Chapters 9 & 10: Memory Management and Virtual Memory

- Important concepts (for final, projects, papers)
  - addressing: physical/absolute, logical/relative/virtual
  - overlays
  - swapping and paging
  - memory protection
  - algorithms: free space fit, page replacement
  - segmentation
  - thrashing
  - locality

Background

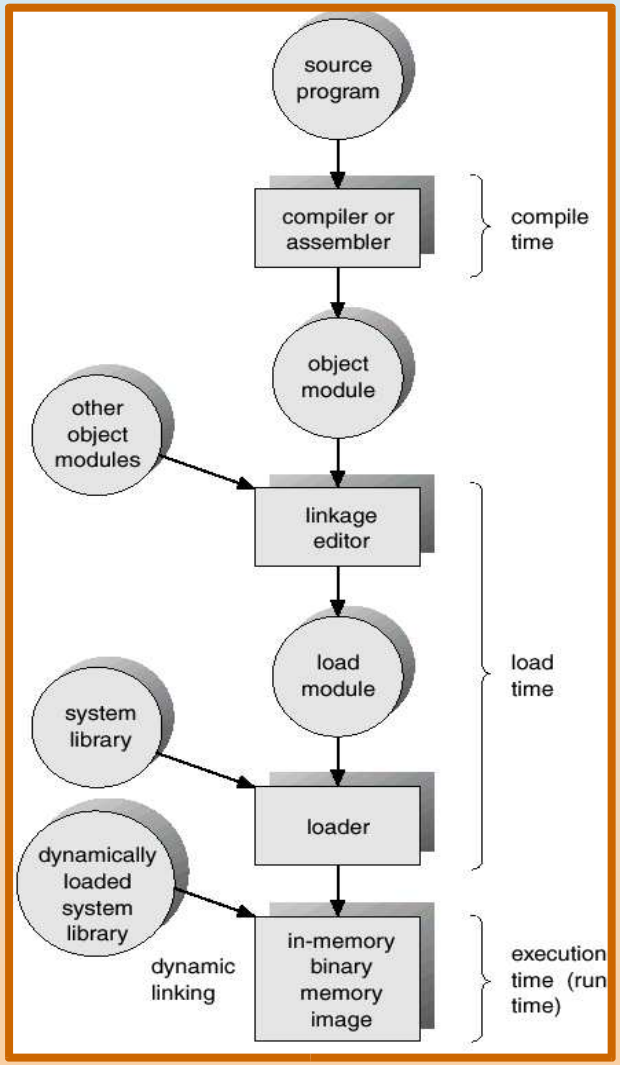
- Program must be brought into memory and placed within a process for it to be run.
- Input queue – collection of processes on the disk that are waiting to be brought into memory to run the program.
- User programs go through several steps before being run.
- process varies by type of OS...following discussion is for shared-resource, multi-user OS
- What is an “address”?
  - How do we represent it?
  - How big can it be?

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 known a priori, absolute code can be generated; must recompile code if starting location changes.
  - when do you need to know absolute address?
- Load time: Must generate relocatable code if memory location is not known at compile time.
  - relocatable? what does that mean?
- Execution time: Binding delayed until run time 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).

Multistep Processing of a User Program



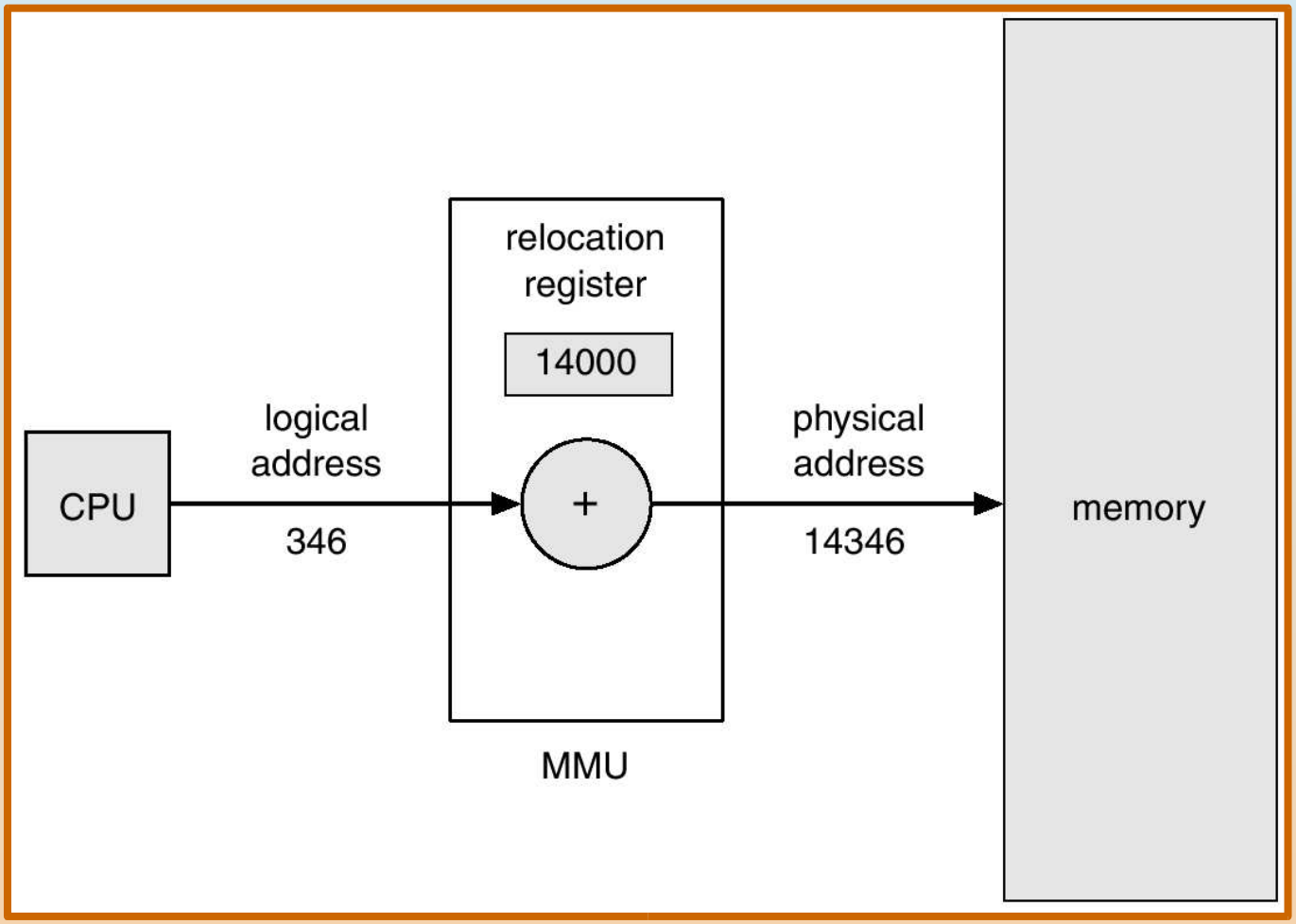
Logical vs. Physical Address Space

- The concept of a logical address space that is bound to a separate physical address space is central to proper memory management.
  - Logical address – generated by the CPU; also referred to as virtual address.
  - Physical address – address seen by the memory unit (also absolute address)
- Logical and physical addresses are the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme.

Memory-Management Unit (MMU)

- Hardware device that maps virtual to physical address.
  - WHERE is the MMU?
- 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 addresses.

Dynamic relocation using a relocation register



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.
- Dynamic linking is particularly useful for libraries.

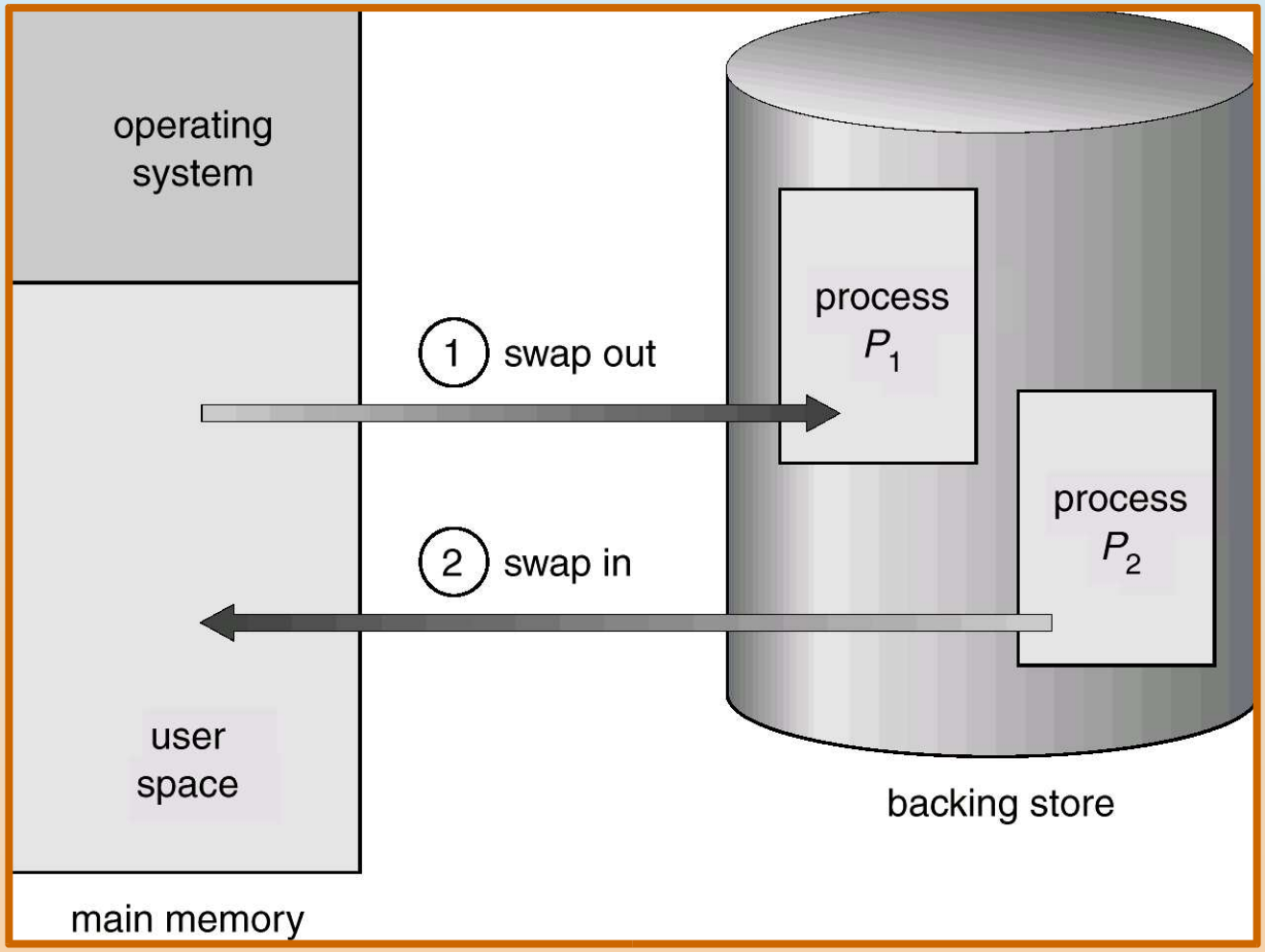
Overlays

- 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 needed from operating system, programming design of overlay structure is complex
- still used for some OS/processor architectures

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.
  - must it be disk?
- 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, Linux, and Windows).

Schematic View of Swapping



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 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.
    - what are the benefits & drawbacks?
  - Compaction is possible only if relocation is dynamic, and is done at execution time.
  - I/O problem
    - Latch job in memory while it is involved in I/O.
    - Do I/O only into OS buffers.
      - why?

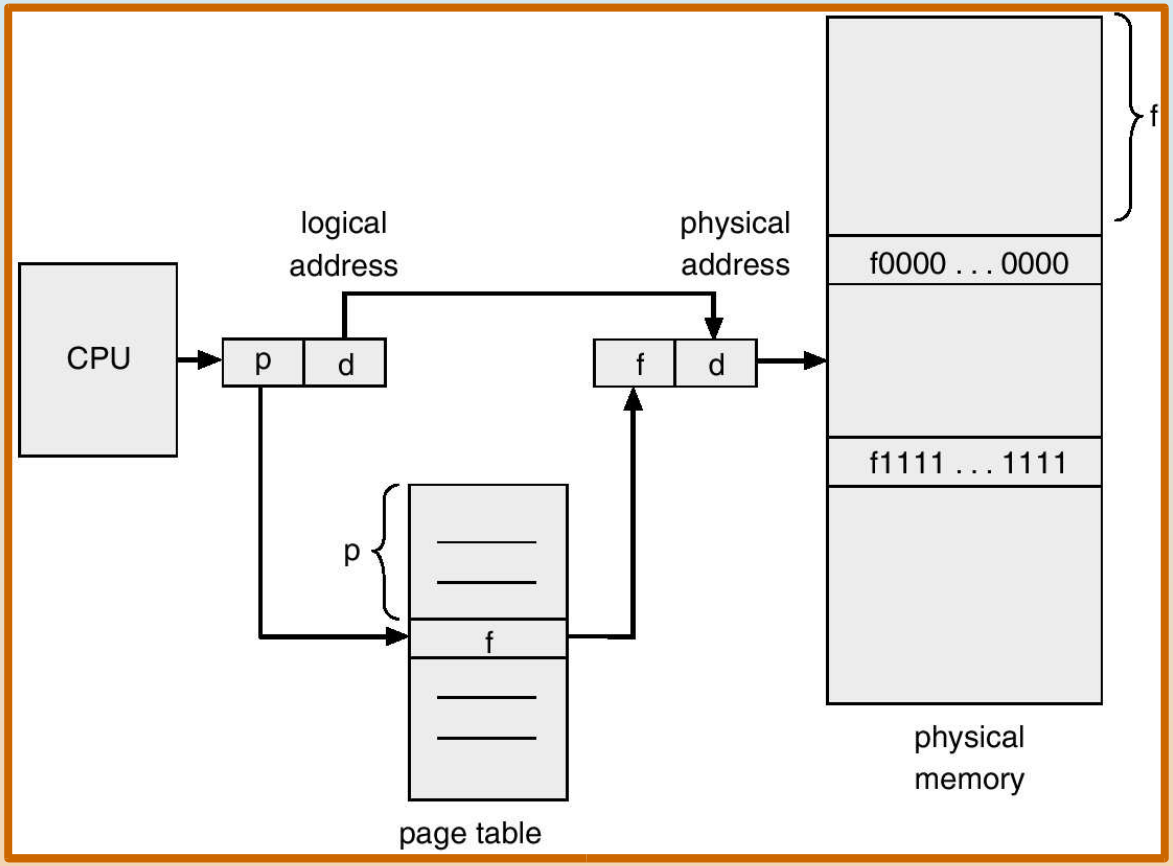
Paging

- Logical address space of a process can be noncontiguous; process is allocated physical memory whenever the latter is available.
- Divide physical memory into fixed-sized blocks called frames (size is power of 2, between 512 bytes and 8192 bytes).
- Divide logical memory into blocks of same size called pages.
- Keep track of all free frames.
- To run a program of size n pages, need to find n free frames and load program.
- Set up a page table to translate logical to physical addresses.
- Internal fragmentation.

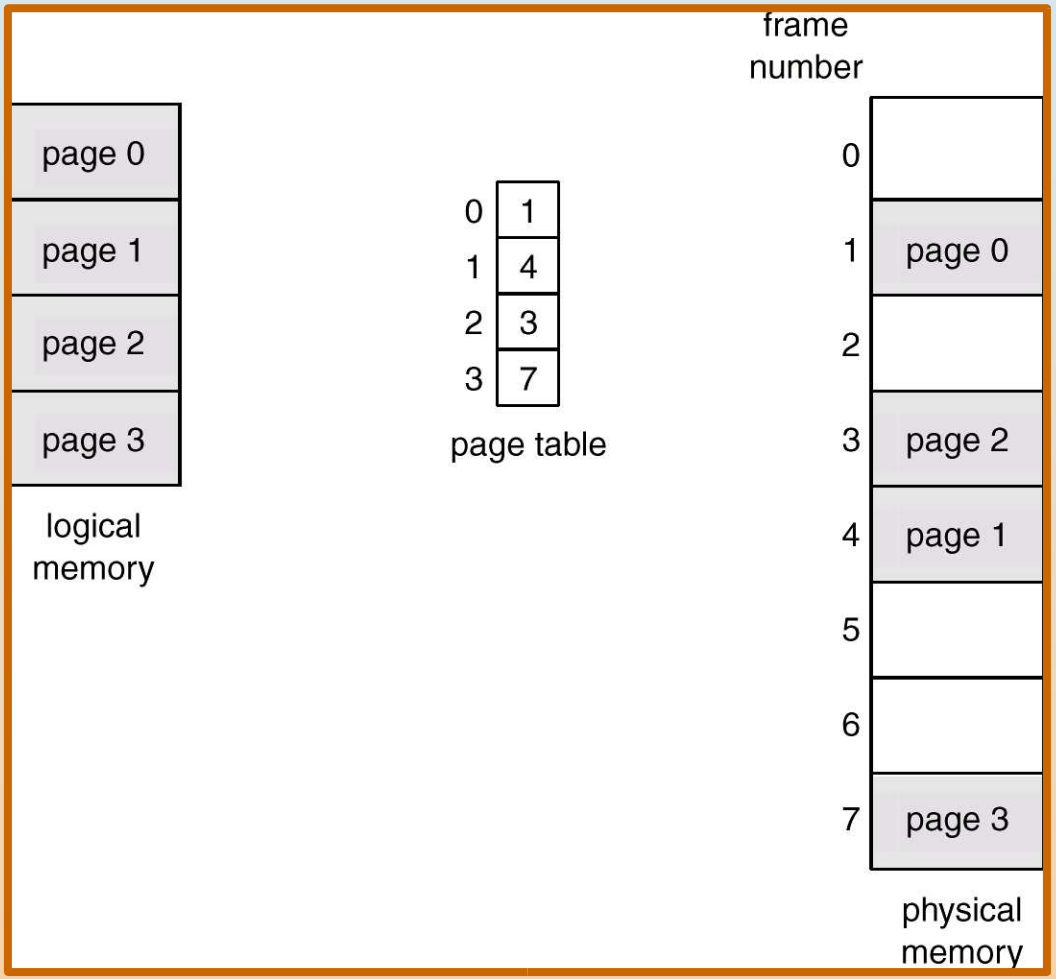
Address Translation Scheme

- Address generated by CPU is divided into:
  - Page number (p) – used as an index into a 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



Paging Example



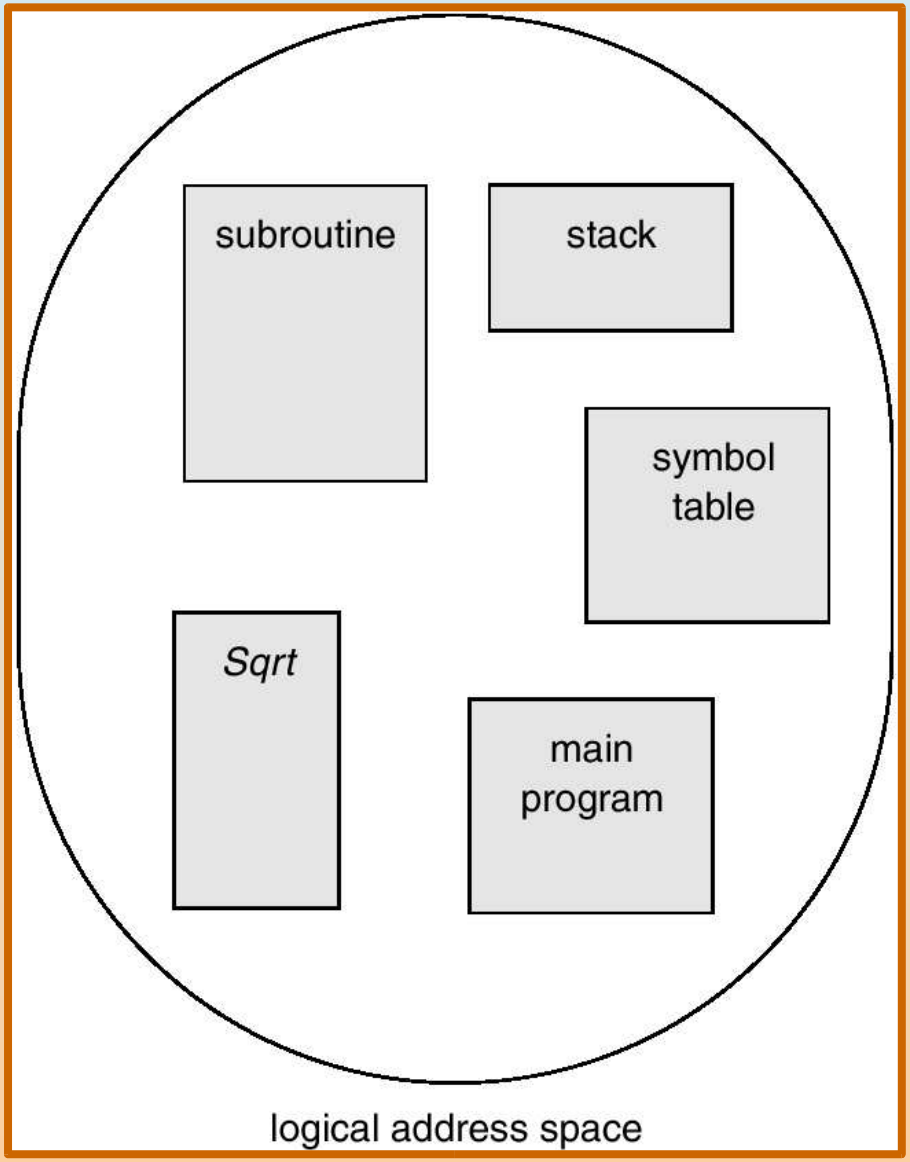
Memory Protection

- Memory protection implemented by associating protection bit with each frame.
- 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.
    - what happens when process tries to access address not in permitted address space?
      - who would DO such a thing?

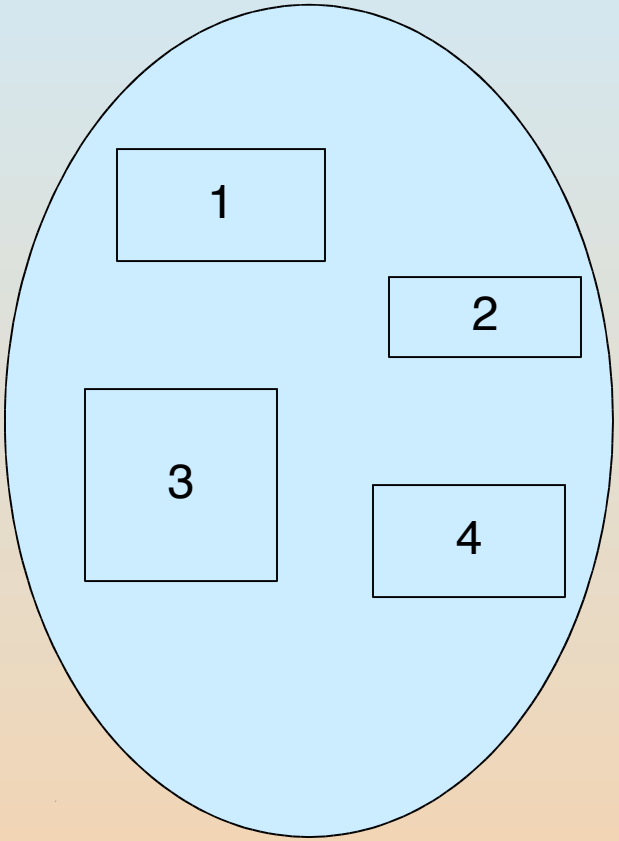
Segmentation

- 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,
  - method,
  - object,
  - local variables, global variables,
  - common block,
  - stack,
  - symbol table, arrays

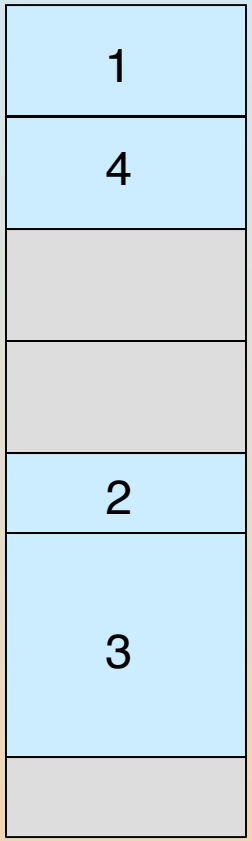
User's View of a Program



Logical View of Segmentation



user space

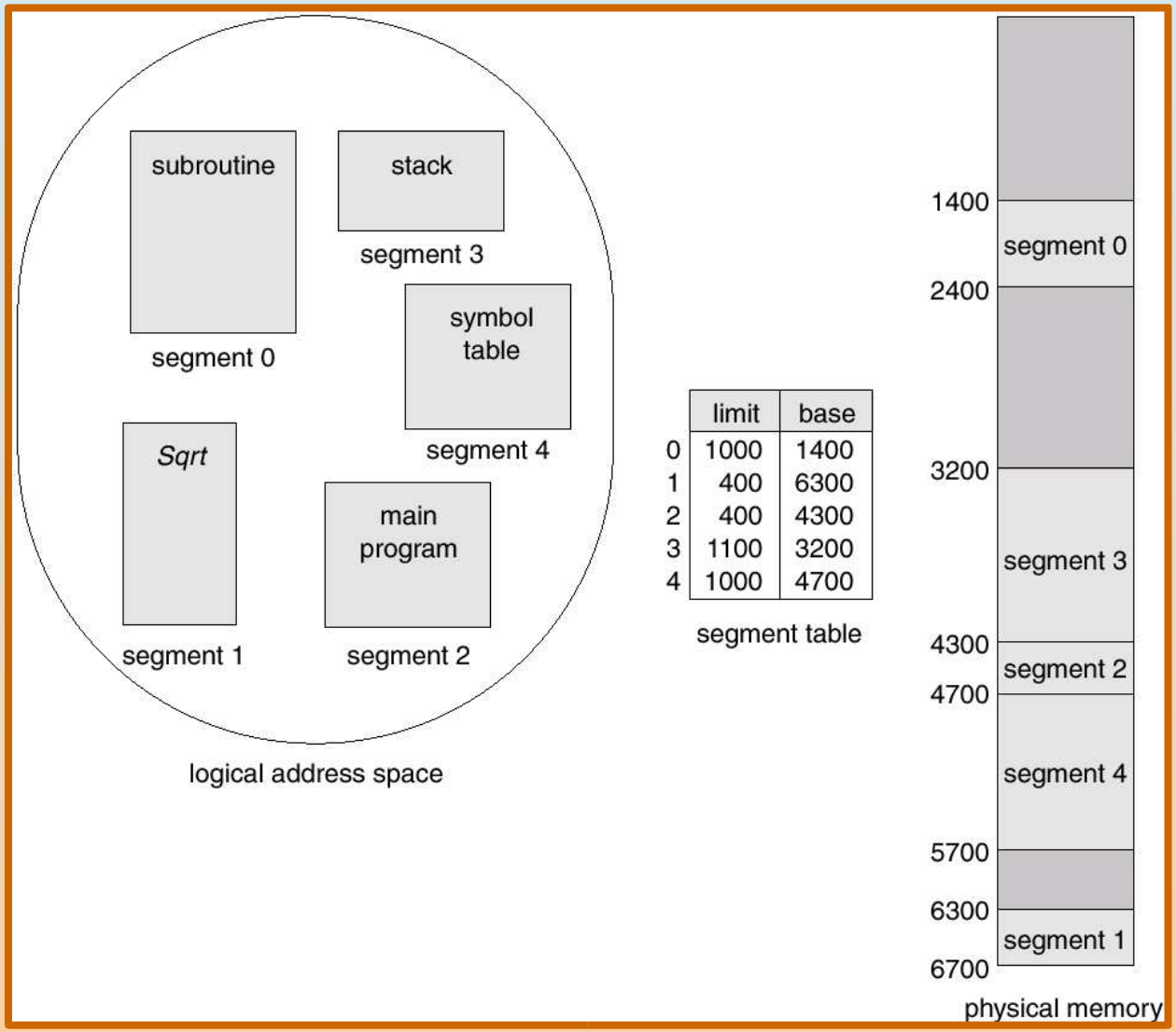


physical memory space

Segmentation Architecture

- Logical address consists of a two tuple:
- <segment-number, offset>,
- Segment table – maps two-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 number of segments used by a program
  - segment number s is legal if s < STLR.

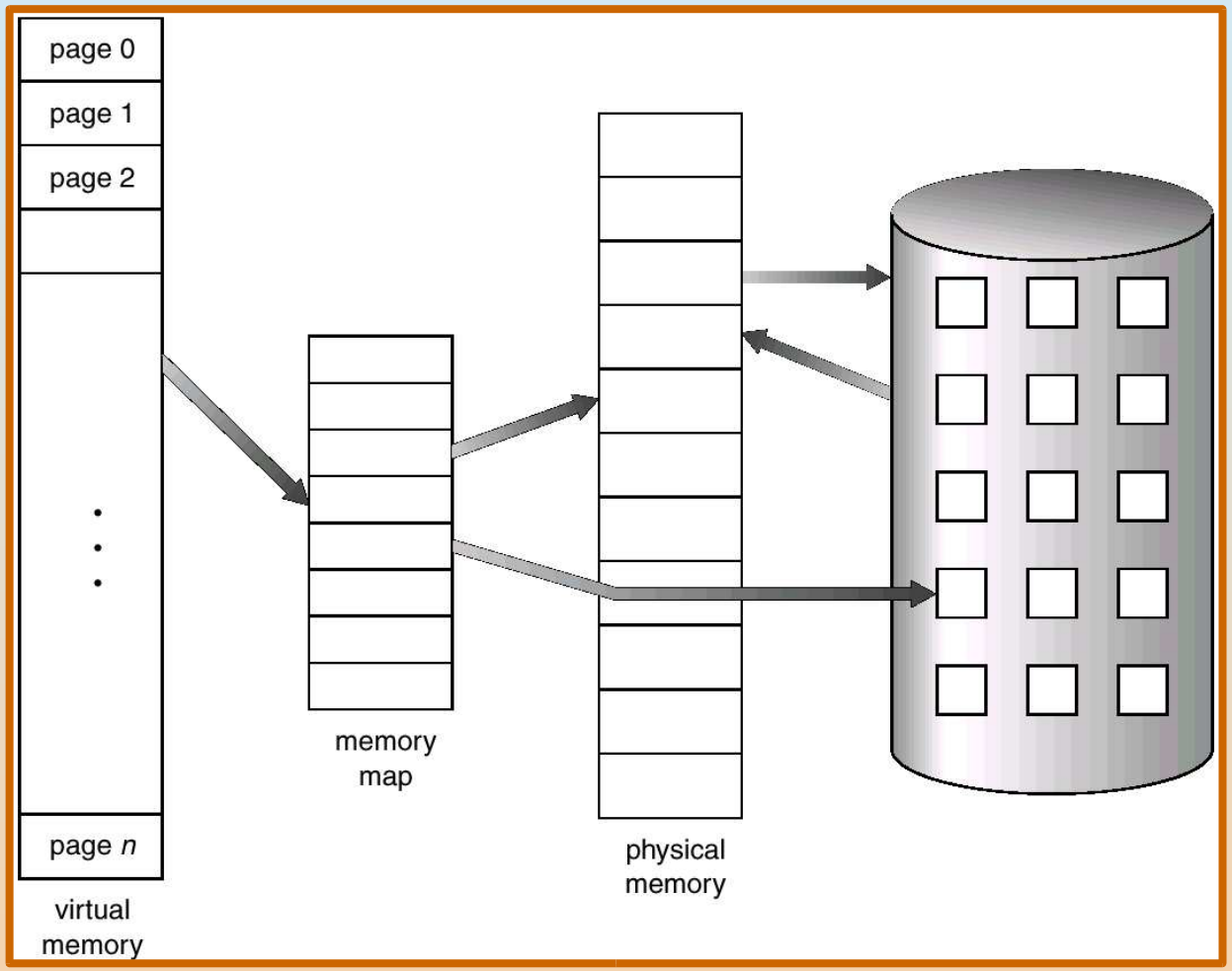
Example of Segmentation



Virtual Memory

- Virtual memory – separation of user logical memory from physical memory.
  - Only part of the program needs to be in memory for execution.
  - Logical address space can therefore be much larger than physical address space.
    - larger? how much larger?
  - Allows address spaces to be shared by several processes.
  - Allows for more efficient process creation.
- Virtual memory can be implemented via:
  - Demand paging
  - Demand segmentation

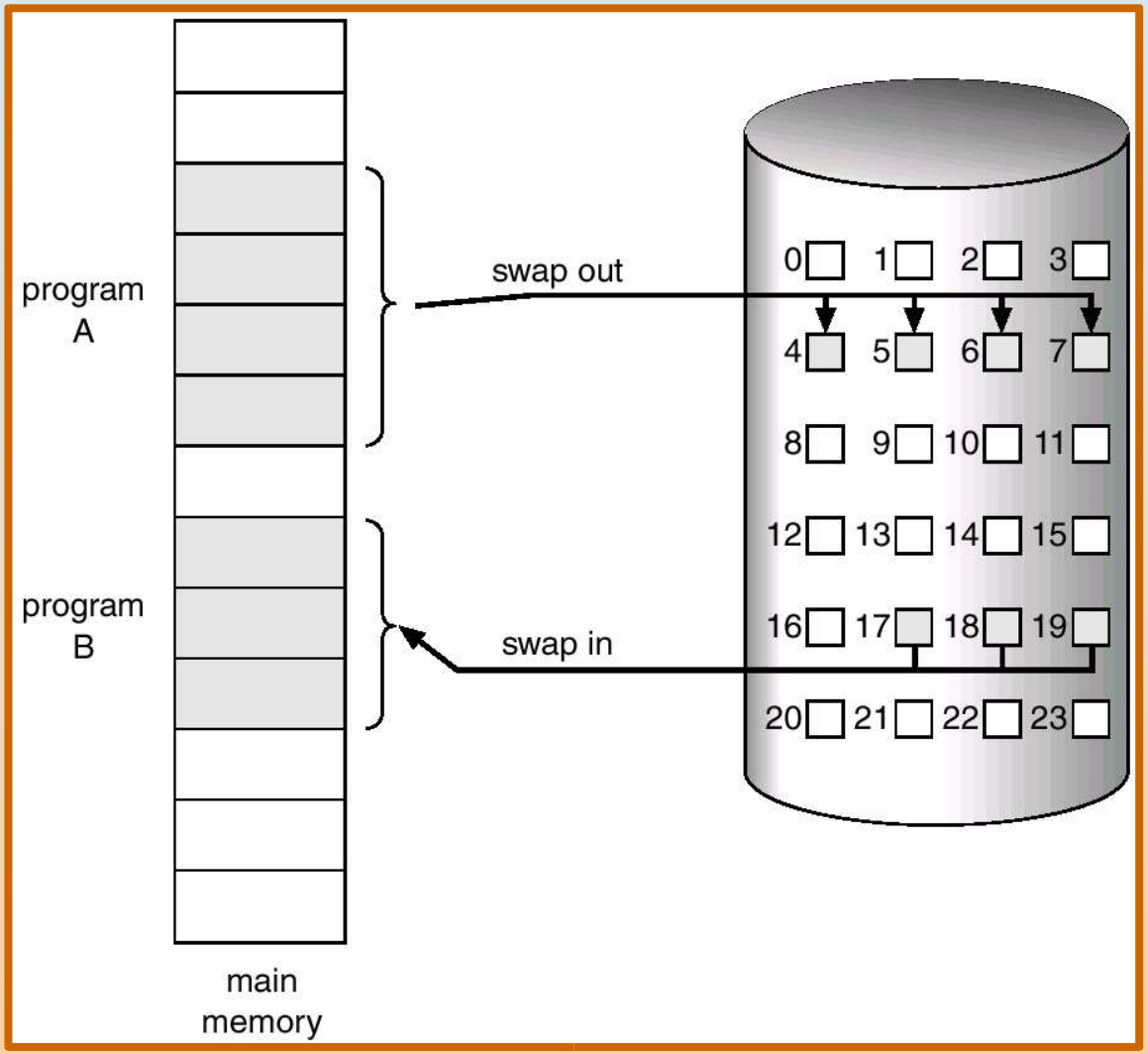
Virtual Memory That is Larger Than Physical Memory



Demand Paging

- Bring a page into memory only when it is needed.
  - Less I/O needed
  - Less memory needed
  - Faster response
  - More users
- Page is needed => reference to it
  - invalid reference => abort
  - not-in-memory => bring to memory

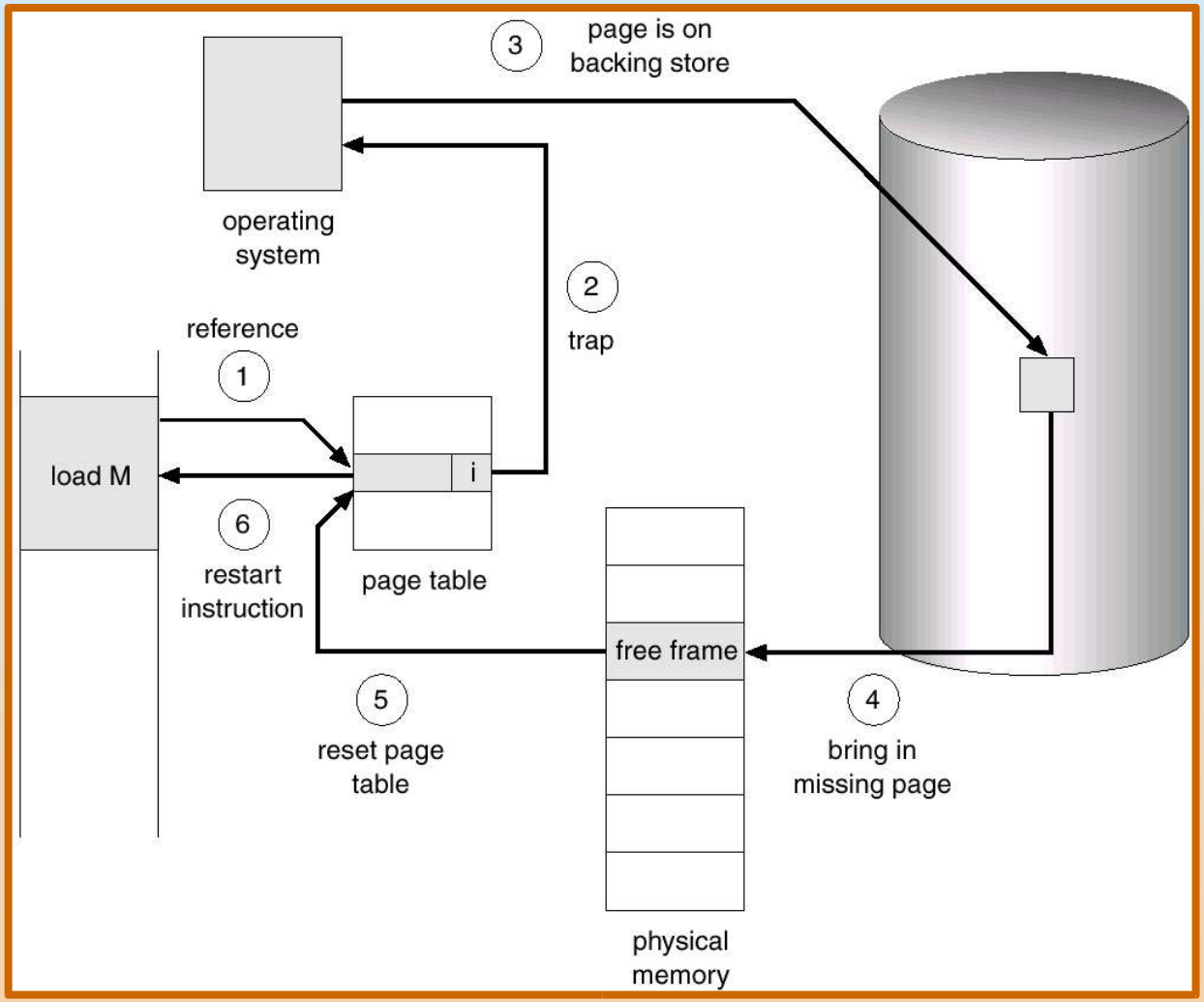
Transfer of a Paged Memory to Contiguous Disk Space
Why contiguous?



Page Fault

- If there is ever a reference to a page, first reference will trap to OS => page fault
- OS looks at another table to decide:
  - Invalid reference => abort.
  - Just not in memory.
- Get empty frame.
- Swap page into frame.

Steps in Handling a Page Fault



What happens if there is no free frame?

- Page replacement – find some page in memory, but not really in use, swap it out.
  - algorithm?
  - performance – want an algorithm which will result in minimum number of page faults.
- Same page may be brought into memory several times.

Process Creation

- Virtual memory allows other benefits during process creation:
- - Copy-on-Write
- - Memory-Mapped Files

Copy-on-Write

- Copy-on-Write (COW) allows both parent and child processes to initially share the same pages in memory.

  If either process modifies a shared page, only then is the page copied.
- COW allows more efficient process creation as only modified pages are copied.
- Free pages are allocated from a pool of zeroed-out pages.

Memory-Mapped Files

- Memory-mapped file I/O allows file I/O to be treated as routine memory access by mapping a disk block to a page in memory.
- A file is initially read using demand paging. A page-sized portion of the file is read from the file system into a physical page. Subsequent reads/writes to/from the file are treated as ordinary memory accesses.
- Simplifies file access by treating file I/O through memory rather than read() write() system calls.
- Also allows several processes to map the same file allowing the pages in memory to be shared.

Page Replacement

- Prevent over-allocation of memory by modifying page-fault service routine to include page replacement.
- Use modify (dirty) bit to reduce overhead of page transfers – only modified pages are written to disk.
- Page replacement completes separation between logical memory and physical memory – large virtual memory can be provided on a smaller physical memory.

Basic Page Replacement

1. Find the location of the desired page on disk.

2. Find a free frame:
   - If there is a free frame, use it.
   - If there is no free frame, use a page replacement algorithm to select a victim frame.
   - how can we efficiently select a victim?

3. Read the desired page into the (newly) free frame. Update the page and frame tables.

4. Restart the process.

Page Replacement Algorithms

- Want lowest page-fault rate.
- Evaluate algorithm by running it on a particular string of memory references (reference string) and computing the number of page faults on that string.
- In all our examples, the reference string is
  - 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.

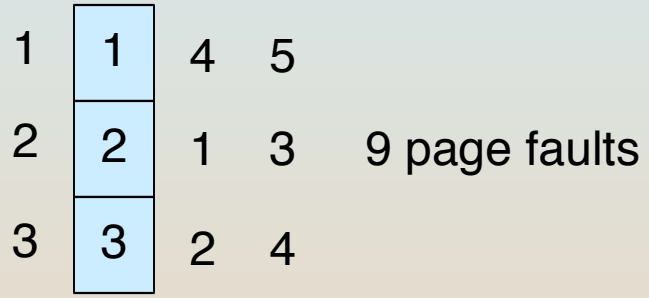
First-In-First-Out (FIFO) Algorithm

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames (3 pages can be in memory at a time per process)

1	1	4	5	
2	2	1	3	9 page faults
3	3	2	4	

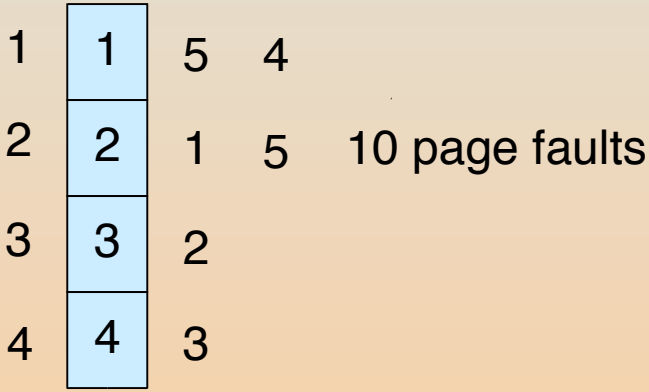
First-In-First-Out (FIFO) Algorithm

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames (3 pages can be in memory at a time per process)



9 page faults

- 4 frames



10 page faults

- FIFO Replacement – Belady's Anomaly
  - more frames does not always result in fewer page faults

Optimal Algorithm

- Replace page that will not be used for longest period of time.
- 4 frames example

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

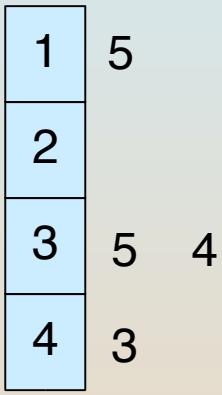
1	4
2	
3	
4	5

6 page faults

- Can we know this?
- Used for measuring how well your algorithm performs.

Least Recently Used (LRU) Algorithm

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



- Counter implementation
  - Every page entry has a counter; every time page is referenced through this entry, copy the clock into the counter.
  - When a page needs to be changed, look at the counters to determine which are to change.

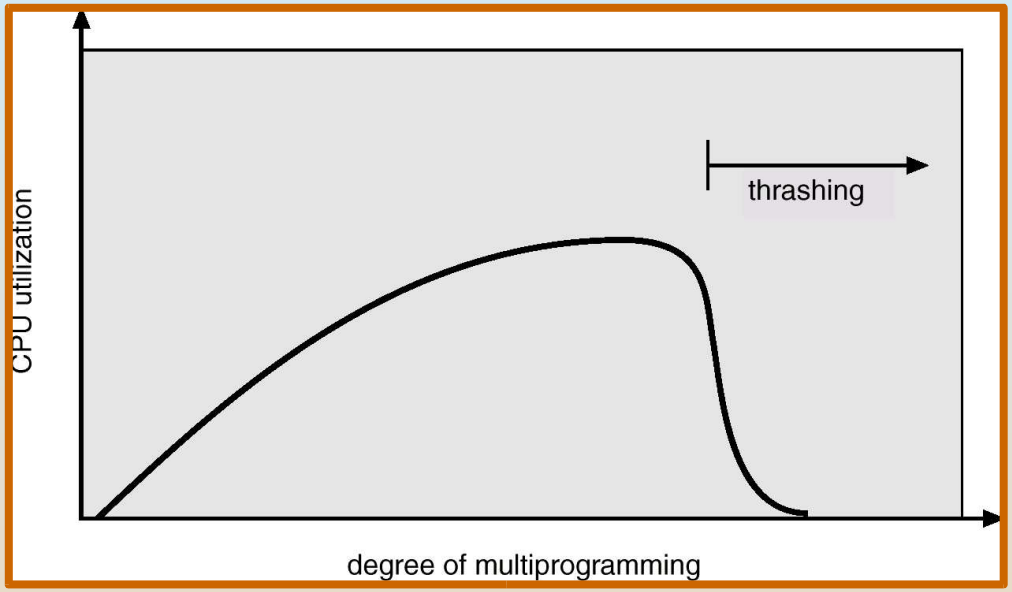
Counting Algorithms

- Keep a counter of the number of references that have been made to each page.
- LFU Algorithm: replaces page with smallest count.
- MFU Algorithm: based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

Thrashing

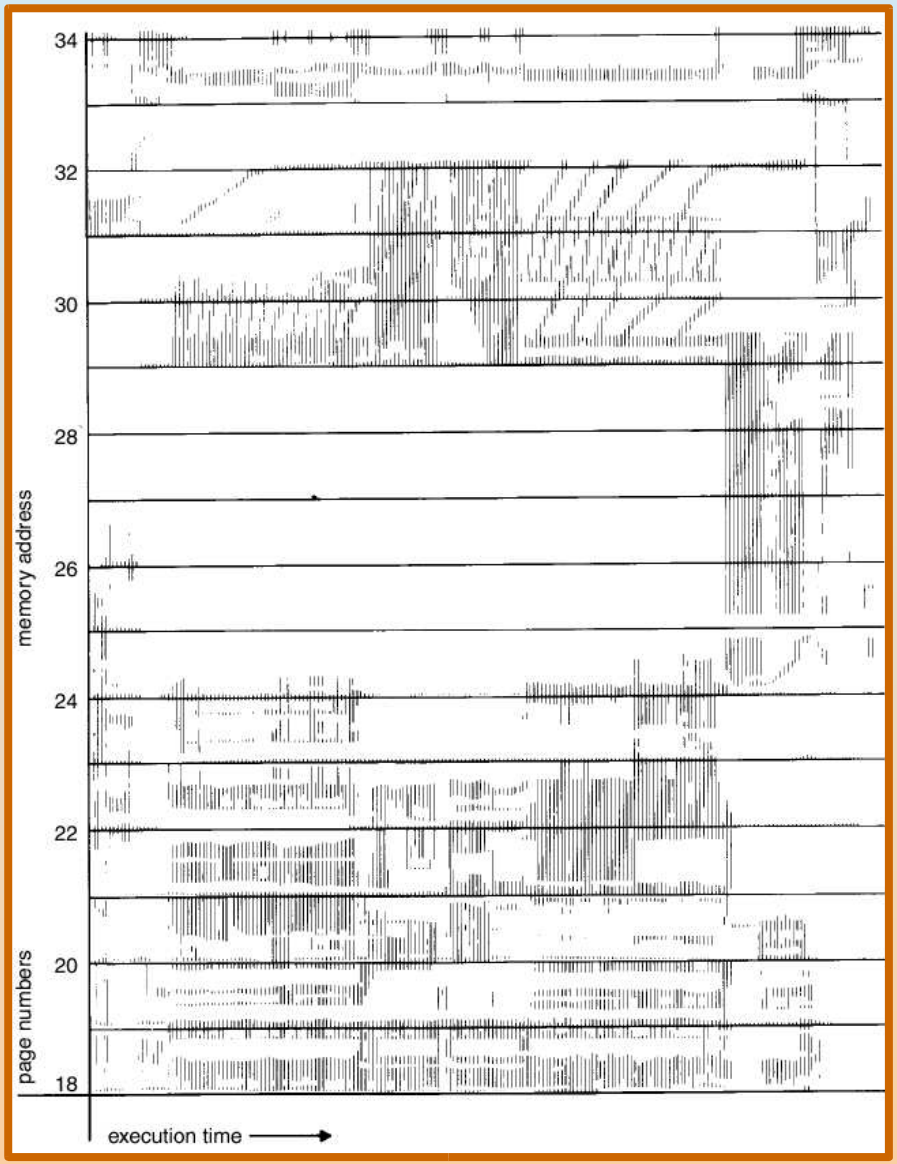
- If a process does not have “enough” pages, the page-fault rate is very high. This leads to:
  - low CPU utilization.
  - operating system thinks that it needs to increase the degree of multiprogramming.
  - another process added to the system.
- Thrashing => a process is busy swapping pages in and out.

Thrashing



- Why does paging work?
  Locality model
  - Process migrates from one locality to another.
  - Localities may overlap.
- Why does thrashing occur?
  size of locality > total memory size

Locality In A Memory-Reference Pattern



< what's happening here?

Other Considerations

- Program data structure
  - `int A[ ][ ] = new int[1024][1024];`
  - Each row is stored in one page
  - Program 1 :

```
for (j = 0; j < A.length; j++)
   for (i = 0; i < A.length; i++)
         A[i,j] = 0;
```

    1024 x 1024 page faults

  - Program 2

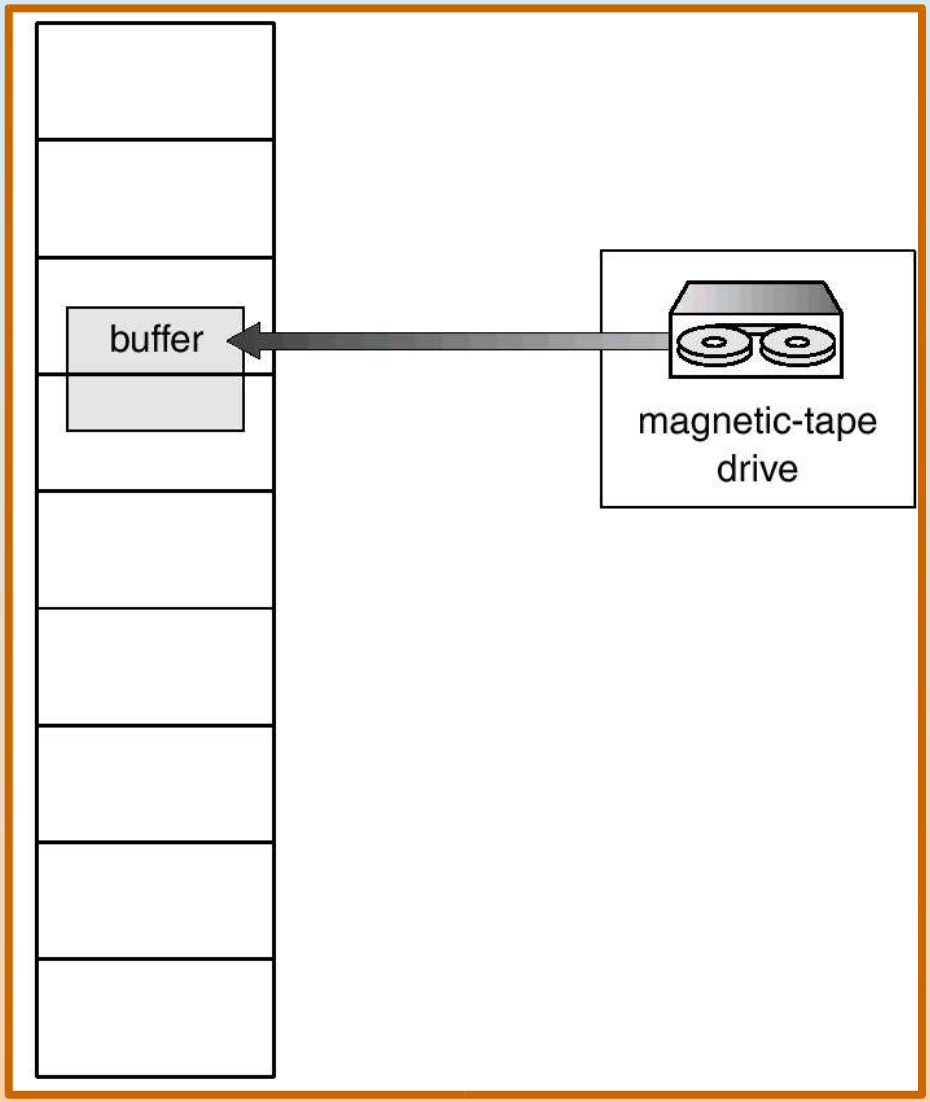
```
for (i = 0; i < A.length; i++)
   for (j = 0; j < A.length; j++)
         A[i,j] = 0;
```

    1024 page faults

Other Considerations (Cont.)

- I/O Interlock – Pages must sometimes be locked into memory.
- Consider I/O. Pages that are used for copying a file from a device must be locked from being selected for eviction by a page replacement algorithm.
  - Why?

Reason Why Frames Used For I/O Must Be In Memory



Some interesting memory issues

- where can the system get memory resources?
  - motherboard, system bus, network, ...
- how big should pages be?
  - depends on type/size of system
- what is NUMA?
- should the OS or application be memory locality aware?
- how to get significant functionality in small memory?
  - PDAs, cell phones, ...
- what problems occur with extremely large memory systems?
  - for example, Sun F15K can have 500MB+ memory
- what are advantages/disadvantages of "RAM disks"?
- Reading assignment:
  - cne.gmu.edu/pjd/PUBS/bvm.pdf