**Memory Management**

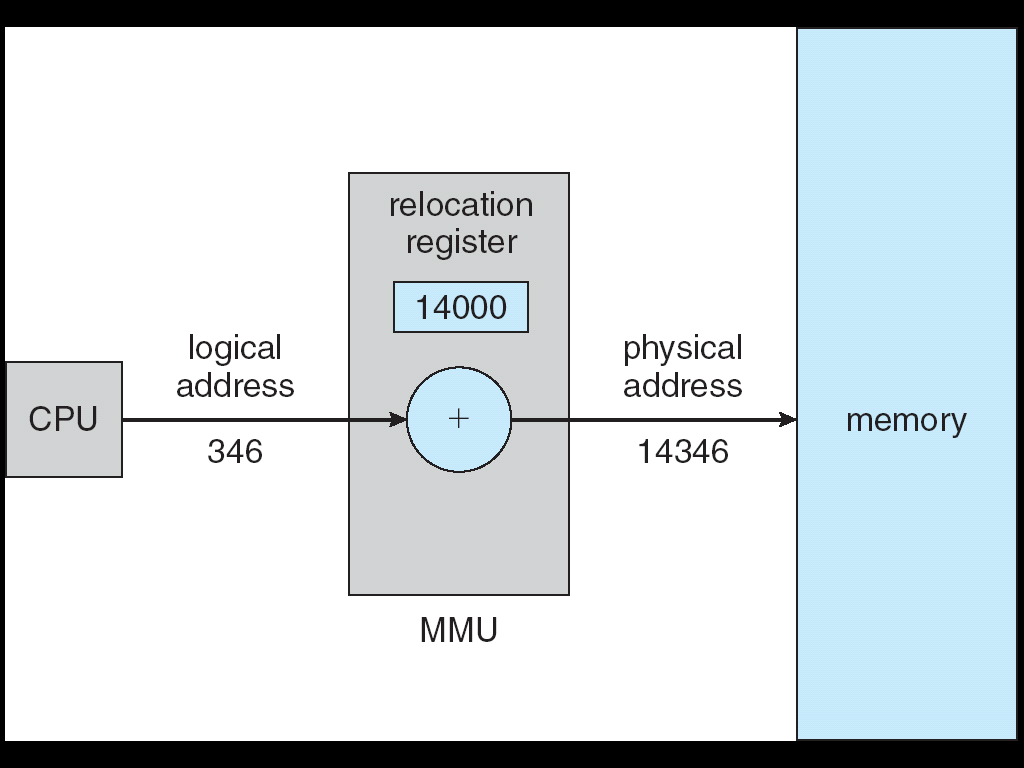
**Logical vs. Physical Address Space**

* 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.
* Logical and physical addresses are the same in compile-time and load-time; but differ in execution-time.

**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 addresses.

**Dynamic relocation using 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.

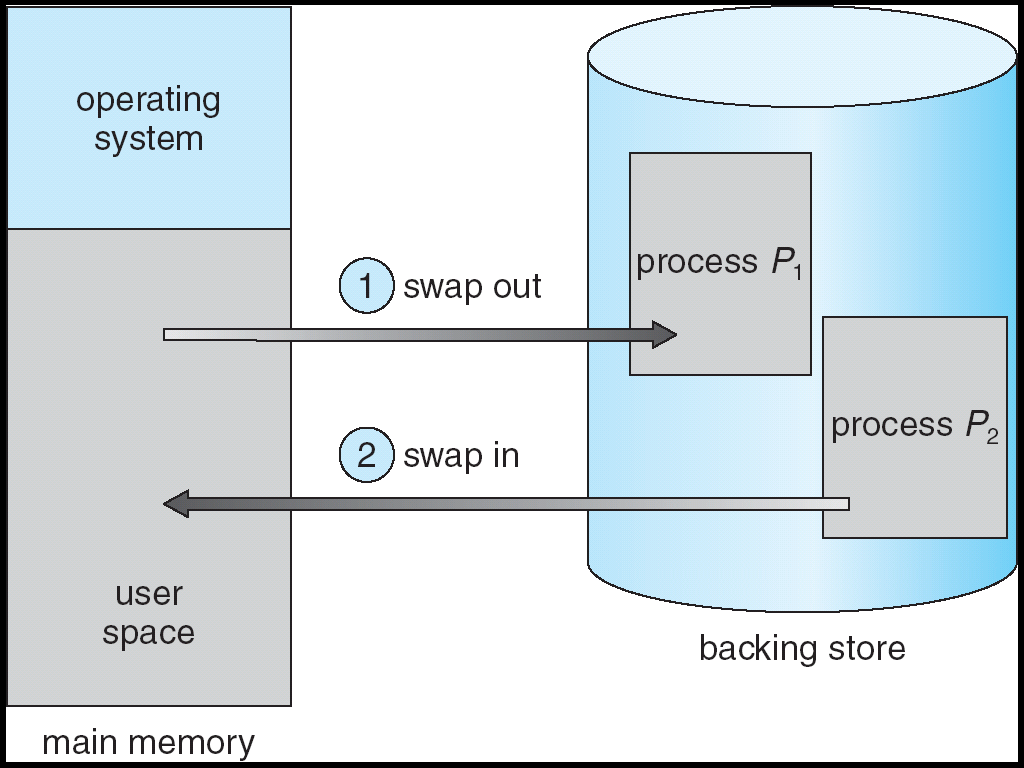
**Dynamic Linking**

* 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.

**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, Linux, and Windows)

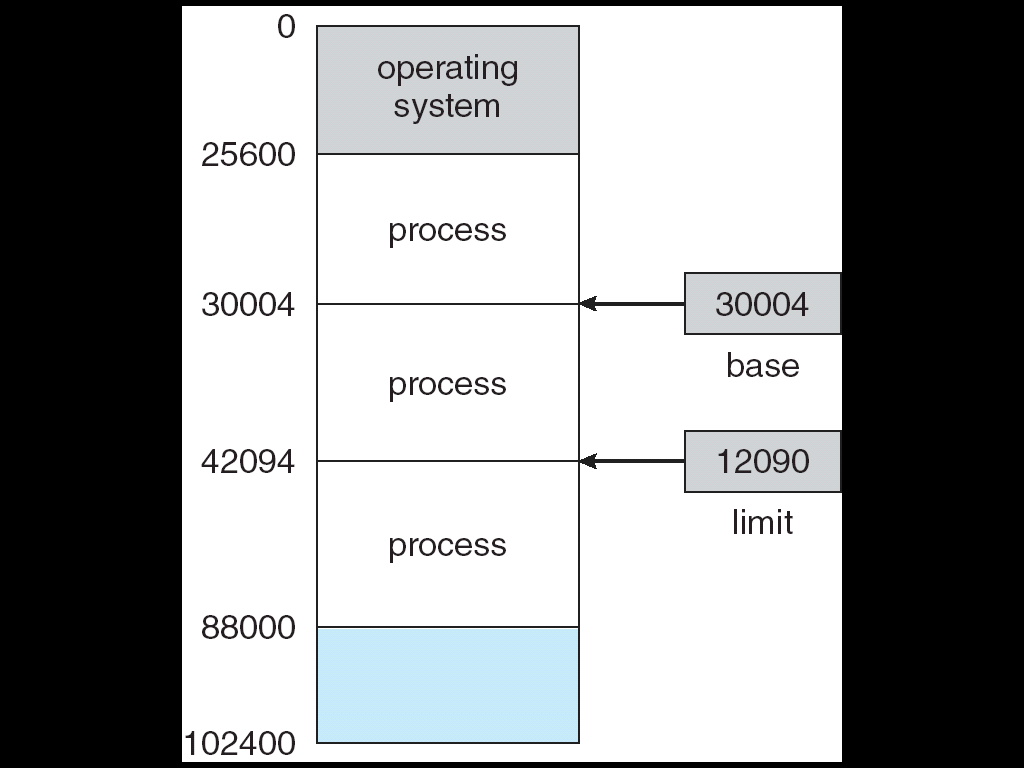
**Schematic View of Swapping**



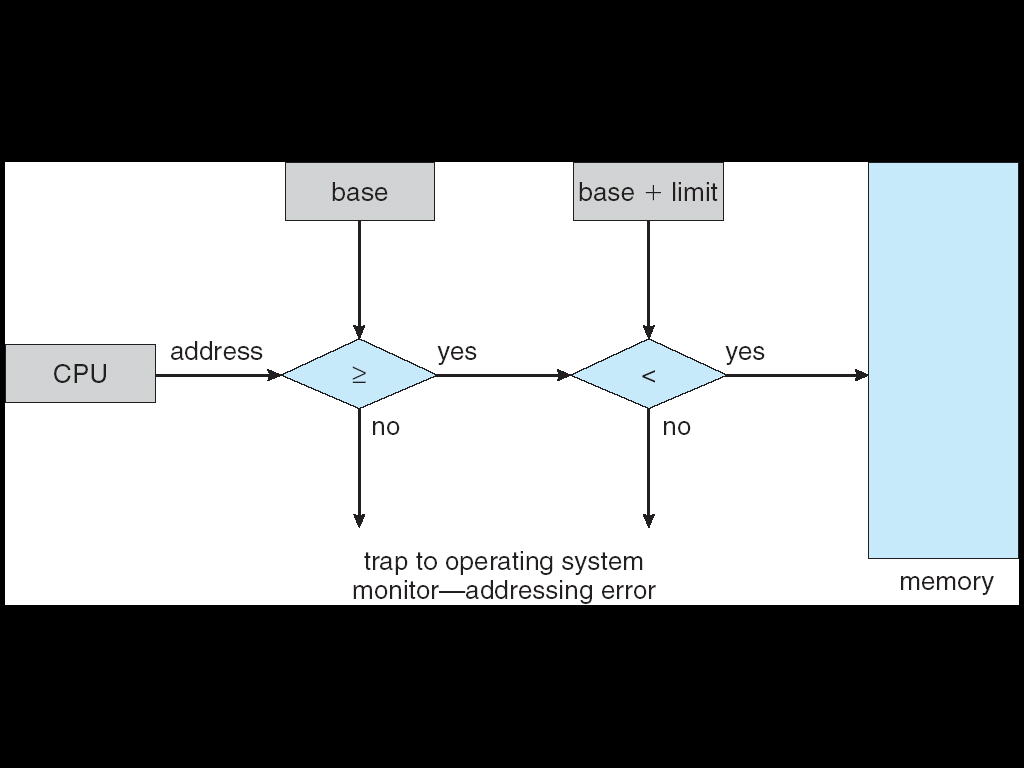
**Contiguous Allocation**

* Main memory usually divide into two partitions:
  1. **Resident operating system**, usually held in low memory with interrupt vector. (System area)
  2. **User processes** then held in high memory. (User area)
* **Relocation-register** scheme used to protect user processes from each other, and from changing operating-system 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

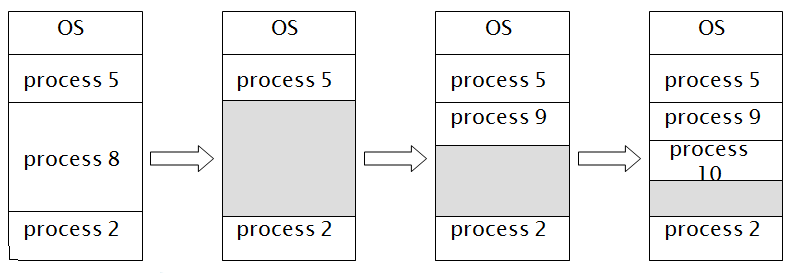
**A base and a limit register define a logical address space:**



**HW address protection with base and limit registers**



* **Multiple-partition allocation**
  + **Hole –** block of available memory; holes of various size 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: **A**) allocated partitions **B**) free partitions (hole)



**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.

**Note**: First-fit and Best-fit better than Worst-fit in terms of speed and storage utilization.

**2-Fragmentation problem**

* **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.

**Note**: Reduce external fragmentation by compaction Shuffle memory contents to place all free memory together in one large block.