Memory Management
Lecture 4

Paging Hardware Implementation

Minakshi R.

Operating System Concepts 8th edition silberschatz Galvin

Paging Implementation & Hardware Requirement

- ✔ OS maintains a list of status of all the frames in the memory
- ✔ Process are loaded in free frames and its status are **marked allocated**
- ✔ The address of the page table, where it is stored in the memory is also stored in the PCB of the process
- ✔ This information is useful at the time of execution of a process

Paging Implementation & Hardware Requirement

Paging problem:

- ✔ Increase Cost
- ✔ Increase access time

Solution:

1. Use Fast access Register
2. Keep the page table in memory

   Use a base Register : **Page Table Base Register (PTBR)**

Paging Implementation & Hardware Requirement

Execute a instruction in a page

1. The Page table is accessed
2. The frame address is combined with the offset and the actual frame is accessed

Total memory access time=

Time to access the page table

+

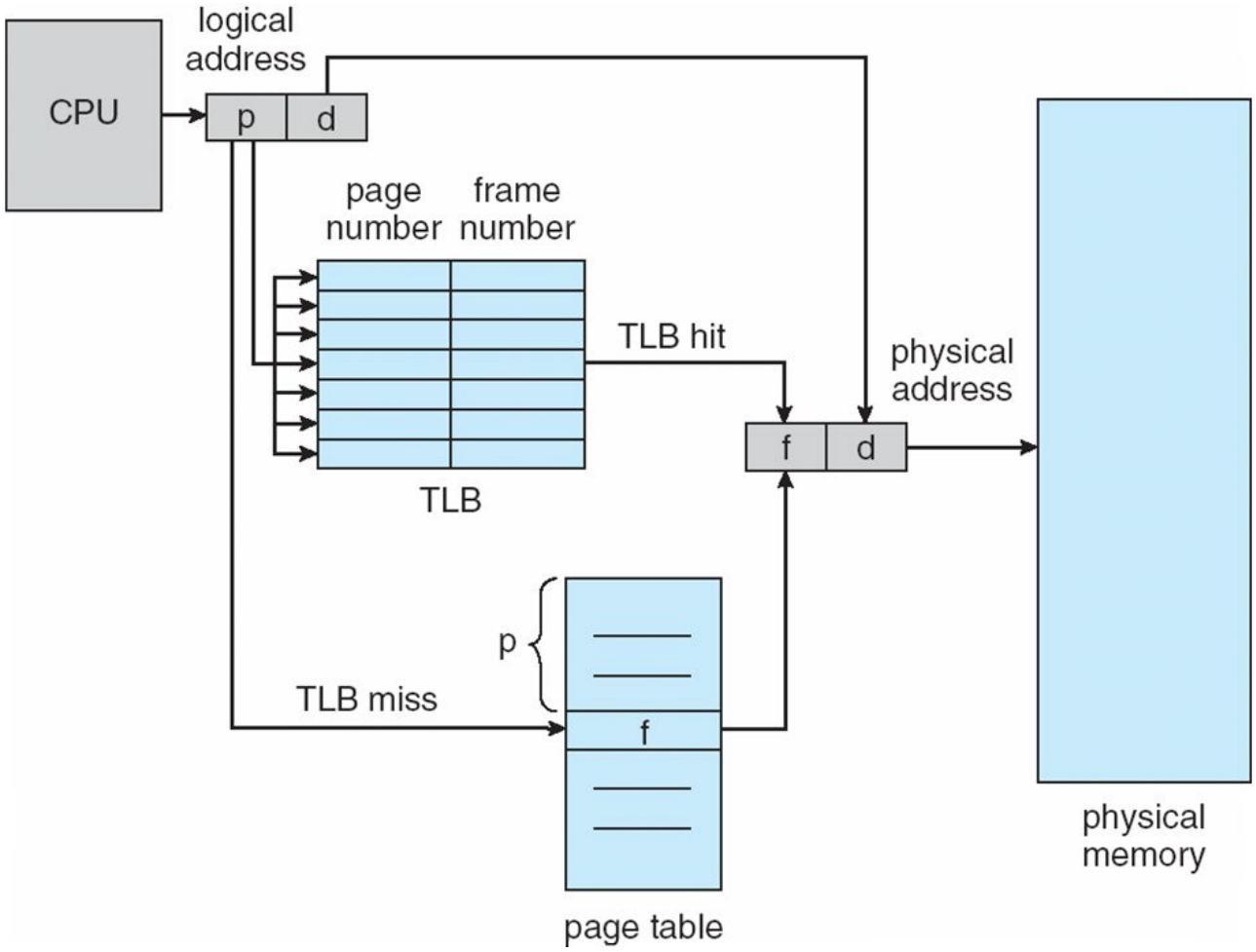
Time to access the memory location

Paging Implementation & Hardware Requirement

How to Reduce this Problem:

- Use of high speed **cache memory**
- This cache memory is also called as Translation Look-aside Buffer(**TLB**)
- TLB consist of page No. & Frame No.
- This type of memory mapping through TLB is called as **Associative mapping**

Paging Hardware With TLB



Paging Implementation & Hardware Requirement

- ✔ TLB Hit : If the page is found
- ✔ TLB Miss : If not

Based on probability of TLB Hit & Miss

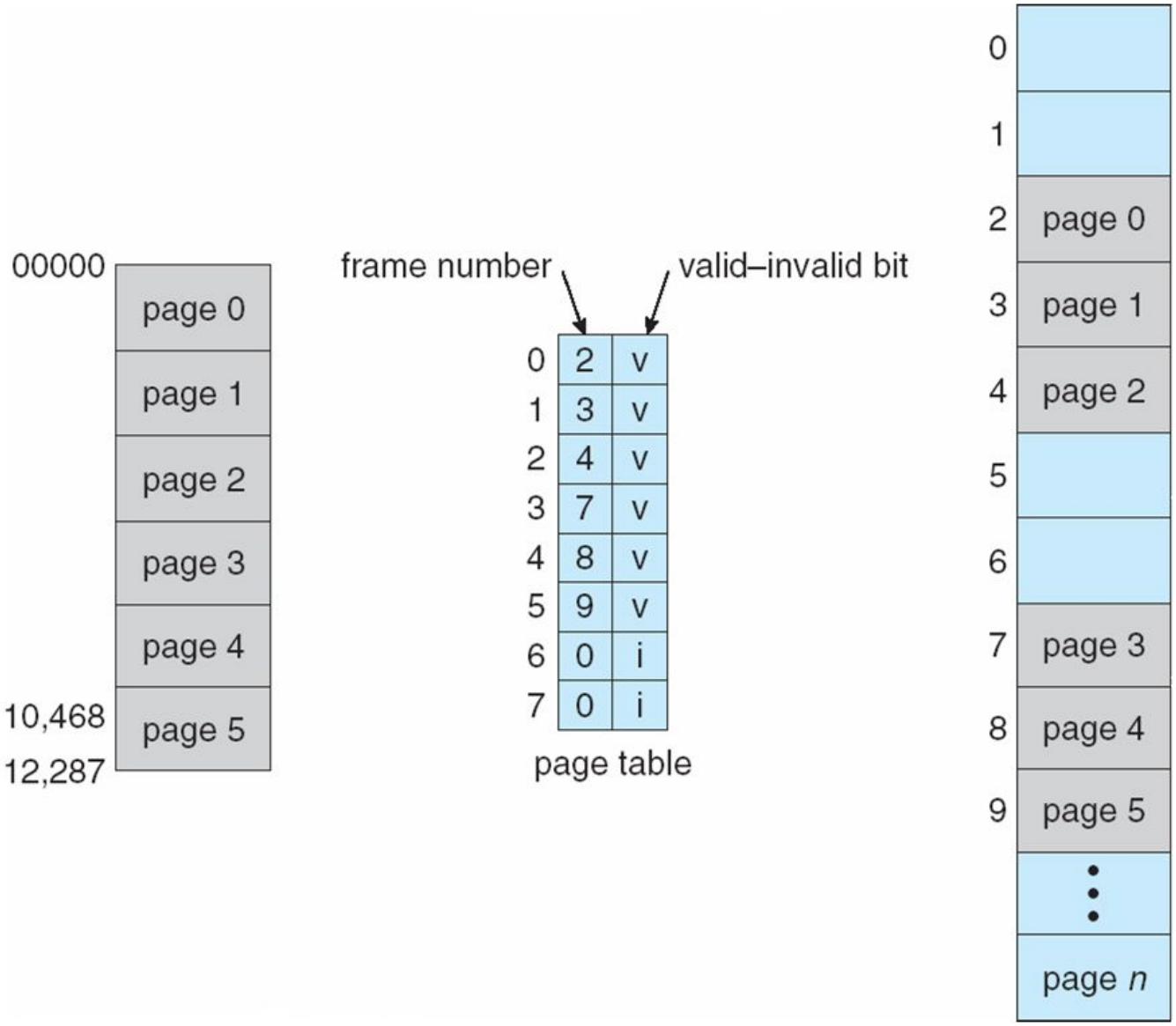
Effective Memory access time=P(H) *(Time to access TLB + Time to access the memory location) +{(1-P(H) * (Time to access TLB + 2(Time to access the memory Location))}

Memory Protection

- ✔ Memory protection implemented by associating protection bit with each frame to indicate if **read-only or read-write access** is allowed
  - Can also add more bits to indicate page execute-only, and so on

- ✔ **Valid-invalid** bit attached to each entry in the page table:
  - "valid" indicates that the associated page is in the process' logical address space, and is thus a legal page
  - "invalid" indicates that the page is not in the process' logical address space
  - Another method to have protection against these valid pages is to have a **Page Table Length Register(PTLR)**

Page Protection



Shared Pages

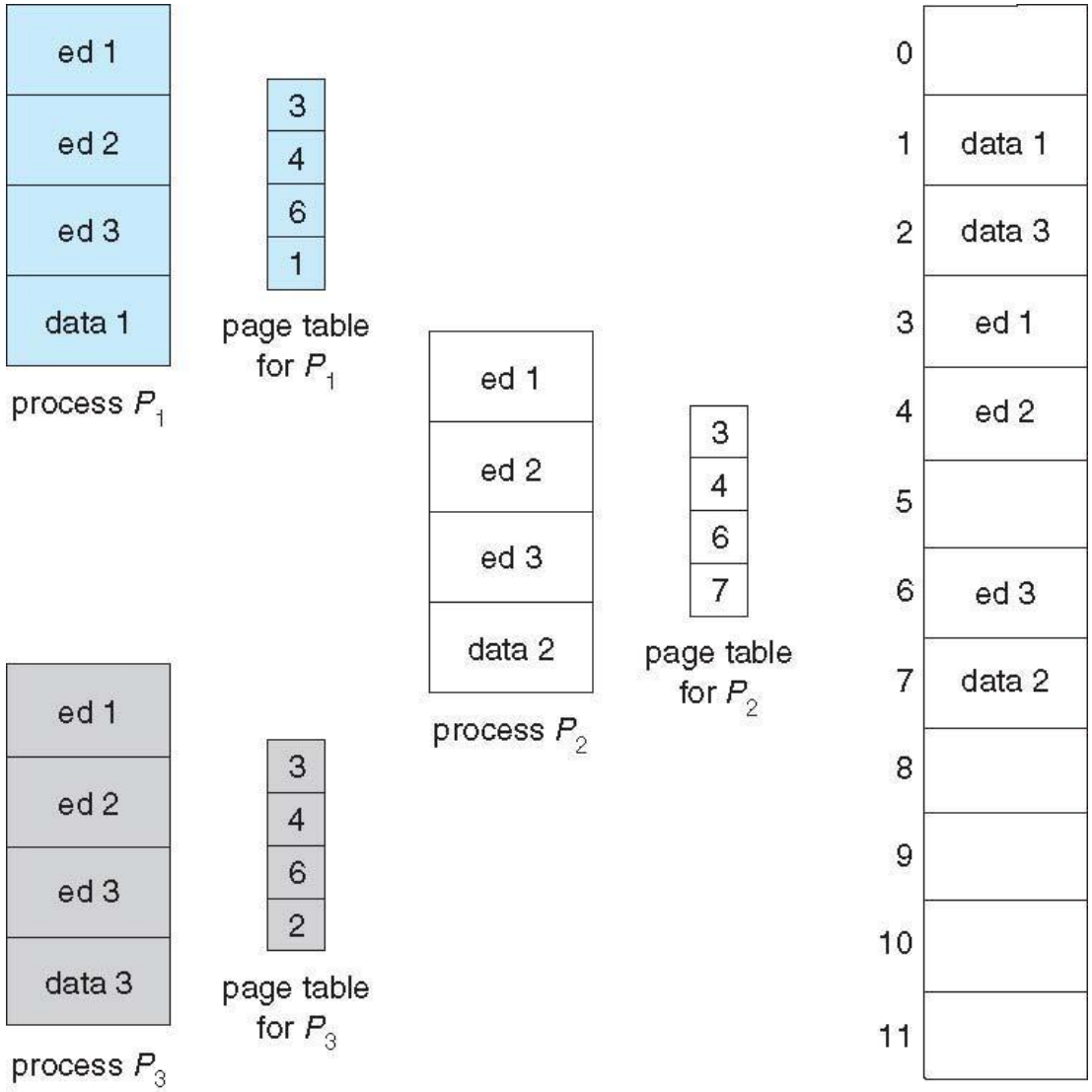
Shared code

- One copy of read-only (**reentrant**) code shared among processes (i.e., text editors, compilers, window systems)
- Similar to multiple threads sharing the same process space
- Also useful for inter-process communication if sharing of read-write pages is allowed

Private code and data

- Each process keeps a separate copy of the code and data
- The pages for the private code and data can appear anywhere in the logical address space

Shared Pages Example



Thank You