OPERATING SYSTEMS

MEMORY MANAGEMENT

OPERATING SYSTEM
Memory Management

What is memory?

- ✓ Memory is central to the operation of a modern computer system.
- ✓ memory consists of a large array of words or bytes.
- ✓ Each with its own address.
- ✓ The CPU fetches instructions from memory according to the value of the program counter.
- ✓ These instructions may cause additional loading from and storing to specific memory addresses.

- The main memory is central to the operation of a modern computer.
- Main Memory is a large array of words or bytes, ranging in size from hundreds of thousands to billions.
- Main memory is a repository of rapidly available information shared by the CPU and I/O devices. Main memory is the place where programs and information are kept when the processor is effectively utilizing them.
- Main memory is associated with the processor, so moving instructions and information into and out of the processor is extremely fast.
    - Main memory is also known as RAM(Random Access Memory). This memory is a volatile memory. RAM lost its data when a power interruption occurs.

MEMORY MANAGEMENT

Definitions

What is Memory Management :

In a multiprogramming computer, the operating system resides in a part of memory and the rest is used by multiple processes. The task of subdividing the memory among different processes is called memory management. Memory management is a method in the operating system to manage operations between main memory and disk during process execution. The main aim of memory management is to achieve efficient utilization of memory.

Why Memory Management is required:

- Allocate and de-allocate memory before and after process execution.
- To keep track of used memory space by processes.
- To minimize fragmentation issues.
- To proper utilization of main memory.
- To maintain data integrity while executing of process.

- The concept of a logical *address space* that is bound to a separate *physical address space* is central to proper memory management.

MEMORY MANAGEMENT

Definitions

- **Logical address** –;It is also referred to as *virtual address.It is generated by CPU only.*
- **Physical address** – 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

MEMORY MANAGEMENT

Definitions

Relocatable Means that the program image can reside anywhere in physical memory.

Binding Programs need real memory in which to reside. When is the location of that real memory determined?

- This is called **mapping** logical to physical addresses.
- This binding can be done at compile/link time. Converts symbolic to relocatable. Data used within compiled source is offset within object module.

Compiler: If it's known where the program will reside, then absolute code is generated. Otherwise compiler produces relocatable code.

Load: Binds relocatable to physical. Can find best physical location.

Execution: The code can be moved around during execution. Means flexible virtual mapping.

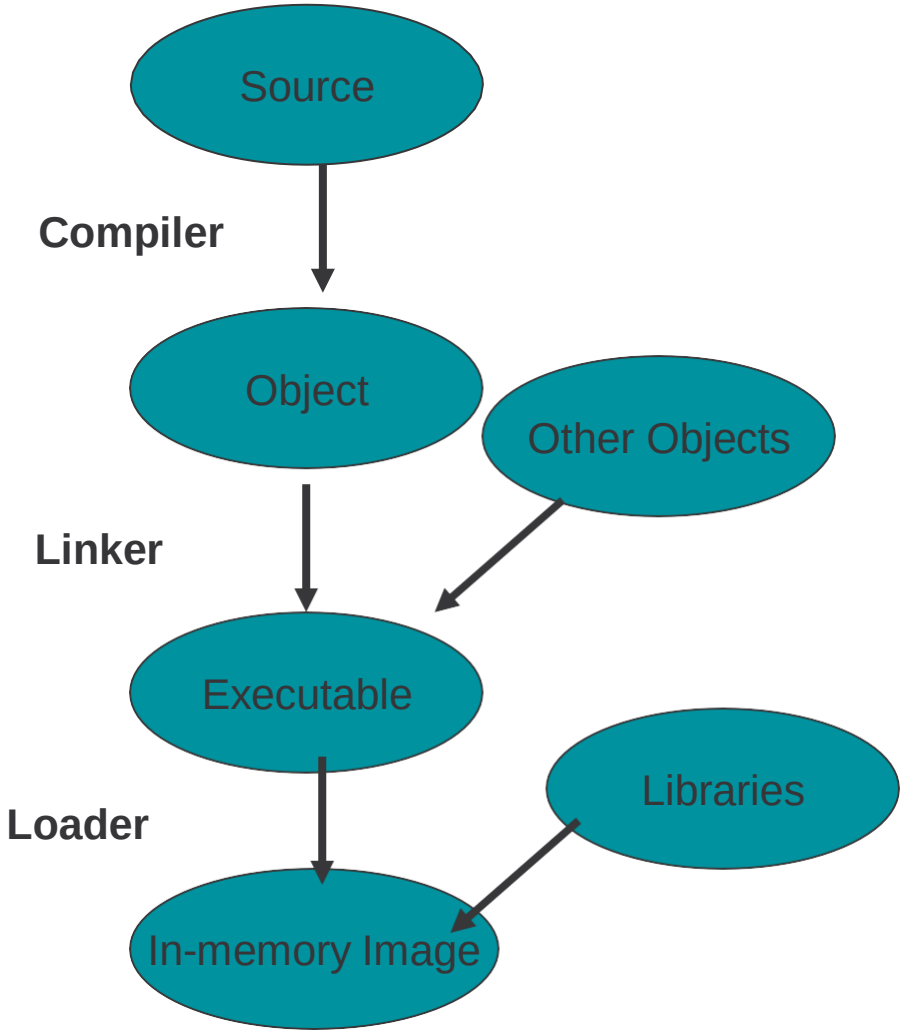
MEMORY MANAGEMENT

Binding Logical To Physical

This binding can be done at compile/link time. Converts symbolic to relocatable. Data used within compiled source is offset within object module.

- Can be done at load time. Binds relocatable to physical.
- Can be done at run time. Implies that the code can be moved around during execution.

The next example shows how a compiler and linker actually determine the locations of these effective addresses.



MEMORY MANAGEMENT

More Definitions

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 OS 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.

Memory Management Performs the above operations. Usually requires hardware support.

MEMORY MANAGEMENT

SINGLE PARTITION ALLOCATION

BARE MACHINE:

- No protection, no utilities, no overhead.
- This is the simplest form of memory management.
- Used by hardware diagnostics, by system boot code, real time/dedicated systems.
- logical == physical
- User can have complete control. Commensurably, the operating system has none.

DEFINITION OF PARTITIONS:

- Division of physical memory into fixed sized regions. (Allows addresses spaces to be distinct = one user can't muck with another user, or the system.)
- The number of partitions determines the level of multiprogramming. Partition is given to a process when it's scheduled.
- Protection around each partition determined by
  bounds (upper, lower)
  base / limit.
- These limits are done in hardware.

MEMORY MANAGEMENT

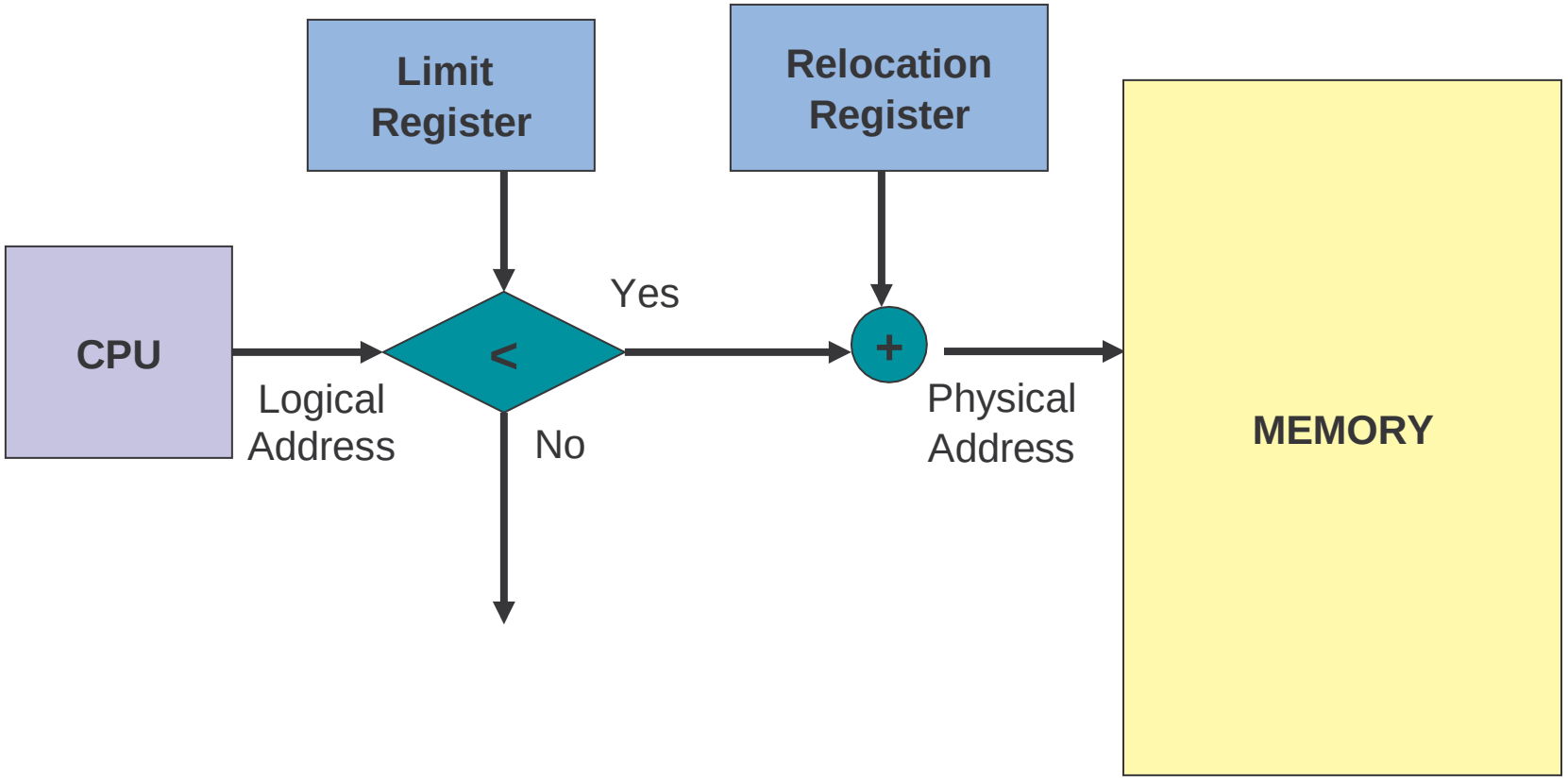
SINGLE PARTITION ALLOCATION

RESIDENT MONITOR:

- Primitive Operating System.
- Usually in low memory where interrupt vectors are placed.
- Must check each memory reference against fence (fixed or variable) in hardware or register. If user generated address < fence, then illegal.
- User program starts at fence -> fixed for duration of execution. Then user code has fence address built in. But only works for static-sized monitor.
- If monitor can change in size, start user at high end and move back, OR use fence as base register that requires address binding at execution time. Add base register to every generated user address.
- Isolate user from physical address space using logical address space.
- Concept of "mapping addresses" shown on next slide.

MEMORY MANAGEMENT

SINGLE PARTITION ALLOCATION



MEMORY MANAGEMENT

CONTIGUOUS ALLOCATION

All pages for a process are allocated together in one chunk.

JOB SCHEDULING

- Must take into account who wants to run, the memory needs, and partition availability. (This is a combination of short/medium term scheduling.)
- Sequence of events:
- In an empty memory slot, load a program
- THEN it can compete for CPU time.
- Upon job completion, the partition becomes available.
- Can determine memory size required (either user specified or "automatically").

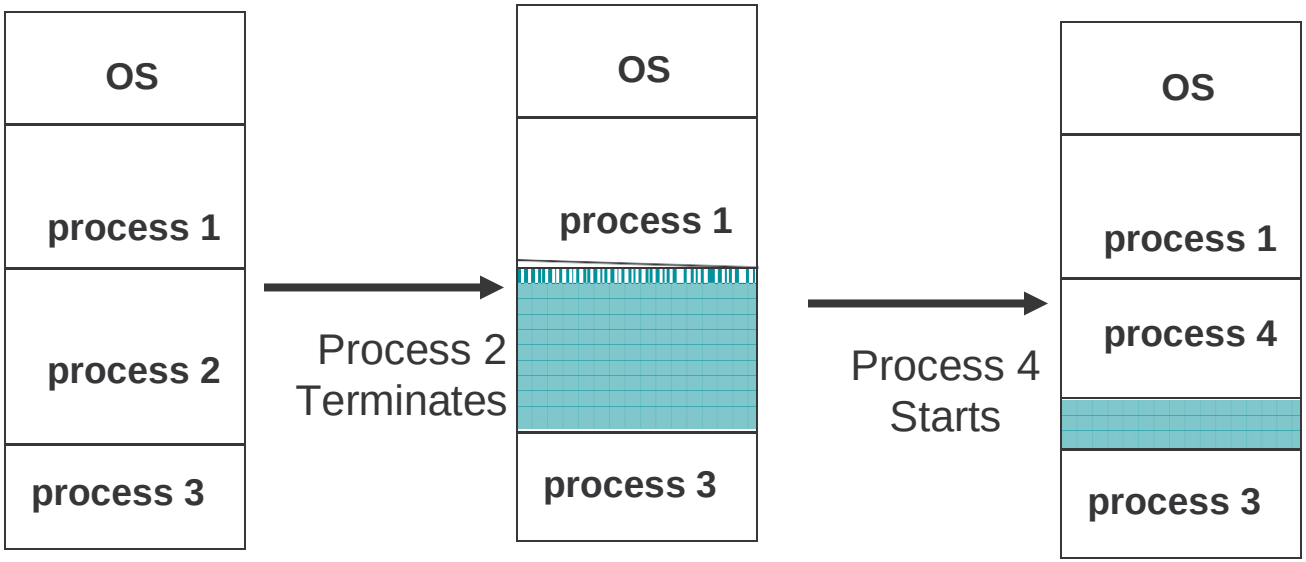
MEMORY MANAGEMENT

CONTIGUOUS ALLOCATION

DYNAMIC STORAGE

- (Variable sized holes in memory allocated on need.)
- Operating System keeps table of this memory - space allocated based on table.
- Adjacent freed space merged to get largest holes - buddy system.

ALLOCATION PRODUCES HOLES



MEMORY MANAGEMENT

CONTIGUOUS ALLOCATION

HOW DO YOU ALLOCATE MEMORY TO NEW PROCESSES?

First fit - allocate the first hole that's big enough.
Best fit - allocate smallest hole that's big enough.
Worst fit - allocate largest hole.

(First fit is fastest, worst fit has lowest memory utilization.)

- Avoid small holes (**external fragmentation**). This occurs when there are many small pieces of free memory.
- What should be the minimum size allocated, allocated in what chunk size?
- Want to also avoid **internal fragmentation.** This is when memory is handed out in some fixed way (power of 2 for instance) and requesting program doesn't use it all.

MEMORY MANAGEMENT

LONG TERM SCHEDULING

If a job doesn't fit in memory, the scheduler can

wait for memory

skip to next job and see if it fits.

What are the pros and cons of each of these?

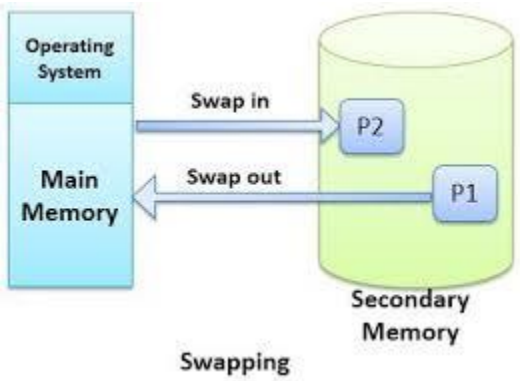
There's little or no internal fragmentation (the process uses the memory given to it - the size given to it will be a page.)

But there can be a great deal of external fragmentation. This is because the memory is constantly being handed cycled between the process and free.

MEMORY MANAGEMENT

Swapping:

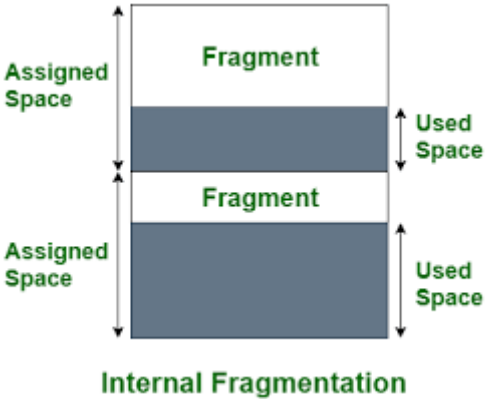
It is a mechanism in which a process can be swapped tempararly out of main memory to secondary memory.later its swap back to main memory



Swapping

Fragmentation:

An unwanted problem that occurs in the os in which a process loaded and unloaded from memory.



Internal Fragmentation