Operating Systems

BITS Pilani
K K Birla Goa Campus

Dr. Lucy J. Gudino
Dept. of CS and IS

Last Class

- **Major responsibilities of OS**
- **Speed, Access time and Capacity**
- **Classification of information**
  - By role in program
  - By protection status
  - Addresses vs. Data
  - Uniprogramming or multiprogramming
- **Functions of memory manager**
  - keep track of which parts of memory are in use and which parts are not in use,
  - allocation memory to processes and deallocate
  - swapping between main memory and disk

Contd…

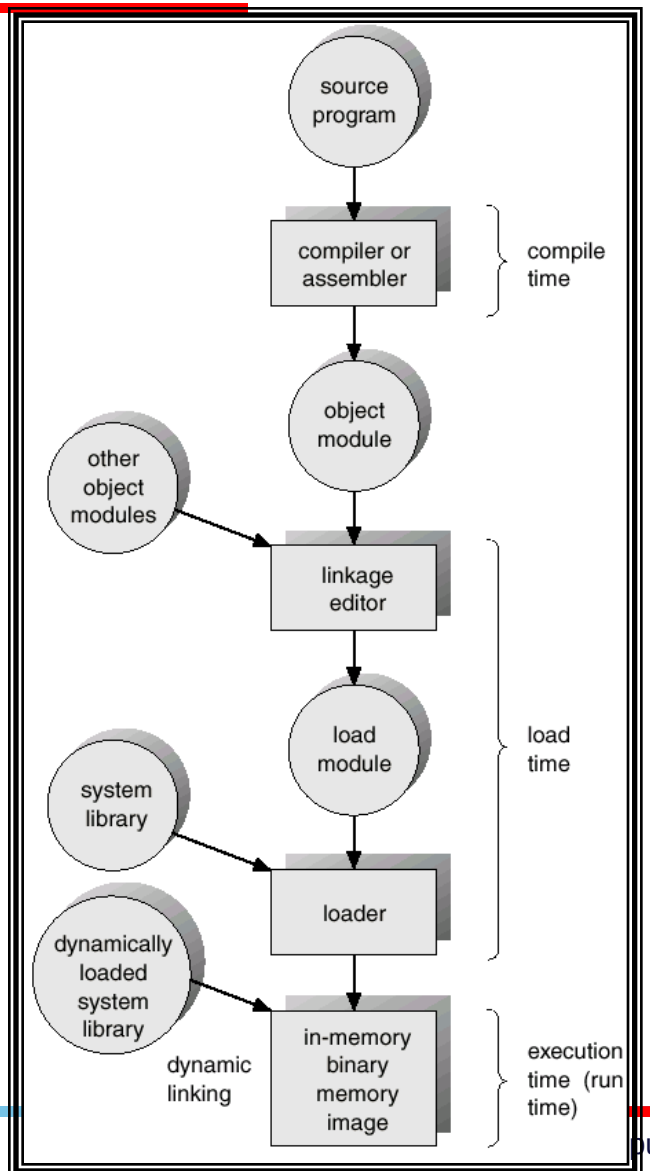
- **Five requirements**
  - Relocation
  - Protection
  - Sharing
  - Logical organization
  - Physical organization

Important Terms

- Physical address : address generated by the memory management unit
- Virtual address or Logical address : address generated by CPU
- Relative Address
- Absolute Address

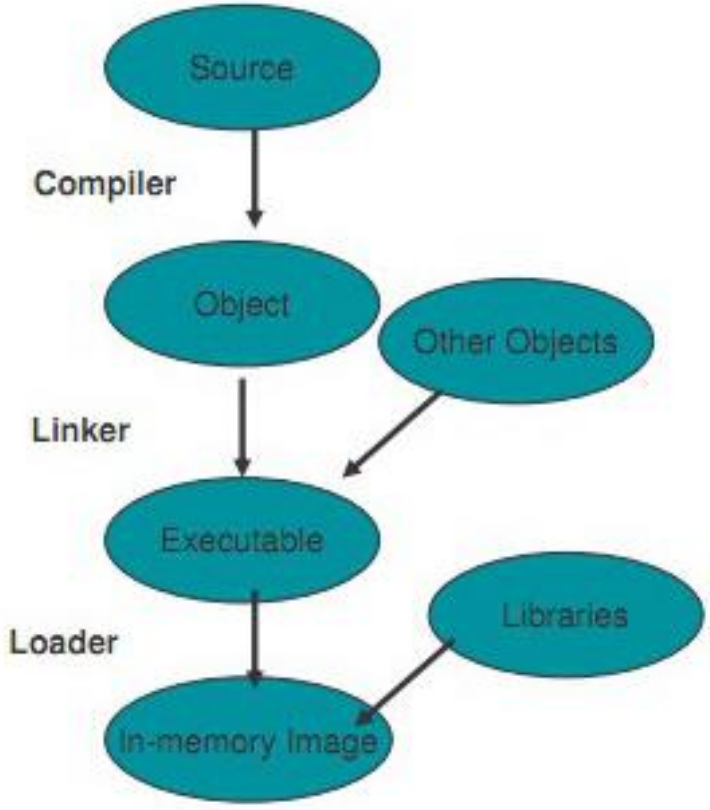
Address Binding

- Mapping from one address space to another
- 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 to run the program.
- User programs go through several steps before being run.



Contd…

- Source Program uses symbolic address
- Compiler bind these symbolic addresses to relocatable addresses
- Linkage editor or loader bind the relocatable addresses to absolute addresses



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.
  - **Load time**: Compiler must generate *relocatable* code if memory location is not known at compile time.
    - Final address binding will be done at load time
    - If starting address changes then reload the program
  - **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*).

Logical vs. Physical Address Space

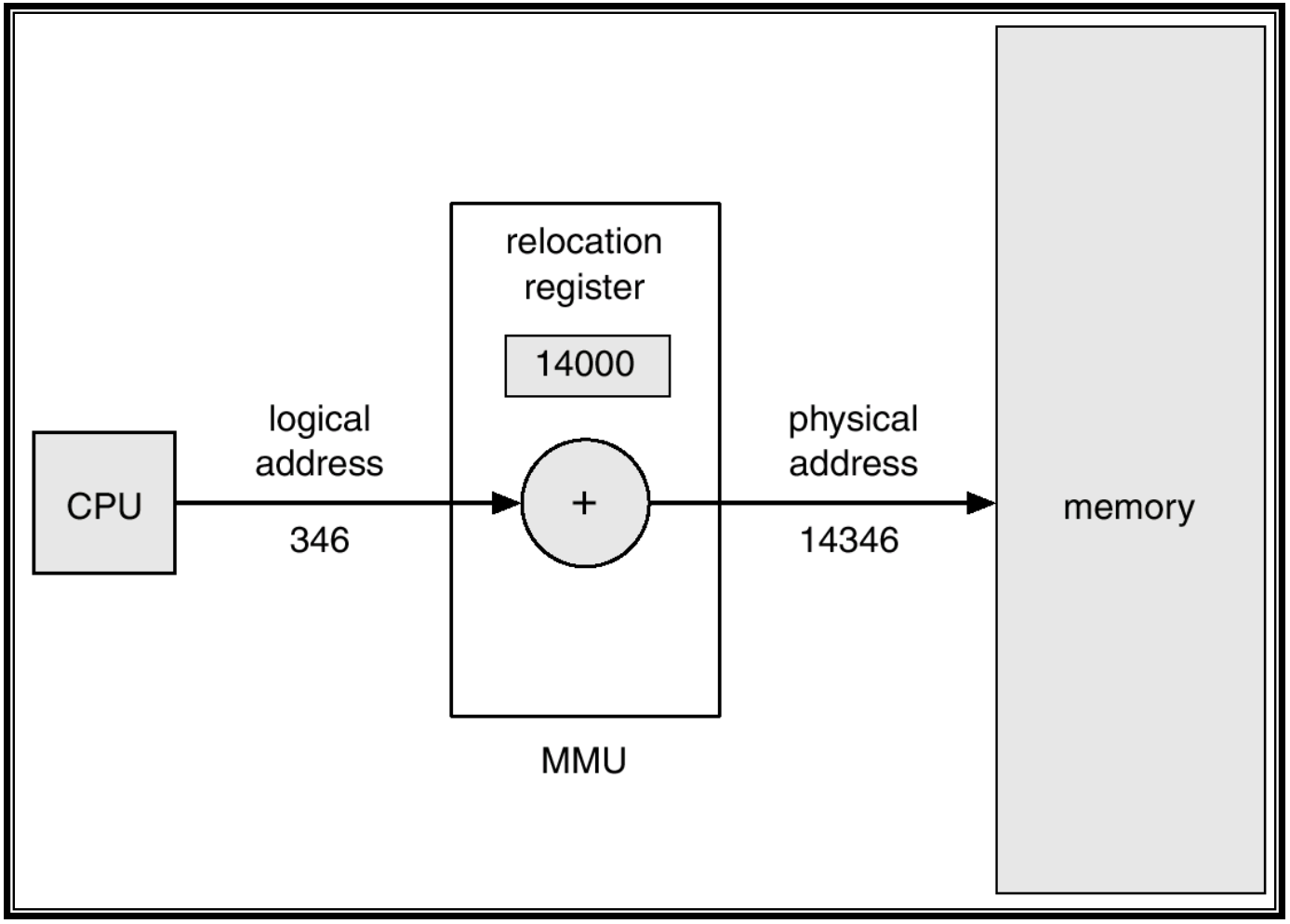
- *Logical address* – generated by the CPU; also referred to as *virtual address*.
- *Physical address* – MAR register- address seen by the memory unit.

- 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.
- The set of all logical address generated by a program

Memory-Management Unit (MMU)

- Hardware device that maps virtual to physical address.
- The user program deals with *logical* addresses; it never sees the *real* physical addresses.
- Two different types of addresses:
  - Logical: range 0 to max.
  - Physical: range R+0 to R+max; where R is a base value.
- Note: The user generates only logical addresses and thinks that the process runs in locations 0 to max.
- 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.

Dynamic relocation using a relocation register

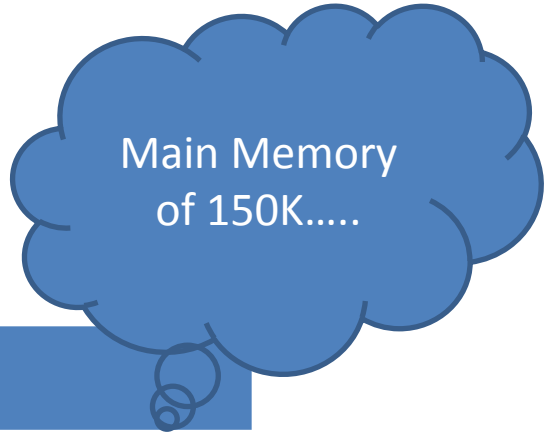


Overlays

- The entire program and data of a process must be in the physical memory for the process to execute.
- The size of a process is limited to the size of physical memory.
- If a process is larger than the amount of memory, a technique called overlays can be used.
- Overlays is to keep in memory only those instructions and data that are needed at any given time.
- When other instructions are needed, they are loaded into space that was occupied previously by instructions that are no longer needed.
- Overlays are implemented by user, no special support needed from operating system, programming design of overlay structure is complex.

Example : Assembler

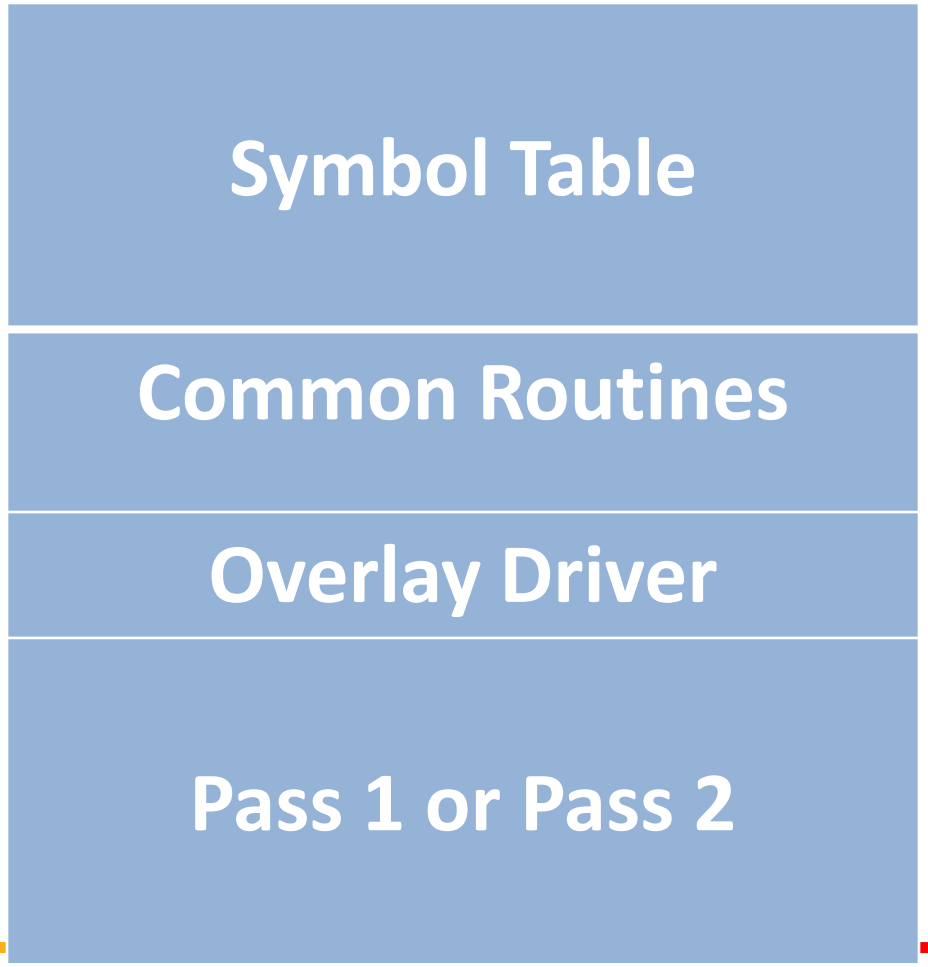
- Two passes : Pass 1 and Pass 2
  - Pass 1: Constructs Symbol Table
  - Pass 2: Generates Machine code



	Size
Pass 1	70K
Pass 2	80K
Symbol Table	25K
Common Routines	25K
Total	200K

Contd…

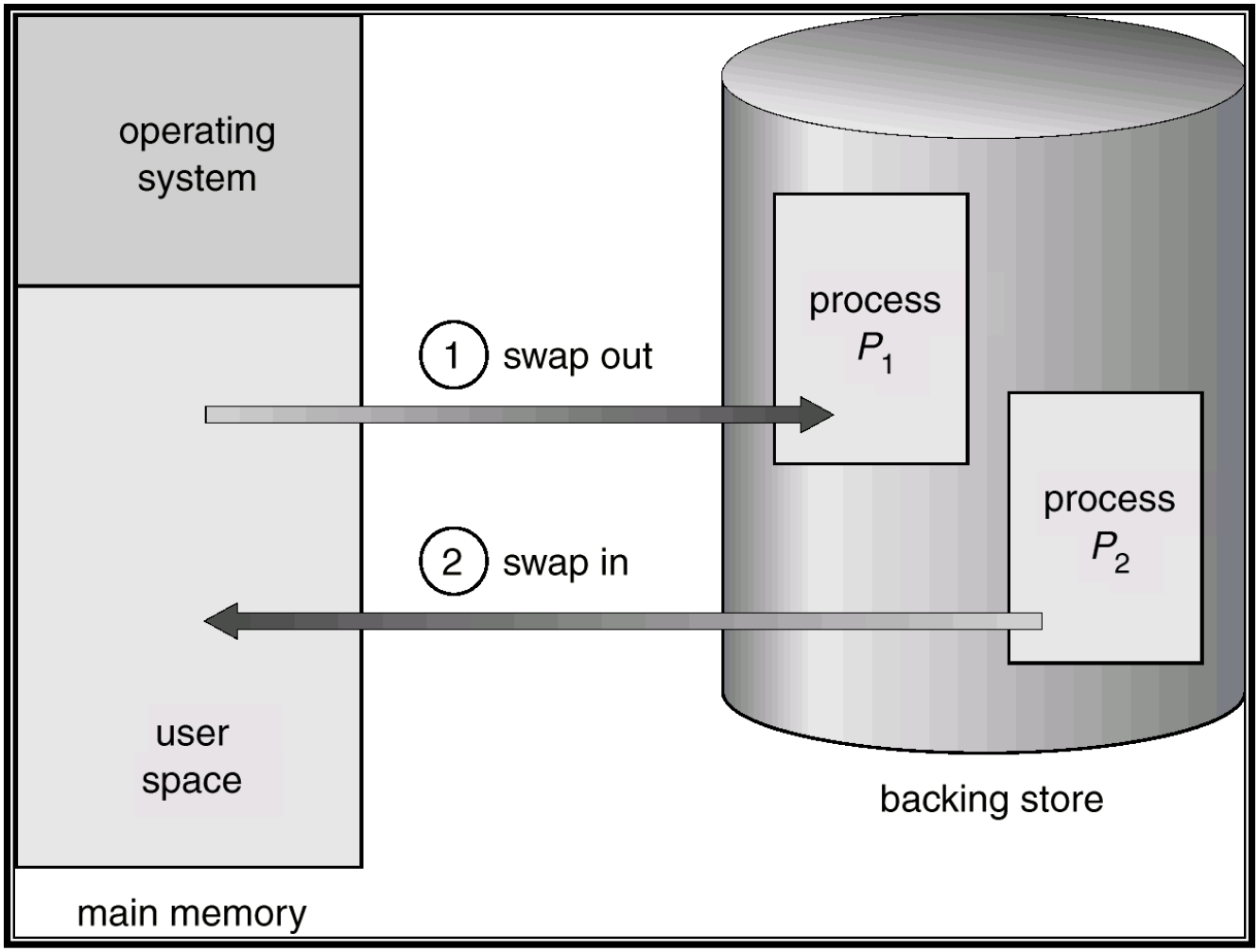
Overlay A requires 130K and Overlay B requires 140K



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

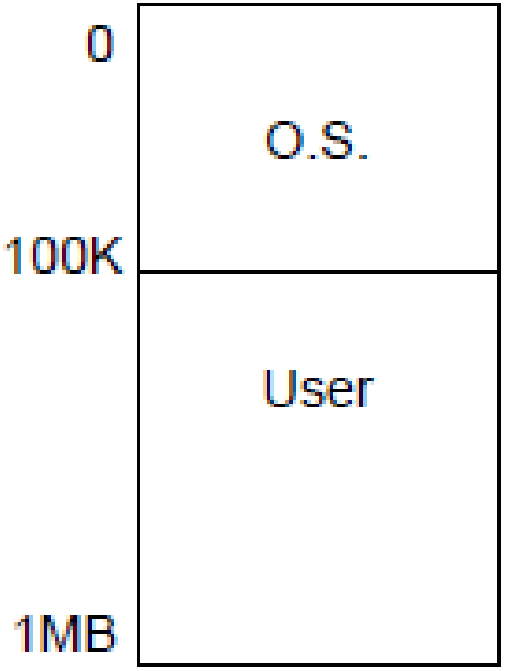


Example

- Consider a process P1 of size 100KB
- Transfer rate of a hard disk is 1MB/sec
- Average Latency is 8 msec
- How much time required to transfer P1 to and from memory?
- Ans: 216 msec
- For efficient CPU utilization, execution time for each process should be long relative to the swap time.
- A Round Robin scheduling algorithm, the time slice should be larger than 216 milliseconds (from the above example).

Contiguous Allocation

- Main memory usually into two partitions:
  - Resident operating system,
  - User processes
- Resident OS usually held in low memory (deciding factor is the position of interrupt vector.)
- Example : Swap time for 100K process and 900K process



Single – Partition Allocation

- Each process is contained in a single contiguous section of memory
  - Protect O.S. code and data from changes by the user process.
  - We can provide protection by using a relocation register with a limit register.
  - 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.

Hardware Support for Relocation and Limit Registers