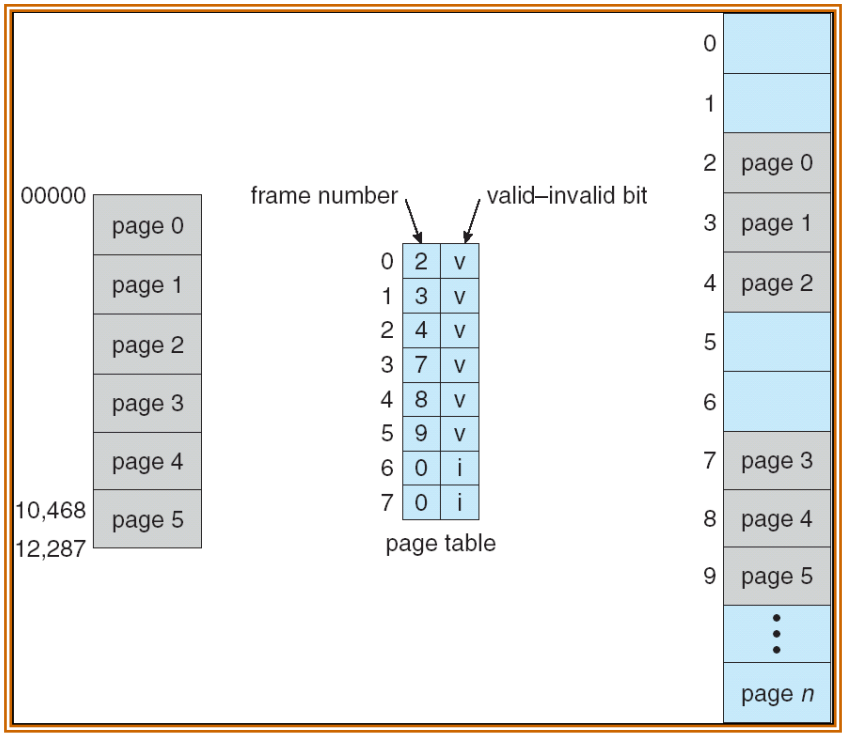
# Protection

- Memory protection in a paged environment is accomplished by protection bits associated with each frame.
- These bits are kept in the page table.
- One bit can define a page to be read-write or read-only.
- Every reference to memory goes through the page table to find the correct frame number.
- At the same time that the physical address is being computed, the protection bits can be checked to verify that no writes are being made to a read-only page.

- An attempt to write to a read-only page causes a hardware trap to the operating system.
- One additional bit is generally attached to each entry in the page table: a **valid-invalid bit.**
- When this bit is set to "valid," the associated page is in the process's logical address space and is thus a legal (or valid) page.
- When the bit is set to "invalid," the page is not in the process's logical address space.

- Illegal addresses are trapped by use of the valid-invalid bit.
- The operating system sets this bit for each page to allow or disallow access to the page.

# Valid (v) or Invalid (i) Bit In A Page Table

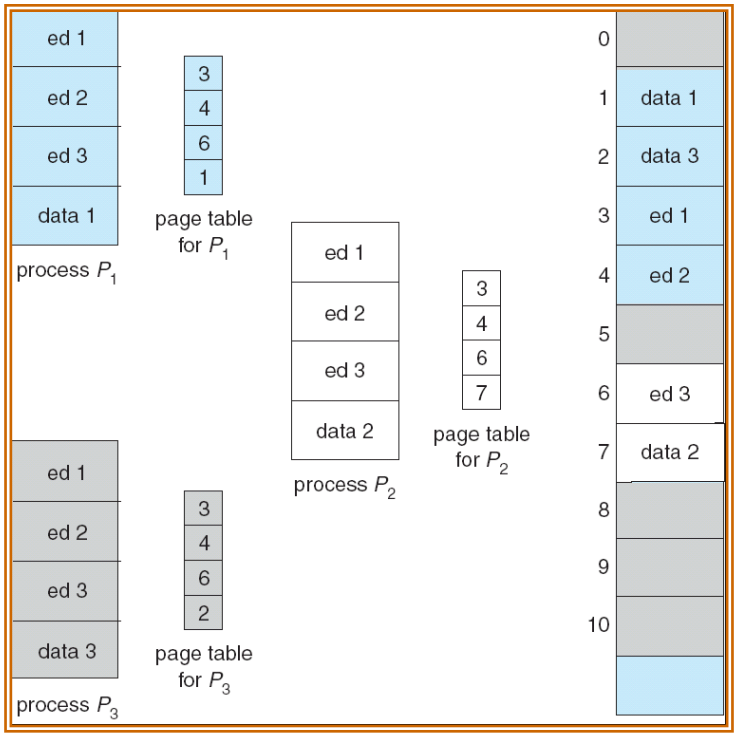


# Shared Pages

- An advantage of paging is the possibility of **sharing** common code.
- It is particularly important in a time-sharing environment
- One copy of read-only (reentrant) code shared among processes (i.e., text editors, compilers, window systems).
- Shared code must appear in same location in the logical address space of all processes

- Consider a system that supports 40 users, each of whom executes a text editor.
- If the text editor consists of 150 KB of code and 50 KB of data space,
- we need 8,000 KB to support the 40 users.
- If the code is reentrant code (or pure code), it can be shared.

# Sharing of code in a paging environment



- Here three-page editor—each page 50 KB in size being shared among three processes.
- Each process has its own data page.
- Reentrant code never changes during execution.
- Thus, two or more processes can execute the same code at the same time.

- Each process has its own copy of registers and data storage to hold the data for the process's execution.
- The data for two different processes will, be different.
- Only one copy of the editor need to be kept in physical memory.
- Each user's page table maps onto the same physical copy of the editor, but data pages are mapped onto different frames.