ICS 143 - Principles of Operating Systems

Memory Management (Part 1)

Prof. Nalini Venkatasubramanian

nalini@ics.uci.edu

(with slides from Prof. Sani, Silberschatz book slides etc.)

Outline

- Background
- Logical versus Physical Address Space
- Swapping
- Contiguous Allocation
- Paging
- Segmentation
- Segmentation with Paging

Background

- Program must be brought into memory and placed within a process for it to be executed.
- Input Queue - collection of processes on the disk that are waiting to be brought into memory for execution.
- User programs go through several steps before being executed.

Virtualizing Resources

- Physical Reality: Processes/Threads share the same hardware
  - Need to multiplex CPU (CPU Scheduling)
  - Need to multiplex use of Memory (Today)

- Why worry about memory multiplexing?
  - The complete working state of a process and/or kernel is defined by its data in memory (and registers)
  - Consequently, cannot just let different processes use the same memory
  - Probably don't want different processes to even have access to each other's memory (protection)

Important Aspects of Memory Multiplexing

- Controlled overlap:
  - Processes should not collide in physical memory
  - Conversely, would like the ability to share memory when desired (for communication)
- Protection:
  - Prevent access to private memory of other processes
    - Different pages of memory can be given special behavior (Read Only, Invisible to user programs, etc)
    - Kernel data protected from user programs
- Translation:
  - Ability to translate accesses from one address space (virtual) to a different one (physical)
  - When translation exists, process uses virtual addresses, physical memory uses physical addresses

Names and Binding

- Symbolic names → Logical names → Physical names
  - Symbolic Names: known in a context or path
    - file names, program names, printer/device names, user names
  - Logical Names: used to label a specific entity
    - inodes, job number, major/minor device numbers, process id (pid), uid, gid..
  - Physical Names: address of entity
    - inode address on disk or memory
    - entry point or variable address
    - PCB address

Binding of instructions and data to memory

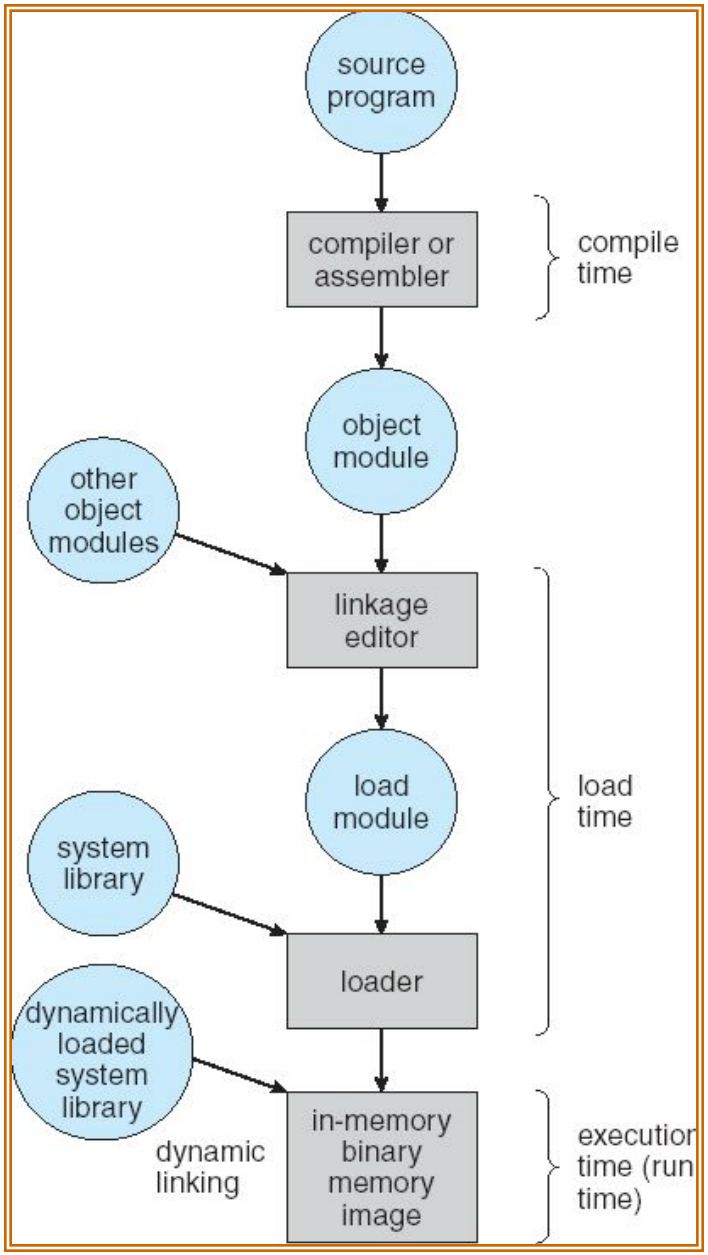
- Address binding of instructions and data to memory addresses can happen at three different stages.
  - Compile time:
    - If memory location is known a priori, absolute code can be generated; must recompile code if starting location changes.
  - Load time:
    - Must generate relocatable code if memory location is not known at compile time.
  - Execution time:
    - Binding delayed until runtime if the process can be moved during its execution from one memory segment to another. Need hardware support for address maps (e.g. base and limit registers).

Binding time tradeoffs

- Early binding
  - compiler - produces efficient code
  - allows checking to be done early
  - allows estimates of running time and space
- Delayed binding
  - Linker, loader
  - produces efficient code, allows separate compilation
  - portability and sharing of object code
- Late binding
  - VM, dynamic linking/loading, overlaying, interpreting
  - code less efficient, checks done at runtime
  - flexible, allows dynamic reconfiguration

Multi-step Processing of a Program for Execution

- Preparation of a program for execution involves components at:
  - Compile time (i.e., “gcc”)
  - Link/Load time (unix “ld” does link)
  - Execution time (e.g. dynamic libs)
- Addresses can be bound to final values anywhere in this path
  - Depends on hardware support
  - Also depends on operating system
- Dynamic Libraries
  - Linking postponed until execution
  - Small piece of code, *stub*, used to locate appropriate memory-resident library routine
  - Stub replaces itself with the address of the routine, and executes routine



Dynamic Loading

- Routine is not loaded until it is called.
- Better memory-space utilization; unused routine is never loaded.
- Useful when large amounts of code are needed to handle infrequently occurring cases.
- No special support from the operating system is required; implemented through program design.

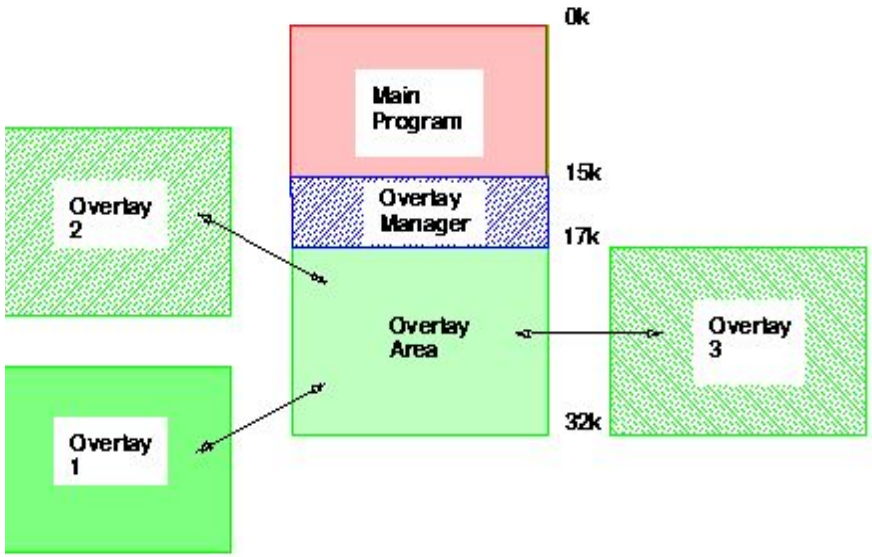
Dynamic Linking

- Linking postponed until execution time.
- Small piece of code, *stub*, used to locate the appropriate memory-resident library routine.
- Stub replaces itself with the address of the routine, and executes the routine.
- Operating system needed to check if routine is in processes' memory address.

Overlays

- Keep in memory only those instructions and data that are needed at any given time.
- Needed when process is larger than amount of memory allocated to it.
- Implemented by user, no special support from operating system; programming design of overlay structure is complex.

Overlaying


0k
Main Program
15k
Overlay Manager
17k
Overlay 2
Overlay Area
Overlay 3
Overlay 1
32k

Logical vs. Physical Address Space

- The concept of a logical address space that is bound to a separate physical address space is central to proper memory management.
  - Logical Address: or virtual address - generated by CPU
  - Physical Address: address seen by memory unit.
- Logical and physical addresses are the same in compile time and load-time binding schemes
- Logical and physical addresses differ in execution-time address-binding scheme.

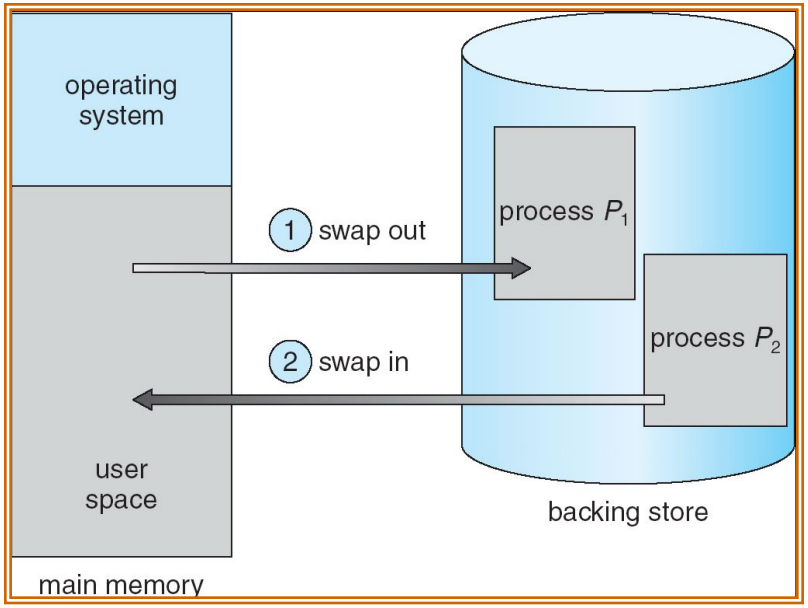
Memory Management Unit (MMU)

- Hardware device that maps virtual to physical address.
- In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory.
- The user program deals with logical addresses; it never sees the real physical address.

Swapping

- A process can be swapped temporarily out of memory to a backing store and then brought back into memory for continued execution.
  - Backing Store - fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images.
  - Roll out, roll in - swapping variant used for priority based scheduling algorithms; lower priority process is swapped out, so higher priority process can be loaded and executed.
  - Major part of swap time is transfer time; total transfer time is directly proportional to the amount of memory swapped.
  - Modified versions of swapping are found on many systems, i.e. UNIX and Microsoft Windows.

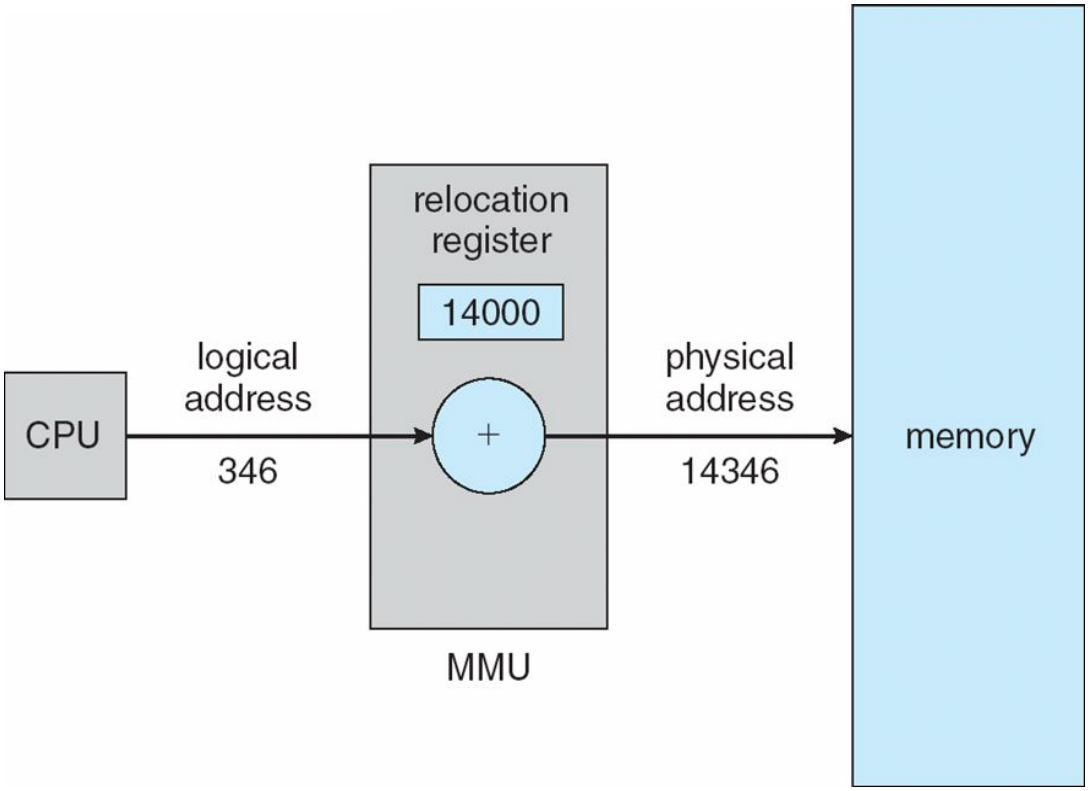
Schematic view of swapping



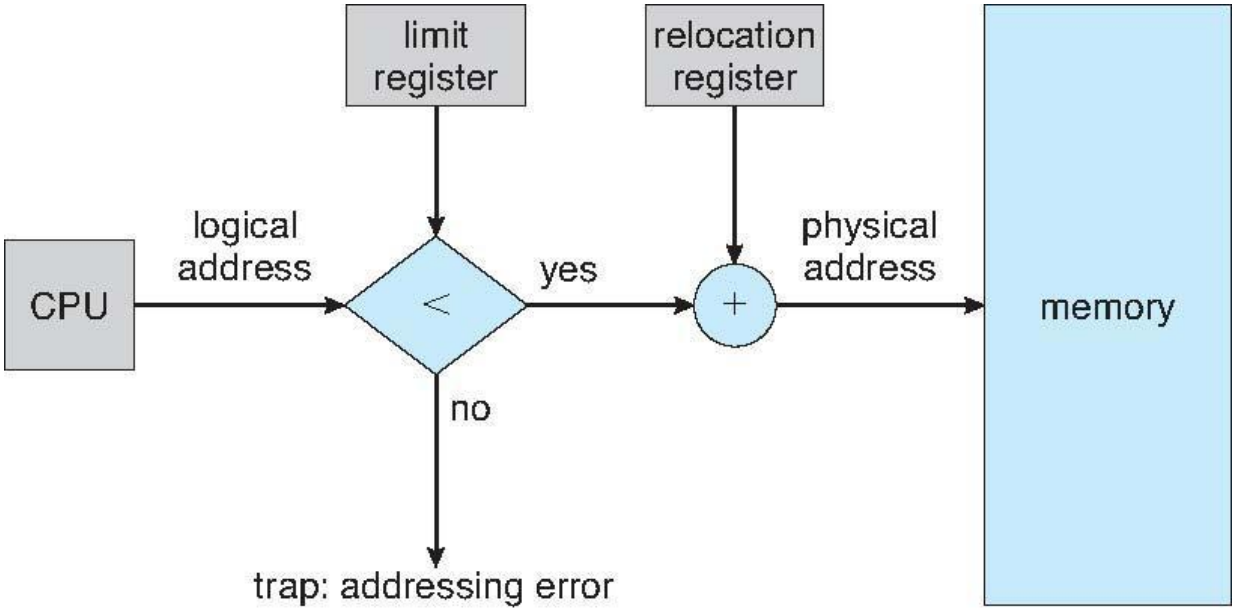
Contiguous Allocation

- Main memory usually into two partitions
  - Resident Operating System, usually held in low memory with interrupt vector.
  - User processes then held in high memory.
- Single partition allocation
  - Relocation register scheme used to protect user processes from each other, and from changing OS code and data.
  - Relocation register contains value of smallest physical address; limit register contains range of logical addresses - each logical address must be less than the limit register.

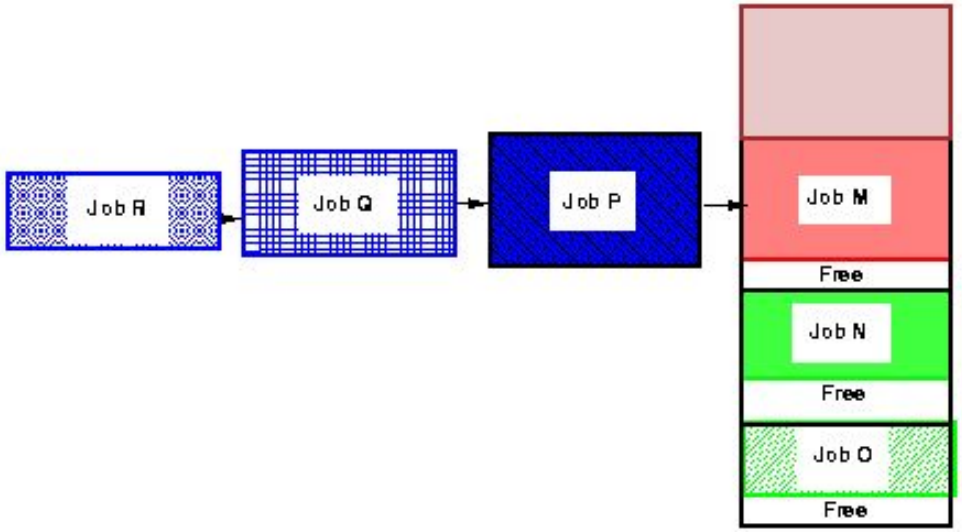
Relocation Register



Relocation and Limit Registers



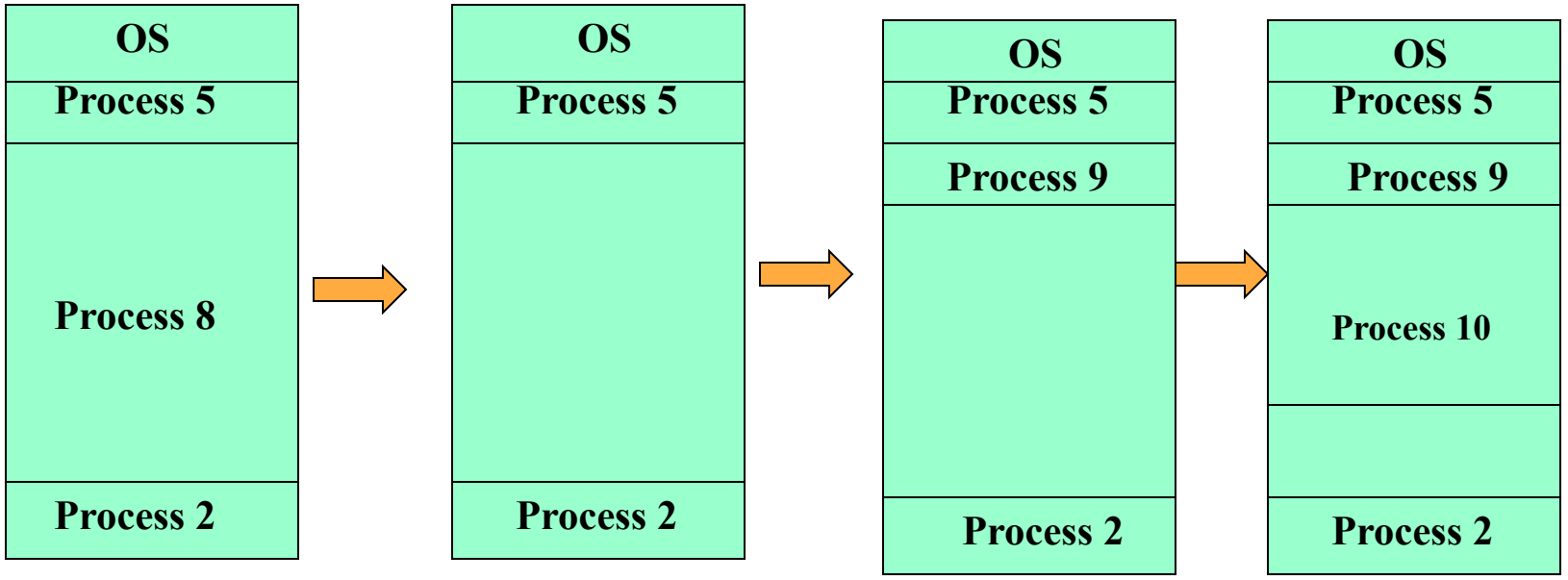
Fixed partitions



Contiguous Allocation (cont.)

- Multiple partition Allocation
  - Hole - block of available memory; holes of various sizes are scattered throughout memory.
  - When a process arrives, it is allocated memory from a hole large enough to accommodate it.
  - Operating system maintains information about
    - allocated partitions
    - free partitions (hole)

Contiguous Allocation example



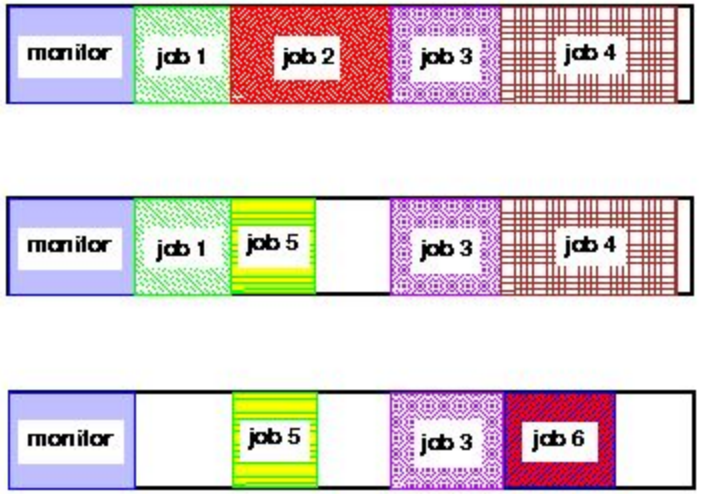
Dynamic Storage Allocation Problem

- How to satisfy a request of size n from a list of free holes.
  - First-fit
    - allocate the first hole that is big enough
  - Best-fit
    - Allocate the smallest hole that is big enough; must search entire list, unless ordered by size. Produces the smallest leftover hole.
  - Worst-fit
    - Allocate the largest hole; must also search entire list. Produces the largest leftover hole.
- First-fit and best-fit are better than worst-fit in terms of speed and storage utilization.

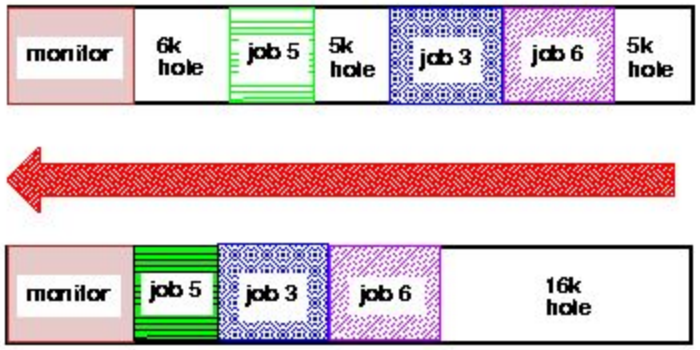
Fragmentation

- External fragmentation
  - total memory space exists to satisfy a request, but it is not contiguous.
- Internal fragmentation
  - allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used.
- Reduce external fragmentation by compaction
  - Shuffle memory contents to place all free memory together in one large block
  - Compaction is possible only if relocation is dynamic, and is done at execution time.
  - I/O problem - (1) latch job in memory while it is in I/O (2) Do I/O only into OS buffers.

Fragmentation example



Compaction


monitor
6k hole
job 5
5k hole
job 3
job 6
5k hole
monitor
job 5
job 3
job 6
16k hole

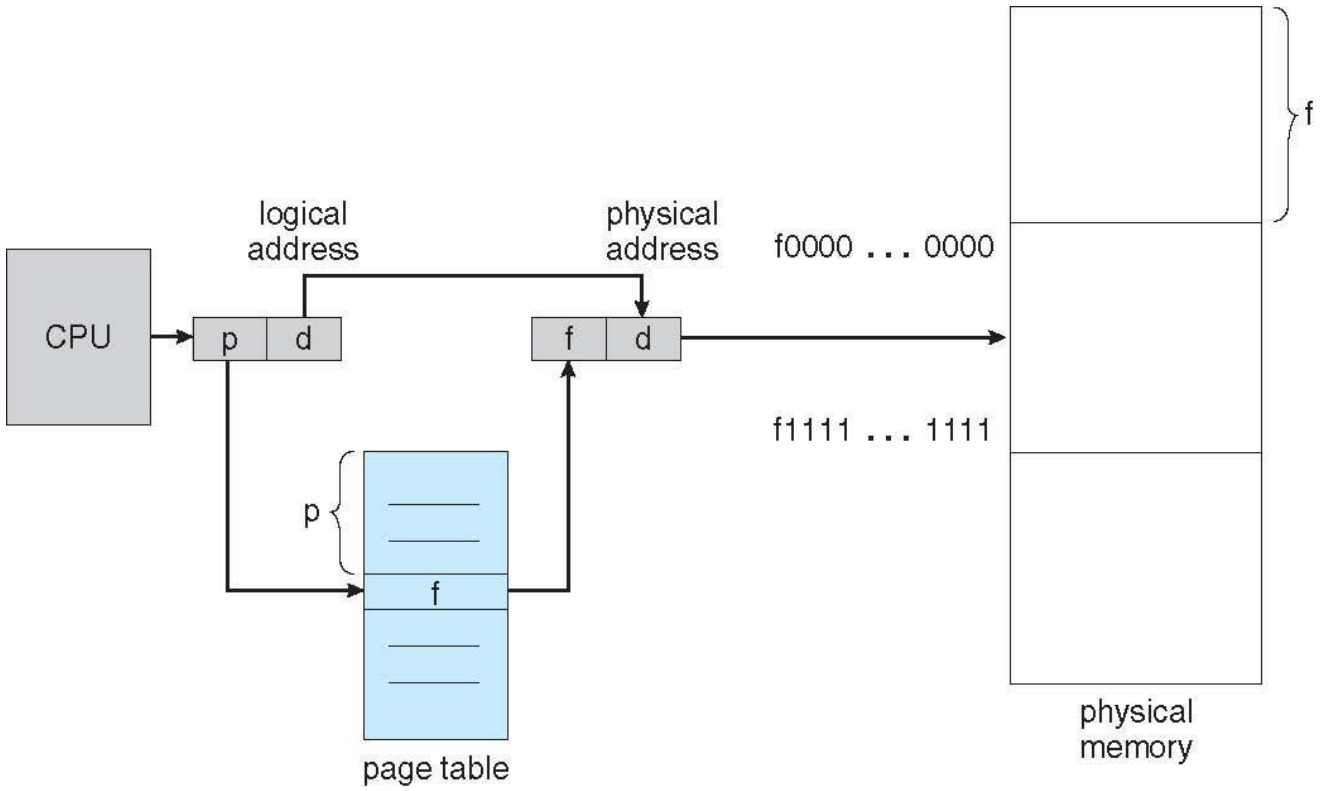
Paging

- Logical address space of a process can be non-contiguous;
  - process is allocated physical memory wherever the latter is available.
  - Divide physical memory into fixed size blocks called ***frames***
    - size is power of 2, 512 bytes - 8K
  - Divide logical memory into same size blocks called ***pages.***
    - Keep track of all free frames.
    - To run a program of size n pages, find n free frames and load program.
  - Set up a page table to translate logical to physical addresses.
  - Note:: Internal Fragmentation possible!!

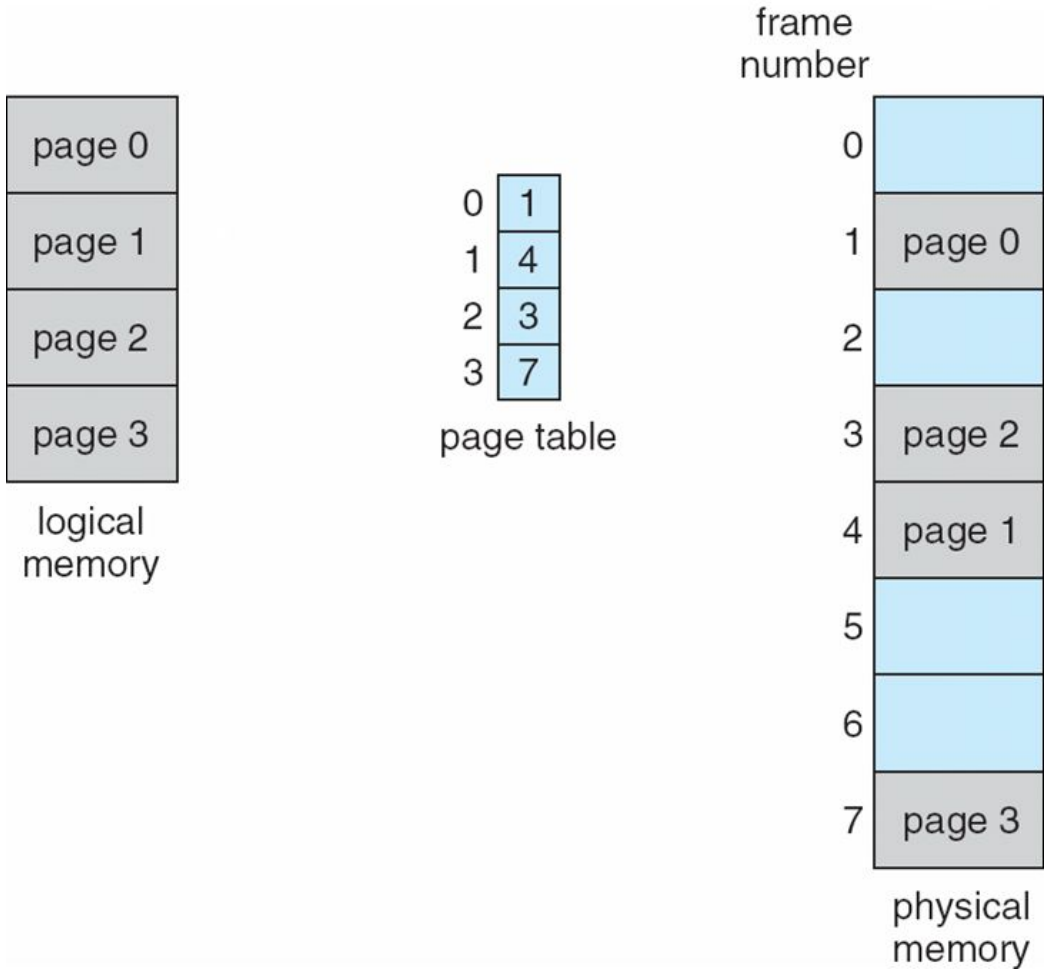
Address Translation Scheme

- Address generated by CPU is divided into:
  - Page number(p)
    - used as an index into page table which contains base address of each page in physical memory.
  - Page offset(d)
    - combined with base address to define the physical memory address that is sent to the memory unit.

Address Translation Architecture



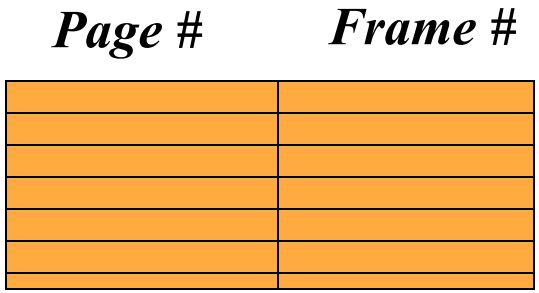
Example of Paging



Page Table Implementation

- Page table is kept in main memory
  - Page-table base register (PTBR) points to the page table.
  - Page-table length register (PTLR) indicates the size of page table.
  - Every data/instruction access requires 2 memory accesses.
    - One for page table, one for data/instruction
    - Two-memory access problem solved by use of special fast-lookup hardware cache (i.e. cache page table in registers)
      - associative registers or translation look-aside buffers (TLBs)

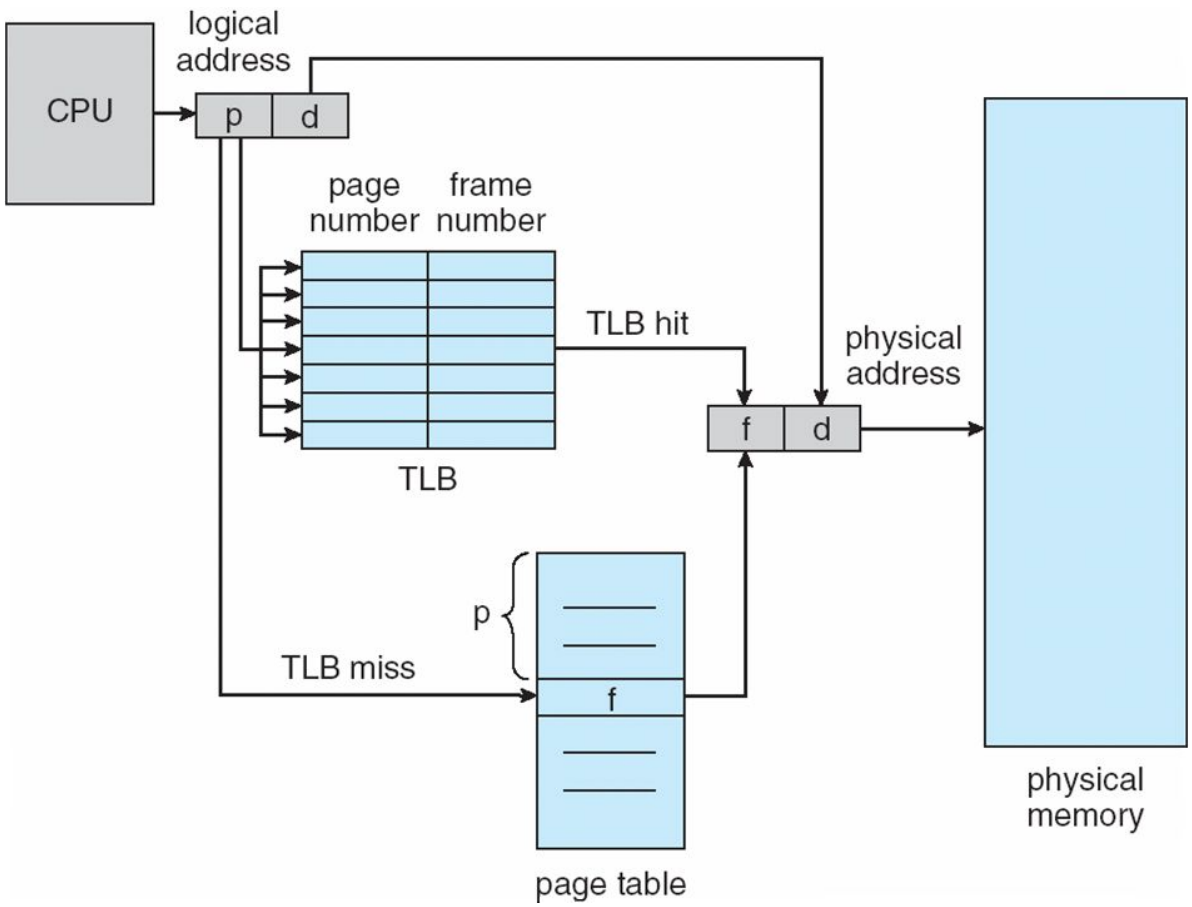
Translation Lookaside Buffer (TLB) (aka Associative Registers)



Address Translation
(A, A')

- If A is in TLB, get frame #
- Otherwise, need to go to page table for frame#
  - requires additional memory reference
  - Page Hit ratio - percentage of time page is found in TLB.

Paging hardware with TLB



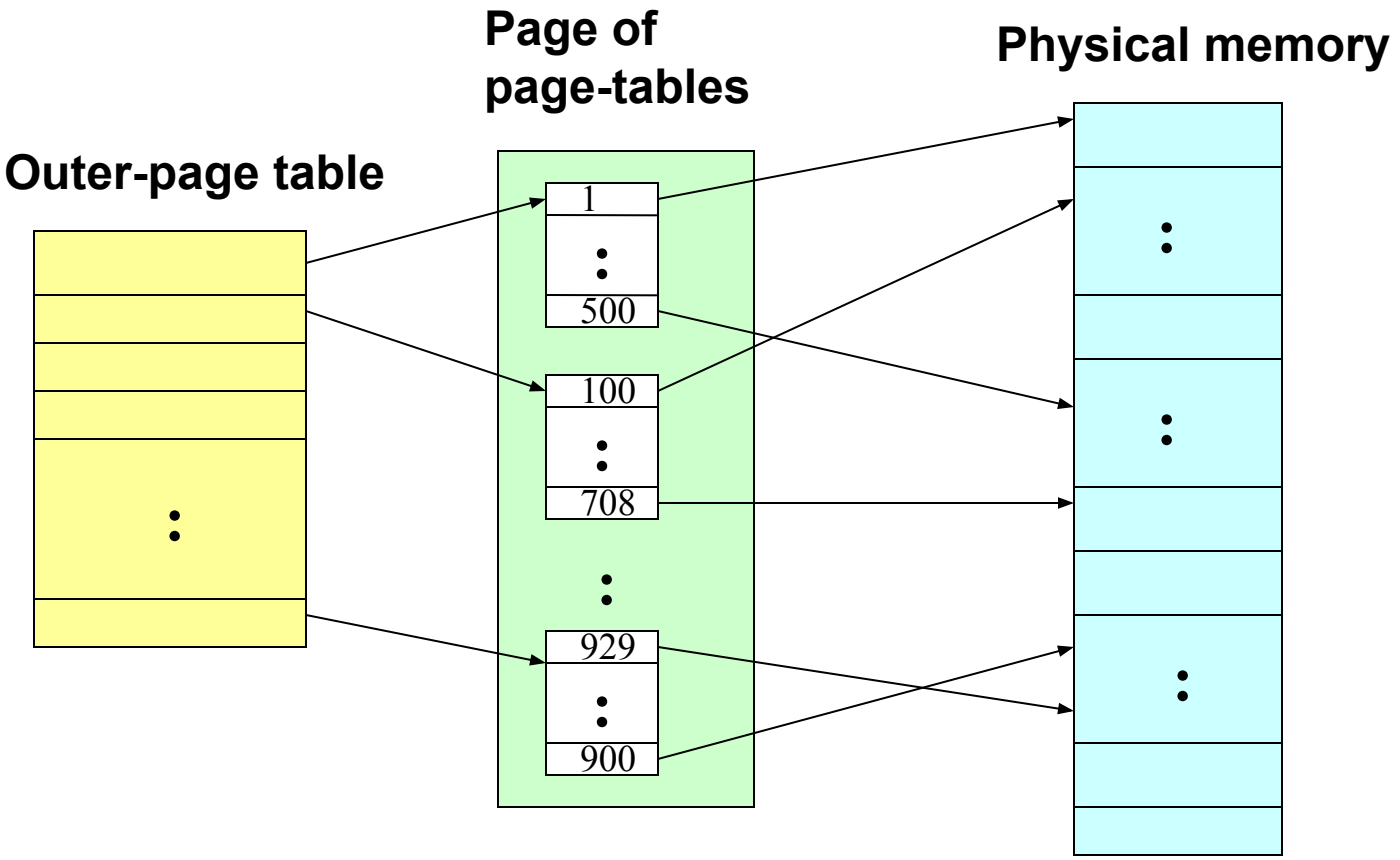
Effective Access time

- TLB lookup time = ε time unit
- Assume Memory cycle time = 1 microsecond
- Hit ratio = α
- Effective access time (EAT)
  - EAT = (1+ ε) α + (2+ ε) (1-α)
  - EAT = 2+ ε - α

Memory Protection

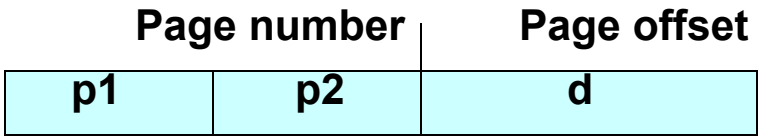
- Implemented by associating protection bits with each frame.
- Valid/invalid bit attached to each entry in page table.
  - Valid: indicates that the associated page is in the process' logical address space.
  - Invalid: indicates that the page is not in the process' logical address space.

Two Level Page Table Scheme

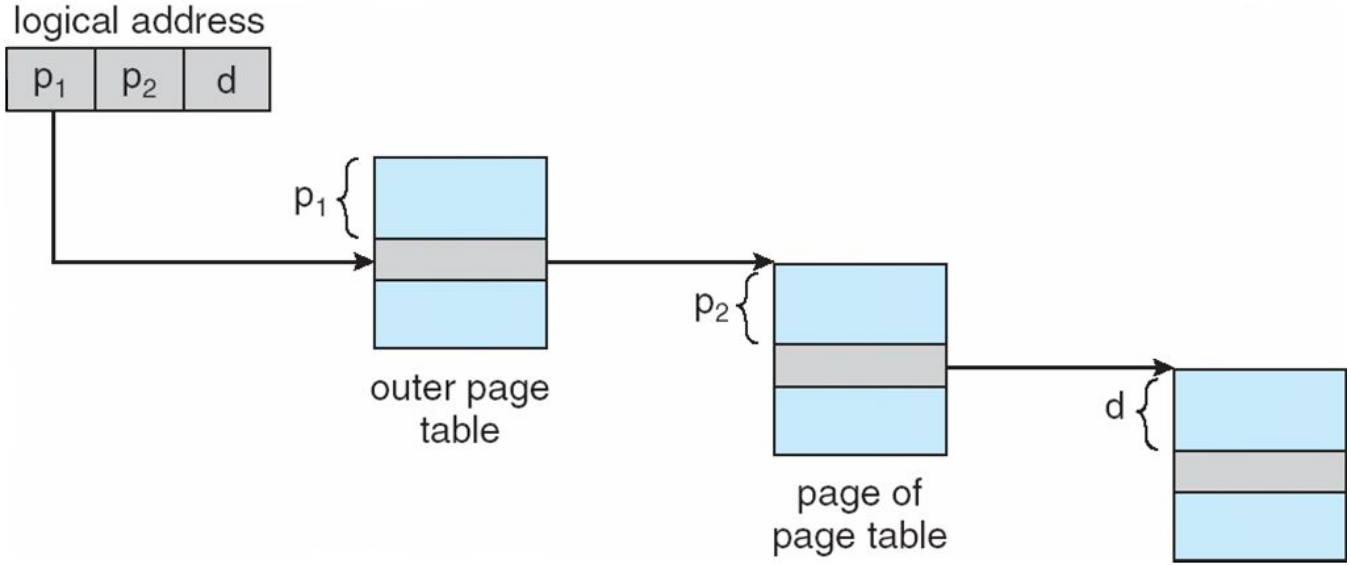


Two Level Paging Example

- A logical address (32bit machine, 4K page size) is divided into
  - a page number consisting of 20 bits, a page offset consisting of 12 bits
- Since the page table is paged, the page number consists of
  - a 10-bit page number, a 10-bit page offset
- Thus, a logical address is organized as (p1,p2,d) where
  - p1 is an index into the outer page table
  - p2 is the displacement within the page of the outer page table



Two Level Paging Example



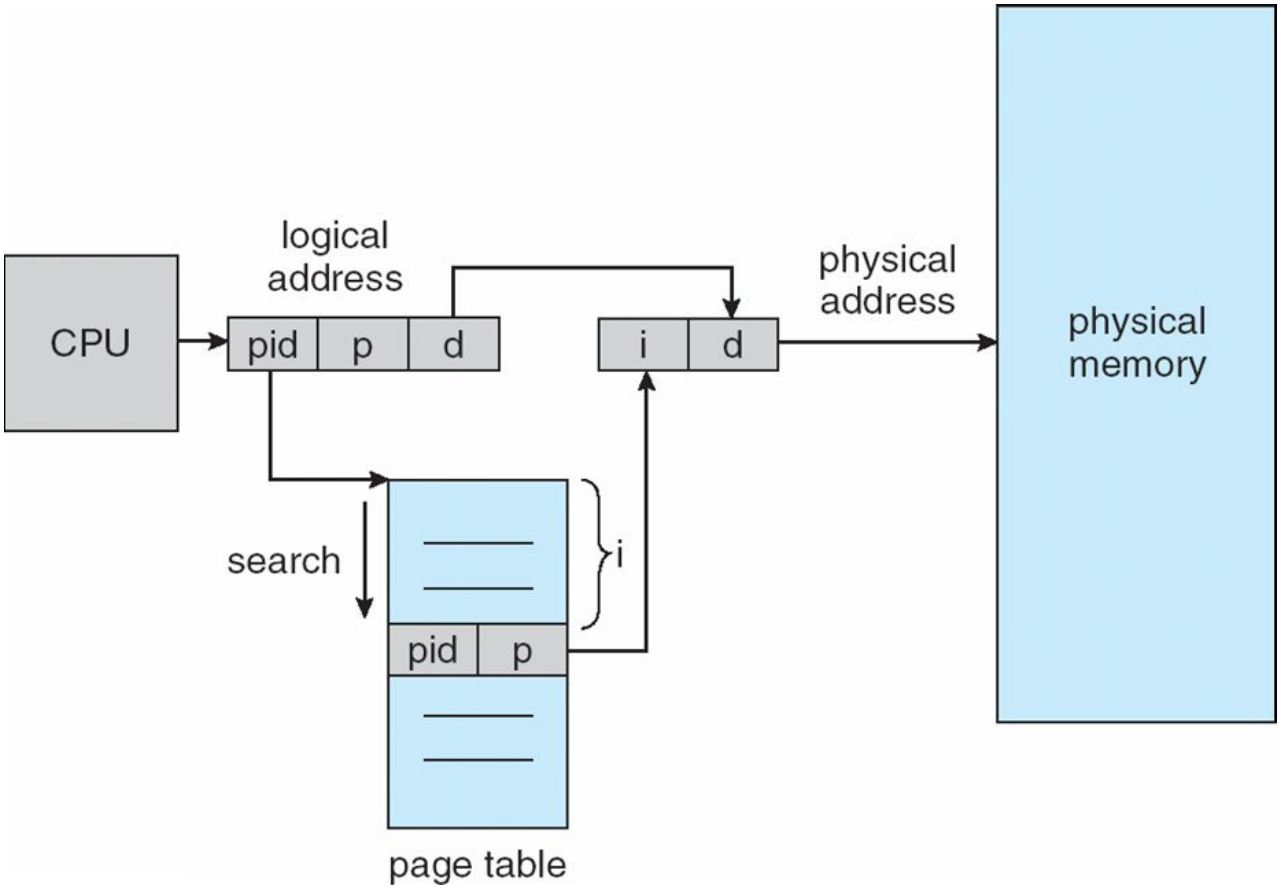
Multilevel paging

- Each level is a separate table in memory
  - converting a logical address to a physical one may take 4 or more memory accesses.
  - Caching can help performance remain reasonable.
    - Assume cache hit rate is 98%, memory access time is quintupled (100 vs. 500 nanoseconds), cache lookup time is 20 nanoseconds
    - Effective Access time = 0.98 * 120 + .02 * 520 = 128 ns
    - This is only a 28% slowdown in memory access time...

Inverted Page Table

- One entry for each real page of memory
  - Entry consists of virtual address of page in real memory with information about process that owns page.
  - Decreases memory needed to store page table
  - Increases time to search table when a page reference occurs
    - table sorted by physical address, lookup by virtual address
  - Use hash table to limit search to one (maybe few) page-table entries.

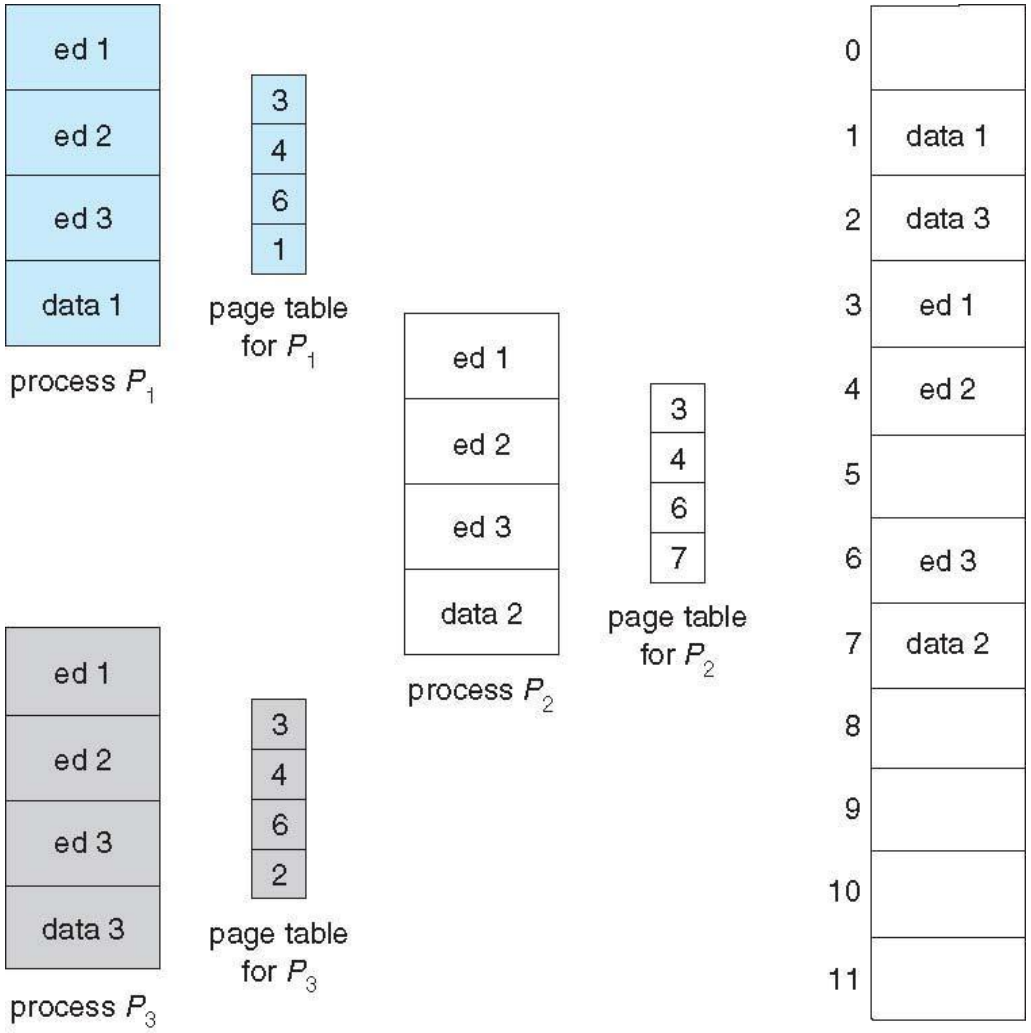
Inverted Page Table



Shared pages

- Code and data can be shared among processes
  - Reentrant (non self-modifying) code can be shared.
  - Map them into pages with common page frame mappings
  - Single copy of read-only code - compilers, editors etc..
- Shared code must appear in the same location in the logical address space of all processes
- Private code and data
  - Each process keeps a separate copy of code and data
  - Pages for private code and data can appear anywhere in logical address space.

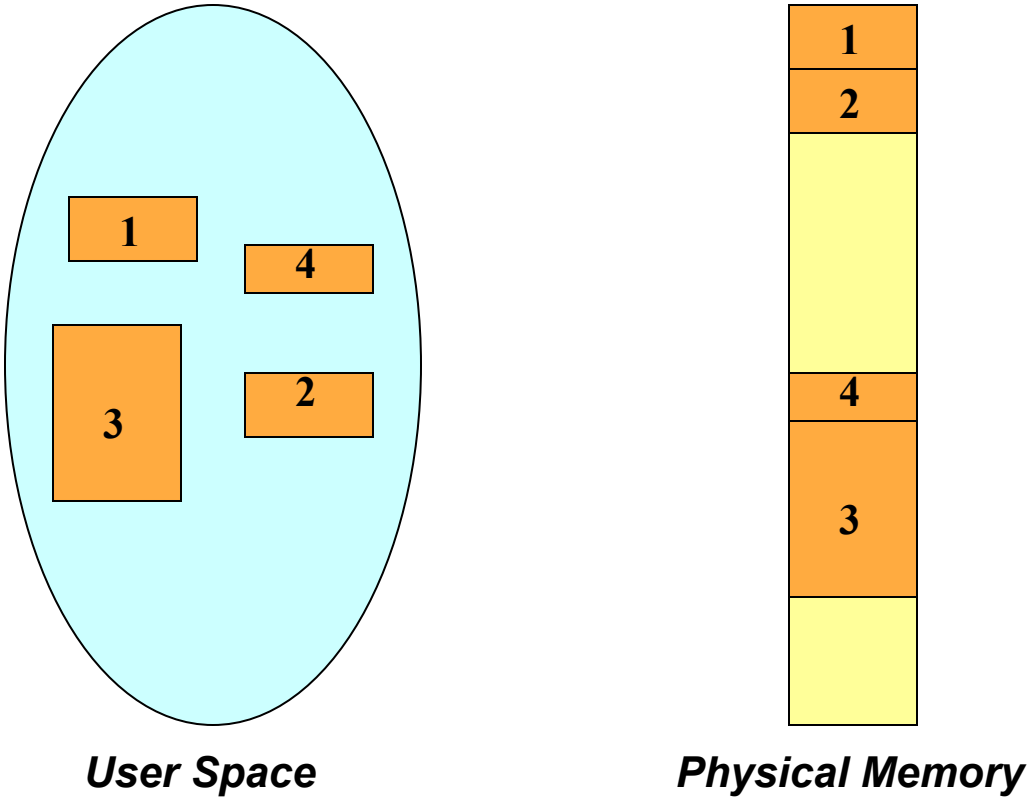
Shared Pages



Segmentation

- Memory Management Scheme that supports user view of memory.
- A program is a collection of segments.
- A segment is a logical unit such as
  - main program, procedure, function
  - local variables, global variables,common block
  - stack, symbol table, arrays
  - Protect each entity independently
  - Allow each segment to grow independently
  - Share each segment independently

Logical view of segmentation



Segmentation Architecture

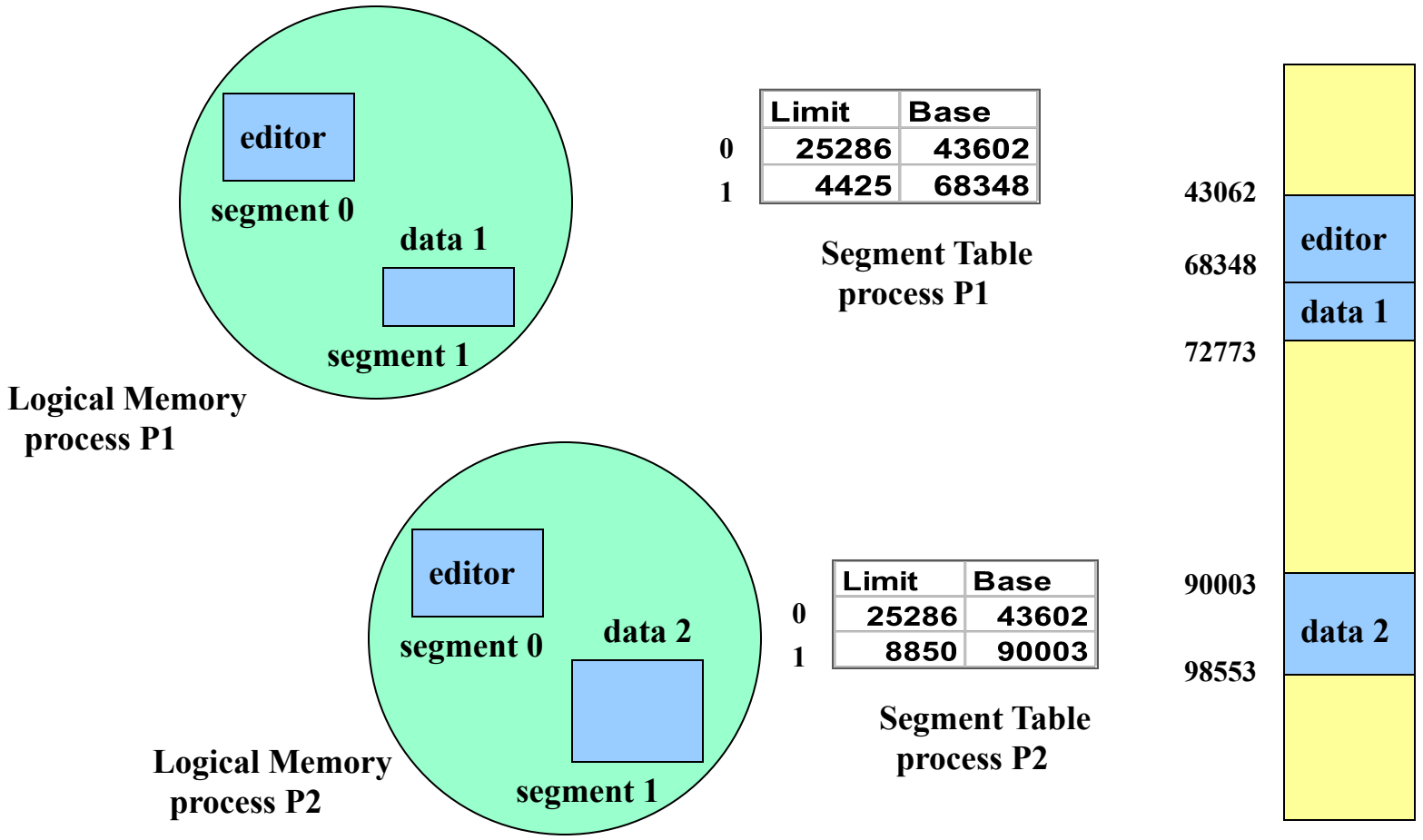
- Logical address consists of a two tuple

  <segment-number, offset>
- Segment Table
  - Maps two-dimensional user-defined addresses into one-dimensional physical addresses. Each table entry has
    - Base - contains the starting physical address where the segments reside in memory.
    - Limit - specifies the length of the segment.
  - *Segment-table base register* (STBR) points to the segment table's location in memory.
  - *Segment-table length register* (STLR) indicates the number of segments used by a program; segment number is legal if s < STLR.

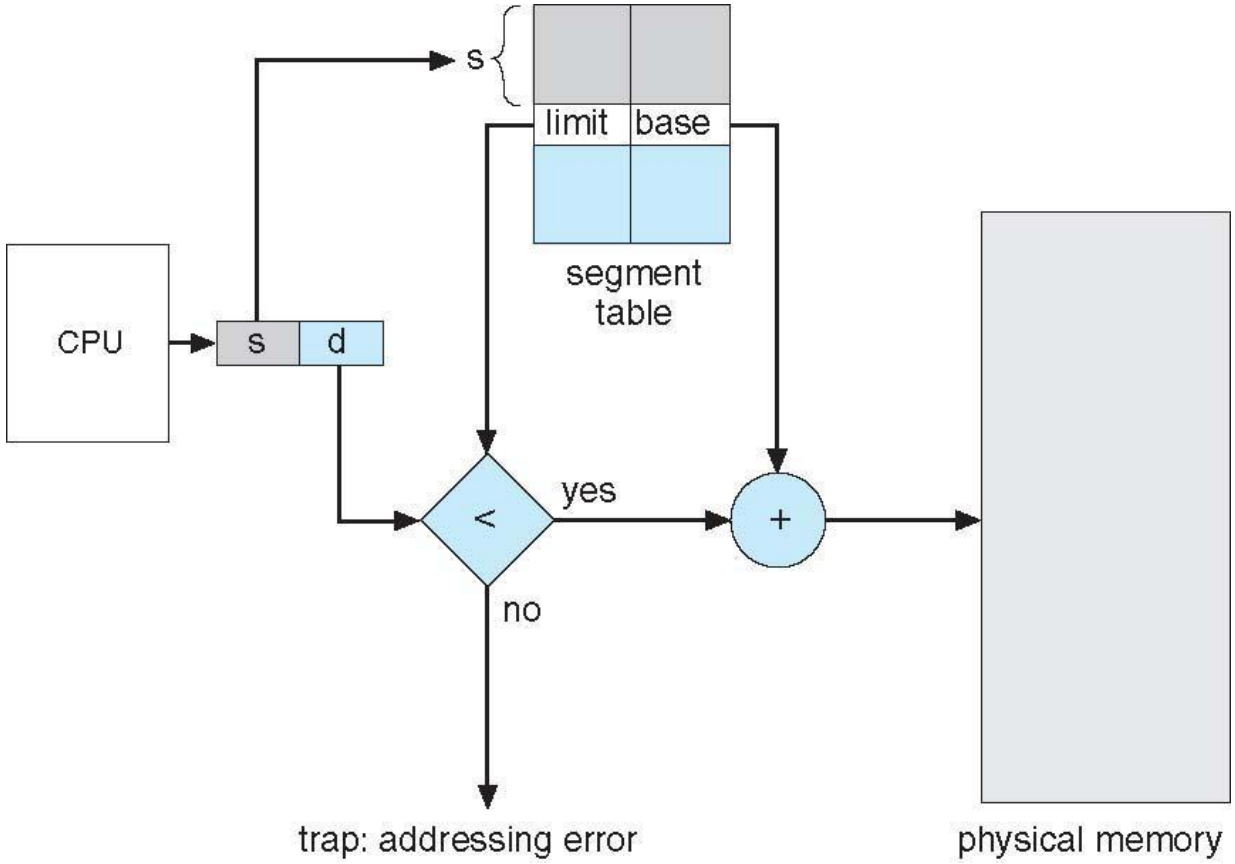
Segmentation Architecture (cont.)

- Relocation is dynamic - by segment table
- Sharing
  - Code sharing occurs at the segment level.
  - Shared segments must have same segment number.
- Allocation - dynamic storage allocation problem
  - use best fit/first fit, may cause external fragmentation.
- Protection
  - protection bits associated with segments
    - read/write/execute privileges
    - array in a separate segment - hardware can check for illegal array indexes.

Shared segments



Segmentation hardware



Segmented Paged Memory

- Segment-table entry contains not the base address of the segment, but the base address of a page table for this segment.
  - Overcomes external fragmentation problem of segmented memory.
  - Paging also makes allocation simpler; time to search for a suitable segment (using best-fit etc.) reduced.
  - Introduces some internal fragmentation and table space overhead.
- Multics - single level page table
- IBM OS/2 - OS on top of Intel 386
  - uses a two level paging scheme

MULTICS address translation scheme