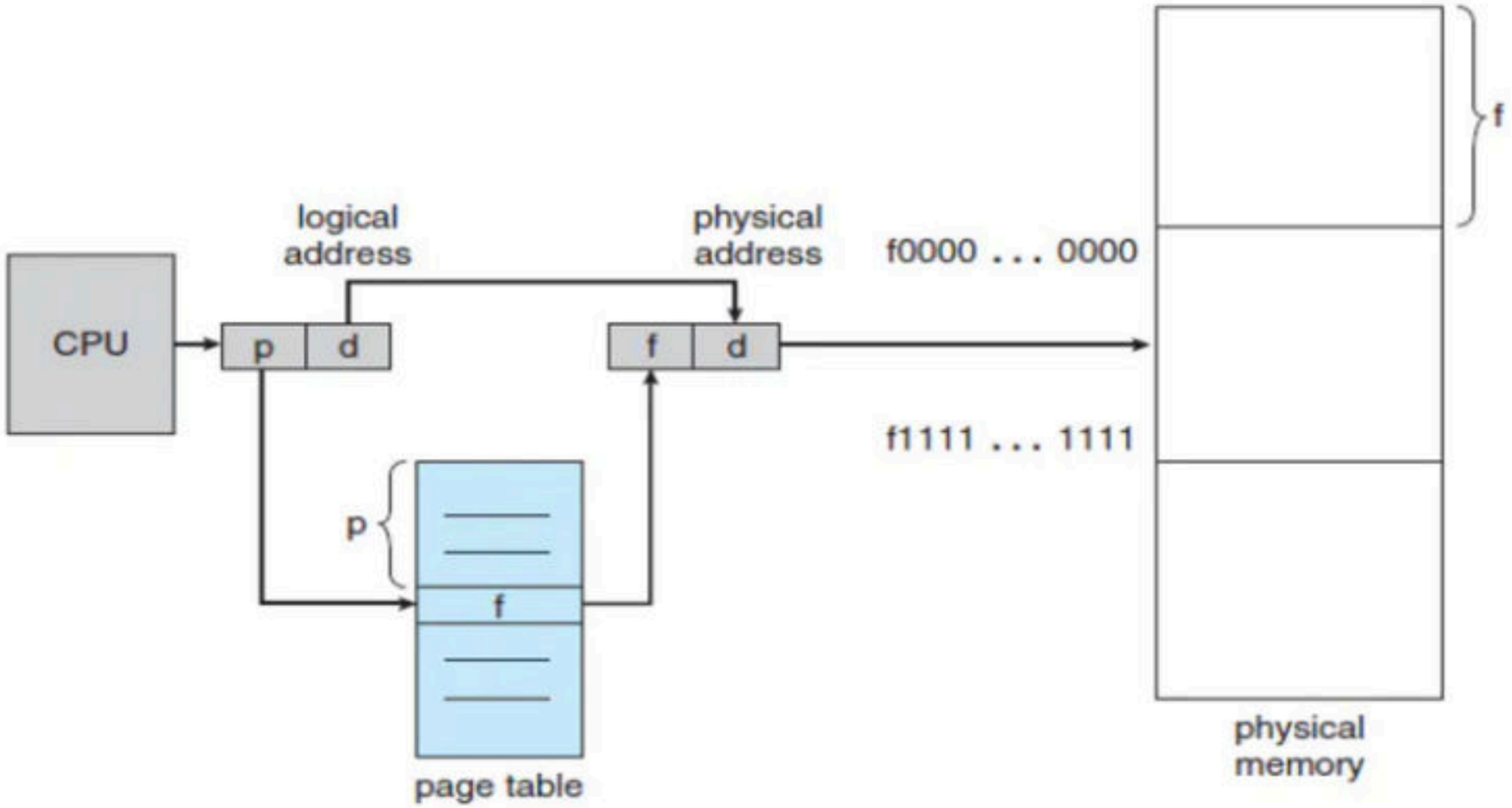


Figure 8.10 Paging hardware.

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.

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

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

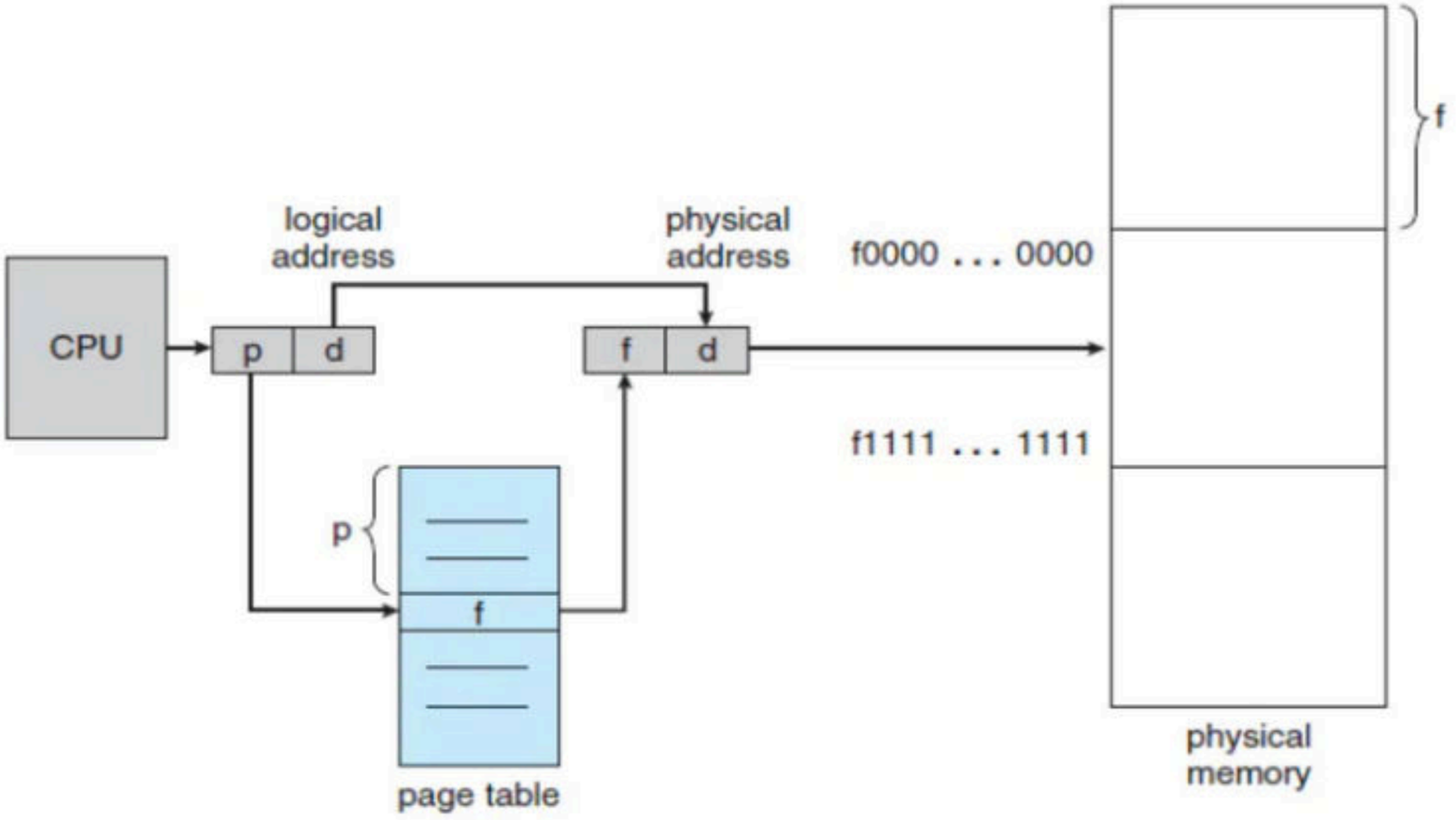


Figure 8.10 Paging hardware.

Break

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	

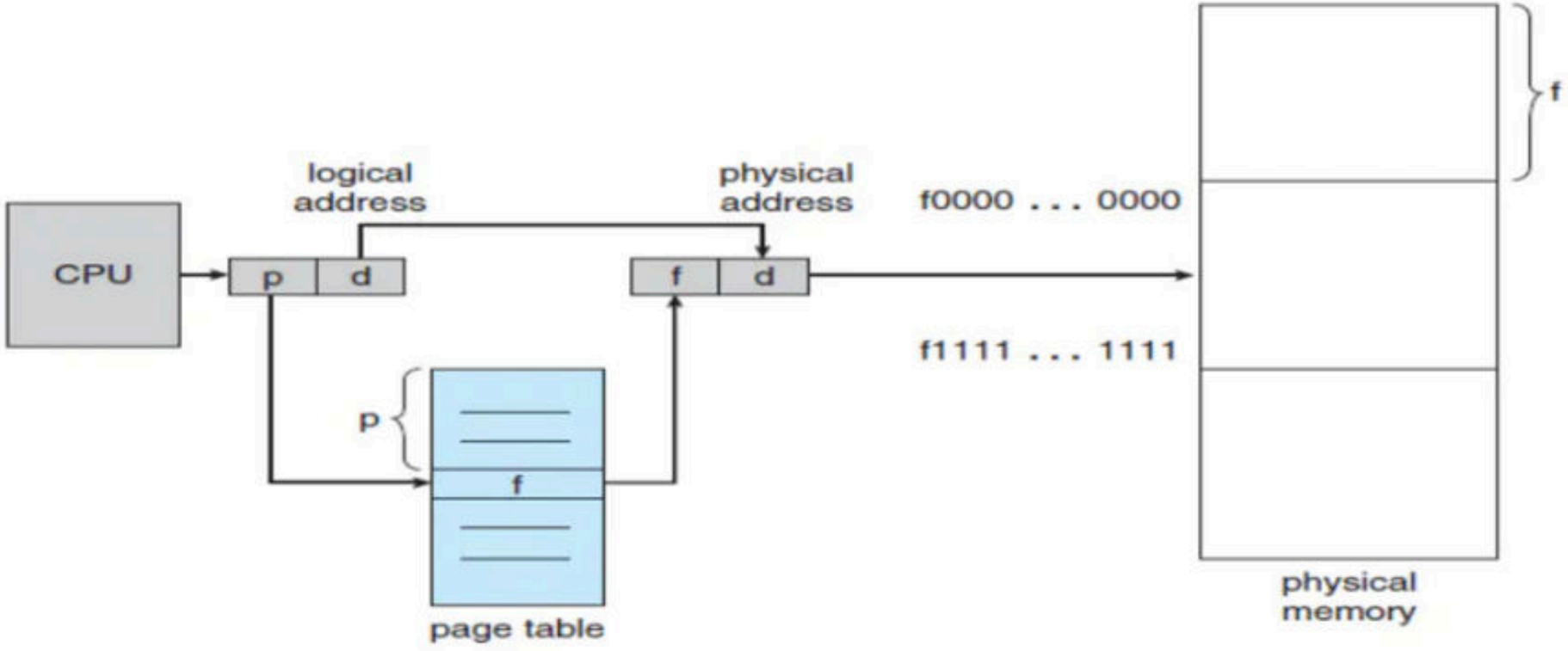
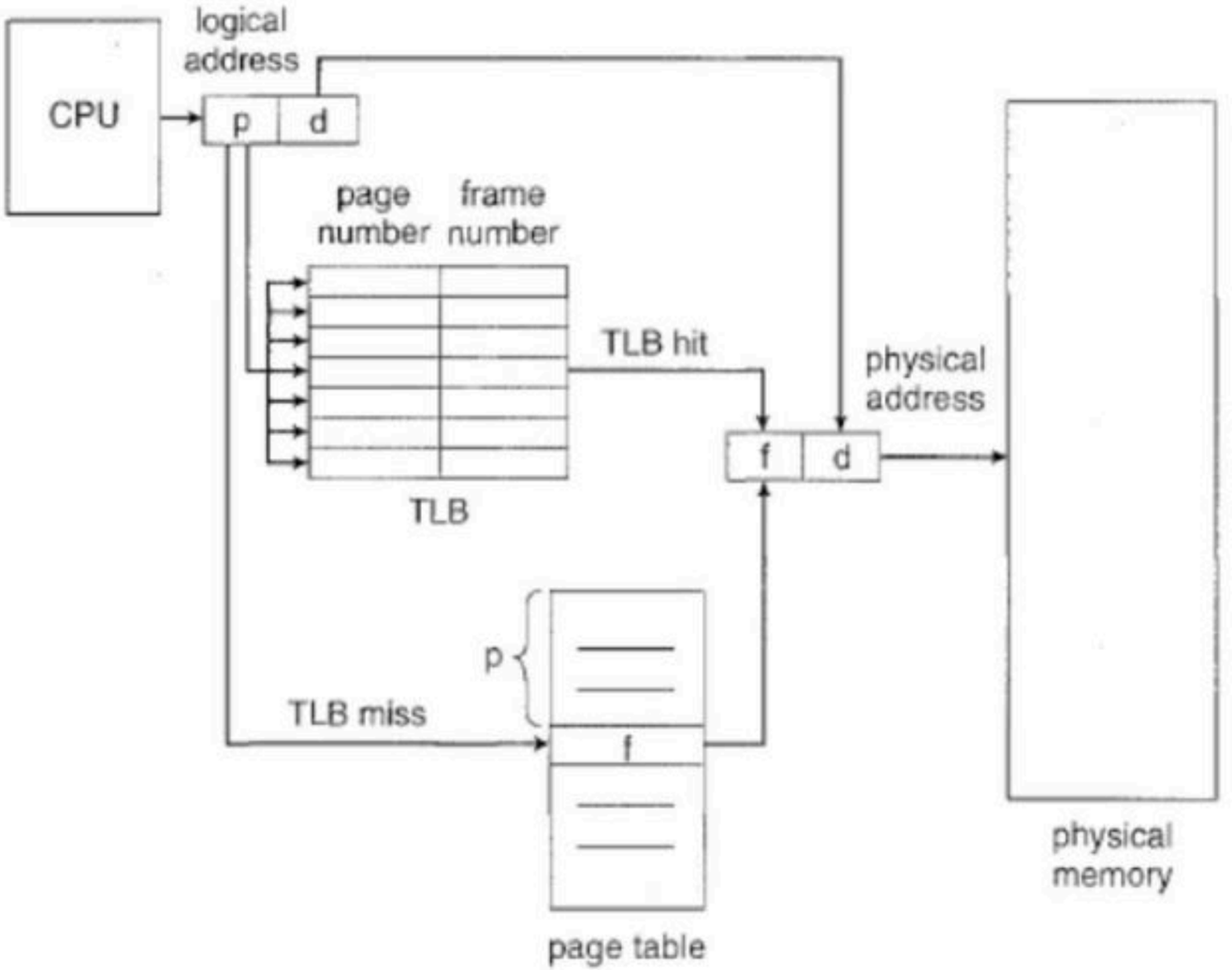


Figure 8.10 Paging hardware.

Break

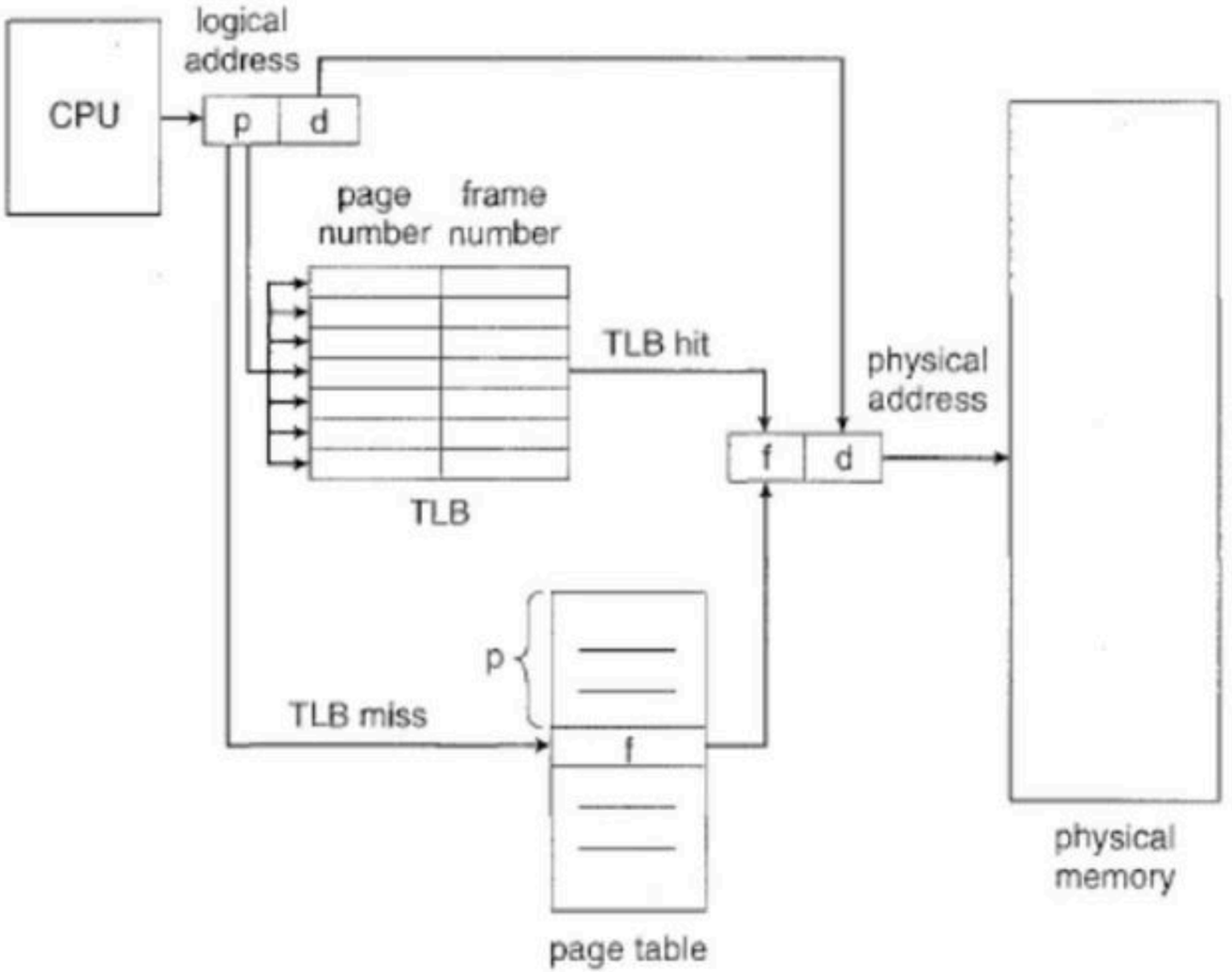
- A serious problem with page is, translation process is slow as page table is accessed two times (one for page table and other for actual access).

- To solve the problems in paging we take the help of TLB. The TLB is associative, high-speed memory.



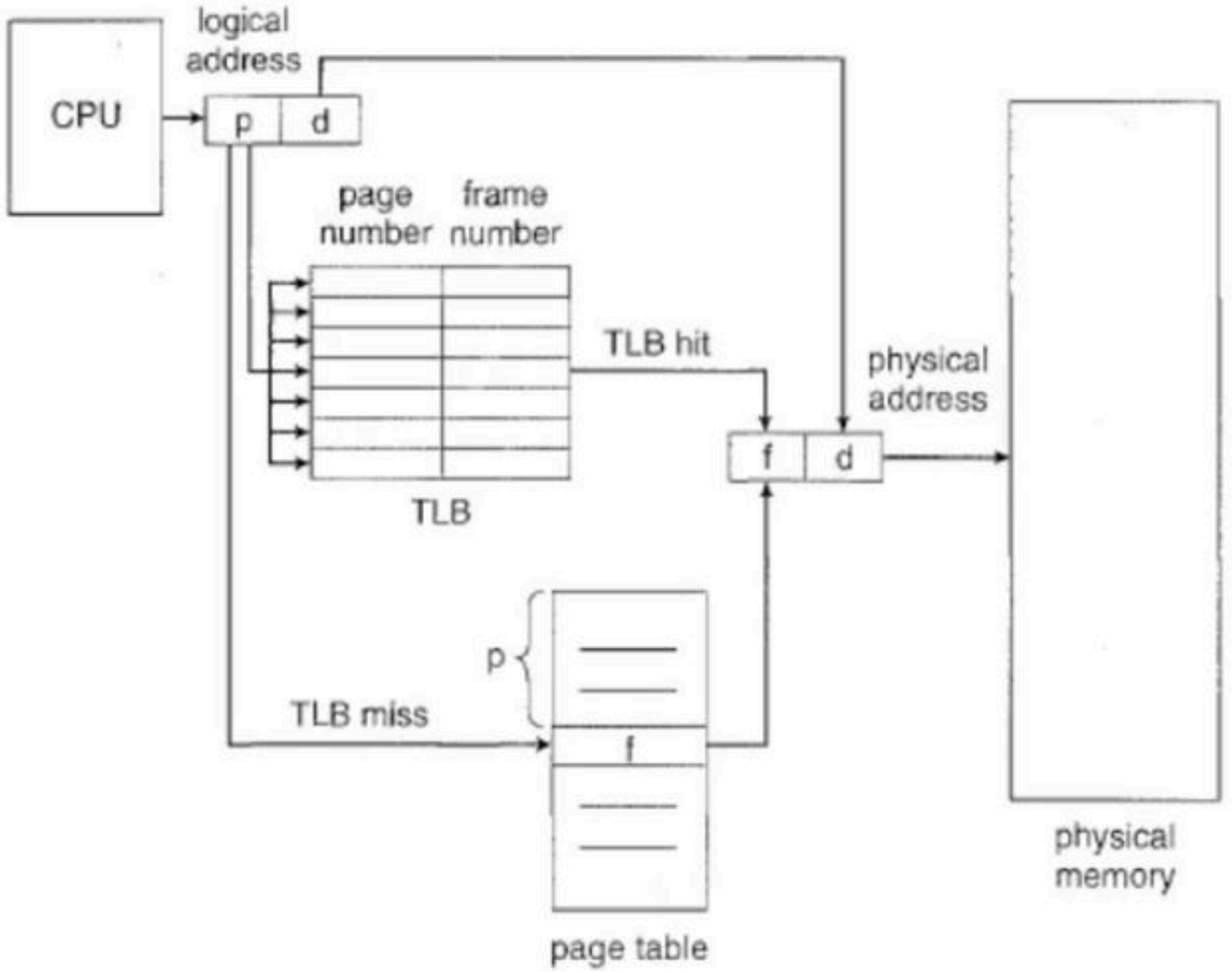
- Each entry in the TLB consists of two parts: a key (Page no) and a value (frame no). When the associative memory is search for page no, the page no is compared with all page no simultaneously. If the item is found, the corresponding frame no field is returned.

- 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.

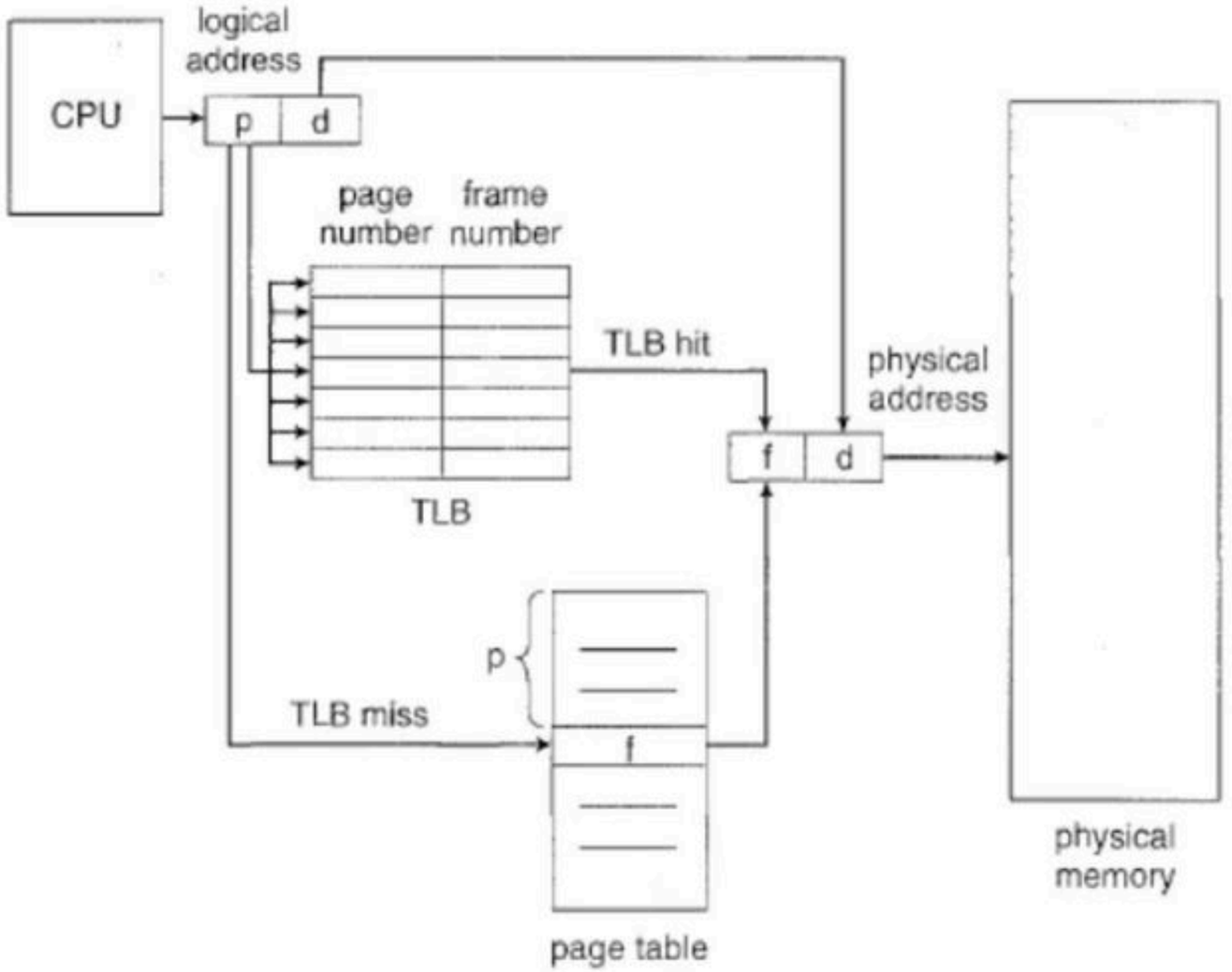
- If the page number is not in the TLB (known as a **TLB Miss**), then a memory reference to the page table must be made.



- Also we add the page number and frame number to the TLB, so that they will be found quickly on the next reference.

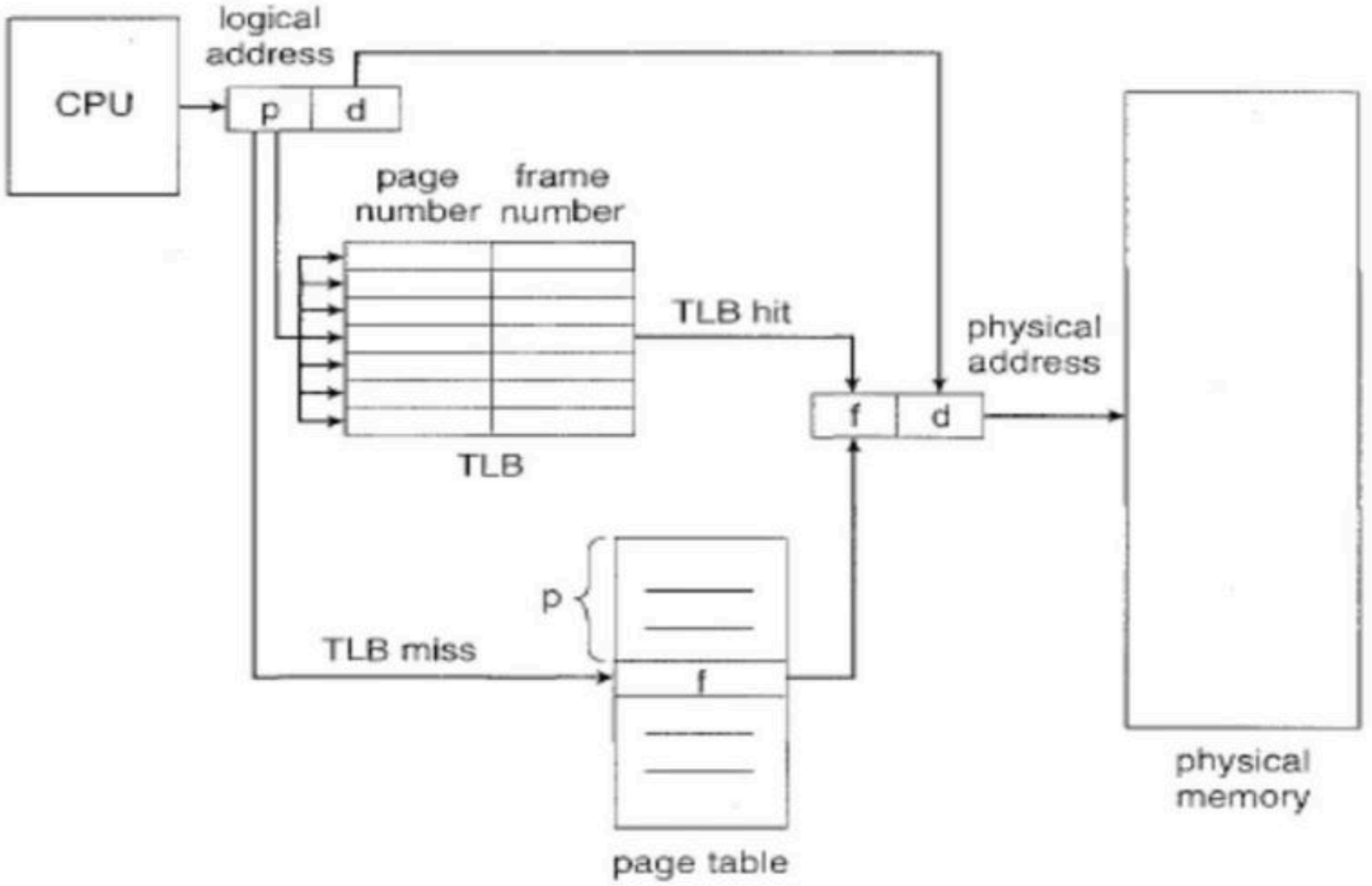
- If the TLB is already full of entries, the operating system must select one for replacement i.e. Page replacement policies.

- The percentage of times that a particular page number is found in the TLB is called the **Hit Ratio**.



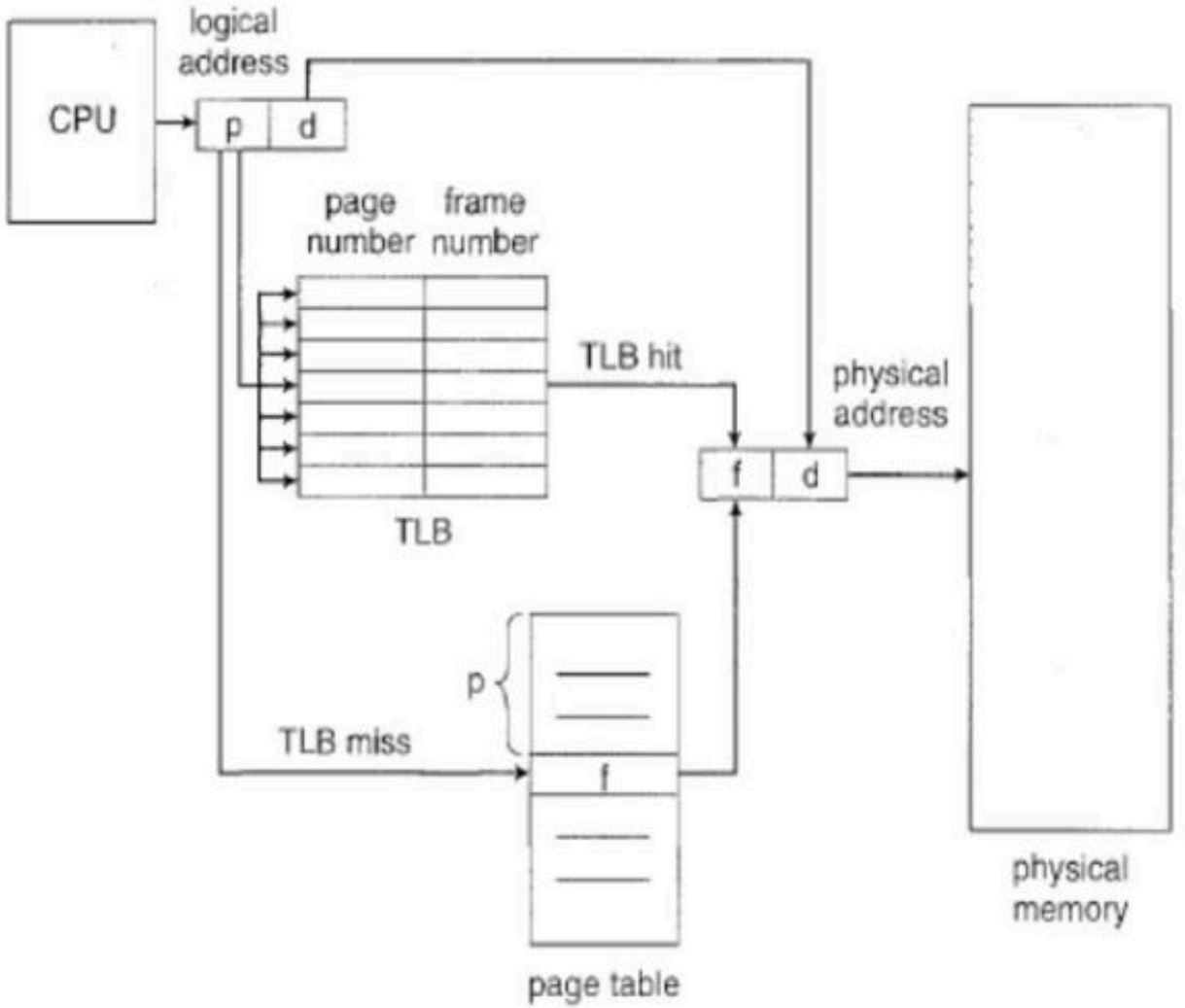
- **Effective Memory Access Time:**
  - Hit [TLB + Main Memory] + 1-Hit [TLB + 2 Main Memory]

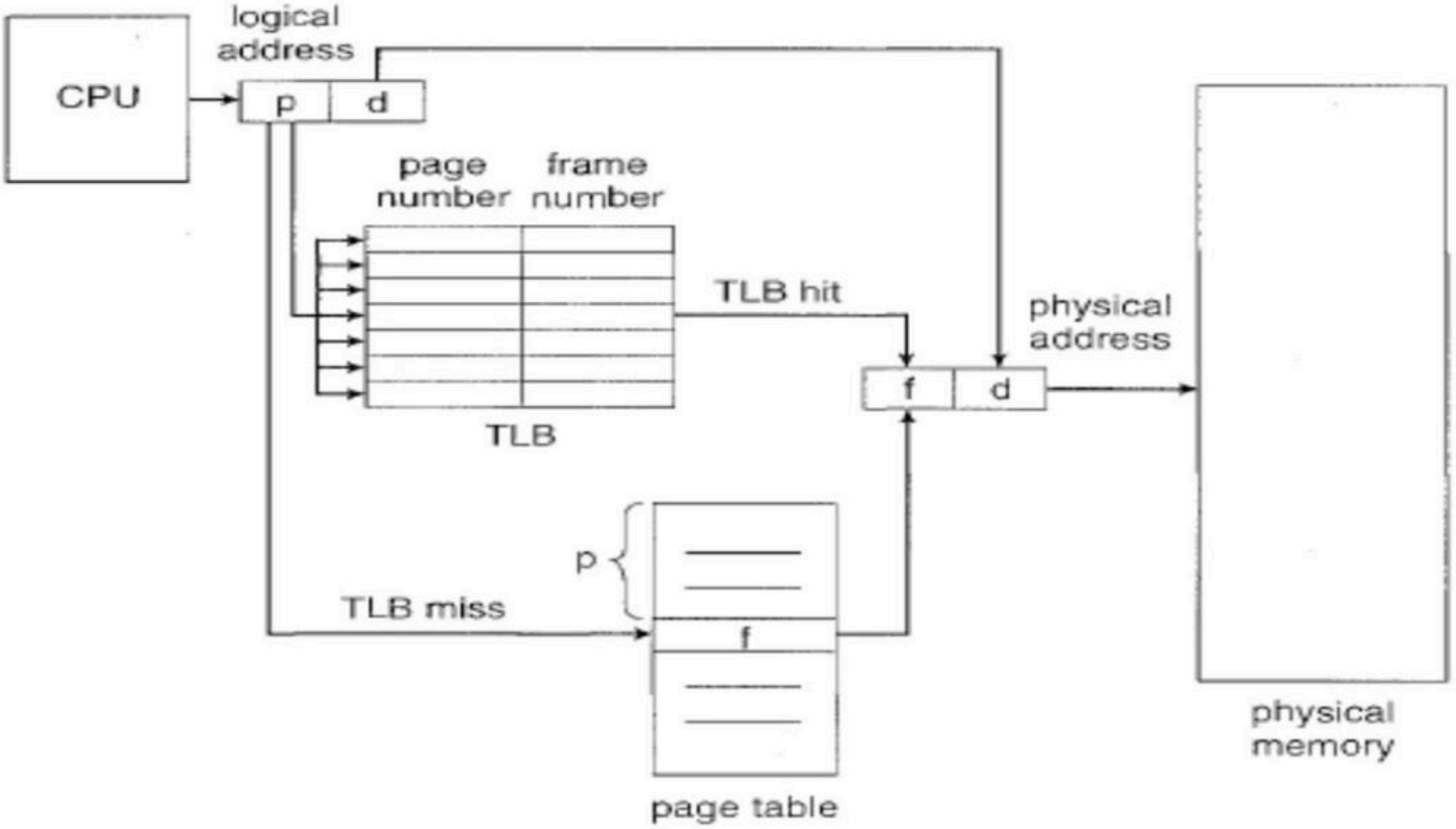
- **TLB** removes the problem of slow access.



- **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.




CPU
logical address
p
d
page number
frame number
TLB hit
TLB
f
d
physical address
physical memory
p
TLB miss
f
page table

Break

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 8kB and 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)**

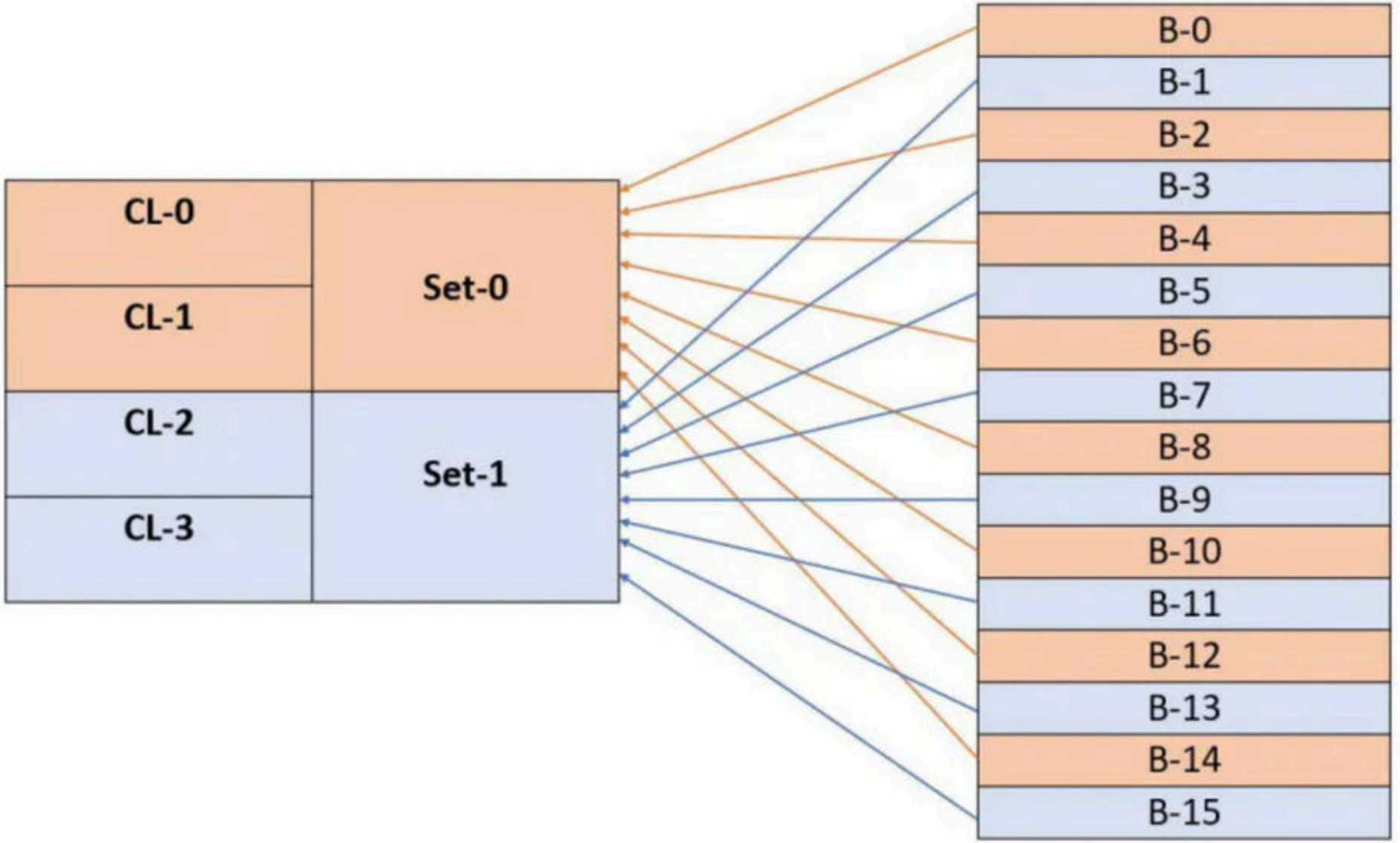
(A) 16×2^{10} **(B)** 8×2^{20} **(C)** 4×2^{20} **(D)** 256×2^{10}

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

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


CL-0
CL-1
CL-2
CL-3
Set-0
Set-1
B-0
B-1
B-2
B-3
B-4
B-5
B-6
B-7
B-8
B-9
B-10
B-11
B-12
B-13
B-14
B-15

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)**

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

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

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.

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

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

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

Break

Size of Page

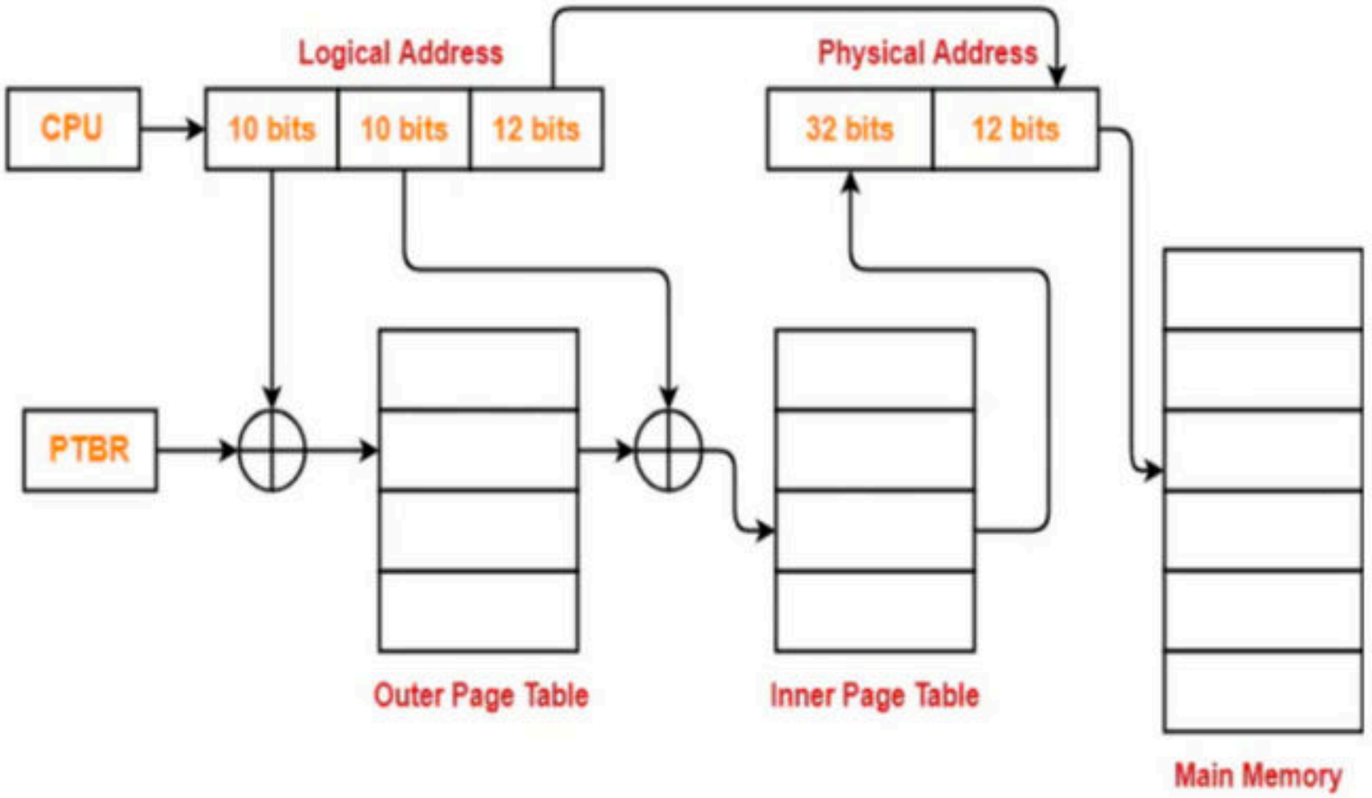
- If we increase the size of page table then internal fragmentation increase but size of page table decreases.
- If we decrease the size of page then internal fragmentation decrease but size of page table increases.
- So we have to find what should be the size of the page, where both cost are minimal.

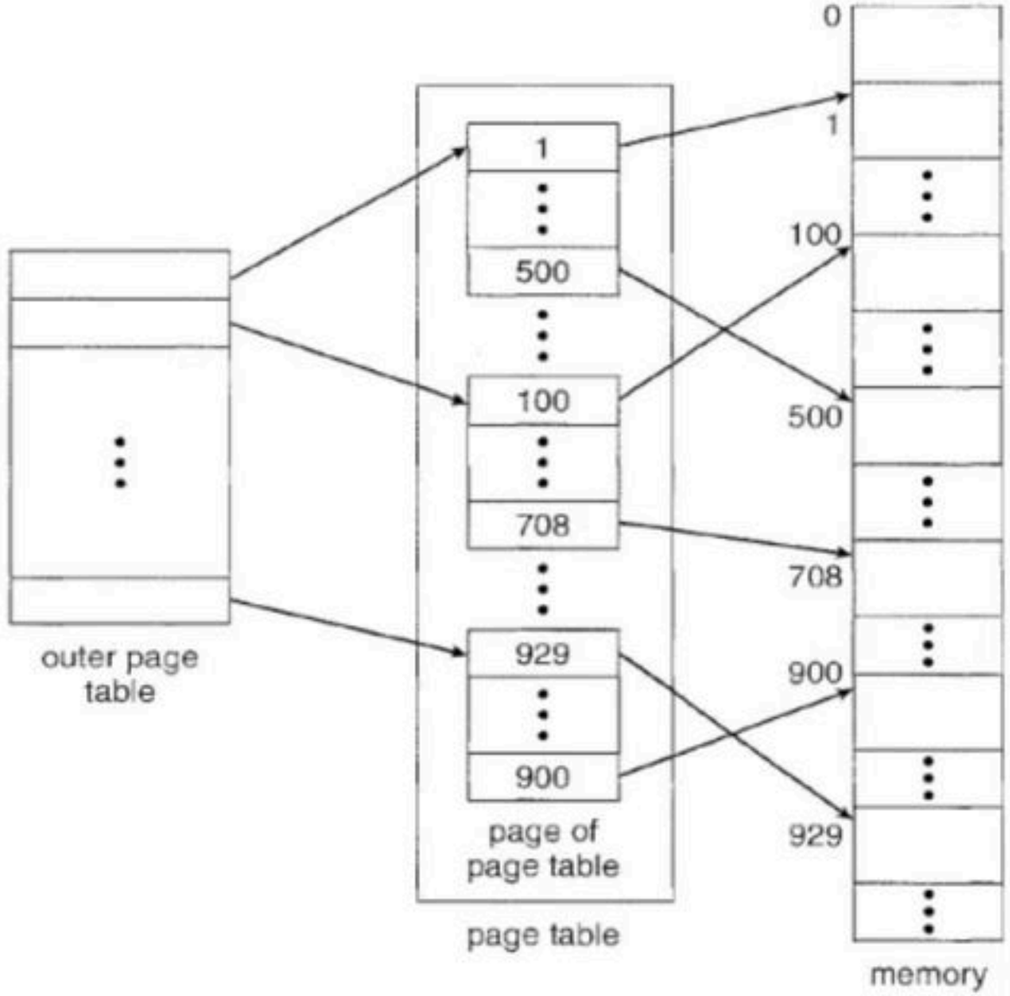
Q For the implementation of a paging scheme, suppose the average process size be x bytes, the page size be y bytes, and each page entry requires z bytes. The optimum page size that minimizes the total overhead due to the page table and the internal fragmentation loss is given by **(NET-DEC-2014)**

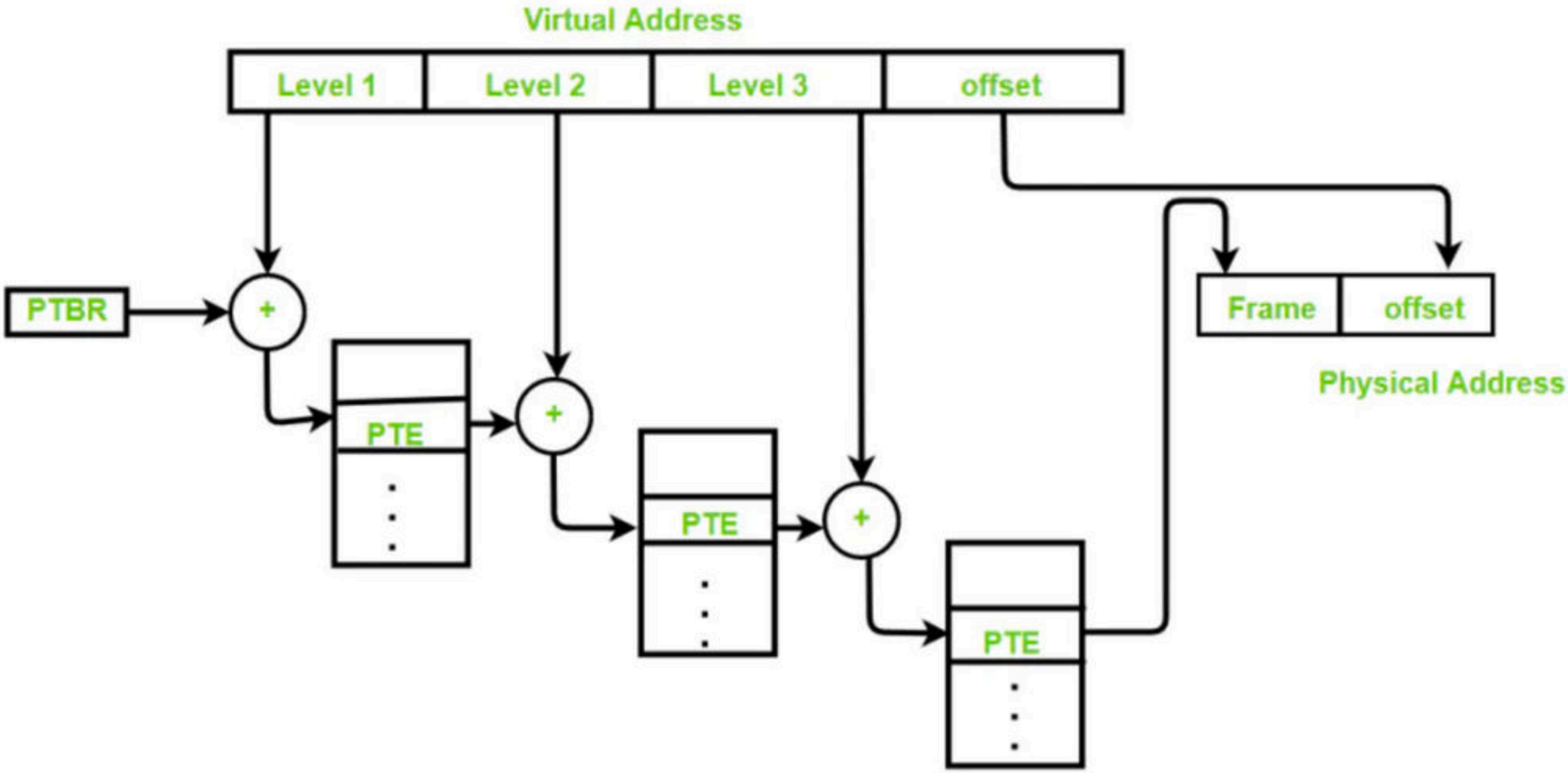
A) x/2 **b)** xz/2 **c)** root(2xz) **d)** root(xz)/2

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.







3 Level paging system

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.

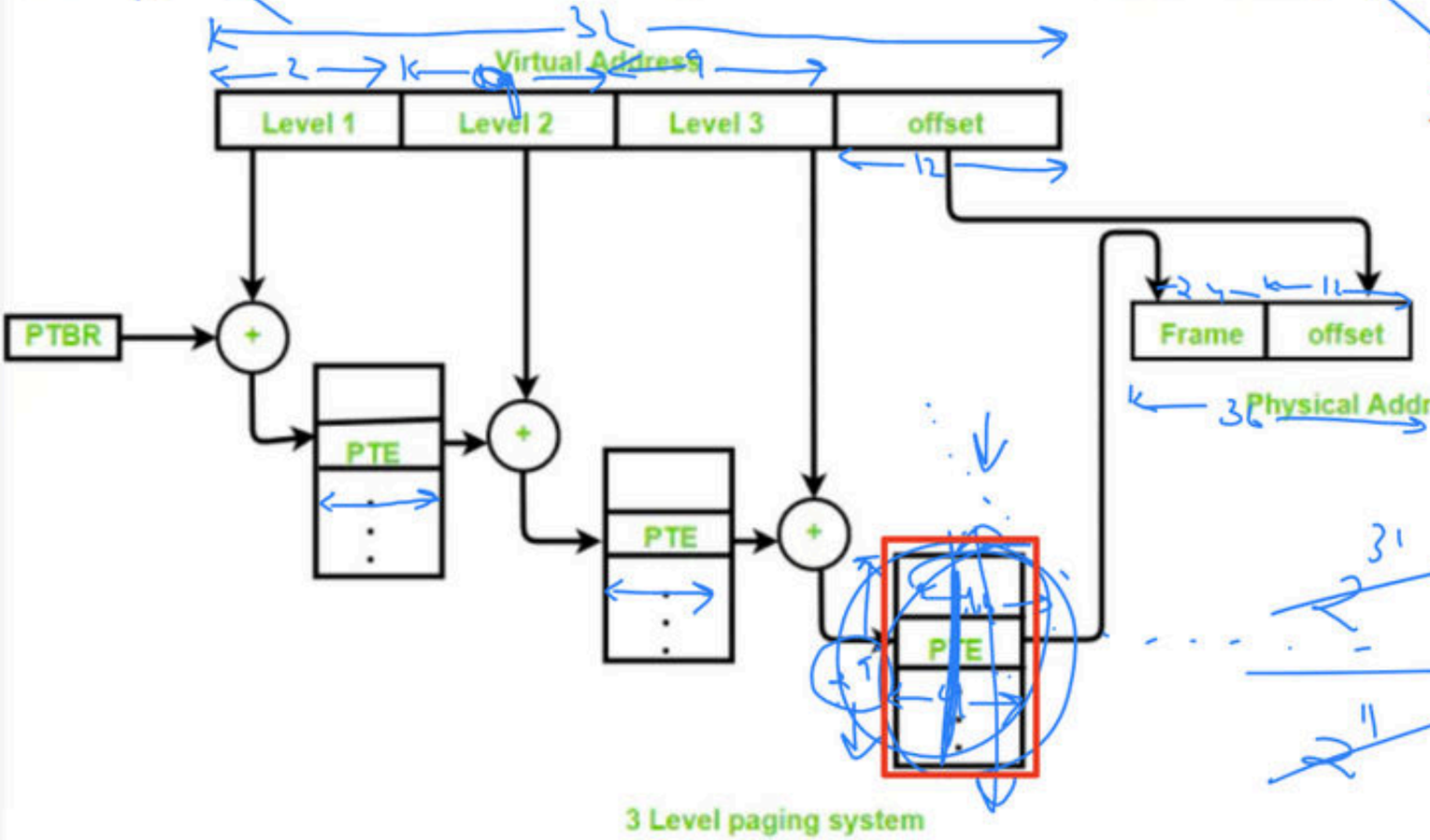
NCERT

R.S.A

RDS

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



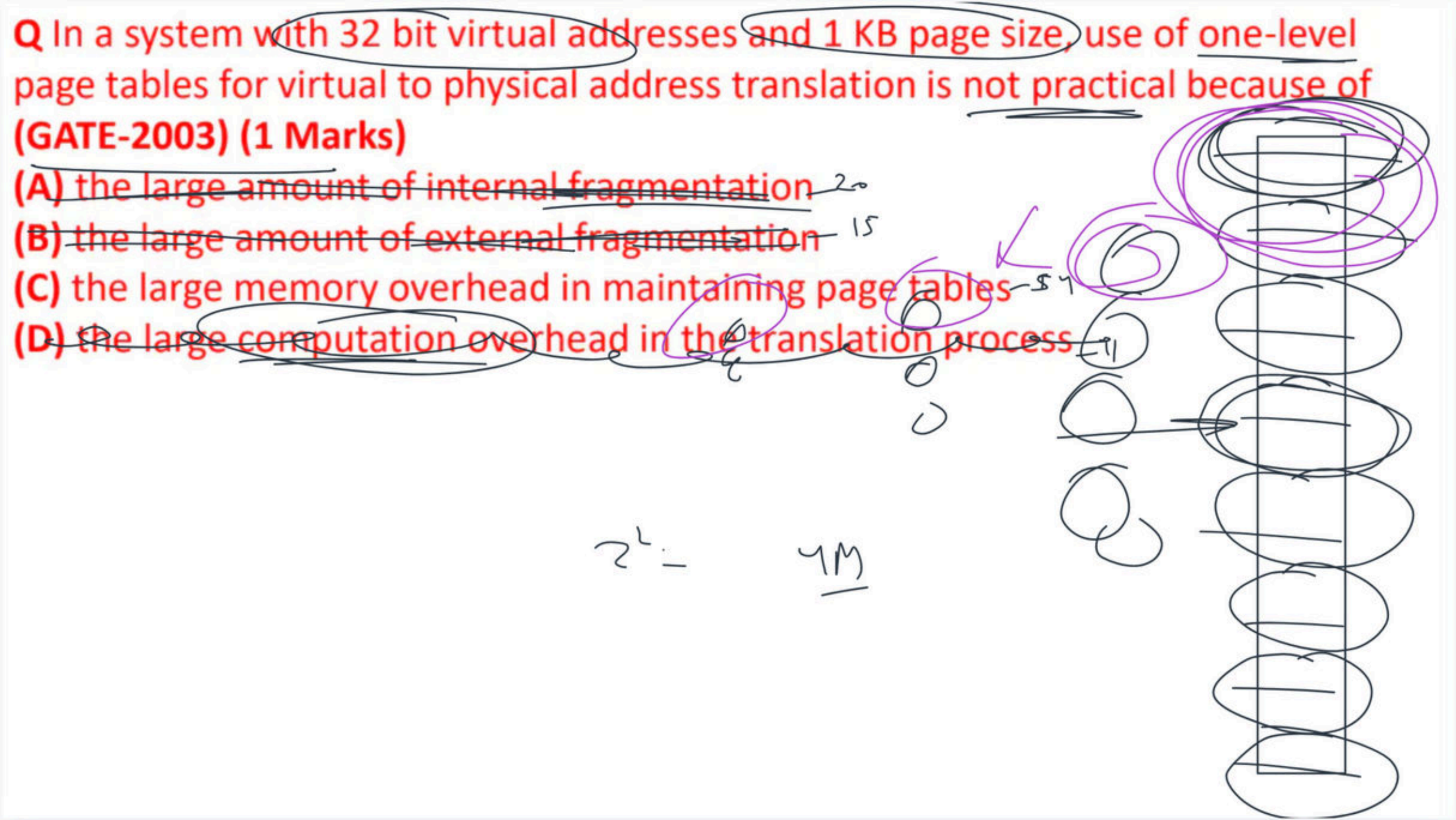
Q In a system with 32 bit virtual addresses and 1 KB page size, use of one-level page tables for virtual to physical address translation is not practical because of **(GATE-2003) (1 Marks)**

(A) the large amount of internal fragmentation

(B) the large amount of external fragmentation

(C) the large memory overhead in maintaining page tables

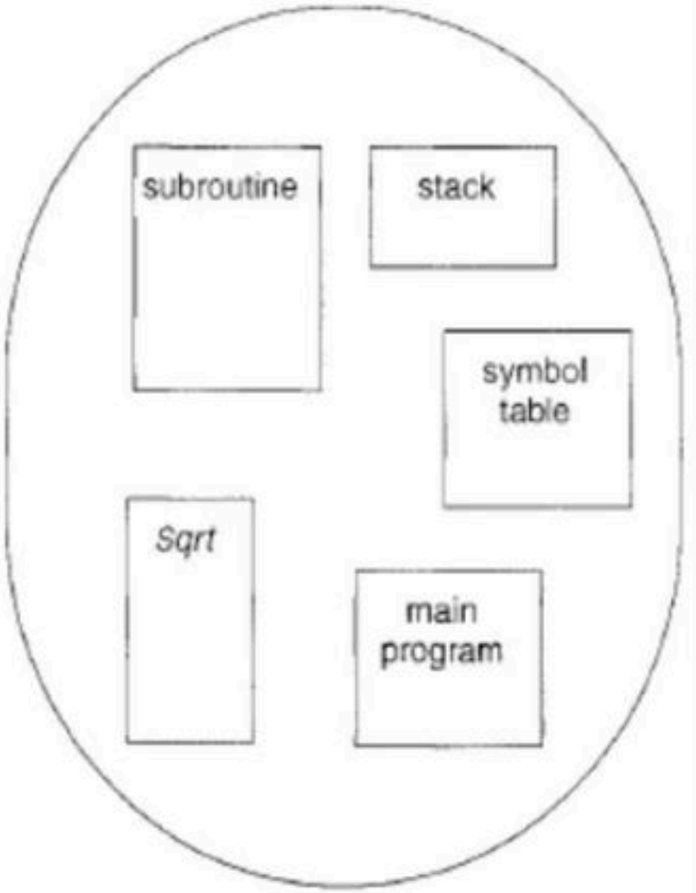
(D) the large computation overhead in the translation process



Break

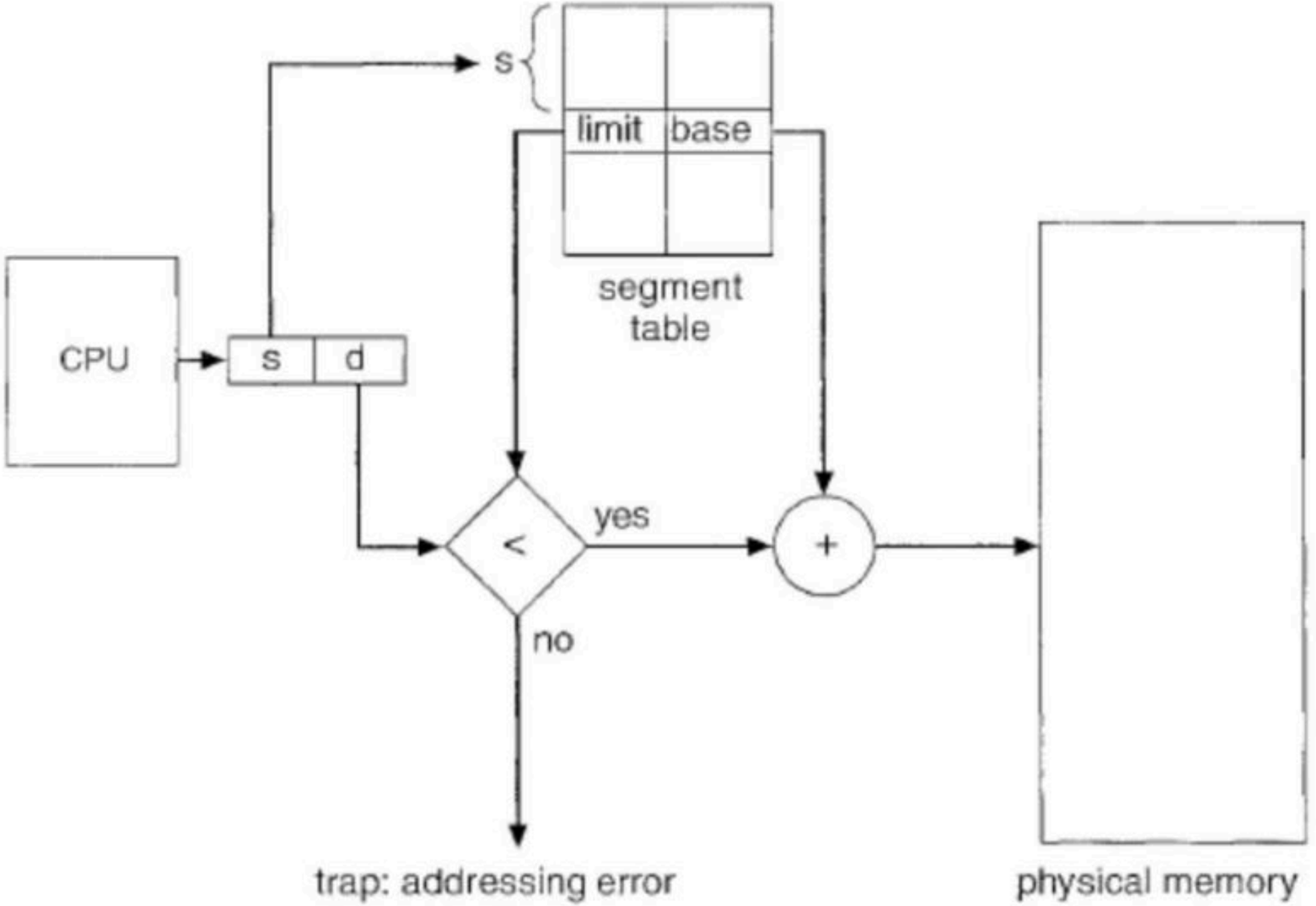
Segmentation

- 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.
- 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.



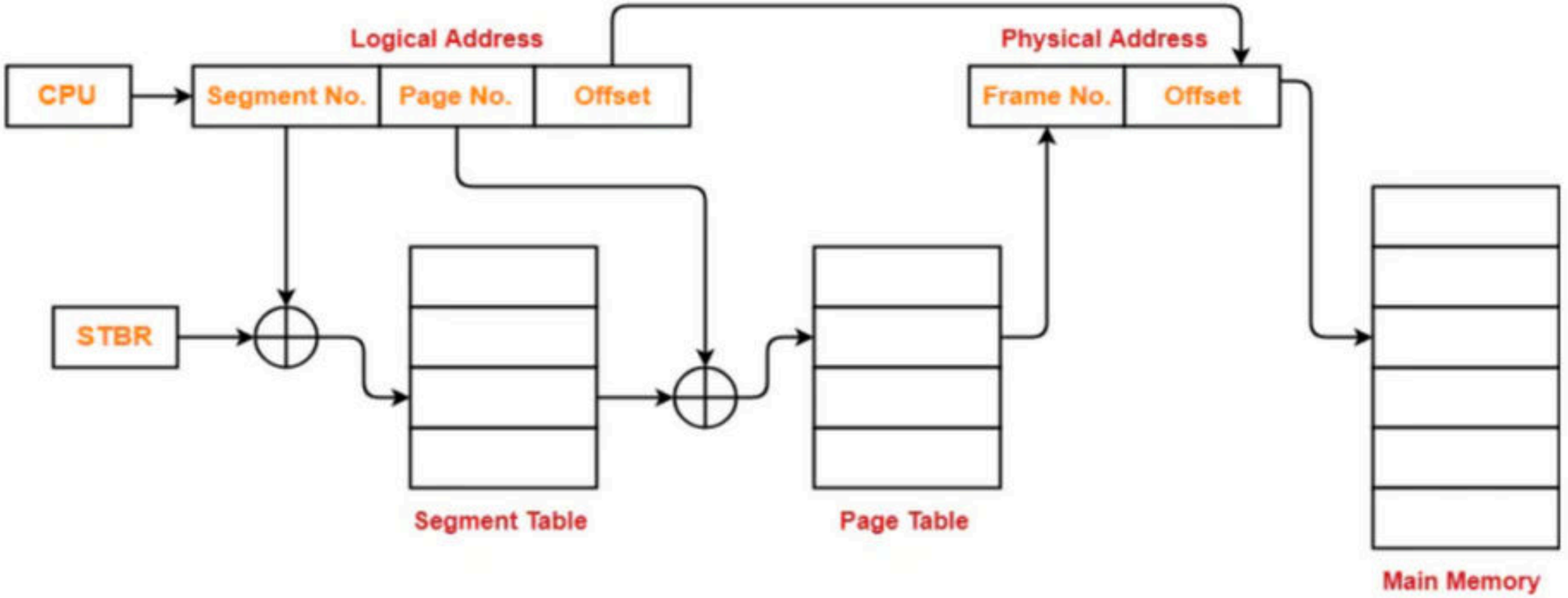
logical address

- **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).

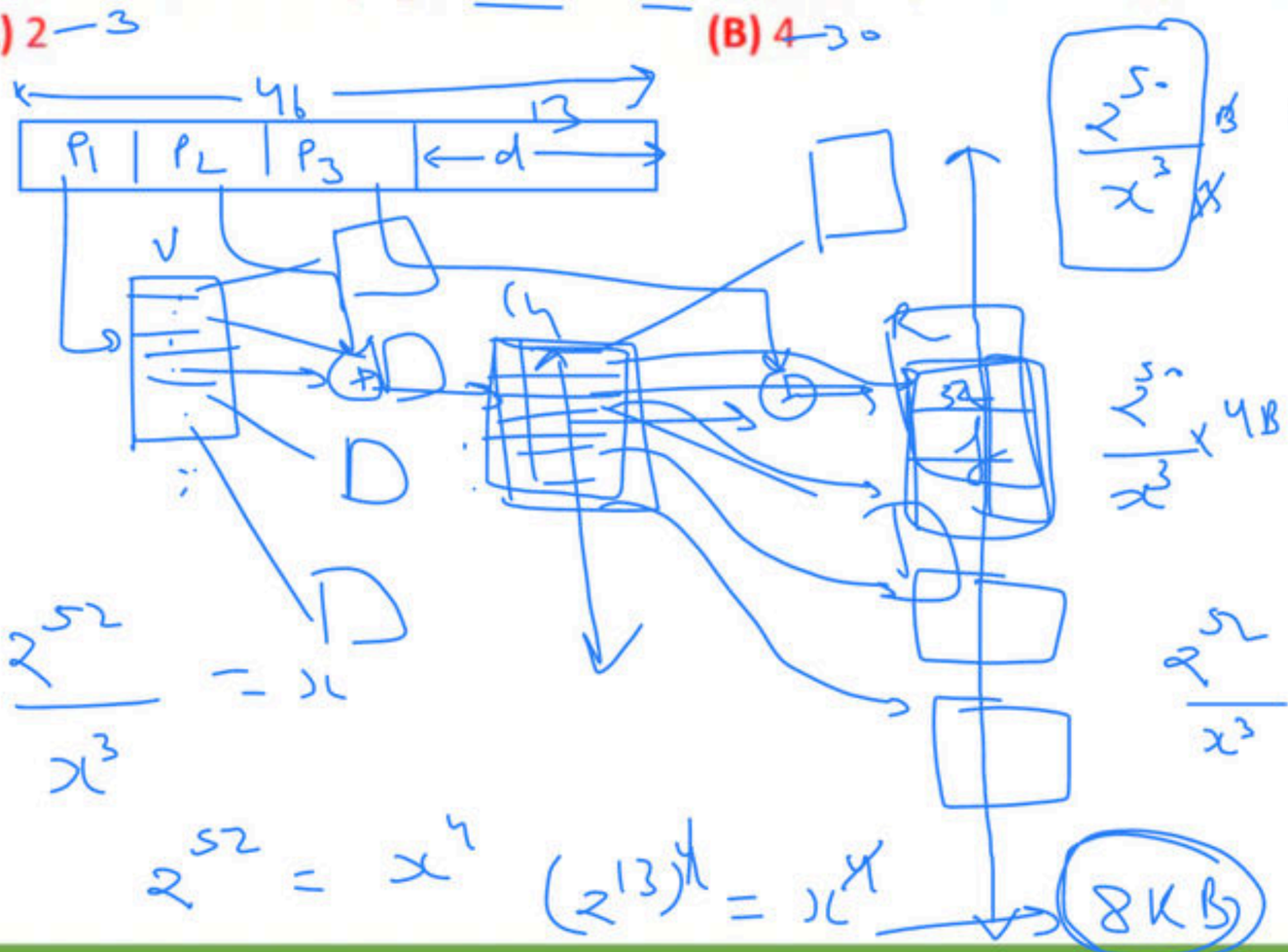


Translating Logical Address into Physical Address

- All properties of segmentation with paging is same as multilevel paging

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. Page size = x
What is the size of a page in KB in this computer? **(GATE-2013) (2 Marks)**

(A) 2 — 3 **(B)** 4 — 30 **(C)** 8 — 64 **(D)** 16



$\frac{2^{48}}{x^2} B$

$\frac{2^{48}}{x^2} \times 4B$

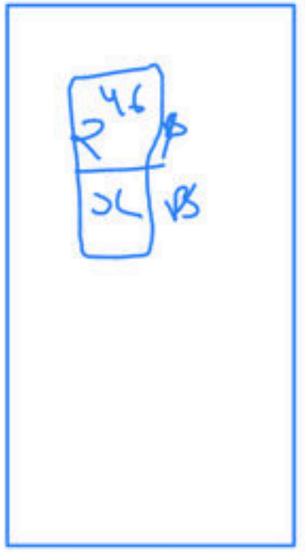
$\frac{2^{48}}{x^2} B$

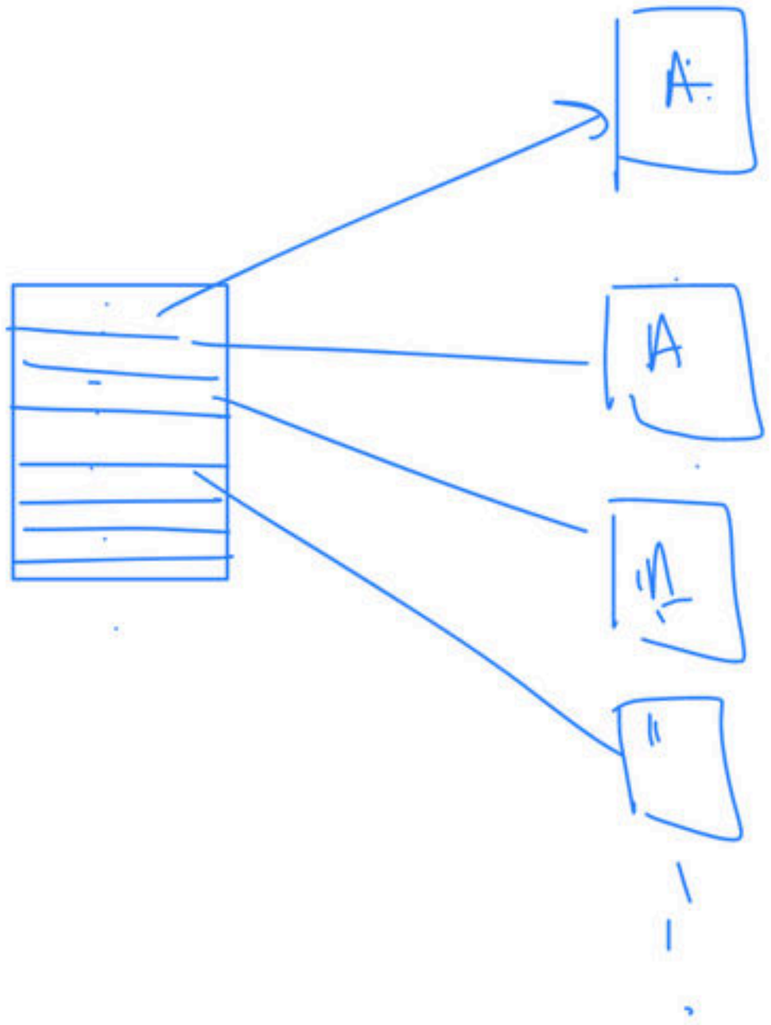
$\frac{2^{46}}{x} \times 4B$

$\frac{2^{48}}{x} B$

$2^{46} \times 1B$

$= 2^{41} B$





Q.21 In the context of operating systems, which of the following statements is/are correct with respect to paging? **(Gate-2021) (1 Marks)**

(a) Paging incurs memory overheads.

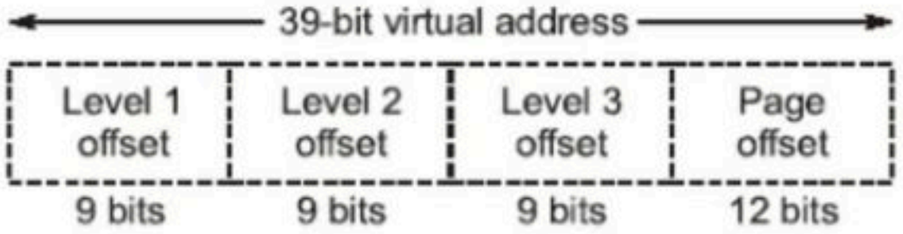
(b) Multi-level paging is necessary to support pages of different sizes.

(c) Page size has no impact on internal fragmentation.

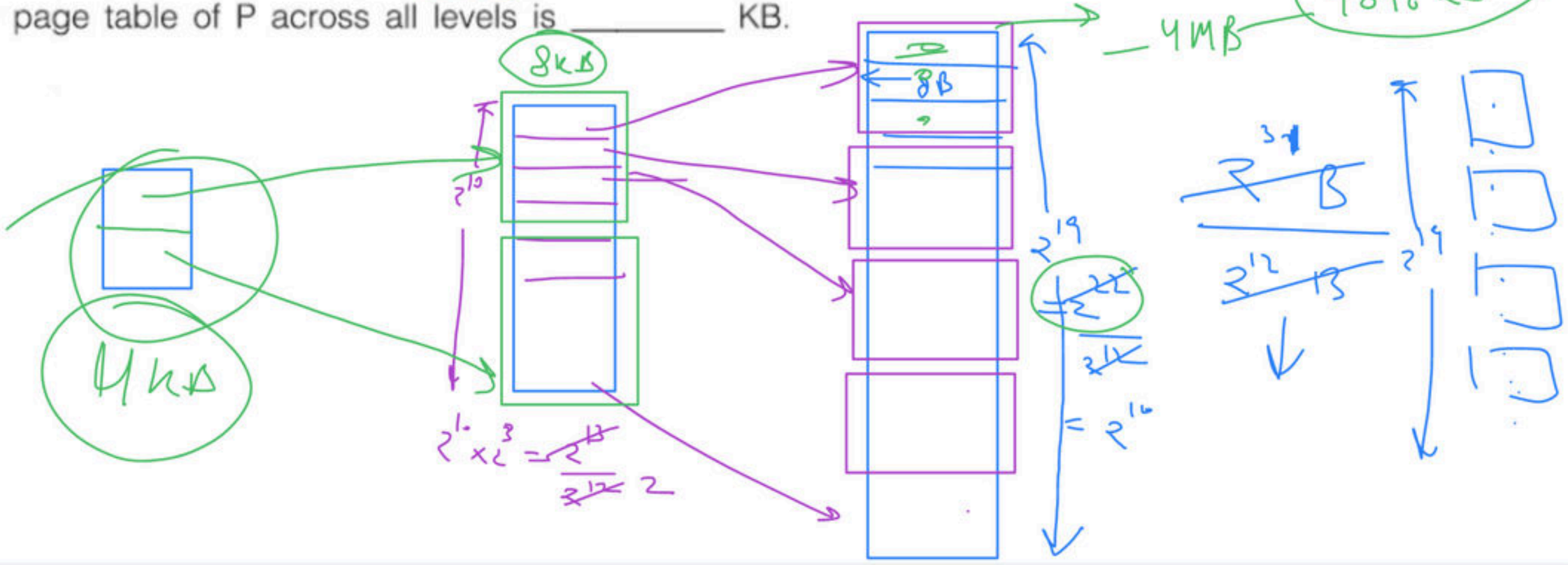
(d) Paging helps solve the issue of external fragmentation.

Q.52 Consider a three-level page table to translate a 39-bit virtual address to a physical address as shown below:

(GATE- 2021)

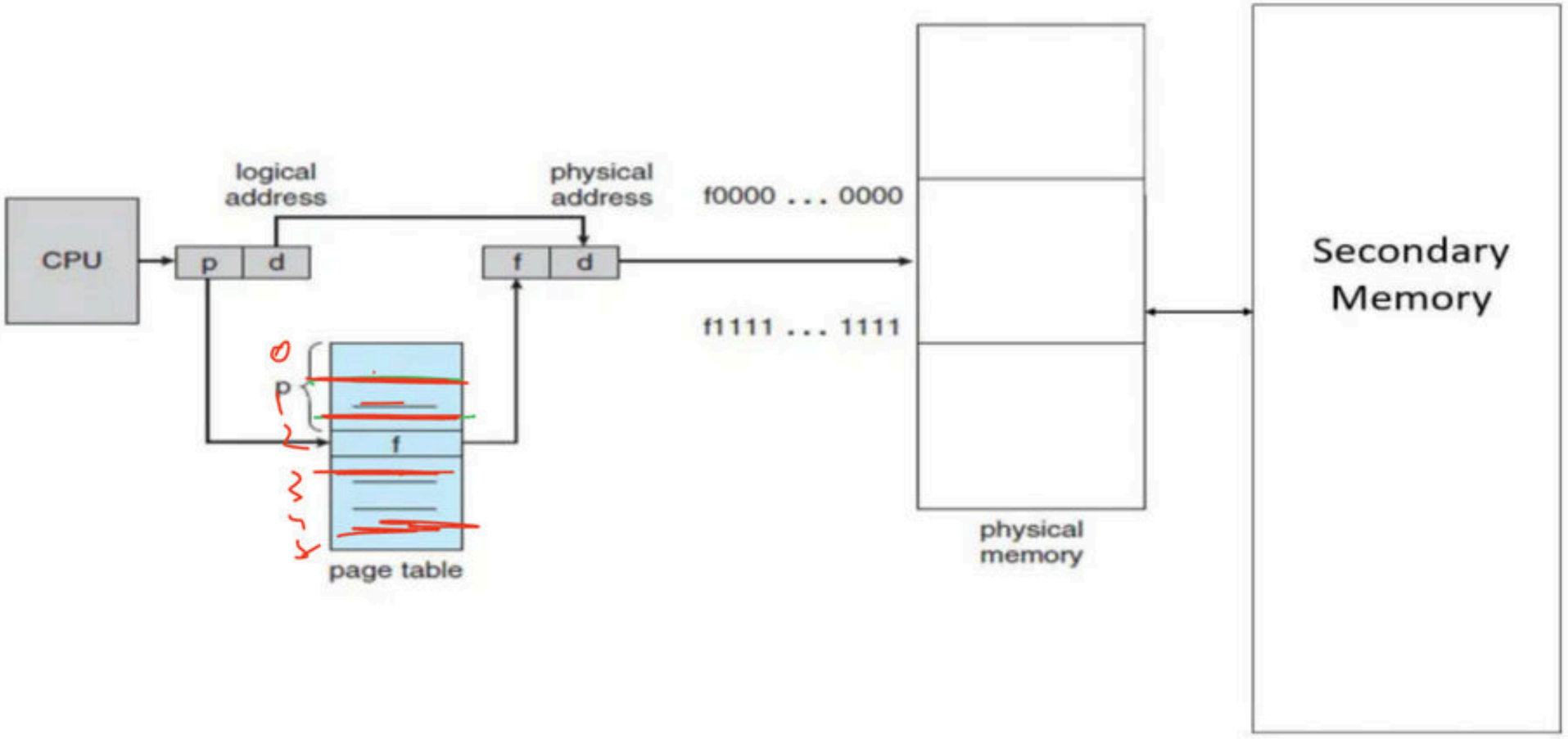


The page size is 4 KB (1 KB = 2^{10} bytes) and page table entry size at every level is 8 bytes. A process *P* is currently using 2 GB (1 GB = 2^{30} bytes) virtual memory which is mapped to 2 GB of physical memory. The minimum amount of memory required for the page table of P across all levels is _____ KB.

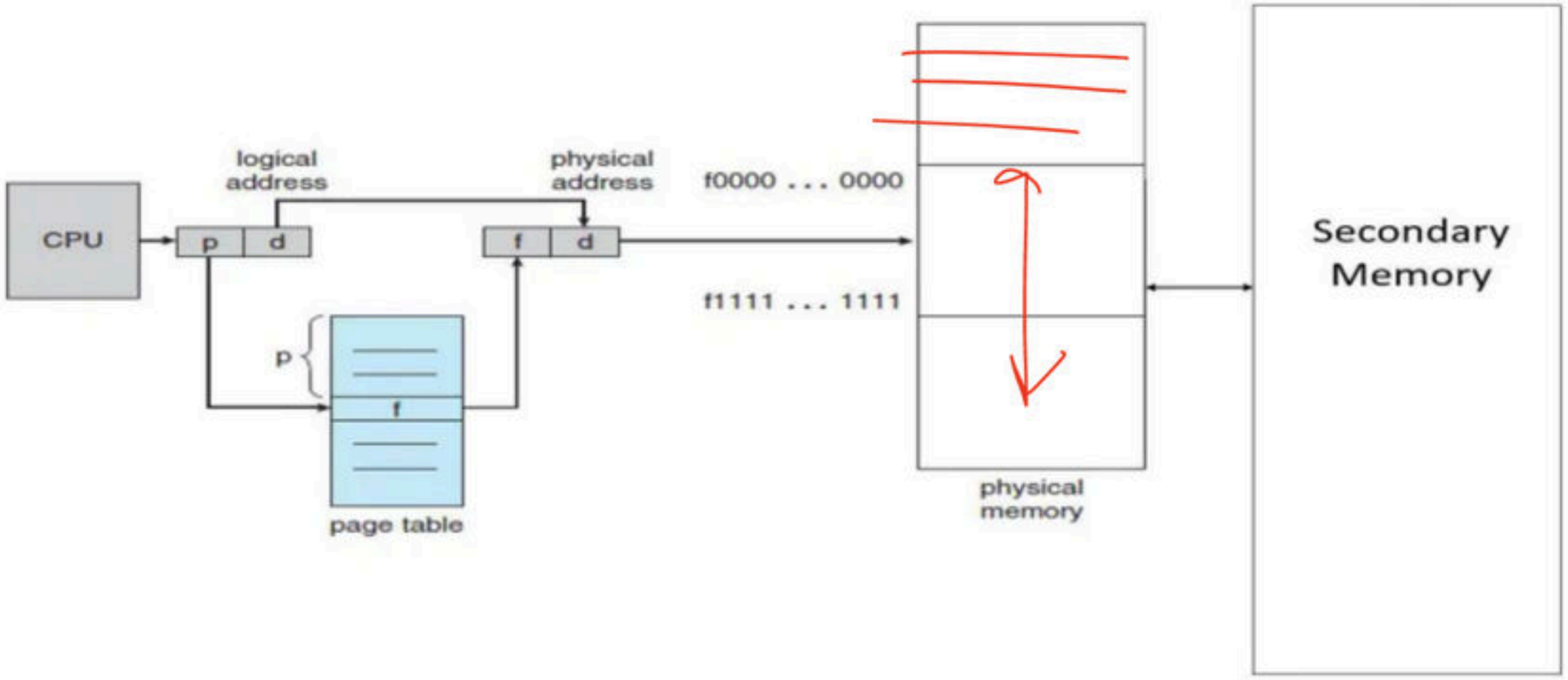


Virtual Memory

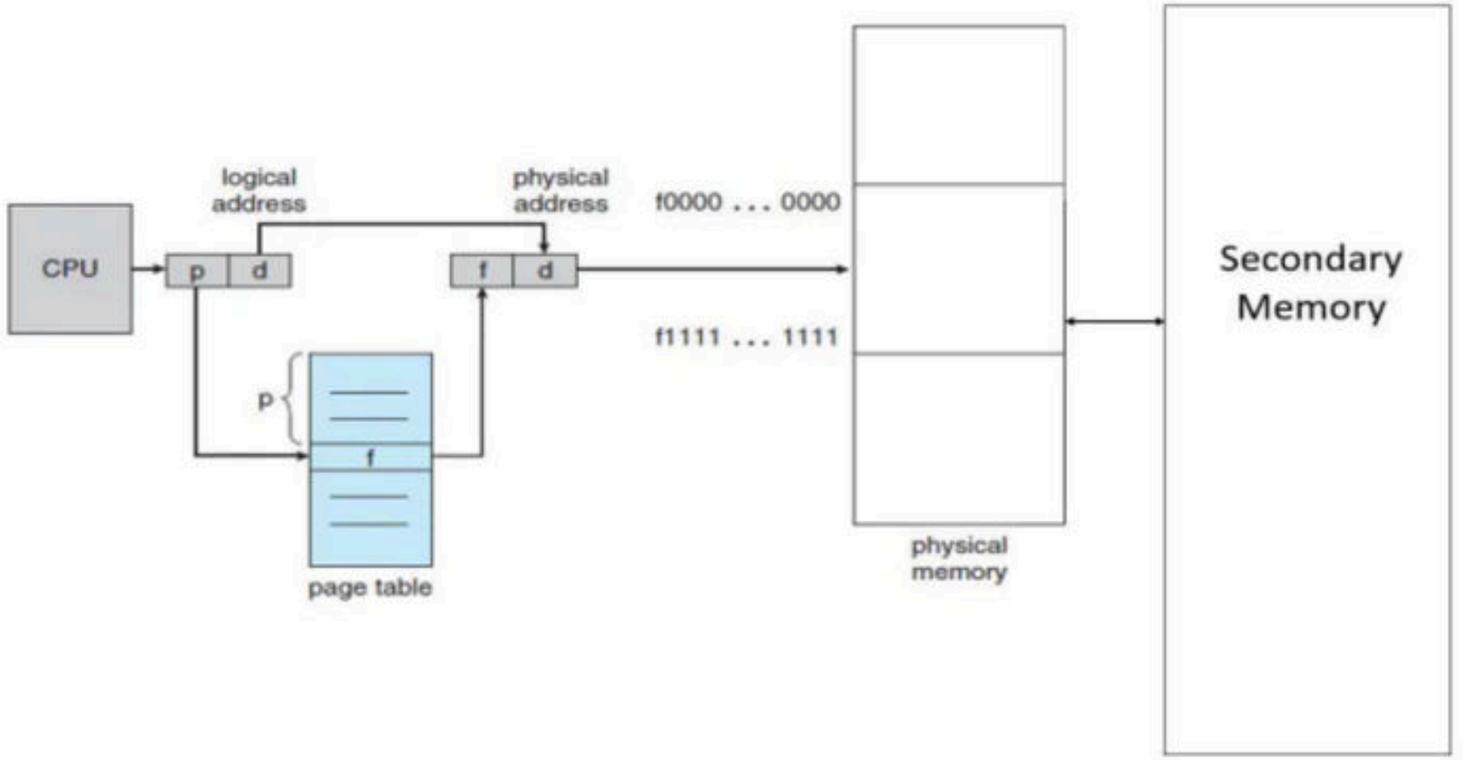
- One important goal in now-a-days computing environment is to keep many processes in memory simultaneously to follow multiprogramming, and use resources efficiently especially main memory.



- **Pure Demand Paging/Demand Paging:** We can start executing a process with no pages in memory. When the operating system sets the instruction pointer to the first instruction of the process, which is on a non-memory-resident page, the process immediately faults for the page.
- After this page is brought into memory, the process continues to execute, faulting as necessary until every page that it needs is in memory. At that it can execute with no more faults. This scheme is **Pure Demand Paging:** never bring a page into memory until it is required.



- The **Locality of Reference** helps Demand Paging in performing reasonably.
- A demand-paging system is extension to a paging system with swapping.
- But rather than swapping the entire process into memory, we use a **Lazy Swapper**. A lazy swapper never swaps a page into memory unless that page will be needed.
- We use the term **Pager** instead of **swapper** in demand paging.



Q Which of the following is incorrect for virtual memory? **(NET-JAN-2017)**

a) Large programs can be written

b) More I/O is required

c) More addressable memory available

d) Faster and easy swapping of process

Q Page making process from main memory to disk is called **(NET-DEC-2010)**

(A) Interruption

(B) Termination

(C) Swapping

(D) None of the above

Q If the executing program size is greater than the existing RAM of a computer, it is still possible to execute the program if the OS supports:
(NET-DEC-2008)

(A) multitasking

(B) virtual memory

(C) paging system

(D) none of the above

Q Moving Process from main memory to disk is called: **(NET-JUNE-2005)**

(A) Caching

(B) Termination

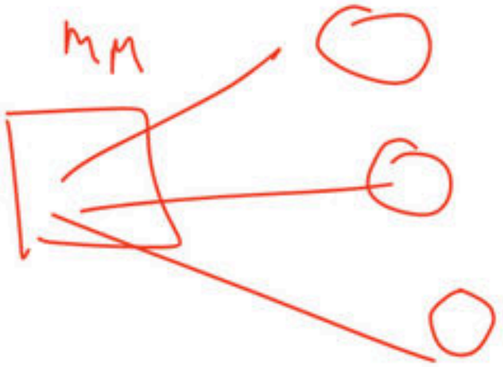
(C) Swapping

(D) Interruption

Break

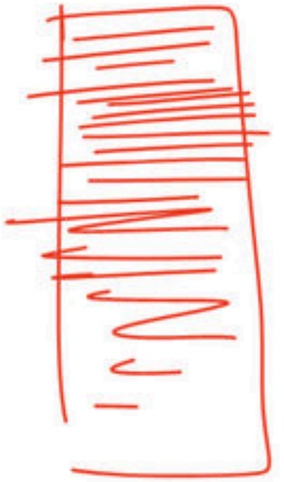
Advantage

- A program would no longer be constrained by the amount of physical memory that is available, Allows the execution of processes that are not completely in main memory, i.e. process can be larger than main memory.
- More programs could be run at the same time as use of main memory is less.
- Less I/O would be needed to load or swap user programs into memory, so each user program would run faster.
- Virtual memory also allows processes to share files easily and to implement shared memory.



Disadvantages

- Virtual memory is not easy to implement.
- It may substantially decrease performance if it is used carelessly (Thrashing)



Q Which of the following memory allocation scheme suffers from external fragmentation? **(NET-DEC-2012)**

(A) Segmentation

(B) Pure demand paging

(C) Swapping

(D) Paging

Q Which of the following statements is false? **(GATE-2001) (1 Marks)**

(A) Virtual memory implements the translation of a program‘s address space into physical memory address space

(B) Virtual memory allows each program to exceed the size of the primary memory

(C) Virtual memory increases the degree of multiprogramming

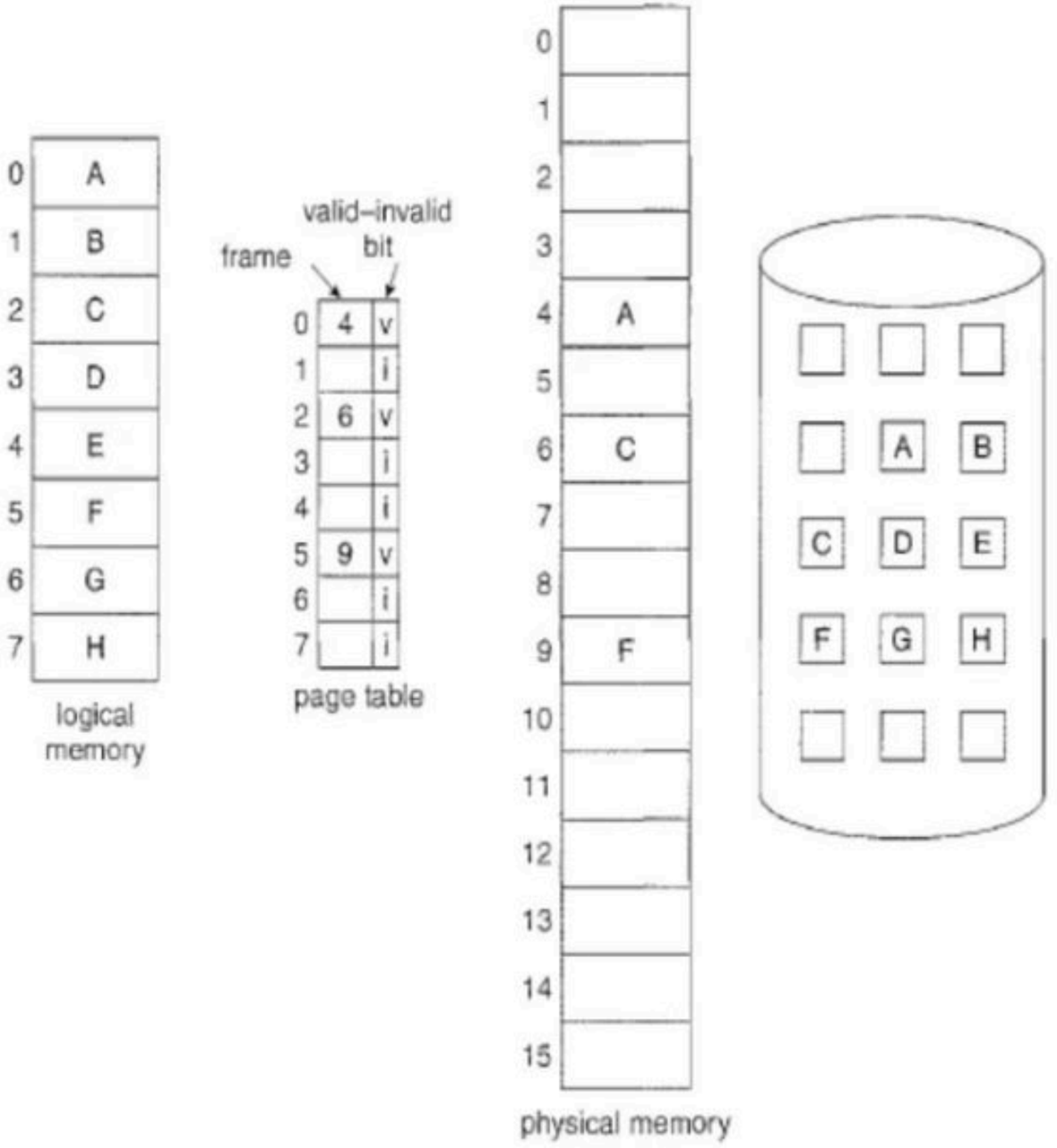
(D) Virtual memory reduces the context switching overhead

Break

Basic Concepts

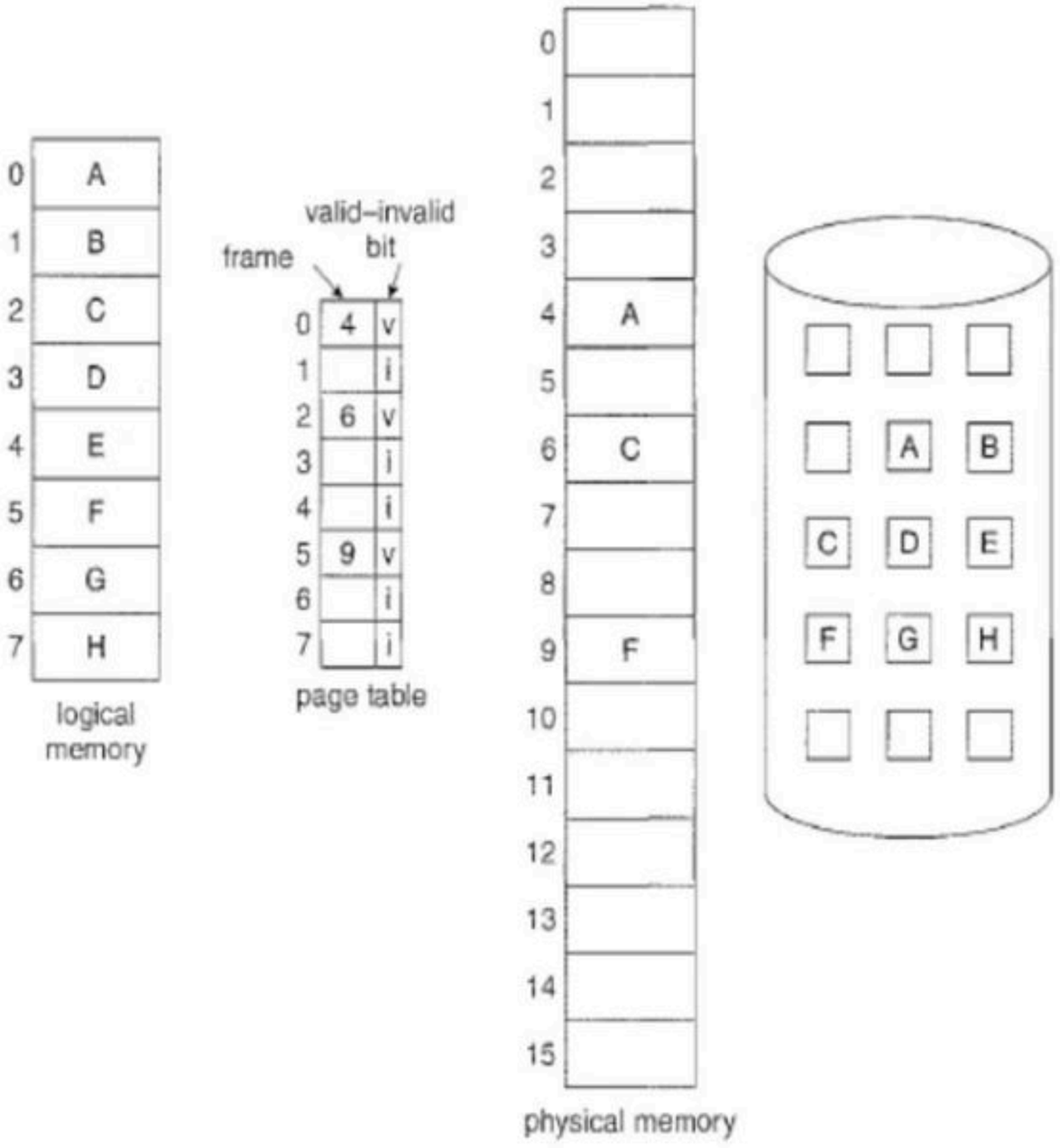
- When a process is started execution, no page is loaded into main memory before demand so, it avoids reading into memory pages that will not be used anyway, decreasing the swap time and the amount of physical memory needed.

- Now we need some hardware to distinguish between which pages are in memory and which are not, so **valid -invalid bit scheme** can be used for it.



- This time, however, when this bit is set to "**valid**" the associated page is both legal and in memory. If the bit is set to "**invalid**" the page either is not valid (that is, not in the logical address space of the process) or is valid but is currently on the disk.

- The page-table entry for a page that is brought into memory is set as usual but the page-table entry for a page that is not currently in memory is either simply marked invalid.



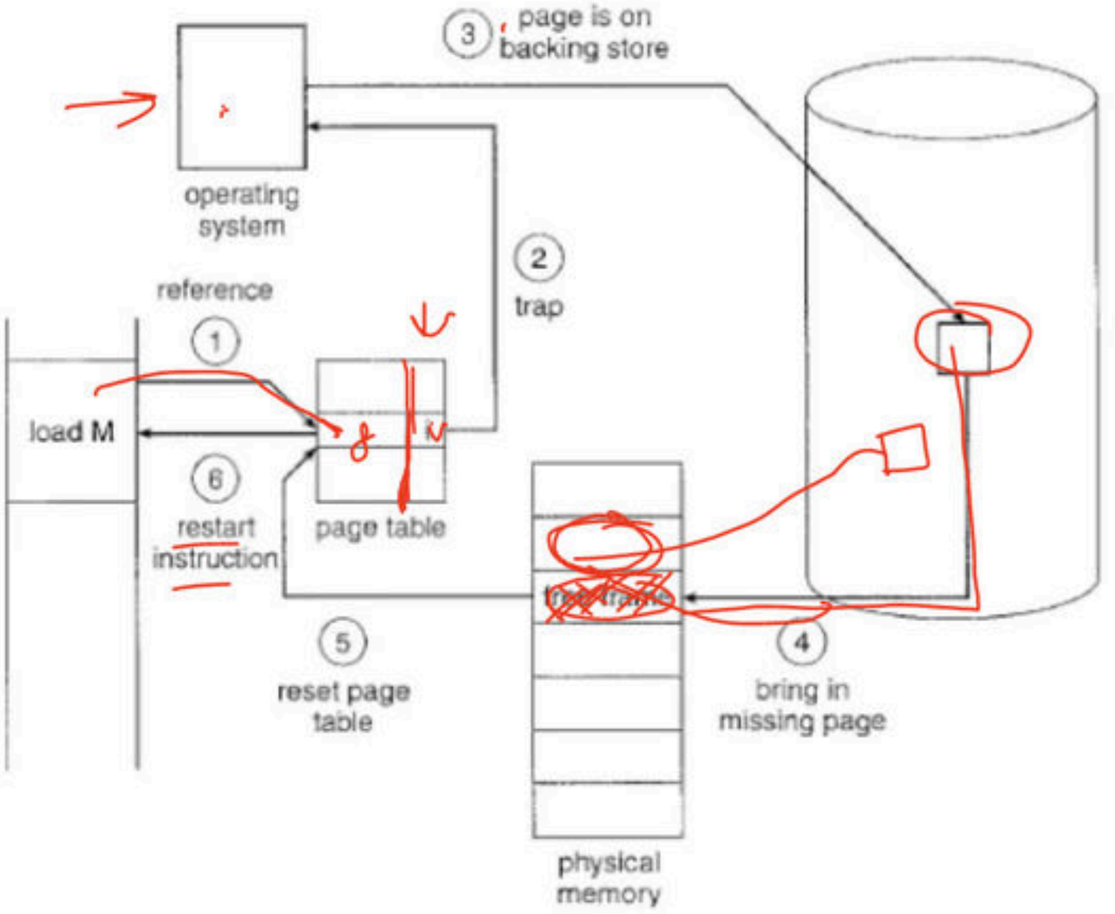
Page Fault: - When a process tries to access a page that is not in main memory then a **Page Fault Occurs.**

Steps to handle Page Fault

- If the reference was invalid, we terminate the process. If it was valid, but we have not yet brought in that page, we now page it in.

- We find a free frame (by taking one from the free-frame list, for example).

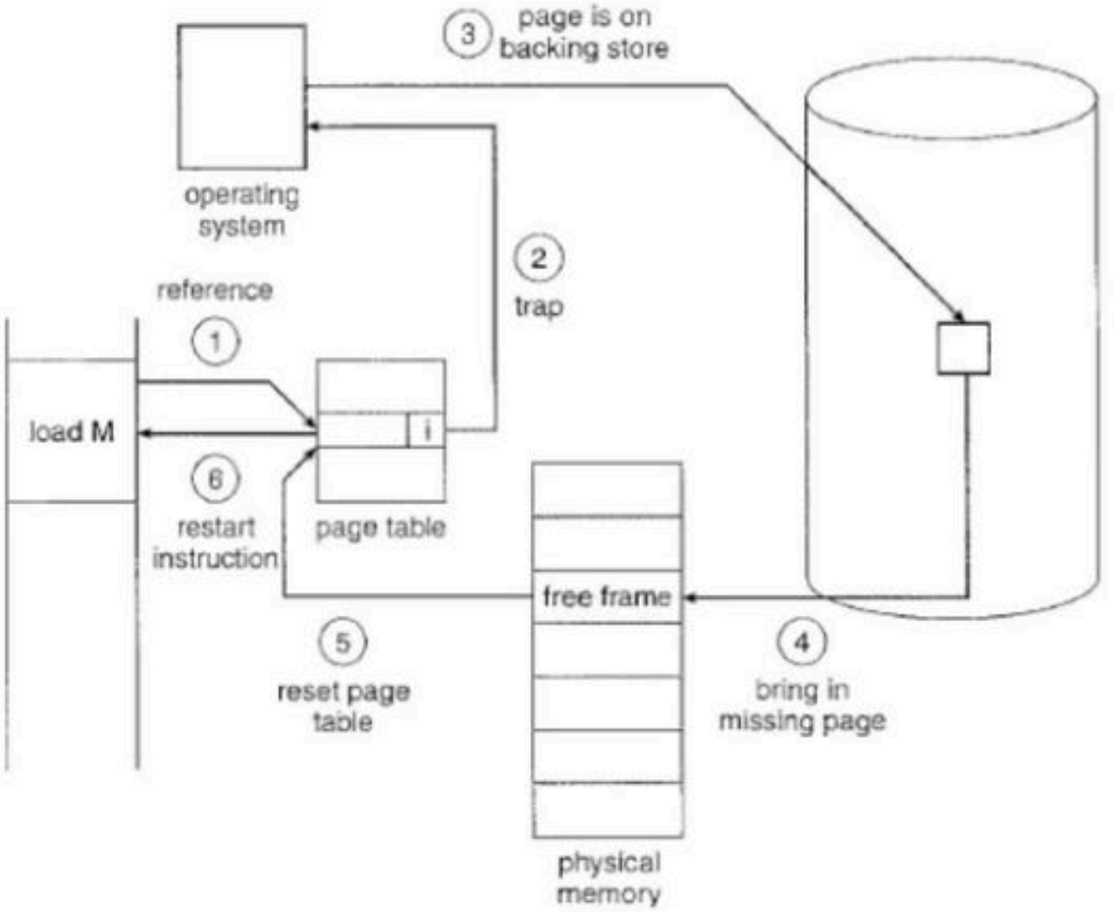
- We schedule a disk operation to read the desired page into the newly allocated frame.



- When the disk read is complete, we modify the internal table kept with the process and the page table to indicate that the page is now in memory.

- We restart the instruction that was interrupted by the trap. The process can now access the page as though it had always been in memory.

- A crucial requirement for demand paging is the ability to restart any instruction after a page fault. Because we save the state (registers, condition code, instruction counter) of the interrupted process when a page fault occurs, we must be able to restart the process in exactly the same place and state.



Q A page fault **(NET-DEC-2009)**

(A) is an error specific page.

(B) is an access to the page not currently in memory.

(C) occur when a page program occur in a page memory.

(D) page used in the previous page reference.

Q A page fault ..... **(NET-JUNE-2006)**

(A) is an error in specific page

(B) is an access to the page not currently in main memory

(C) occurs when a page program accesses a page of memory

(D) is reference to the page which belongs to another program

Break

Performance of Demand Paging

- **Effective Access time for Demand Paging:**
  - (1 - p) x ma + p x page fault service time.

- Here, p: Page fault rate or probability of a page fault.

- ma is memory access time.

Q In a paged memory, the page hit ratio is 0.40. The time required to access a page in secondary memory is equal to 120 ns. The time required to access a page in primary memory is 15 ns. The average time required to access a page is _____.
(NET-JULY-2018)

(a) 105 **(b)** 68 **(c)** 75 **(d)** 78

Q Consider a process executing on an operating system that uses demand paging. The average time for a memory access in the system is M units if the corresponding memory page is available in memory, and D units if the memory access causes a page fault. It has been experimental measured that the average time taken for a memory access in the process is X units.
Which one of the following is the correct expression for the page fault rate experienced by the process? **(GATE-2018) (2 Marks)**

(A) (D – M) / (X – M)

(B) (X – M) / (D – M)

(C) (D – X) / (D – M)

(D) (X – M) / (D – X)

Q Suppose that the number of instructions executed between page fault is directly proportional to the number of page frames allocated to a program. If the available memory is doubled, the mean interval between page faults is also doubled. Further, consider that a normal instruction takes one microsecond, but if a page fault occurs, it takes 2001 microseconds. If a program takes 60 sec to run, during which time it gets 15,000 page faults, how long would it take to run if twice as much memory were available? **(NET-DEC-2015)**

A) 60 sec **b)** 30 sec **c)** 45 sec **d)** 10 sec

Q In a demand paging memory system, page table is held in registers. The time taken to service a page fault is 8 msec. if an empty frame is available or if the replaced page is not modified, and it takes 20 msec., if the replaced page is modified. What is the average access time to service a page fault assuming that the page to be replaced is modified 70 of the time? **(NET-DEC-2014)**

(A) 11.6 msec. **(B)** 16.4 msec. **(C)** 28 msec. **(D)** 14 msec.

Q Virtual memory is **(NET-JUNE-2011)**

(A) related to virtual reality

(B) a form of ROM

(C) a form of RAM

(D) None of the above

Q Let the page fault service time be 10ms in a computer with average memory access time being 20ns. If one-page fault is generated for every 10^6 memory accesses, what is the effective access time for the memory? **(GATE-2011) (1 Marks)**

(A) 21ns **(B)** 30ns **(C)** 23ns **(D)** 35ns

Q The memory allocation scheme subjected to “external” fragmentation is: **(NET-JUNE-2006)**

(A) Segmentation

(B) Swapping

(C) Demand paging

(D) Multiple contiguous fixed partitions

Q Suppose the time to service a page fault is on the average 10 milliseconds, while a memory access takes 1 microsecond. Then a 99.99% hit ratio results in average memory access time of? **(GATE-2000) (1 Marks).**

(A) 1.9999 milliseconds

(B) 1 millisecond

(C) 9.999 microseconds

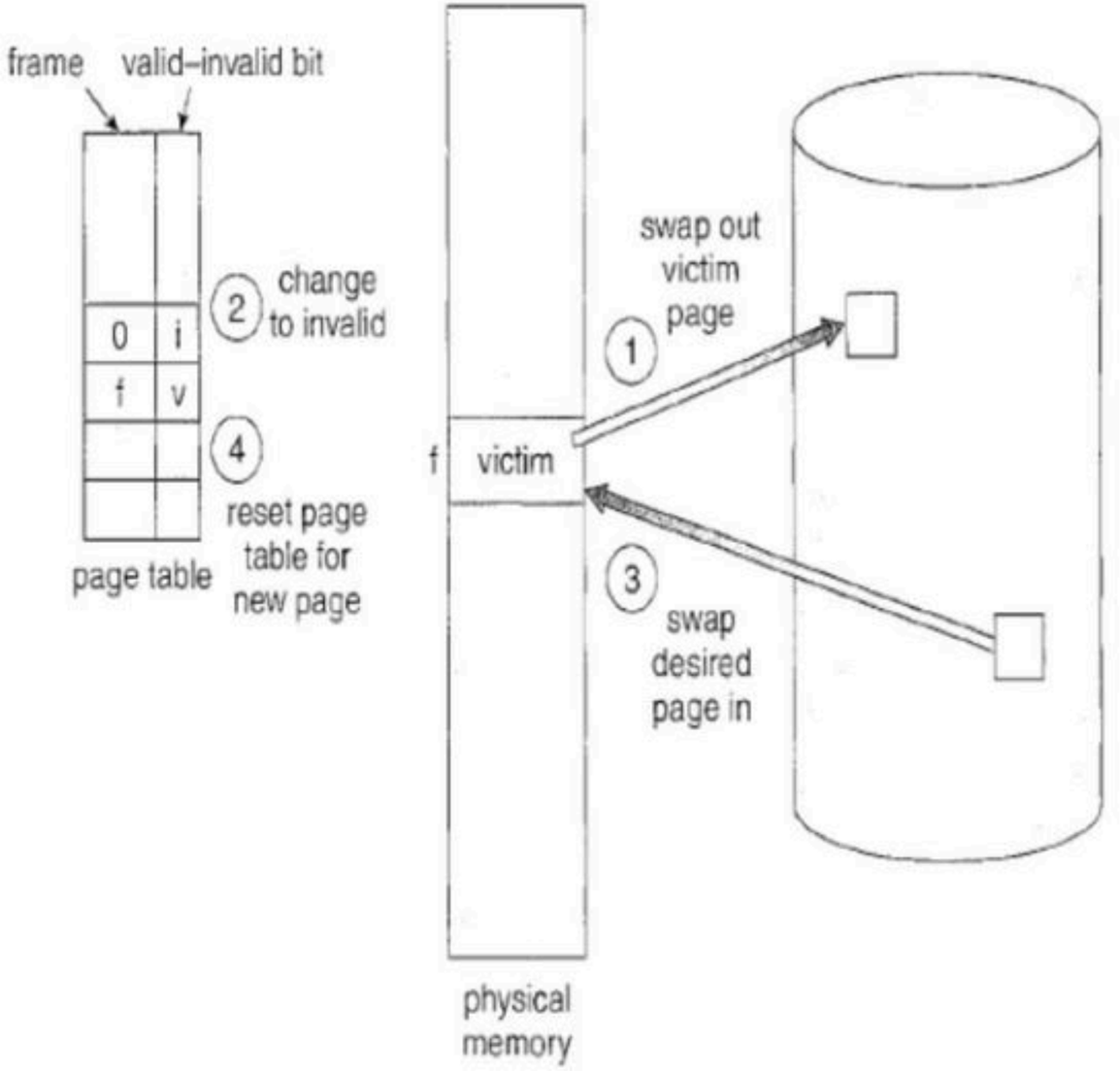
(D) 1.9999 microseconds

Break

Page Replacement

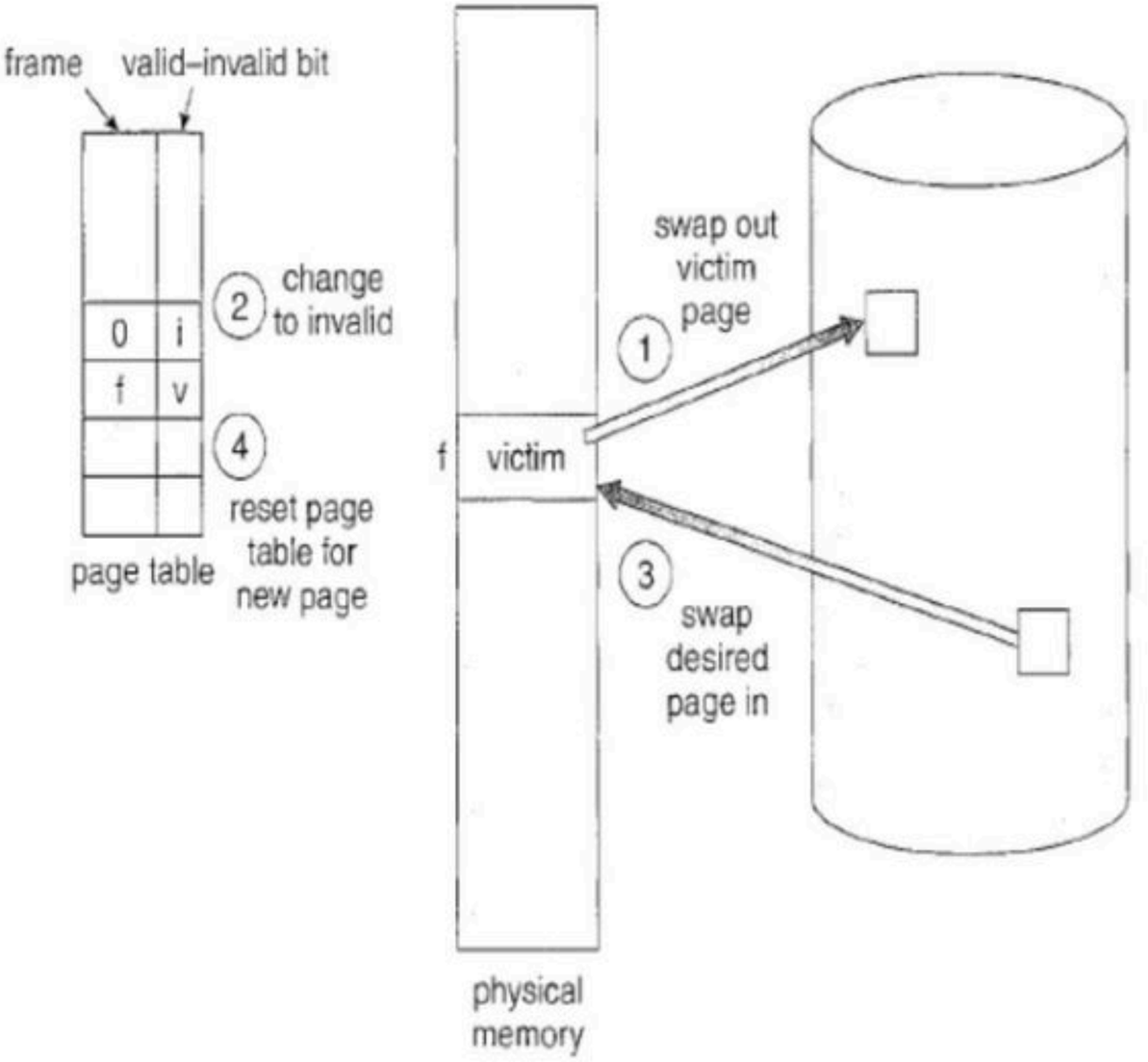
- With the increase in multiprogramming each process will get less amount of space in main memory so, the rate of page faults may rise.

- thus, to reduce the degree of multiprogramming the operating system swaps out processes from the memory freeing the frames and thus the process that requires to execute can now execute.



- If no frames are free, two-page transfers (one out and one in) are required. This situation effectively doubles the page-fault service time and increases the effective access time accordingly.

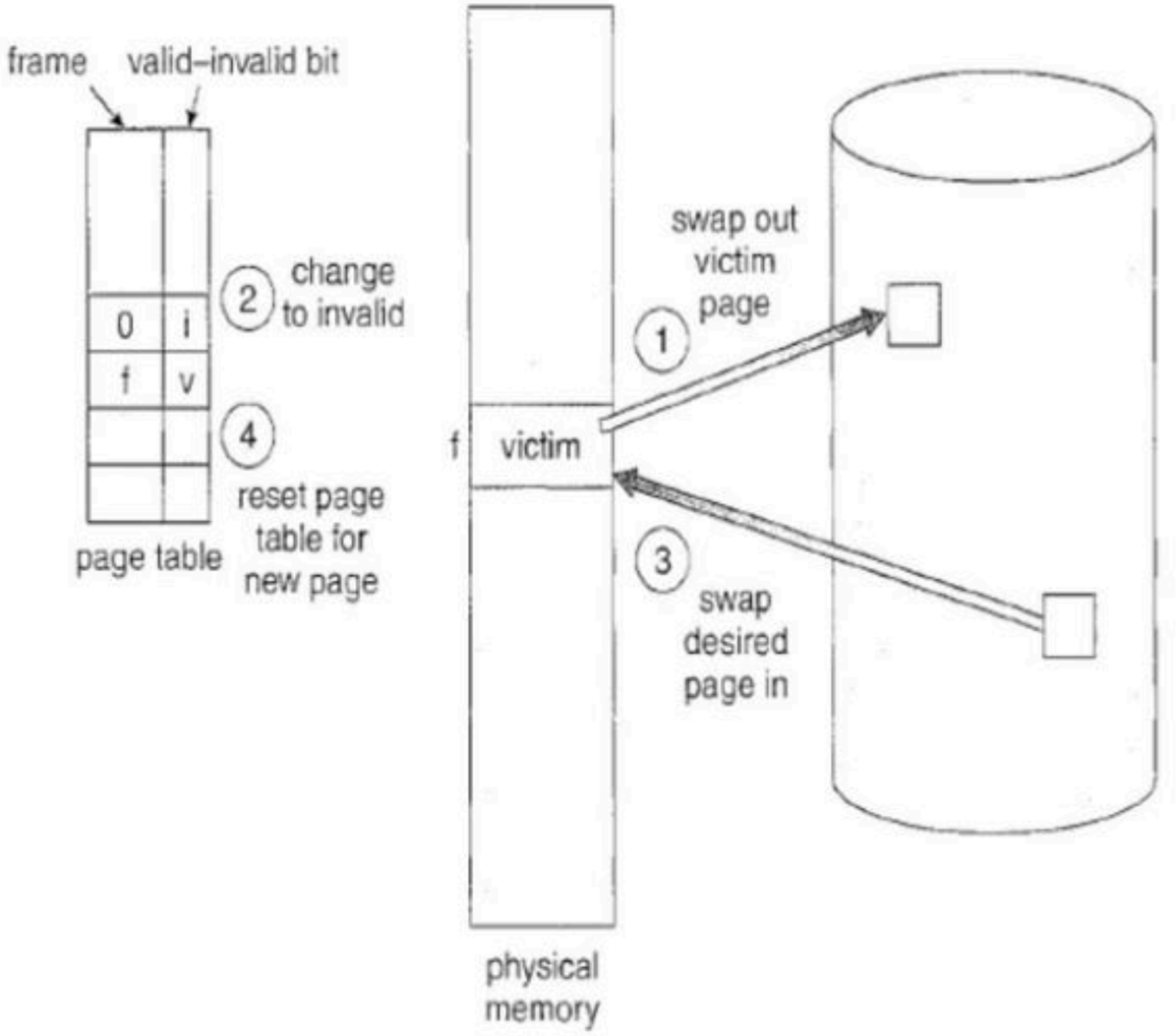
- We can reduce this overhead by using a **Modify bit or Dirty Bit**. When this scheme is used, each page or frame has a modify bit associated with it in the hardware.



- The modify bit for a page is set by the hardware whenever any word or byte in the page is written into, indicating that the page has been modified.

- **If the bit is set**: the page has been modified since it was read in from the disk. In this case, we must write the page to the disk.

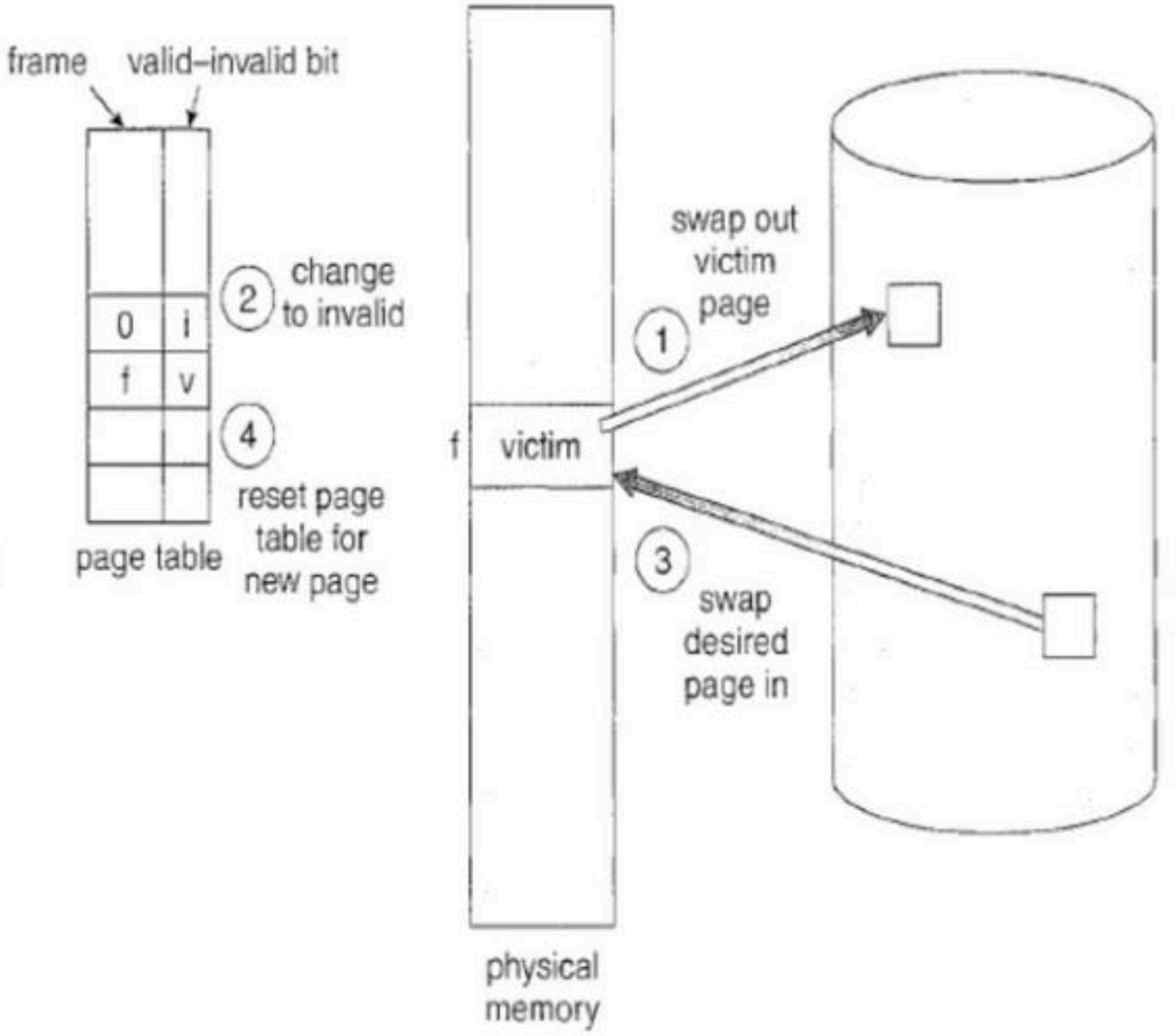
- **If the modify bit is not set**: however, the page has not been modified since it was read into memory. In this case, we need not write the memory page to the disk: it is already there.



- Now, we must solve two major problems to implement demand paging: We must be able to develop a **frame allocation algorithm** and a **page replacement algorithm**.

- Frame Allocation will decide how much frames to allocate to a process.

- Page Replacement will decide which page to replace next.



Q Dirty bit is used to show the **(NET-DEC-2018)**

a) Wrong Page

b) Page with corrupted data

c) Page with low frequency occurrence

d) Page that is modified after being loaded into cache memory

Q A virtual memory has a page size of 1K words. There are eight pages and four blocks. The associative memory page table contains the following entries:
Which of the following list of virtual addresses (in decimal) will not cause any page fault if referenced by the CPU ? **(NET-DEC-2015)**

a) 1024, 3072, 4096, 6144

b) 1234, 4012, 5000, 6200

c) 1020, 3012, 6120, 8100

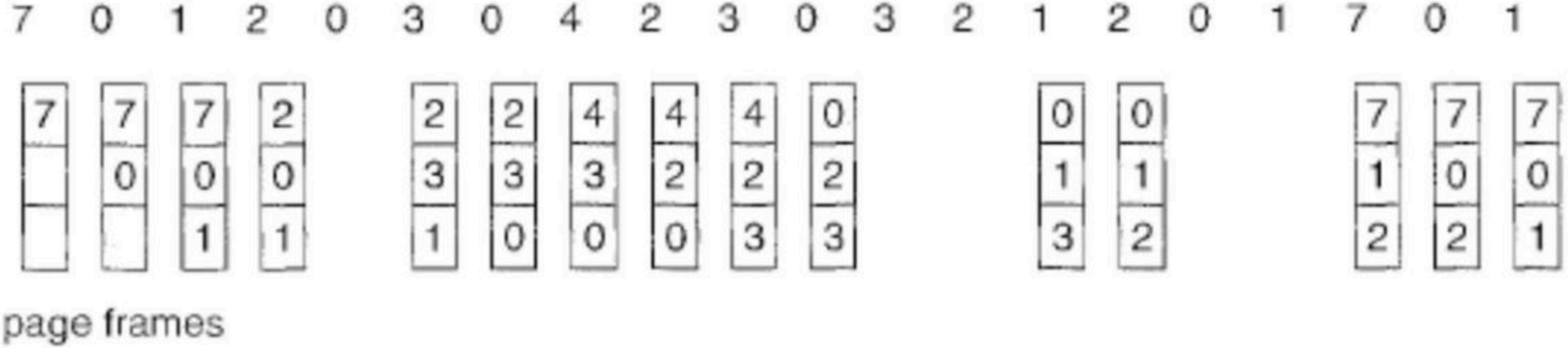
d) 2021, 4050, 5112, 7100

Page	Block
0	3
2	1
5	2
7	0

Break

Page Replacement Algorithms

- **First In First Out Page Replacement Algorithm:** - A FIFO replacement algorithm associates with each page, the time when that page was brought into memory. When a page must be replaced, the oldest page is chosen. i.e. the first page that came into the memory will be replaced first.



- In the above example the number of page fault is 15.

- The FIFO page-replacement algorithm is easy to understand and program. However, its performance is not always good.

- **Belady's Anomaly:** for some page-replacement algorithms, the page-fault rate may increase as the number of allocated frames increases.

Q Suppose for a process P, references to pages occur in order are 1, 2, 4, 5, 2, 1, 2, 4. Assume that the main memory can accommodate 3 pages and the main memory already has the pages 1 and 2 in the order 1-first, 2-second. At this moment, assume fifo page replacement algorithm is used then the number of page faults that occur to complete the execution of process P is **(NET-DEC-2018)**

(A) 4 **(B)** 3 **(C)** 5 **(D)** 6

Q. In which one of the following page replacement algorithms it is possible for the page fault rate to increase even when the number of allocated frames increases?
(GATE-2016) (1 Marks)

(a) LRU (Least Recently Used

(b) OPT (Optimal Page Replacement)

(c) MRU (Most Recently Used)

(d) FIFO (First In First Out)

Q Consider the reference string 0 1 2 3 0 1 4 0 1 2 3 4 If FIFO page replacement algorithm is used, then the number of page faults with three-page frames and four-page frames are _____ and _____ respectively. **(NET-JUNE-2016)**

a) 10, 9 **b)** 9, 9 **c)** 10, 10 **d)** 9,10

Q Consider the following page trace: 4, 3, 2, 1, 4, 3, 5, 4, 3, 2, 1, 5. Percentage of page fault that would occur if FIFO page replacement algorithm is used with number of frames for the JOB m = 4 will be **(NET-JUNE-2012)**

(A) 8 **(B)** 9 **(C)** 10 **(D)** 12

Q A system uses FIFO policy for page replacement. It has 4-page frames with no pages loaded to begin with. The system first accesses 100 distinct pages in some order and then accesses the same 100 pages but now in the reverse order. How many page faults will occur? **(GATE-2010) (1 Marks)**

(A) 196 **(B)** 192 **(C)** 197 **(D)** 195

Q In which one of the following page replacement policies, Belady's anomaly may occur? **(GATE-2009) (1 Marks)**

(A) FIFO **(B)** Optimal **(C)** LRU **(D)** MRU

Q A virtual memory system uses First In First Out (FIFO) page replacement policy and allocates a fixed number of frames to a process. Consider the following statements: **(GATE-2007) (2 Marks)**

P: Increasing the number of page frames allocated to a process sometimes increases the page fault rate.

Q: Some programs do not exhibit locality of reference.

Which one of the following is TRUE?

(A) Both P and Q are true, and Q is the reason for P

(B) Both P and Q are true, but Q is not the reason for P.

(C) P is false, but Q is true

(D) Both P and Q are false.

Q Which page replacement policy suffers from Belady's anomaly? **(NET-JUNE-2007)**

(A) LRU **(B)** LFU **(C)** FIFO **(D)** OPTIMAL

Q A program has five virtual pages, numbered from 0 to 4. If the pages are referenced in the order 012301401234, with three-page frames, the total number of page faults with FIFO will be equal to: **(NET-DEC-2007)**

(A) 0 **(B)** 4 **(C)** 6 **(D)** 9

Q Consider a virtual memory system with FIFO page replacement policy. For an arbitrary page access pattern, increasing the number of page frames in main memory will **(GATE-2001) (1 Marks)**

(A) always decrease the number of page faults

(B) always increase the number of page faults

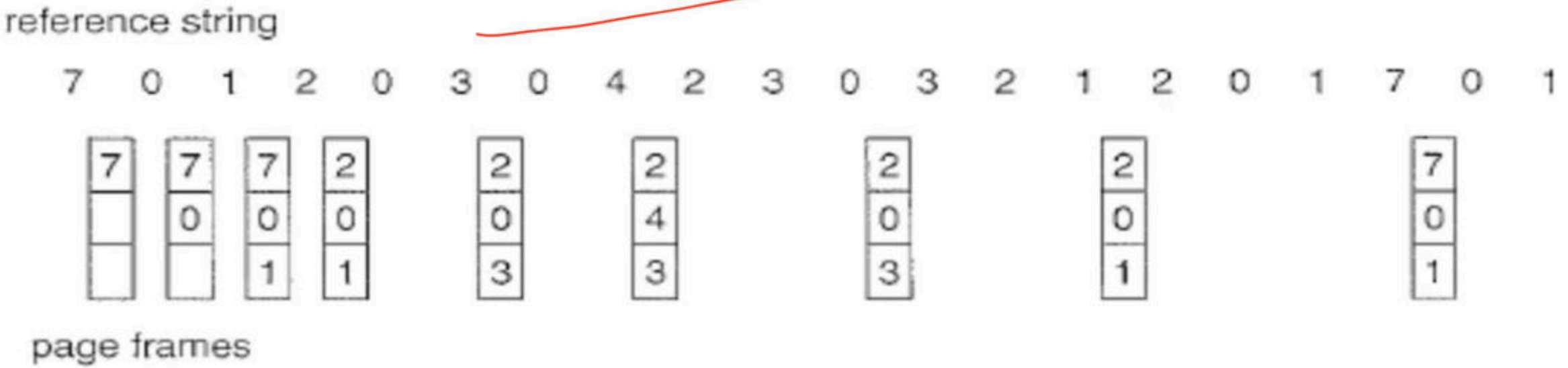
(C) sometimes increase the number of page faults

(D) never affect the number of page faults

Break

Optimal Page Replacement Algorithm

- Replace the page that will not be used for the longest period of time.
- It has the lowest page-fault rate of all algorithms and will never suffer from Belady's anomaly.
- Use of this page-replacement algorithm guarantees the lowest possible page fault rate for a fixed number of frames.



- The number of page faults occurred: 9

- Unfortunately, the optimal page-replacement algorithm is difficult to implement, because it requires future knowledge of the reference string.

- It is mainly used for comparison studies.

Q Assume that there are 3 page frames which are initially empty. If the page reference string is 1, 2, 3, 4, 2, 1, 5, 3, 2, 4, 6, the number of page faults using the optimal replacement policy is_____. **(GATE-2014) (2 Marks)**

Q A process has been allocated 3-page frames. Assume that none of the pages of the process are available in the memory initially. The process makes the following sequence of page references (reference string): 1, 2, 1, 3, 7, 4, 5, 6, 3, 1. If optimal page replacement policy is used, how many page faults occur for the above reference string?
(GATE-2007) (2 Marks)

(A) 7 **(B)** 8 **(C)** 9 **(D)** 10

Q The optimal page replacement algorithm will select the page that **(GATE-2002) (1 Marks)**

(A) Has not been used for the longest time in the past.

(B) Will not be used for the longest time in the future.

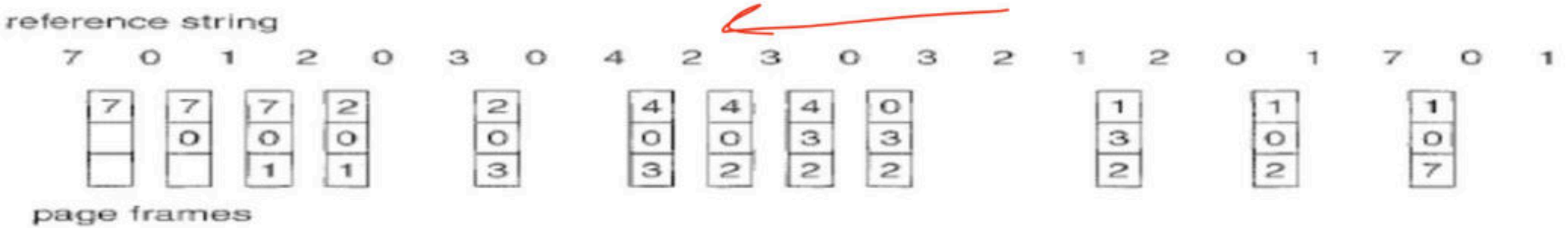
(C) Has been used least number of times.

(D) Has been used most number of times.

Break

Least Recently Used (LRU) Page Replacement Algorithm

- Replace the page that has not been used for the longest period of time.
- We can think of this strategy as the optimal page-replacement algorithm looking backward in time, rather than forward.



- LRU gives us 12-page faults.
- LRU is much better than FIFO replacement.
- The LRU policy is often used as a page-replacement algorithm and is considered to be good.
- LRU also does not suffer from Belady's Anomaly.

- The major problem is how to implement LRU replacement. An LRU page-replacement algorithm may require substantial hardware assistance.

- LRU and OPT belong to a class of page-replacement algorithms, called **stack algorithms** and can never exhibit Belady's anomaly.

- A stack algorithm is an algorithm for which it can be shown that the set of pages in memory for *n* frames is always a subset of the set of pages that would be in memory with *n + 1* frames.

Q Consider a virtual page reference string 1, 2, 3, 2, 4, 2, 5, 2, 3, 4. Suppose LRU page replacement algorithm is implemented with 3 page frames in main memory. Then the number of page faults are_____.
(NET-JULY-2018)

(a) 5 **(b)** 7 **(c)** 9 **(d)** 10

Q Consider a virtual page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1 Suppose a demand paged virtual memory system running on a computer system such that the main memory has 3 page frames. Then _____ page replacement algorithm has minimum number of page faults. **(NET-NOV-2017)**

A) FIFO **b)** LIFO **c)** LRU **d)** Optimal

Q Recall that Belady's anomaly is that the page-fault rate may increase as the number of allocated frames increases. Now, consider the following statements:

S_1: Random page replacement algorithm (where a page chosen at random is replaced) suffers from Belady's anomaly

S_2: LRU page replacement algorithm suffers from Belady's anomaly

Which of the following is CORRECT? **(GATE-2017) (1 Marks)**

(a) S_1 is true, S_2 is true

(b) S_1 is true, S_2 is false

(c) S_1 is false, S_2 is true

(d) S_1 is false, S_2 is false

Q Consider the following page reference string : 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6 Which of the following options, gives the correct number of page faults related to LRU, FIFO, and optimal page replacement algorithms respectively, assuming 05 page frames and all frames are initially empty ? **(NET-AUG-2016)**

a) 10, 14, 8 **b)** 8, 10, 7 **c)** 7, 10, 8 **d)** 7, 10, 7

Q Suppose that the virtual Address space has eight pages and physical memory with four page frames. If LRU page replacement algorithm is used, _____ number of page faults occur with the reference string.

0 2 1 3 5 4 6 3 7 4 7 3 3 5 5 3 1 1 1 7 2 3 4 1 **(NET-AUG-2016)**

(A) 13 **(B)** 12 **(C)** 11 **(D)** 10

Q Consider a main memory with five page frames and the following sequence of page references: 3, 8, 2, 3, 9, 1, 6, 3, 8, 9, 3, 6, 2, 1, 3. Which one of the following is true with respect to page replacement policies First-In-First Out (FIFO) and Least Recently Used (LRU)? **(GATE-2015) (2 Marks)**

(A) Both incur the same number of page faults

(B) FIFO incurs 2 more-page faults than LRU

(C) LRU incurs 2 more-page faults than FIFO

(D) FIFO incurs 1 more page faults than LRU

Q A LRU page replacement is used with four page frames and eight pages. How many page faults will occur with the reference string 0172327103 if the four frames are initially empty? **(NET-JUNE-2015)**

a) 6 **b)** 7 **c)** 8 **d)** 5

Q A system uses 3-page frames for storing process pages in main memory. It uses the Least Recently Used (LRU) page replacement policy. Assume that all the page frames are initially empty. What is the total number of page faults that will occur while processing the page reference string given below? 4, 7, 6, 1, 7, 6, 1, 2, 7, 2 **(GATE-2014) (2 Marks)**

Q Consider a program that consists of 8 pages (from 0 to 7) and we have 4 page frames in the physical memory for the pages. The page reference string is : 1 2 3 2 5 6 3 4 6 3 7 3 1 5 3 6 3 4 2 4 3 4 5 1. The number of page faults in LRU and optimal page replacement algorithms are respectively (without including initial page faults to fill available page frames with pages) : **(NET-JUNE-2014)**

(A) 9 and 6 **(B)** 10 and 7 **(C)** 9 and 7 **(D)** 10 and 6

Q A job has four pages A, B, C, D and the main memory has two page frames only. The job needs to process its pages in following order : ABACABDBACD. Assuming that a page interrupt occurs when a new page is brought in the main memory, irrespective of whether the page is swapped out or not. The number of page interrupts in FIFO and LRU page replacement algorithms are **(NET-JUNE-2013)**

(A) 9 and 7 **(B)** 7 and 6 **(C)** 9 and 8 **(D)** 8 and 6

Q Consider the virtual page reference string 1, 2, 3, 2, 4, 1, 3, 2, 4, 1 On a demand paged virtual memory system running on a computer system that main memory size of 3 pages frames which are initially empty. Let LRU, FIFO and OPTIMAL denote the number of page faults under the corresponding page replacements policy. Then **(GATE-2012) (2 Marks)**

(A) OPTIMAL < LRU < FIFO

(B) OPTIMAL < FIFO < LRU

(C) OPTIMAL = LRU

(D) OPTIMAL = FIFO

Q A process, has been allocated 3-page frames. Assume that none of the pages of the process are available in the memory initially. The process makes the following sequence of page references (reference string): 1,2,1,3,7,4,5,6,3,1 Least Recently Used (LRU) page replacement policy is a practical approximation to optimal page replacement. For the above reference string, how many more page faults occur with LRU than with the optimal page replacement policy? **(GATE-2007) (2 Marks)**

(A) 0 **(B)** 1 **(C)** 2 **(D)** 3

Break

Solution: Two implementations are feasible:

- **Counters**. In the simplest case, we associate with each page-table entry a time-of-use field and add to the CPU a logical clock or counter.

- The clock is incremented for every memory reference. Whenever a reference to a page is made, the contents of the clock registers are copied to the time-of-use field in the page-table entry for that page.

- In this way, we always have the "time" of the last reference to each page. We replace the page with the smallest time value.

- **Stack:** Another approach to implementing LRU replacement is to keep a stack of page numbers.

- Whenever a page is referenced, it is removed from the stack and put on the top.

- In this way, the most recently used page is always at the top of the stack and the least recently used page is always at the bottom.

Counting-Based Page Replacement

- We keep a counter of the number of references that have been made to each page and develop the following two schemes.

- The **least frequently used (LFU)** page-replacement algorithm requires that the page with the smallest count be replaced.

- The reason for this selection is that an actively used page should have a large reference count.

- A problem arises, however, when a page is used heavily during the initial phase of a process but then is never used again. Since it was used heavily, it has a large count and remains in memory even though it is no longer needed.

- One solution is to shift the counts right by 1 bit at regular intervals, forming an exponentially decaying average usage count.

- **The most frequently used (MFU)** page-replacement algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

- Neither MFU nor LFU replacement is common. The implementation of these algorithms is expensive, and they do not approximate OPT replacement well.

Q Consider a computer system with ten physical page frames. The system is provided with an access sequence (a_1,a_2,…,a_{20},a_1,a_2,…,a_{20}), where each a_i is a distinct virtual page number. The difference in the number of page faults between the last-in-first-out page replacement policy and the optimal page replacement policy is _____. **(GATE-2016) (1 Marks)**

Q Consider a main memory with 3 page frames for the following page reference string : 5, 4, 3, 2, 1, 4, 3, 5, 4, 3, 4, 1, 4. Assuming that the execution of process is initiated after loading page 5 in memory, the number of page faults in FIFO and second chance replacement respectively are **(NET-SEP-2013)**

(A) 8 and 9 **(B)** 10 and 11 **(C)** 7 and 9 **(D)** 9 and 8

Q A computer has twenty physical page frames which contain pages numbered 101 through 120. Now a program accesses the pages numbered 1, 2, …, 100 in that order, and repeats the access sequence THRICE. Which one of the following page replacement policies experiences the same number of page faults as the optimal page replacement policy for this program? **(GATE-2014) (2 Marks)**

(A) Least-recently-used

(B) First-in-first-out

(C) Last-in-first-out

(D) Most-recently-used

Q Increasing the RAM of a computer typically improves performance because **(GATE-2005) (1 Marks) (ISRO-2015)**

(A) Virtual memory increases

(B) Larger RAMs are faster

(C) Fewer page faults occur

(D) Fewer segmentation faults occur

Break

Frame Allocation Algorithms

- Minimum Number of frames to be allocated to each process depends on Instruction Set Architecture, and the maximum number of allocation of frames depends on the size of the process.

- **Equal Allocation:** Frames will be equally allocated to each process. Ex: if we have 30 frames and 3 processes, each will get 10 regardless of their size.

- **<u>Weighted / Proportional Allocation:</u>** Frames will be allocated according to the weights of the process. The allocation of frames per process: **a(i)= s(i)/S x m.**

- Where, a(i) is the number of allocated frames, s(i) is the process and S is the size of the virtual memory in general, **S = Σ s(i),** and m is the total number of frames available.

- Example: If we have 3 processes of size 20 k, 30k and 50 k then 30 frames will be allocated as:

- P1 gets: 20/100*30 = 6 frames, similarly P2 and P3 will get 9 and 15 frames respectively.

Some points to remember

- If the multiprogramming level is increased, each process will lose some frames to provide the memory needed for the new process.

- Conversely, if the multiprogramming level decreases, the frames that were allocated to the departed process can be spread over the remaining processes.

Global vs Local Frame Allocation

With multiple processes competing for frames, we can classify page-replacement algorithms into two broad categories:

- **Global Replacement**
- **Local Replacement**

Global Replacement: Global replacement allows a process to select a replacement frame from the set of all frames, even if that frame is currently allocated to some other process; that is, one process can take a frame from another.

- With global replacement, a process may happen to select only frames allocated to other processes, thus increasing the number of frames allocated to it.

- One problem with a global replacement algorithm is that a process cannot control its own page-fault rate. As The set of pages in memory for a process depends not only on the paging behaviour of that process but also on the paging behaviour of other processes.

- Global replacement generally results in greater system throughput and is therefore the more common method.

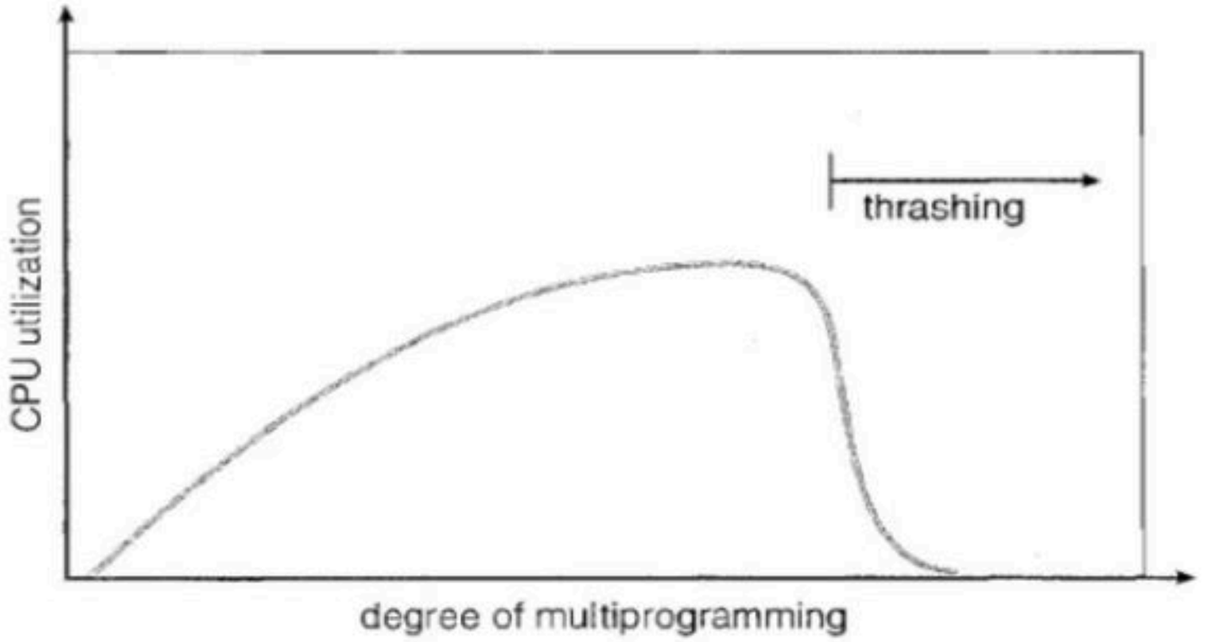
Local Replacement: Local replacement requires that each process select from only its own set of allocated frames.

- With a local replacement strategy, the number of frames allocated to a process does not change.
- Under local replacement, the set of pages in memory for a process is affected by the paging behaviour of only that process.
- Local replacement might hinder a process, however, by not making available to it other, less used pages of memory.

Break

Thrashing

- A process is thrashing if it is spending more time paging than executing. High paging activity is called Thrashing.



Causes of Thrashing

- **High Degree of Multiprogramming:** If CPU utilization is too low, we increase the degree of multiprogramming by introducing a new process to the system.

- A global page-replacement algorithm is used; it replaces pages without regard to the process to which they belong. Now if that a process enters a new phase in its execution and needs more frames. It will start faulting and taking frames away from other processes.

- These processes need those pages, however, and so they also fault, taking frames from other processes. These faulting processes must use the paging device to swap pages in and out. As they queue up for the paging device, the ready queue empties. As processes wait for the paging device, CPU utilization decreases.

- The CPU scheduler sees the decreasing CPU utilization and increases the degree of multiprogramming as a result.

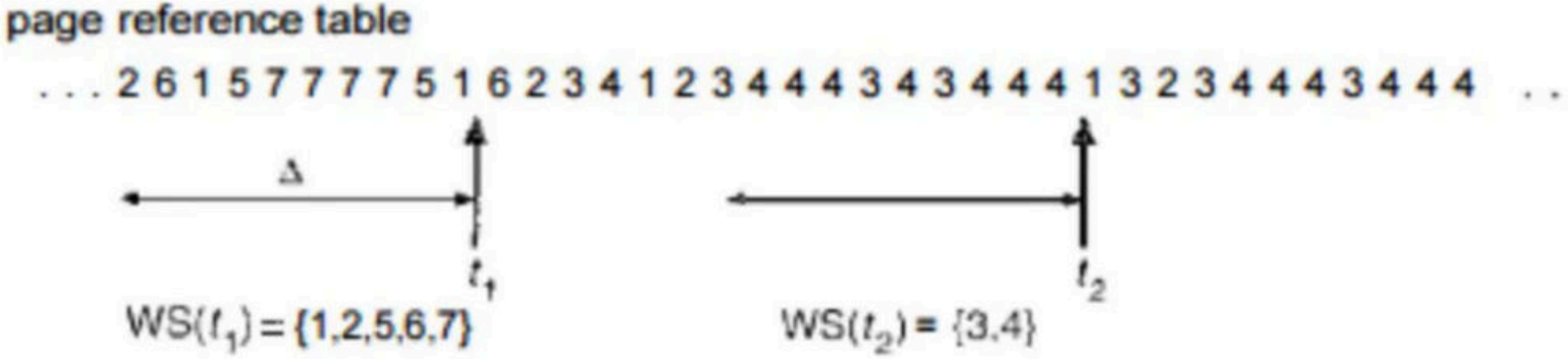
- The new process tries to get started by taking frames from running processes, causing more page faults and a longer queue for the paging device. As a result, CPU utilization drops even further, and the CPU scheduler tries to increase the degree of multiprogramming even more.
-
- Thrashing has occurred, and system throughput plunges. The page fault rate increases tremendously. As a result, the effective memory-access time increases. No work is getting done, because the processes are spending all their time paging.

Solution

- With local replacement, if one process starts thrashing, it cannot steal frames from another process and cause the latter to thrash as well.
- To prevent thrashing, we must provide a process with as many frames as it needs.
- We decide how to allocate the number of frames to a process by **the working-set strategy.**

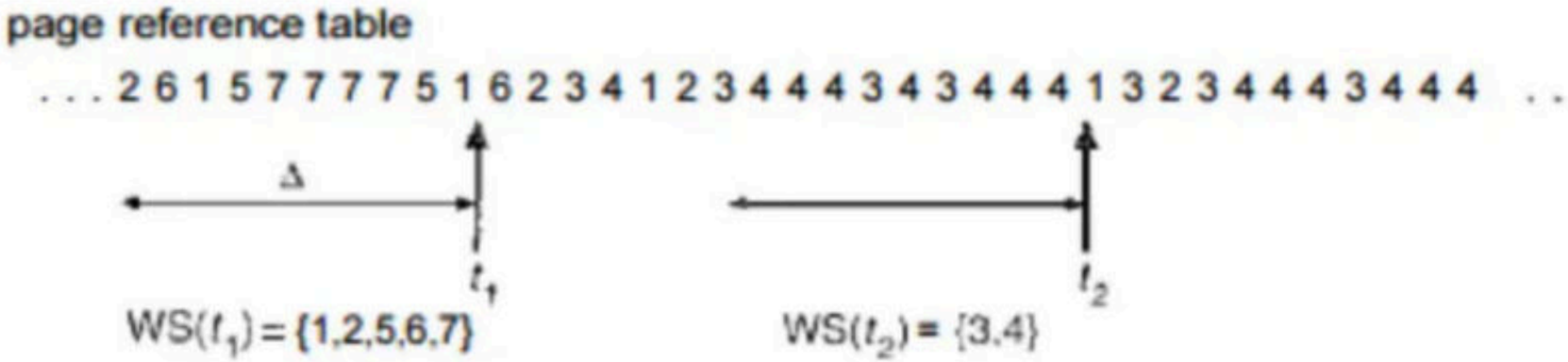
The Working Set Strategy

- This approach defines the **locality model** of process execution.
- The locality model states that, as a process executes, it moves from locality to locality. A locality is a set of pages that are actively used together
- A program is generally composed of several different localities, which may overlap.



Working Set Model

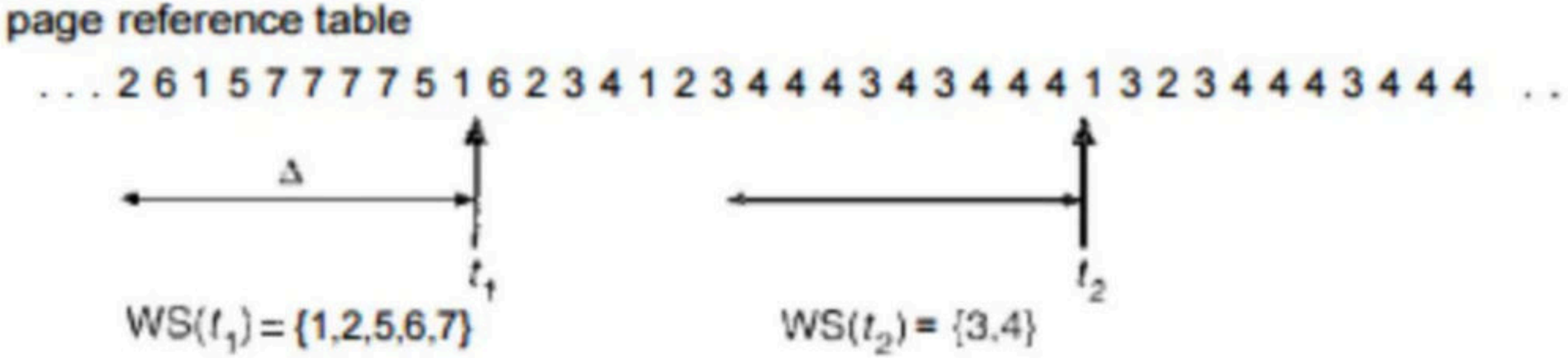
- This model uses a parameter Δ, to define the **working set window.**
- The set of pages in the most recent Δ page references is the working set.
- If a page is in active use, it will be in the working set. If it is no longer being used, it will drop from the working set Δ time units after its last reference.
- The working set is an approximation of the program's locality.



- The accuracy of the working set depends on the selection of Δ . If Δ is too small, it will not encompass the entire locality; if Δ is too large, it may overlap several localities.

- If we can calculate the Working-Set Size (WSS) for each process then: $D = \Sigma WSS_i$, where D is D is the total demand for frames.

- If the total demand is greater than the total number of available frames (D > m), thrashing will occur, because some processes will not have enough frames.



Break

Page-Fault Frequency

- It is a much more direct strategy to control **Page-Fault Frequency.**
- Thrashing has a high page-fault rate and thus we want to control the page-fault rate.
- When it is too high, we know that the process needs more frames. Conversely, if the page-fault rate is too low, then the process may have too many frames.

- **We establish upper and lower bounds on the desired page-fault rate.**

- If the actual page-fault rate exceeds the upper limit, we allocate the process another frame; if the page-fault rate falls below the lower limit, we remove a frame from the process. **With the working-set strategy, we may have to suspend a process. If the page-fault rate increases and no free frames are available, we must select some process and suspend it.**

Q Working set model is used in memory management to implement the concept of **(NET-JUNE-2013)**

(A) Swapping

(B) Principal of Locality

(C) Segmentation

(D) Thrashing