Operating System Concepts

SunBeam Institute of Information & Technology, Hinjwadi, Pune & Karad.

Trainer: Akshita Chanchlani
Email: akshita.chanchlani@sunbeaminfo.com

Fragmentation

As 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.

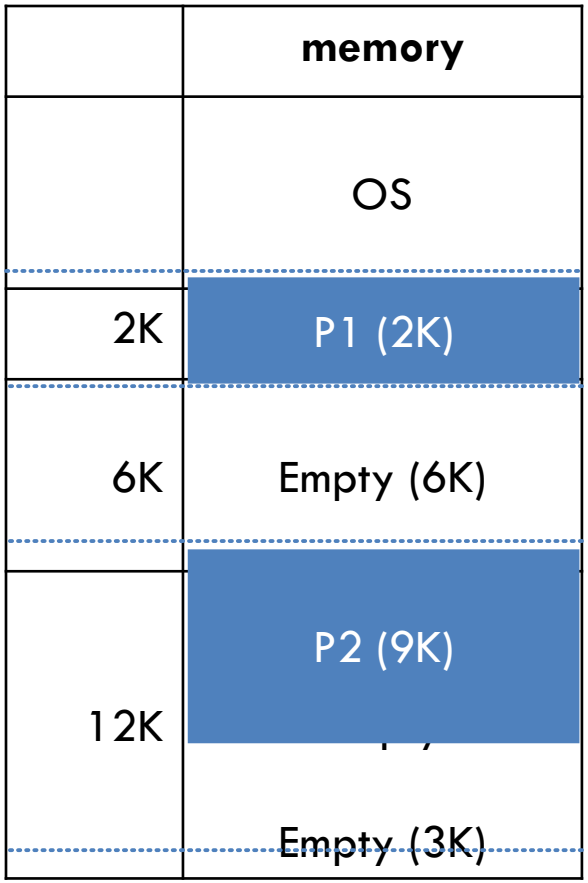
External fragmentation

- 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.

fragmentation



If a whole partition is currently not being used, then it is called *an* ***external fragmentation.***

If a partition is being used by a process requiring some memory smaller than the partition size, then it is called an ***internal fragmentation.***

Memory Allocation Types

Contiguous Memory Allocation **- One process – One piece of memory.**

- **Single** Partition Allocation.
- **Multiple** Partition Allocation
    - **Fixed** size partitioning (equal size, unequal size)
    - **Dynamic** Partitioning (First-Fit, Best-Fit , Worst-fit)
        - Problem (Internal and External Fragmentation)

Non Contiguous Memory Allocation - **One process – Many pieces of memory.**

- **Paging**
- **Segmentation**

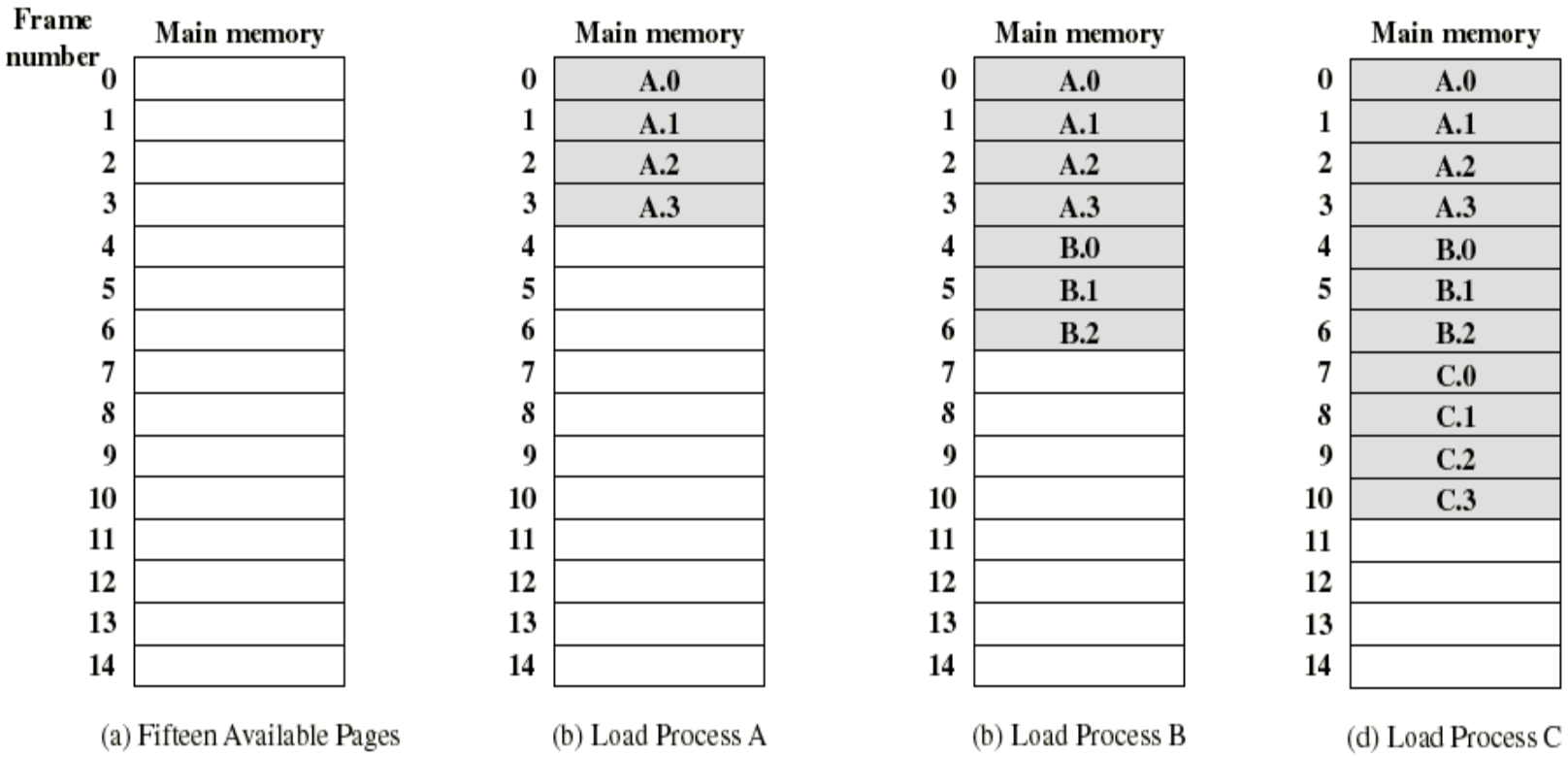
Simple Paging

- Main memory is partition into equal fixed-sized chunks (of relatively small size)
- Trick: each process is also divided into chunks of the same size called **pages**
- The process pages can thus be assigned to the available chunks in main memory called **frames** (or page frames)
- Consequence: a process does not need to occupy a contiguous portion of memory

Paging

- Divide physical memory into fixed-sized blocks called **frames** (size is power of 2, between 512 bytes and 8,192 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

Example of process loading



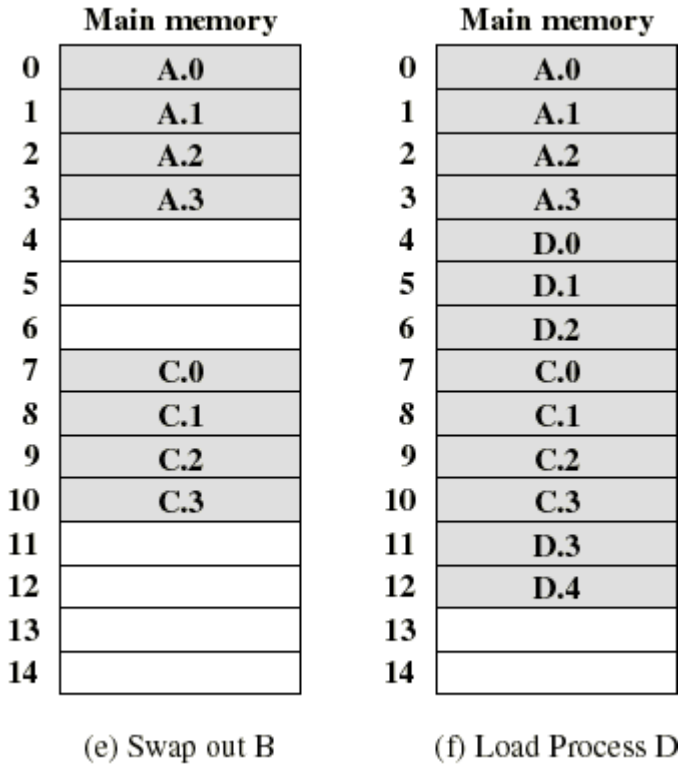
Now suppose that process B is swapped out

Example of process loading (cont.)

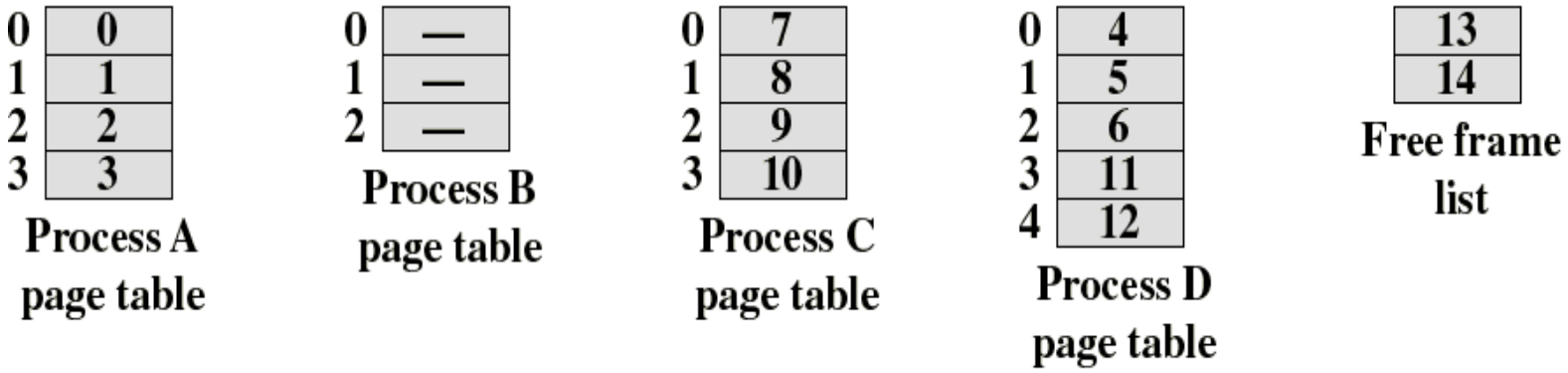
When process A and C are blocked, the pager loads a new process D consisting of 5 pages

Process D does not occupied a contiguous portion of memory

There is no external fragmentation



Page Tables



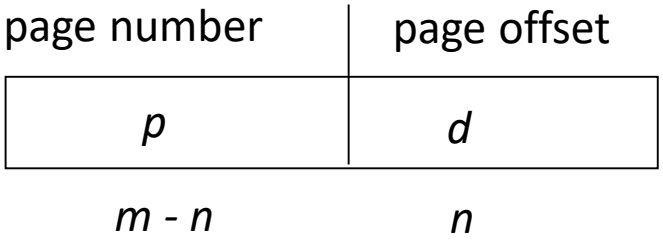
- The OS now needs to maintain (in main memory) a **page table** for each process
- Each entry of a page table consist of the frame number where the corresponding page is physically located
- The page table is indexed by the page number to obtain the frame number
- A free frame list, available for pages, is maintained

Logical address used in paging

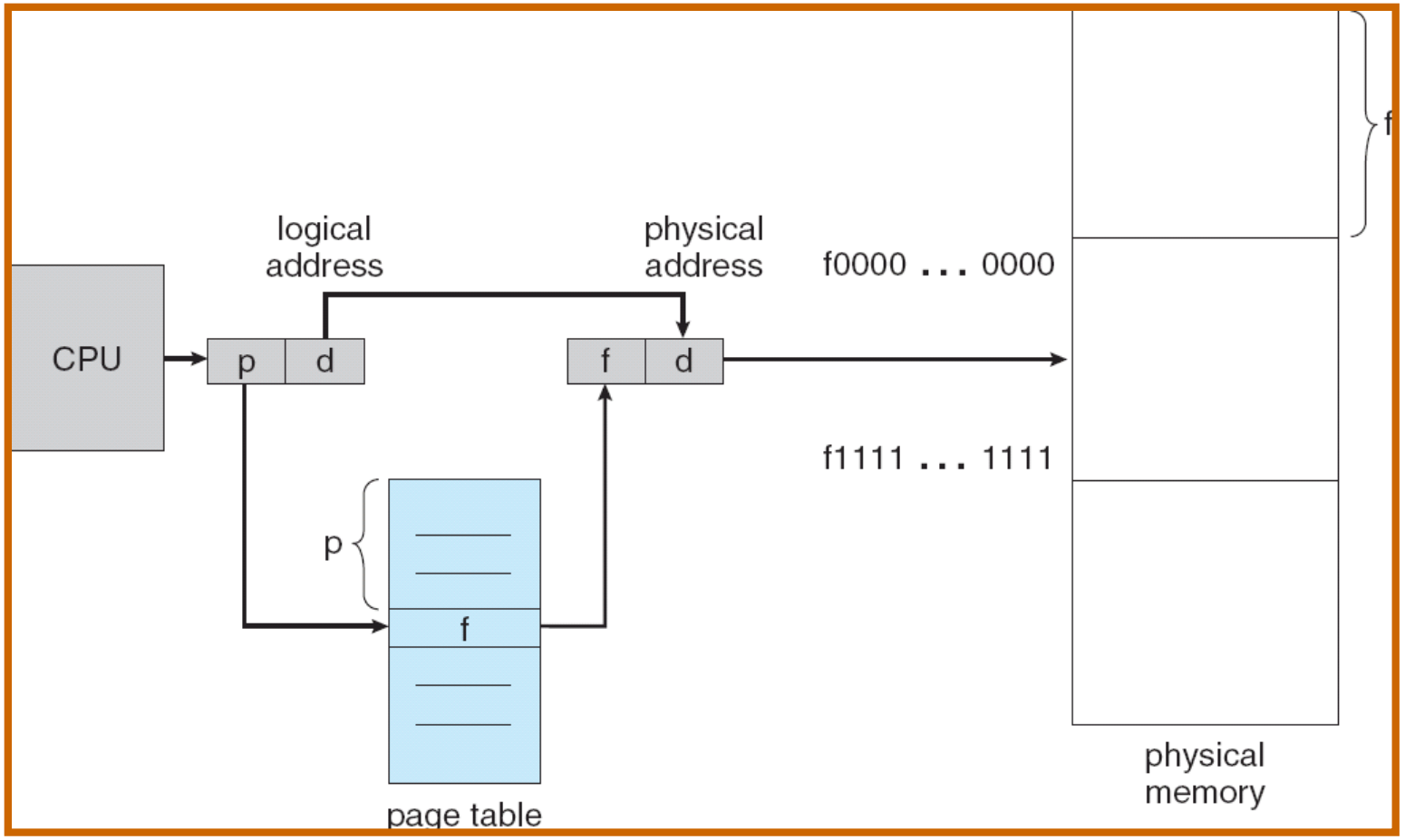
- Within each program, each logical address must consist of a page number and an offset within the page.
- A CPU register always holds the starting physical address of the page table of the currently running process.
- Presented with the logical address (page number, offset) the processor accesses the page table to obtain the physical address (frame number, offset)

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
  - For given logical address space 2^m *and page size* 2^n



Paging Hardware



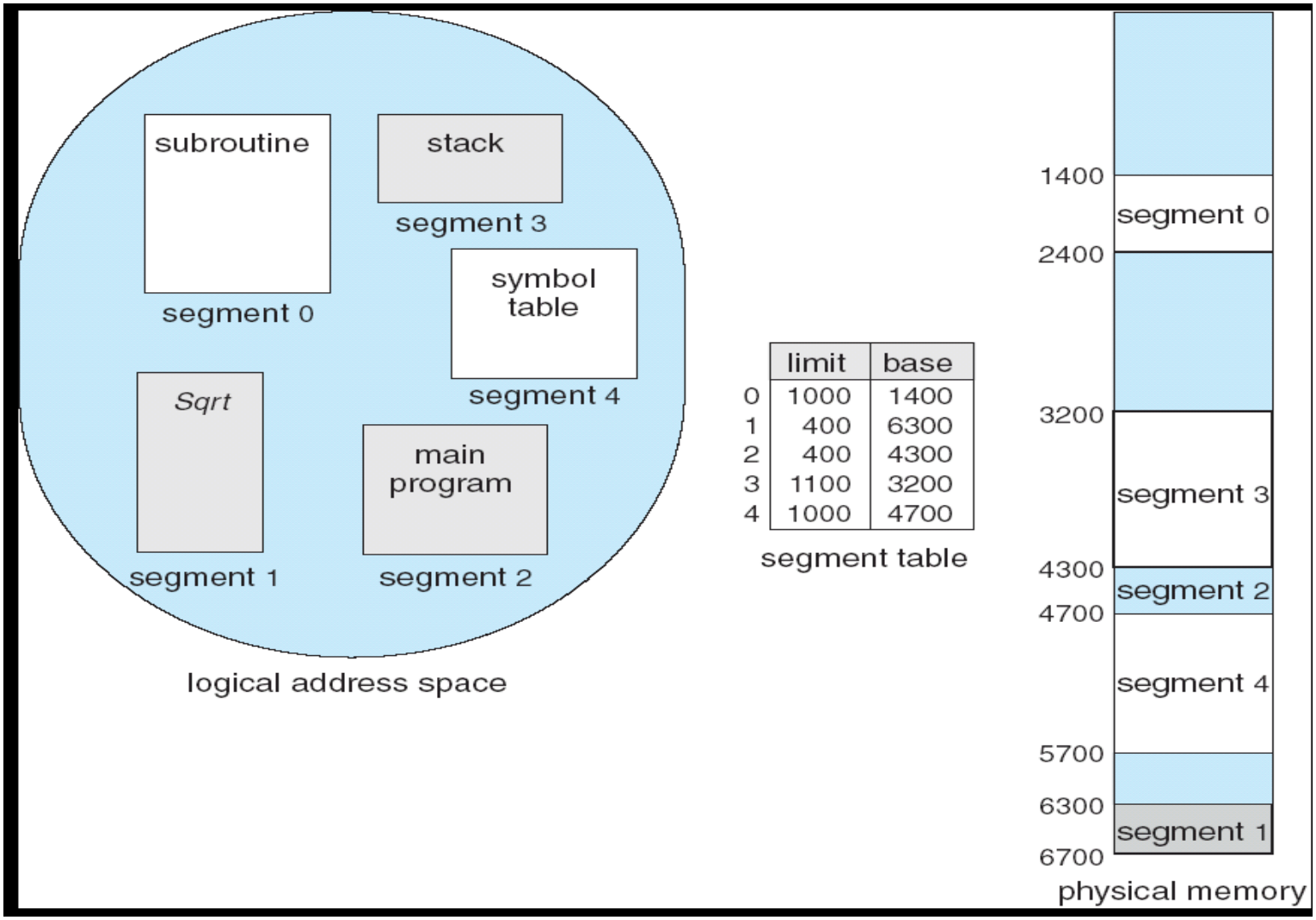
Segmentation

- Each program is subdivided into blocks of non-equal size called segments
- ***Segment***: a region of logically contiguous memory
- ***Segmentation-based transition***: use a table of base-and-bound pairs
- When a process gets loaded into main memory, its different segments can be located anywhere.
- Each segment is fully packed with instructs/data: no internal fragmentation
- There is external fragmentation; it is reduced when using small segments

Segmentation

- In contrast with paging, segmentation is visible to the programmer
  - –provided as a convenience to organize logically programs (ex: data in one segment, code in another segment)
  - –must be aware of segment size limit
- The OS maintains a **segment table** for each process. Each entry contains:
  - – the starting physical addresses of that segment.
  - –the length of that segment (for protection)

Segmentation Example



Logical address used in segmentation

- When a process enters the Running state, a CPU register gets loaded with the starting address of the process's segment table.
- Presented with a **logical address (segment number, offset) = (n,m),** the CPU indexes (with n) the segment table to obtain the starting physical address k and the length l of that segment
- The physical address is obtained by **adding** m to k (in contrast with paging)
  - the hardware also compares the offset m with the length l of that segment to determine if the address is valid

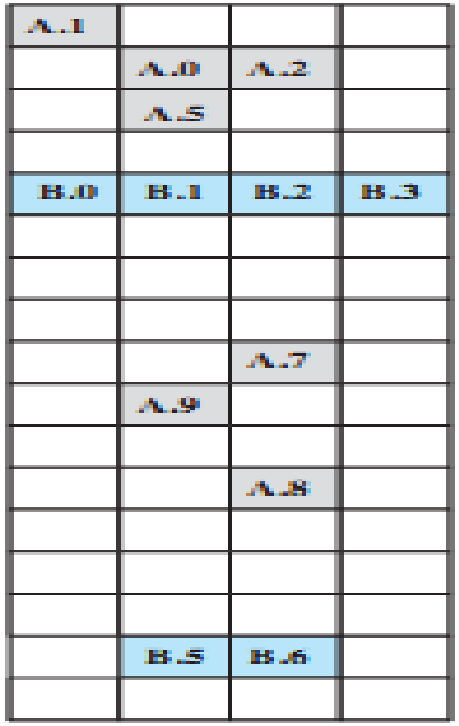
Virtual memory

- **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.
  - –Allows address spaces to be shared by several processes.
  - –Allows for more efficient process creation.
  - –More programs running concurrently.
  - –Less I/O needed to load or swap processes.
- Virtual memory gives the programmer the impression that he/she is dealing with a huge main memory (relying on available disk space). The OS loads automatically and on-demand pages of the running process.

- A process image may be larger than the entire main memory.
- The required pages need to be loaded into memory whenever required.

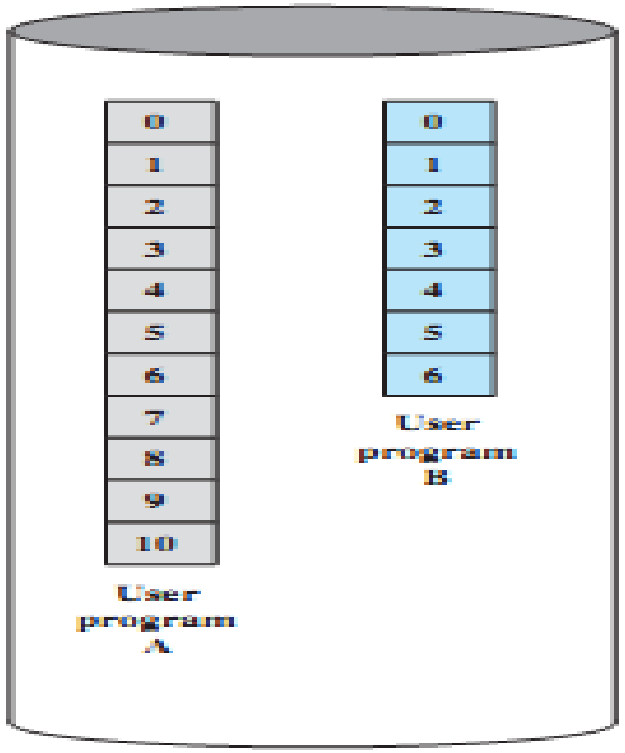
Virtual memory is implemented using Demand Paging or Demand Segmentation.

Virtual memory components



Main Memory

Main memory consists of a number of fixed-length frames, each equal to the size of a page. For a program to execute, some or all of its pages must be in main memory.



Disk

Secondary memory (disk) can hold many fixed-length pages. A user program consists of some number of pages. Pages for all programs plus the operating system are on disk, as are files.

Virtual Memory that is larger than Physical Memory