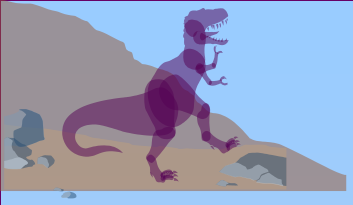
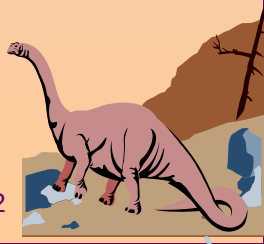
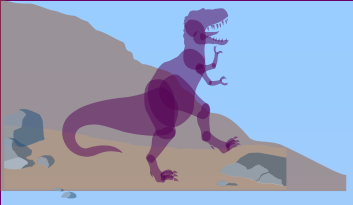


FRAGMENTATION

&

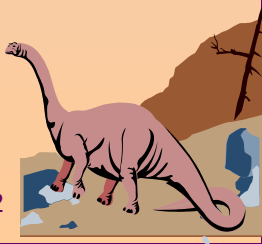
PAGING

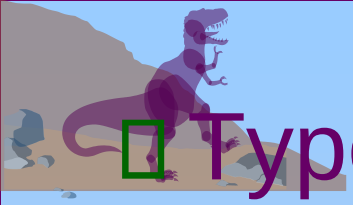




Fragmentation

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





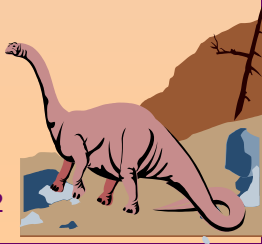
Types of Fragmentation

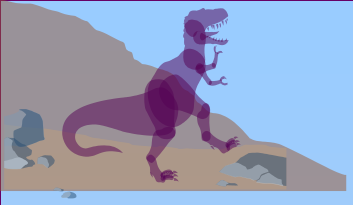
External fragmentation

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

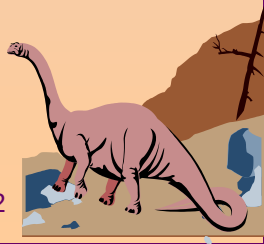
Internal fragmentation

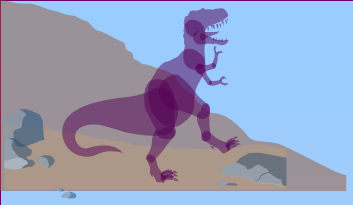
- Memory block assigned to process is bigger. Some portion of memory is left unused, as it cannot be used by another process.





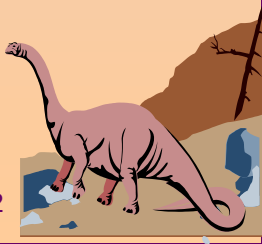
SO 1: Define Paging

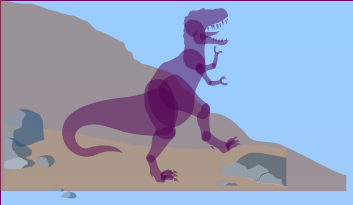




Paging

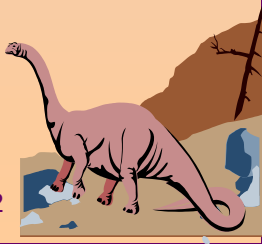
- In a Computer Operating system paging is a memory management scheme by which a computer stores and retrieve a data from secondary storage for use in main memory.
- In this scheme a operating system retrieve a data from secondary storage from same size-block is called as pages.
- The paging is a very important part of virtual memory implementation in operating system ,using secondary storage to let program exceeds the size of available physical memory.
- Non contiguous memory allocation
- Help to prevent external fragmentation.
- Logical address space is divided into equal size pages.
- Physical address space is divided into equal size frames.
- Page size = Frame size.



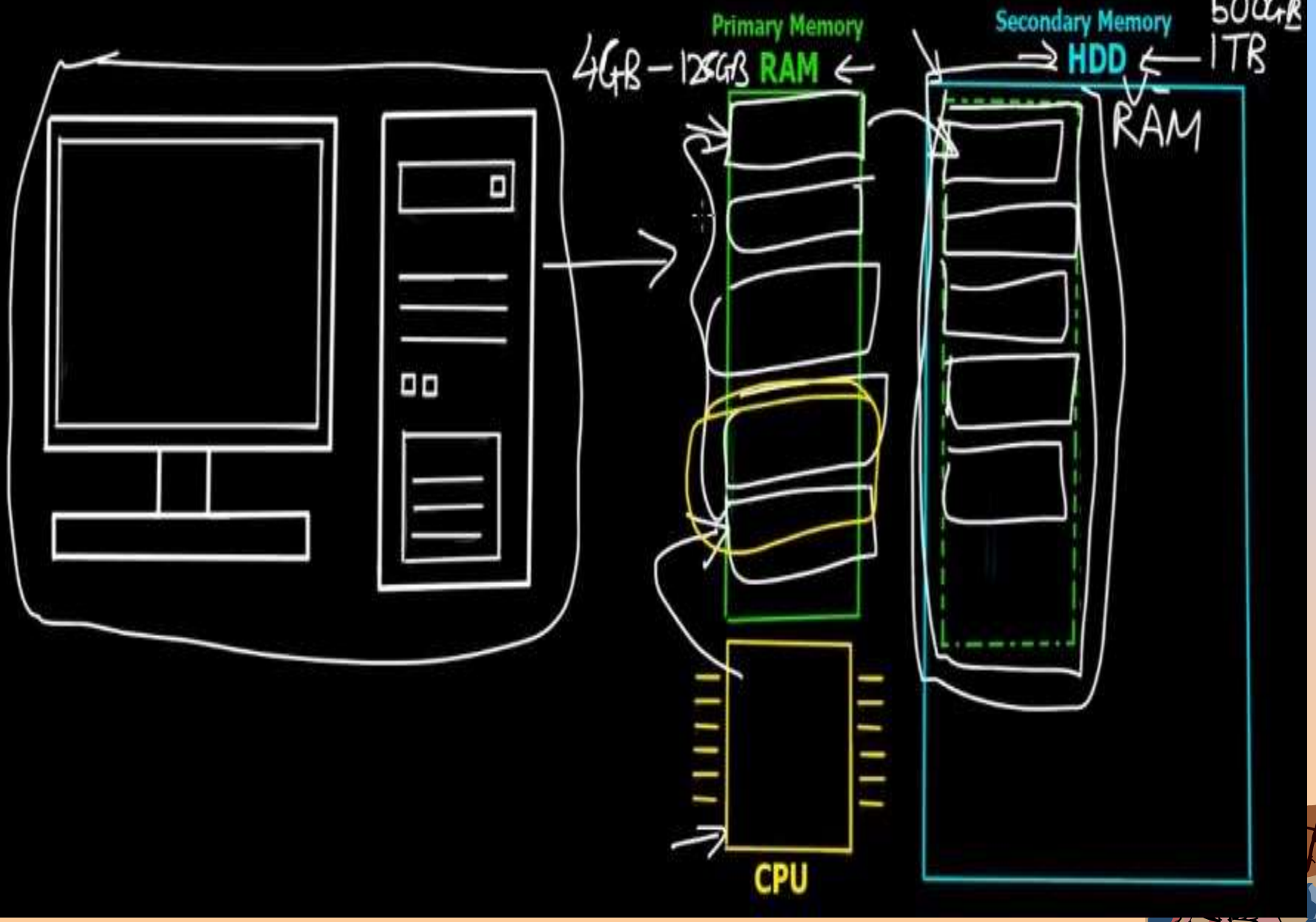


Paging

- Logical Address or virtual address (represented in bits) generated by CPU.
- Physical Address (represented in bits) actually available in memory unit.
- The Mapping from logical address to physical address is done by memory management unit(MMU) which is a hardware device and this mapping is known PAGING Technique.



Paging in Operating Systems - Memory Management



Paging in Operating Systems - Memory Management

Paging Overview -

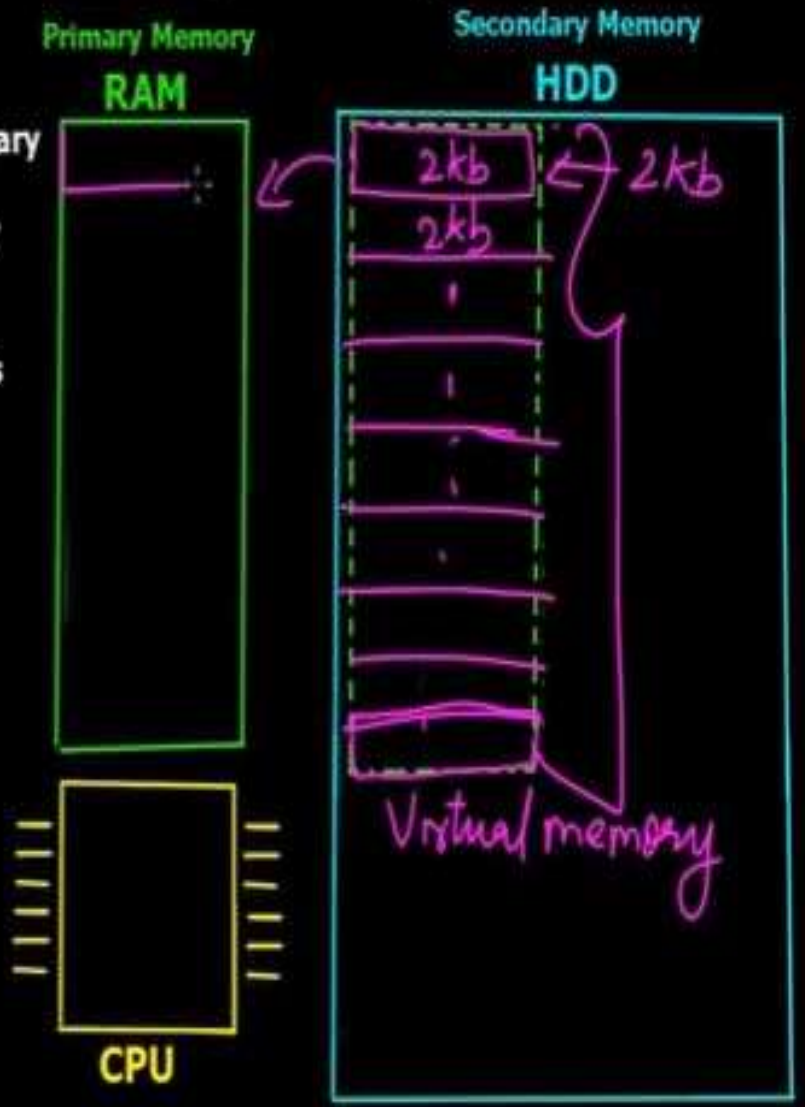
1. In computer operating systems, paging is a memory management scheme by which a computer stores and retrieves data from secondary storage for use in main memory.
2. In this scheme, the operating system retrieves data from secondary storage in same-size blocks called pages.
3. Paging is an important part of virtual memory implementations in modern operating systems, using secondary storage to let programs exceed the size of available physical memory.
4. Non-contiguous memory allocation
5. Helps prevent external fragmentation

6. Logical address space is divided into equal size pages
7. physical address space is divided into equal size frames
8. Page Size = Frame Size

>> Logical Address or Virtual Address (represented in bits) -
An address generated by the CPU

>> Physical Address (represented in bits) -
An address actually available on memory unit

>> The mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device and this mapping is known as paging technique.



Paging in Operating Systems - Memory Management

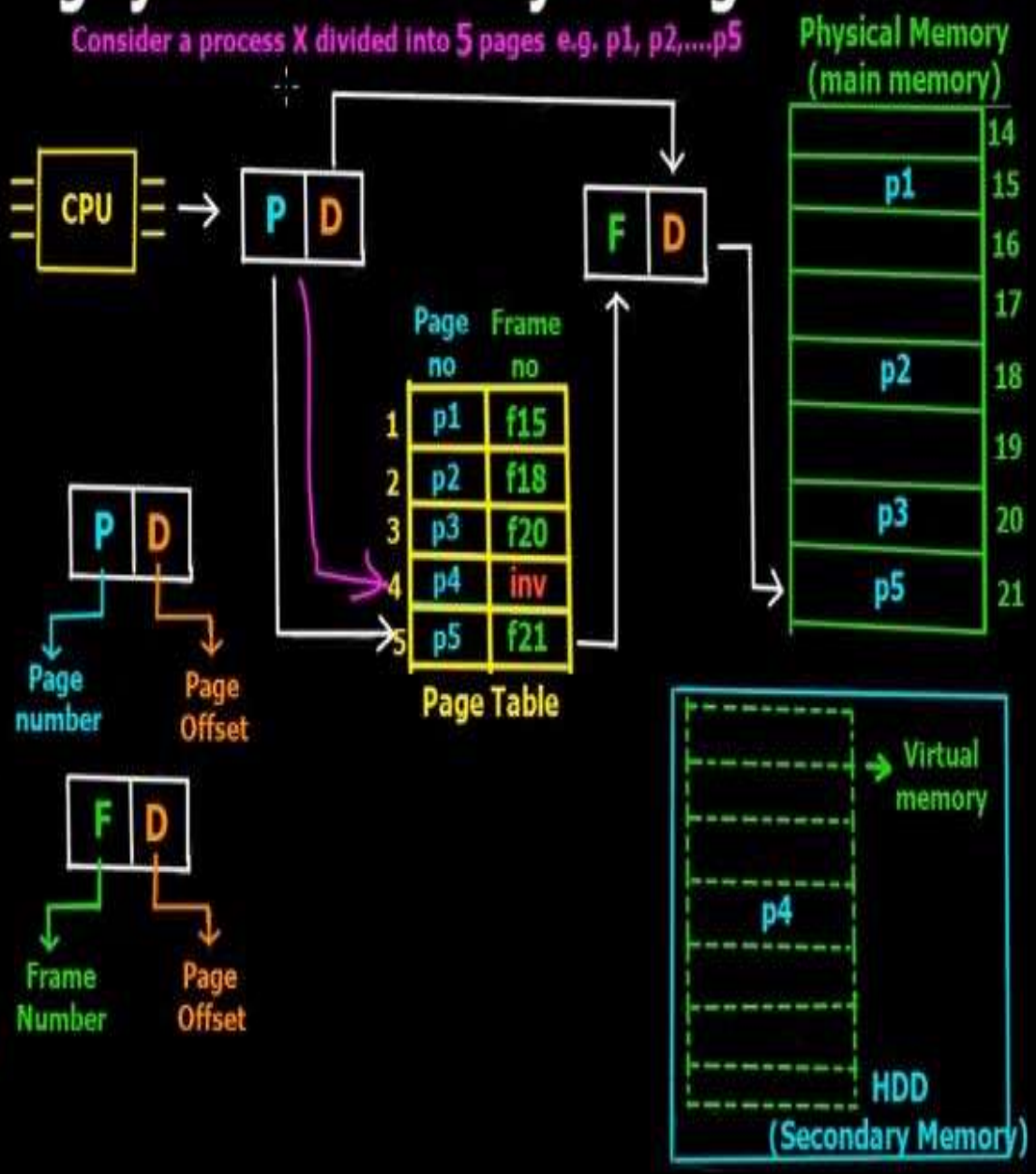
Address generated by CPU is divided into -

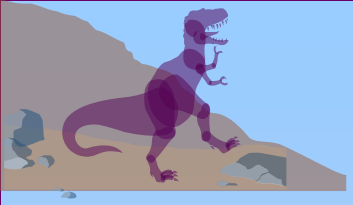
1. Page number (P) – used as an index into a page table which contains base address of each page in physical memory
2. Page offset (D) – combined with base address to define the physical memory address that is sent to the memory unit

Physical address is divided into -

1. Frame number (P) – used as an index into physical memory where process frame is located
2. Frame offset (D) – combined with base address to define the physical memory address that is sent to the memory unit

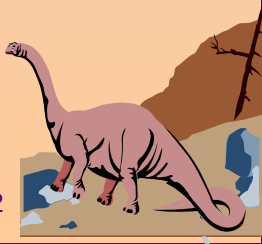
>> Every Process has its own Page Table
>> Process Table is stored in Main Memory (physical memory)

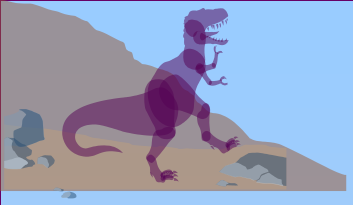




Paging

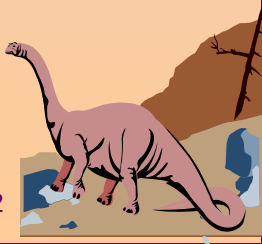
- Paging is a memory management scheme that permits the physical address space of a process to be noncontiguous.
- Physical memory is broken into fixed sized blocks called frames
- Logical memory is also broken into blocks of the same size called pages
- When a process is to be executed, its pages are loaded into any available memory frames.



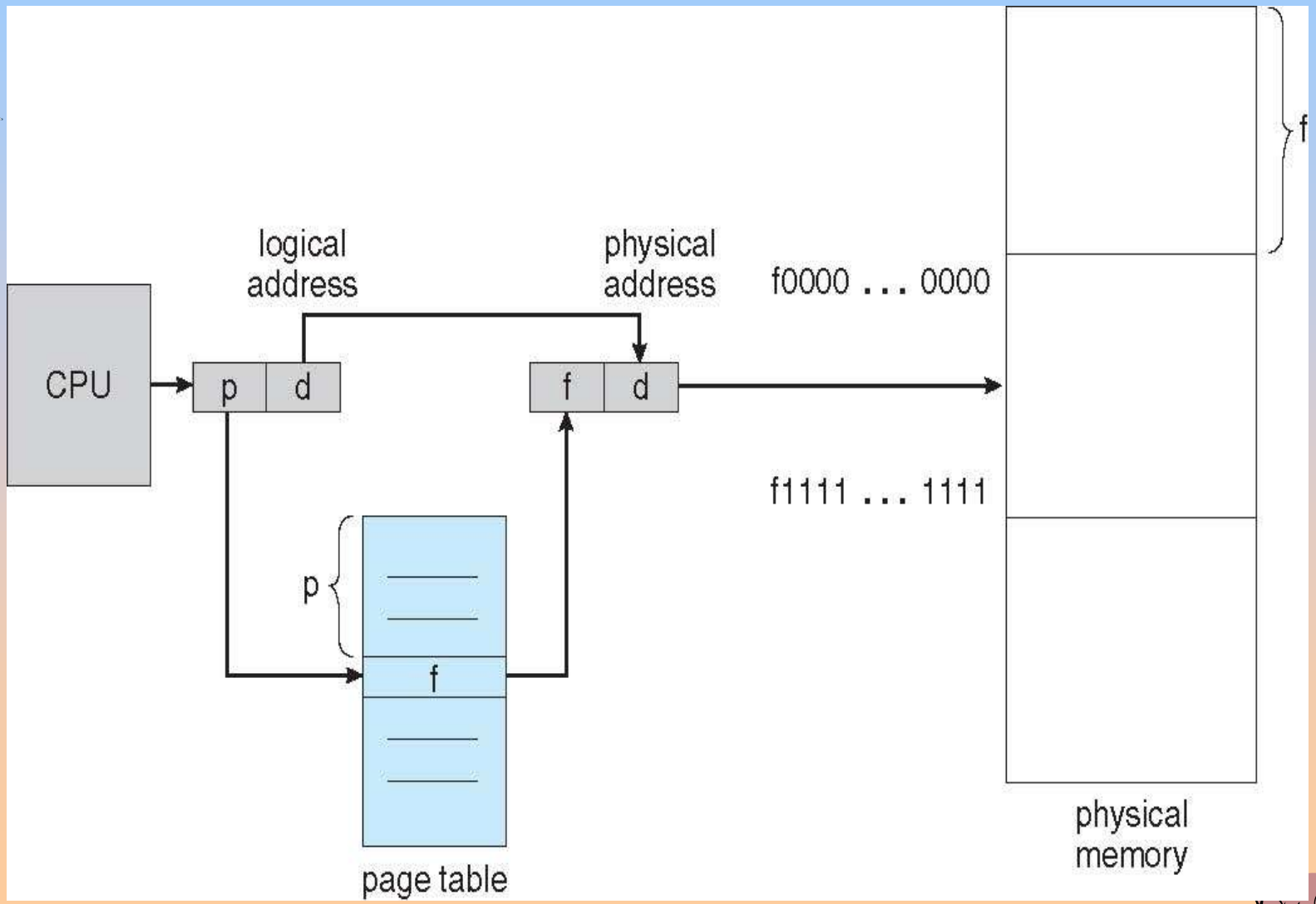


Paging

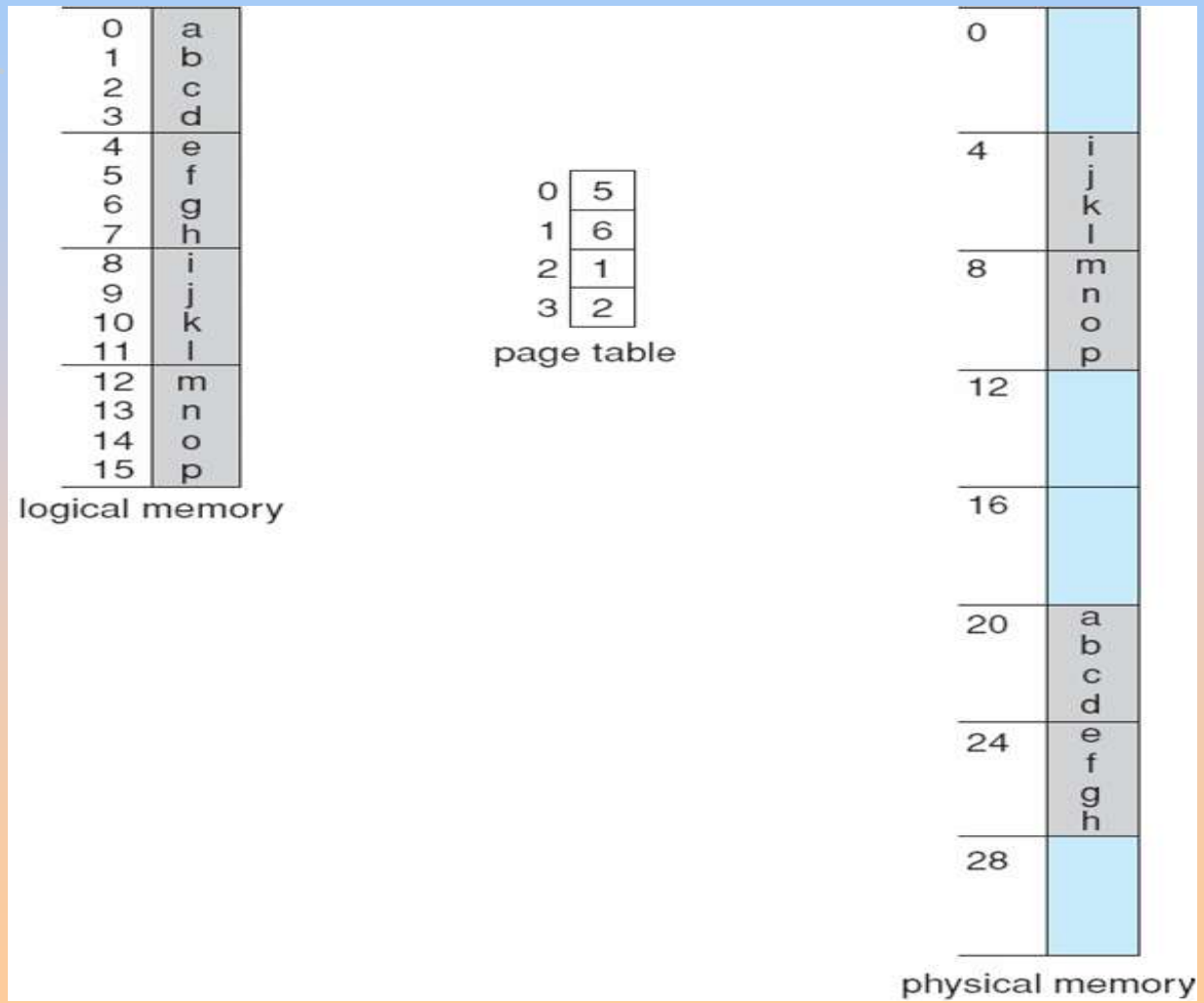
- Every address generated by the CPU is divided into two parts: **page number(p)** and **page offset(d)**
- The page number is used as an index into a page table
- The page table contains the base address of each page in physical memory
- This base address is combined with the page offset to define the physical memory address.
- The page size is defined by the hardware
- The size of a page is typically a power of 2, varying between 512bytes and 16MB per page

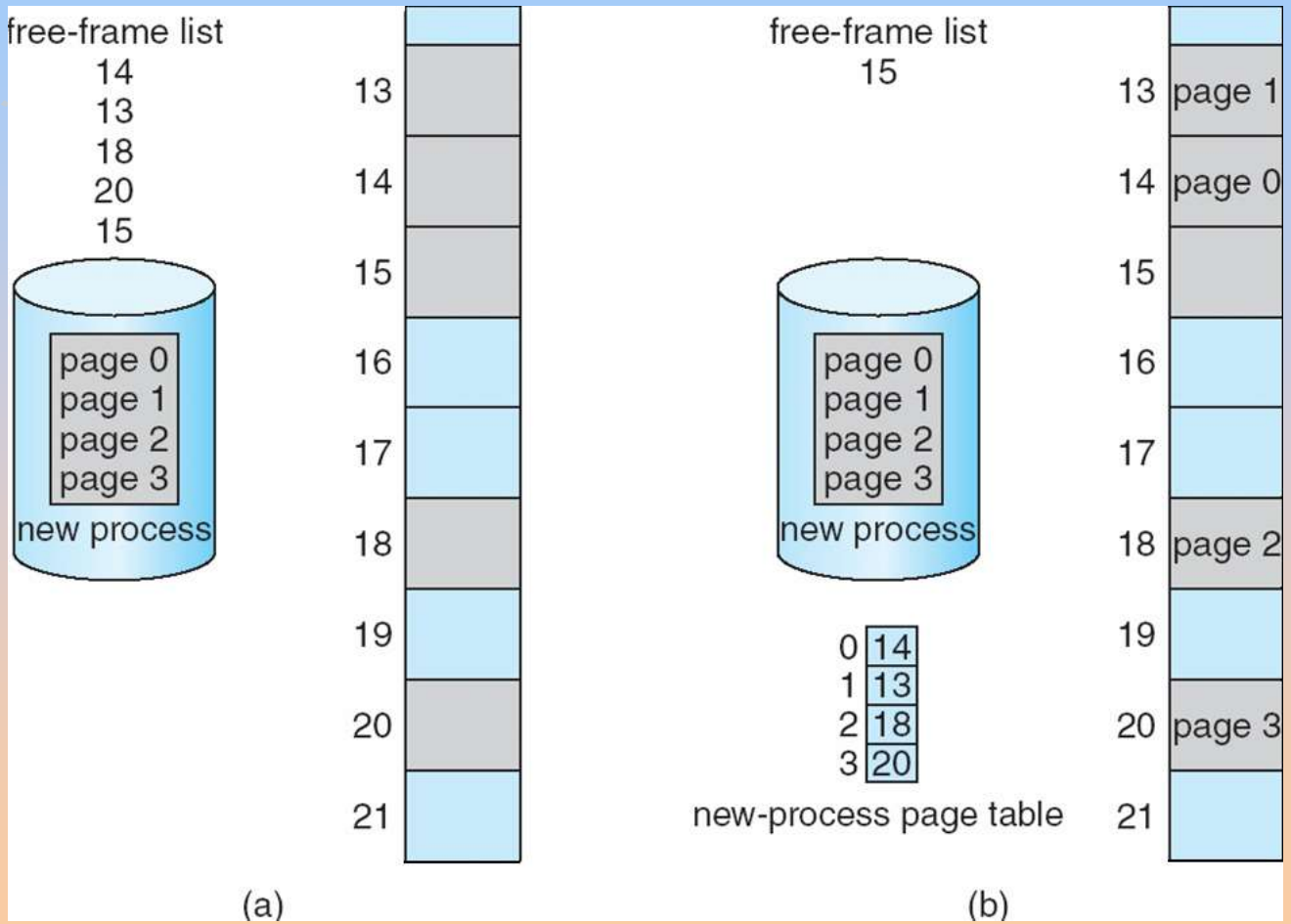


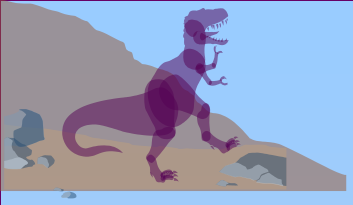
Address Translation



Paging example for 32-byte memory with 4-byte pages

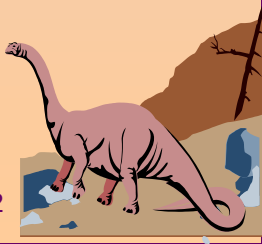


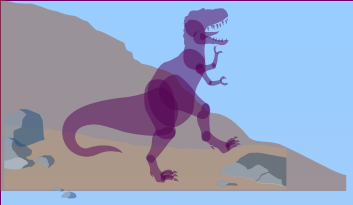

free-frame list
14
13
18
20
15
page 0
page 1
page 2
page 3
new process
13
14
15
16
17
18
19
20
21
(a)
free-frame list
15
page 0
page 1
page 2
page 3
new process
0 14
1 13
2 18
3 20
new-process page table
13 page 1
14 page 0
15
16
17
18 page 2
19
20 page 3
21
(b)



Paging

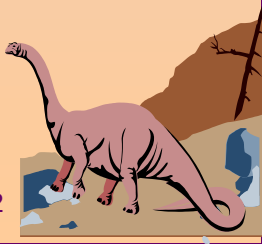
- When we use a paging scheme, we have no external fragmentation
- Any free frame can be allocated to a process that needs it.
- However we may have internal fragmentation
- To avoid internal fragmentation, size of the page may be reduced

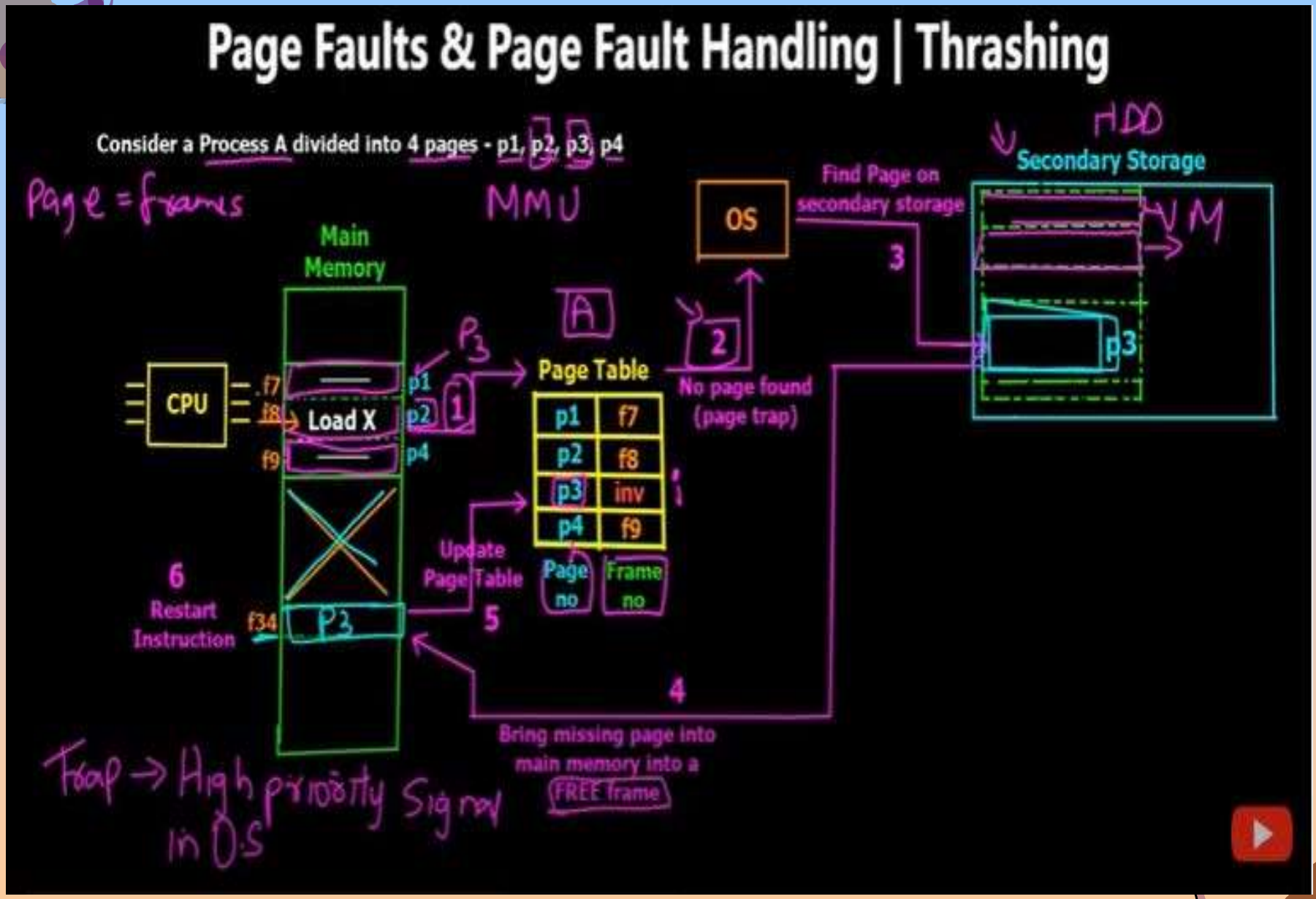




Page Fault

- A page fault occurs when a program attempts to access a block of memory that is not stored in the physical memory, or RAM.
- The fault notifies the operating system that it must locate the data in virtual memory, then transfer it from the storage device, such as an HDD or SSD, to the system RAM.

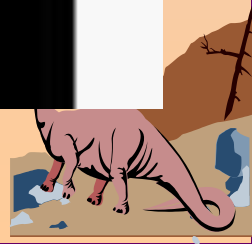


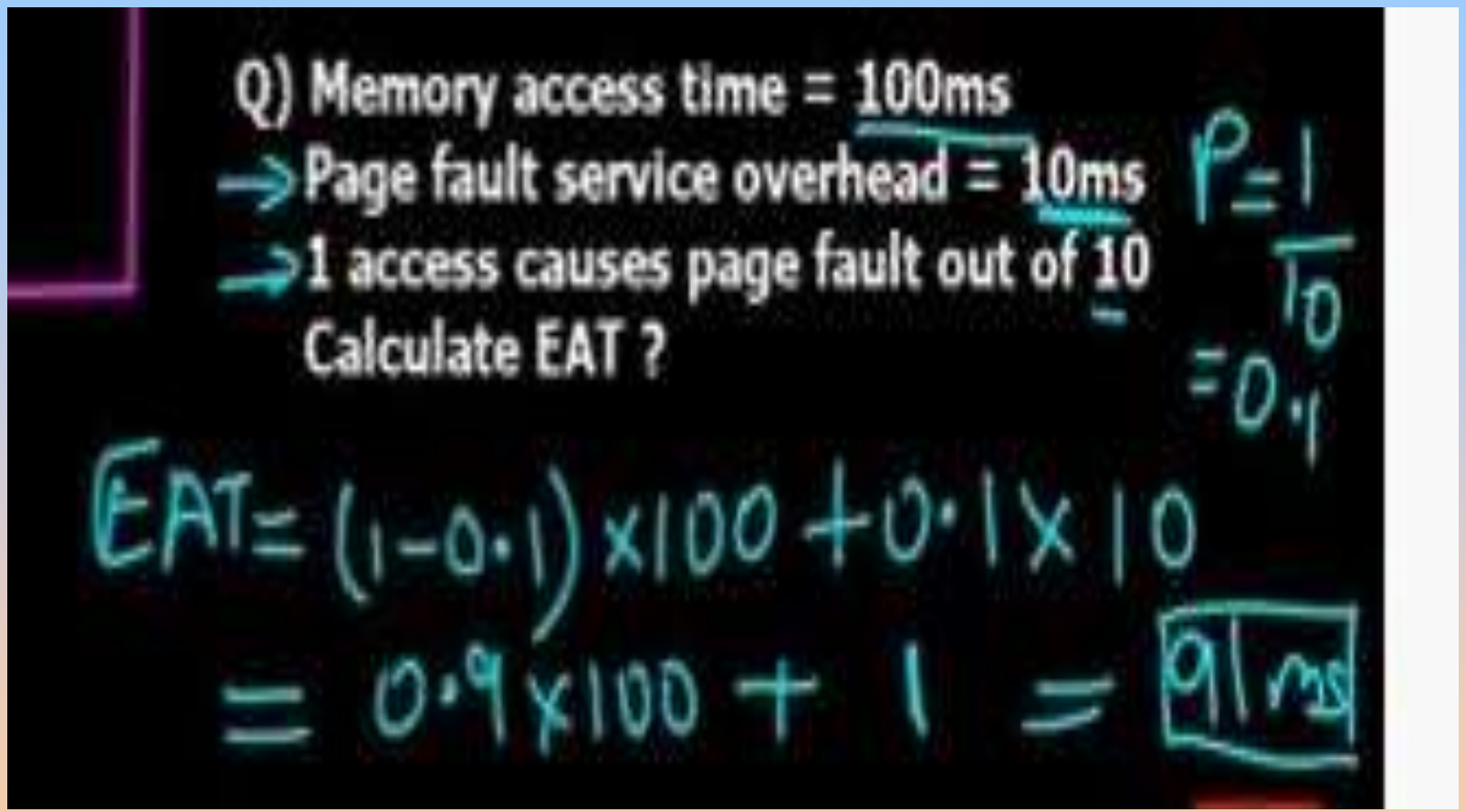

Page Faults & Page Fault Handling | Thrashing
Consider a Process A divided into 4 pages - p1, p2, p3, p4
Page = frames
MMU
HDD
Secondary Storage
Main Memory
OS
Find Page on secondary storage
3
CPU
f7
f8
f9
Load X
p1
p2
p3
p4
P3
1
A
Page Table
2
No page found (page trap)
p1 f7
p2 f8
p3 inv
p4 f9
Page no
Frame no
Update Page Table
5
6
Restart Instruction
f34
P3
4
Bring missing page into main memory into a FREE frame
Trap → High priority Signal in O.S

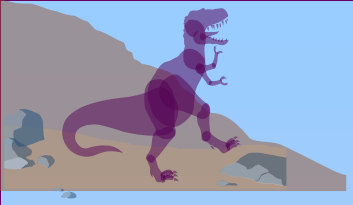
1. 0 <= Page Fault Probability(P)<= 1
2. Effective Access Time (EAT) -

$$EAT = (1-P) \times \text{Memory Acess} + P \times \text{page fault overload}$$

Q) Memory access time = 100ms
Page fault service overhead = 10ms
1 access causes page fault out of 10
Calculate EAT ?




Q) Memory access time = 100ms
→ Page fault service overhead = 10ms
→ 1 access causes page fault out of 10
Calculate EAT ?
p = 1/10 = 0.1
EAT = (1 − 0.1) × 100 + 0.1 × 10
= 0.9 × 100 + 1 = 91 ms



SUMMARY