

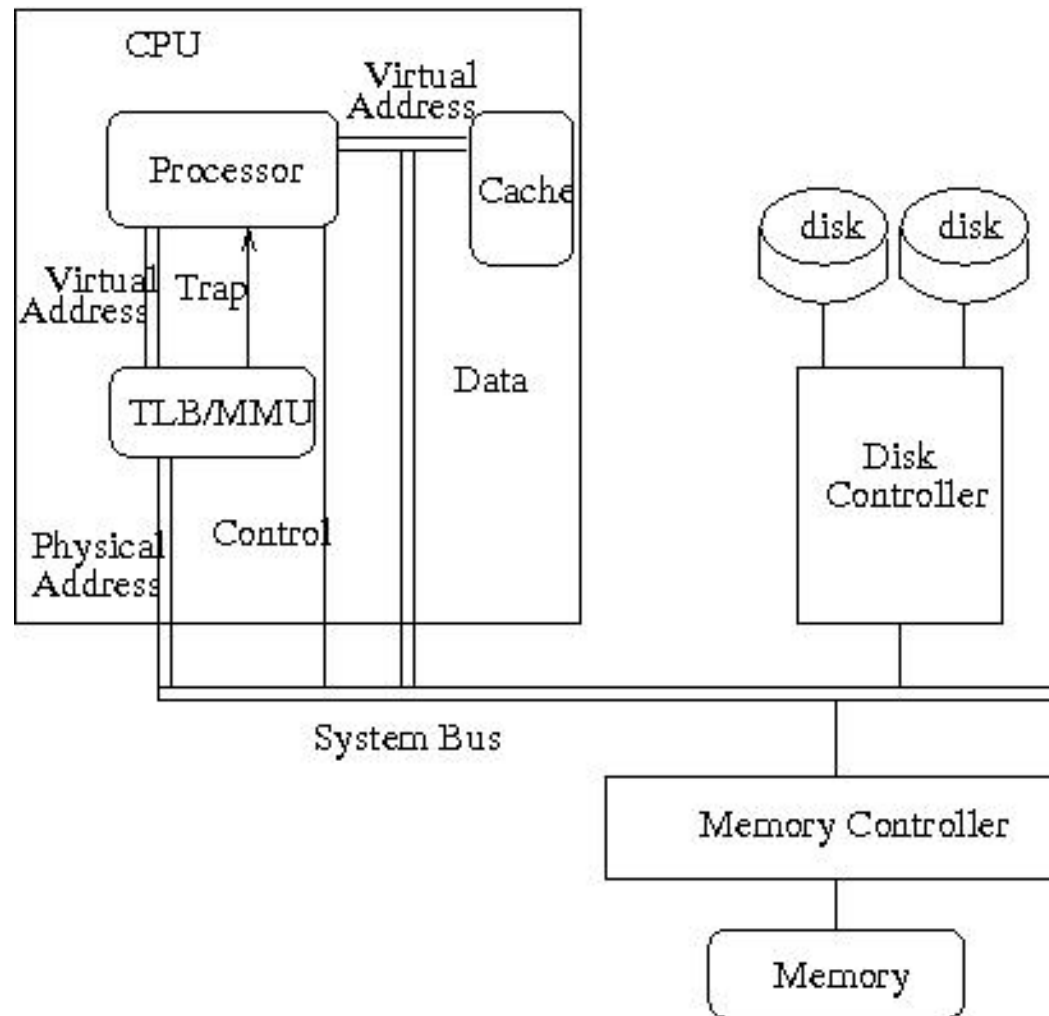
Finding the Executing Process

Basic Concepts: Address Spaces

- ***Physical address space***: Collection of physical memory addresses supported by the hardware
 - From address 0 to address **MAX_{sys}**
- ***Logical/virtual address space***: Collection of addresses that the process can access (this is the process's view)
 - From address 0 to address **MAX_{prog}**
- ***Segment***: A chunk of physical memory assigned to a process



Back to Architecture

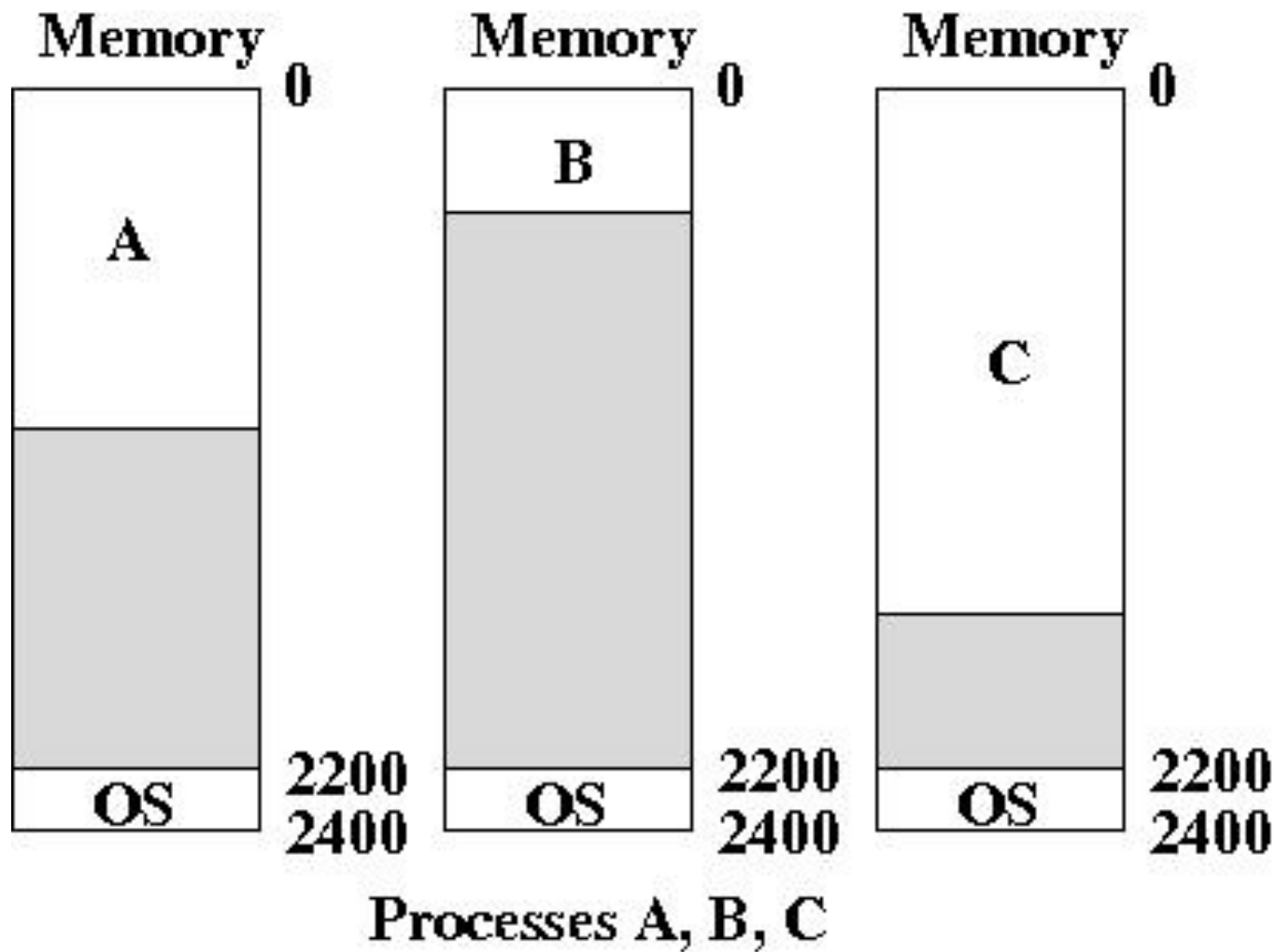


Basic Concepts: Address Generation

Uniprogramming:

- One process executes at a time
- Process is always loaded starting at address 0
- Process executes in a contiguous section of memory
- OS gets fixed part of memory
- Compiler can generate physical addresses
- Maximum address space size = Memory Size – OS Size
- OS is protected from processes by address checking

Uniprogramming Picture



Multiple Programs Sharing Memory: Requirements

Transparency:

- We want multiple processes to coexist in memory
- No process should be aware that memory is shared
- Processes should not care what physical portion of memory they get

Safety:

- Processes must not be able to corrupt each other
- Processes must not be able to corrupt the OS

Efficiency:

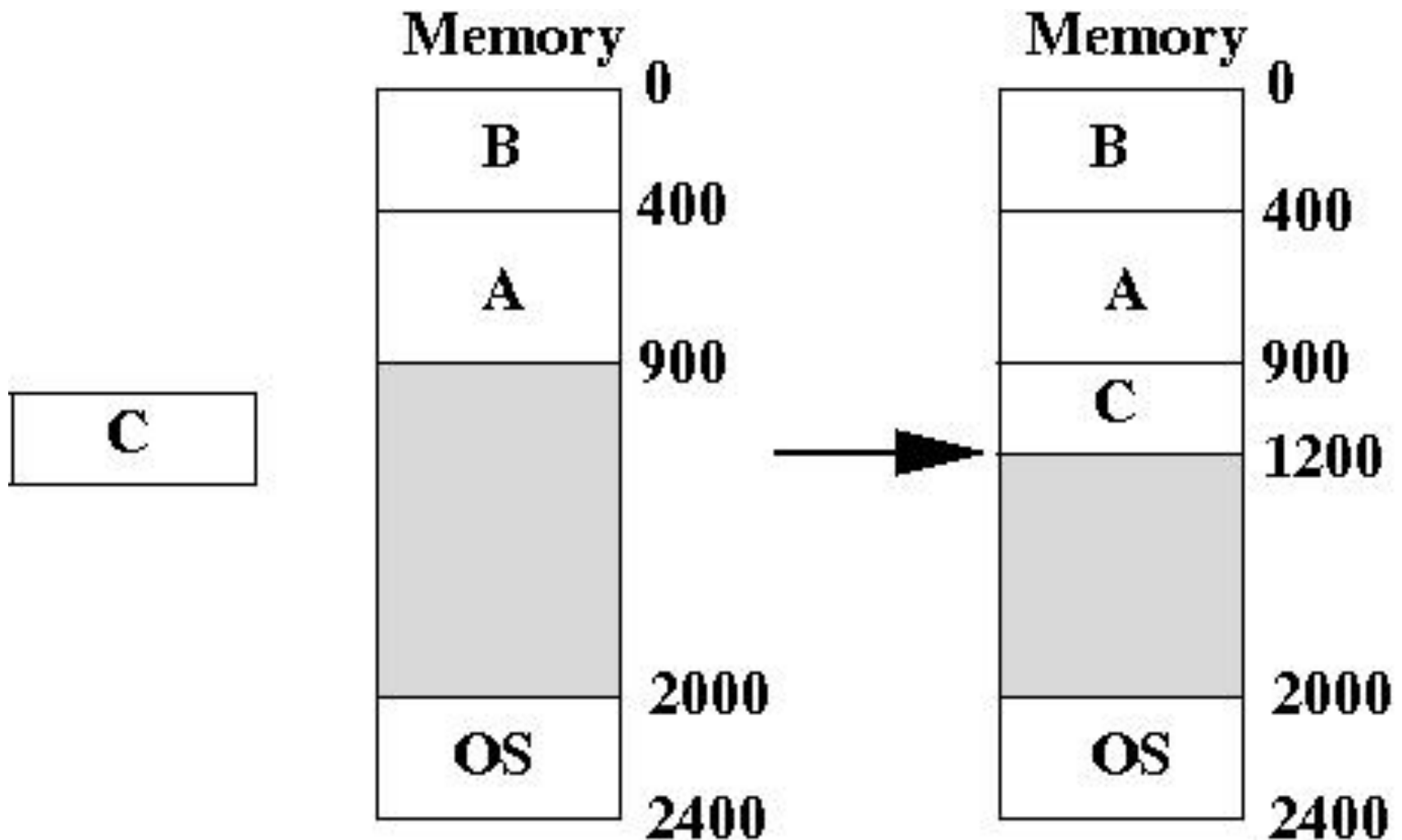
- Performance of the CPU and memory should not be degraded badly due to sharing

Enabling Multiprogramming: Static Relocation

Relocation

- Put the OS in the highest memory
- Assume at compile/link time that the process starts at 0 with a Maximum address = Memory Size – OS Size
- When the OS loads the process, it allocates a contiguous segment of memory in which the process fits. If it does not fit, the OS waits for a process to terminate
- The first (smallest) physical address of the process is the *base* address and the largest physical address the process can access is the *limit* address
- The base address is also known as the *relocation address*

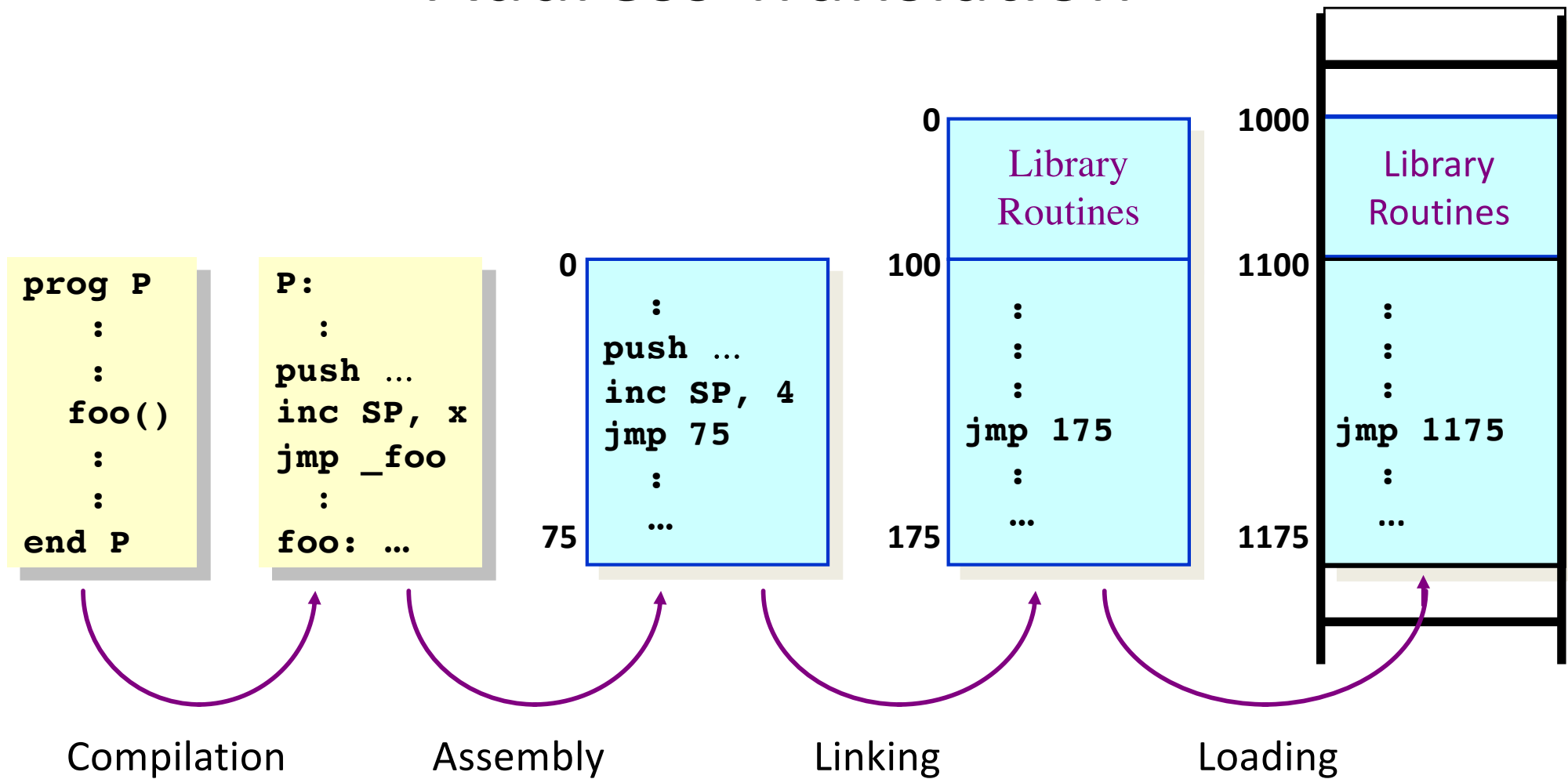
Relocation Picture



Static Relocation

- Loader adjusts the addresses in a process to reflect its location in memory by
 - It add the base address to the address in the binary
- Once process is assigned a place in memory and starts executing, OS cannot move it

Static Relocation: Address Translation



The Compilation Pipeline

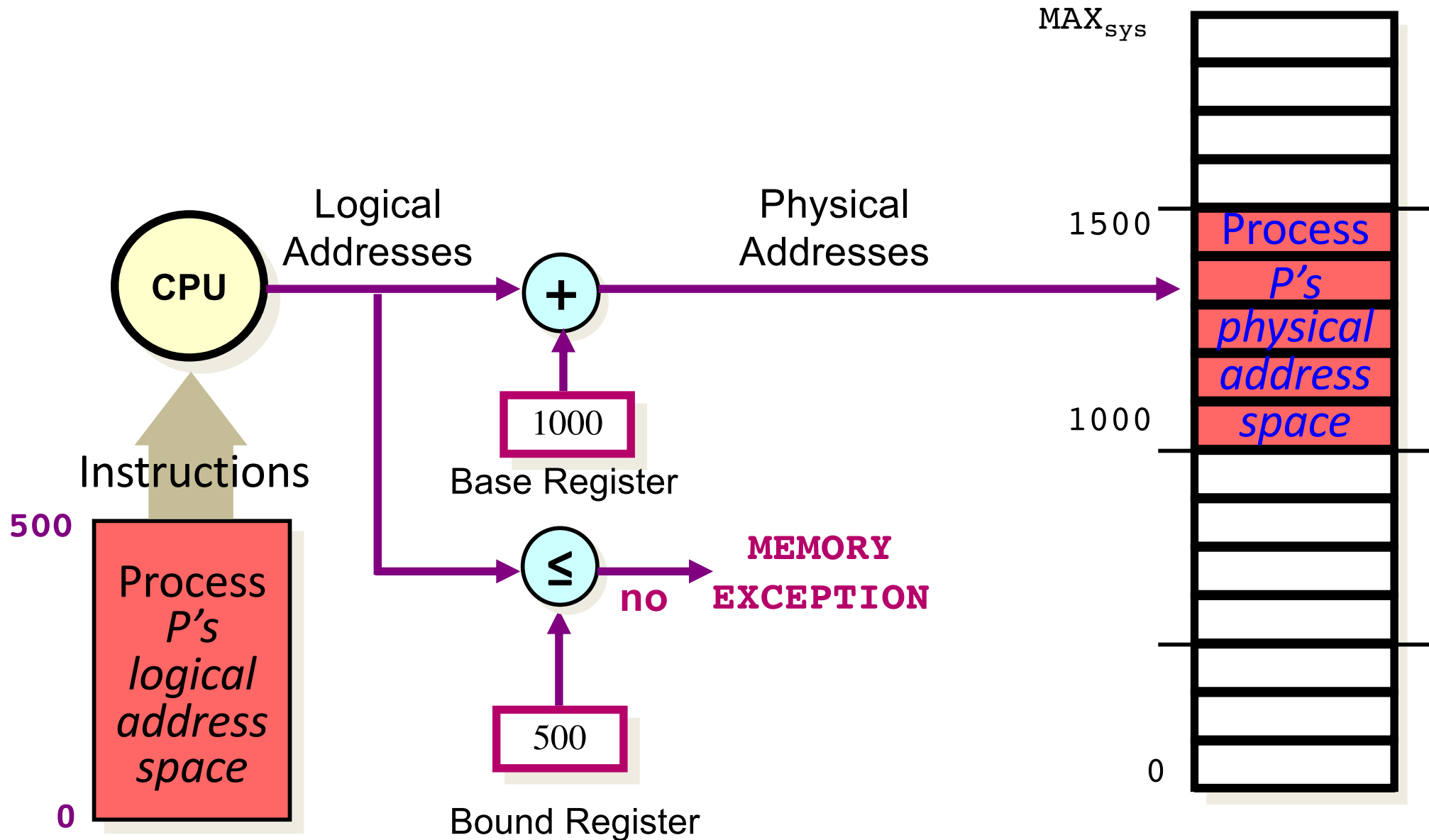
Enabling Multiprogramming: Dynamic Relocation

Dynamic Relocation

In parallel (at the same time):

- Hardware adds relocation register (base) to virtual address to get physical address
- Hardware compares address with limit register
 - address must be less than limit
 - if test fails, the processor raises an exception

Dynamic Relocation: Address Translation



iClicker Question

Address translation in dynamic relocation does the following steps:

- A. Converts address by adding base and then checks against limit
- B. Checks address against limit and then adds base
- C. Checks address against limit and adds base in parallel

Dynamic Relocation

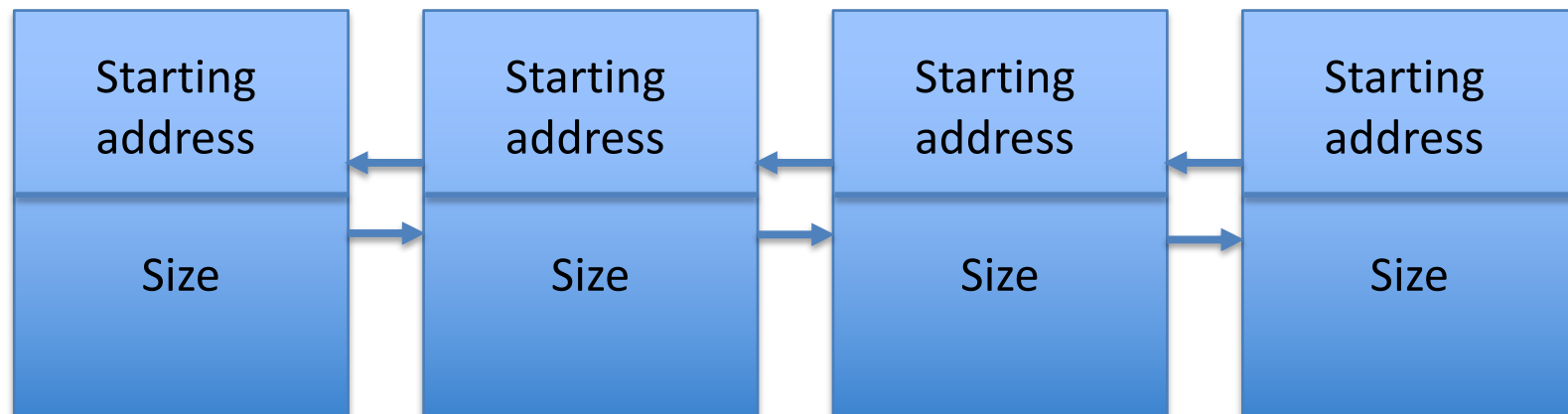
Advantages

Disadvantages

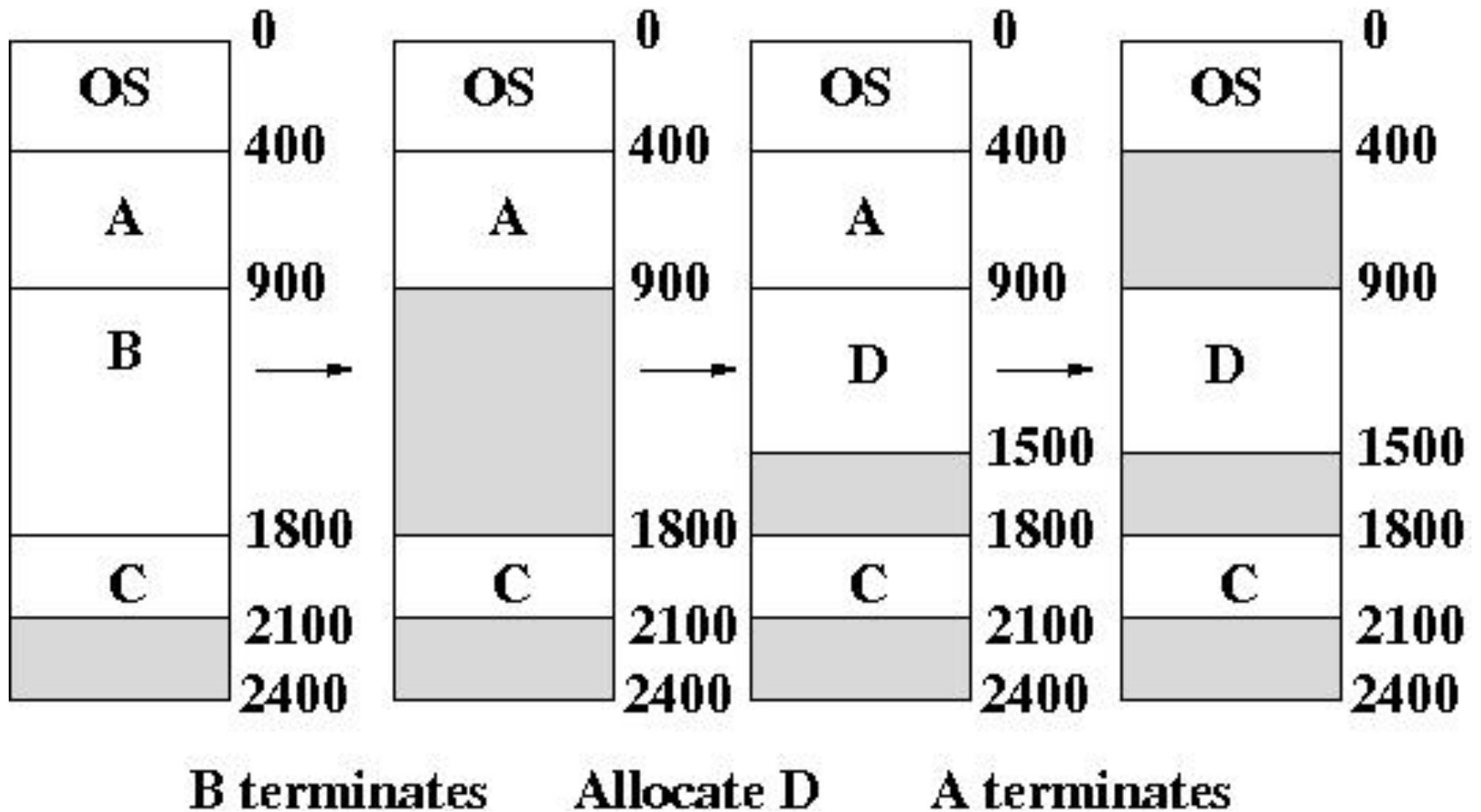
Enabling Multiprogramming: Tracking Free Space

Tracking Free Space

- The OS must track which memory is available (the holes) and which is utilized
- Can change rapidly as processes enter the system, grow, and terminate
- Tracked using a linked list---the *free list*



Memory Management Picture



Enabling Multiprogramming: Placement Policies

Allocating Memory

Given a memory request from a starting process, the OS must decide what space to use for the process

Memory Allocation Policies: Goals

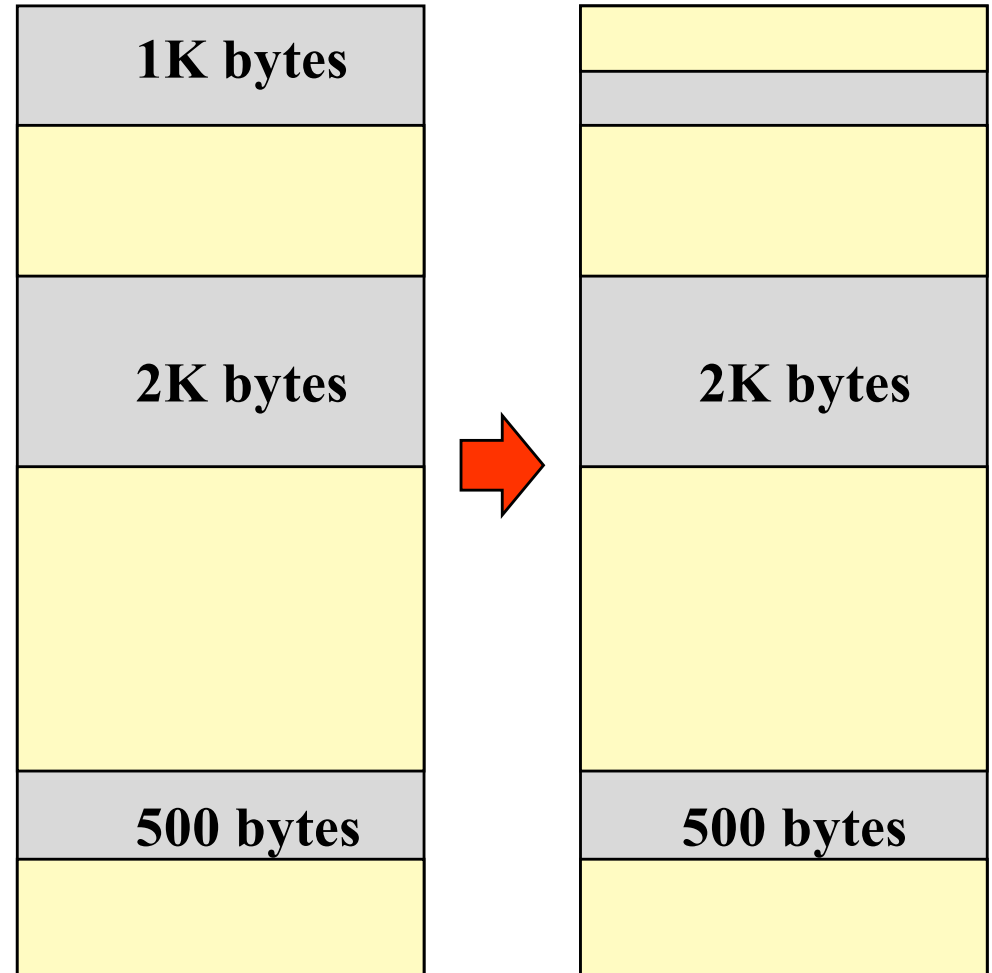
- Minimize wasted space
- Two types of wasted space:
 - **External Fragmentation**
 - Unused memory between units of allocation
 - For example, two fixed tables for two but a party of four
 - **Internal Fragmentation**
 - Unused memory within a unit of allocation
 - For example, a party of three at a table for four

Memory Allocation Policies:

First-Fit Example

To allocate n bytes, use the *first* available free block such that the block size is larger than or equal to n .

To allocate 400 bytes, we use the first free block



Memory Allocation Policies:

First-Fit

Goal: Simplicity of Implementation

Requirements:

- Free block list sorted by address
- Allocation requires a search for a suitable position
- De-allocation requires a check to see if the freed partition could be merged with any adjacent free partitions

Advantages?

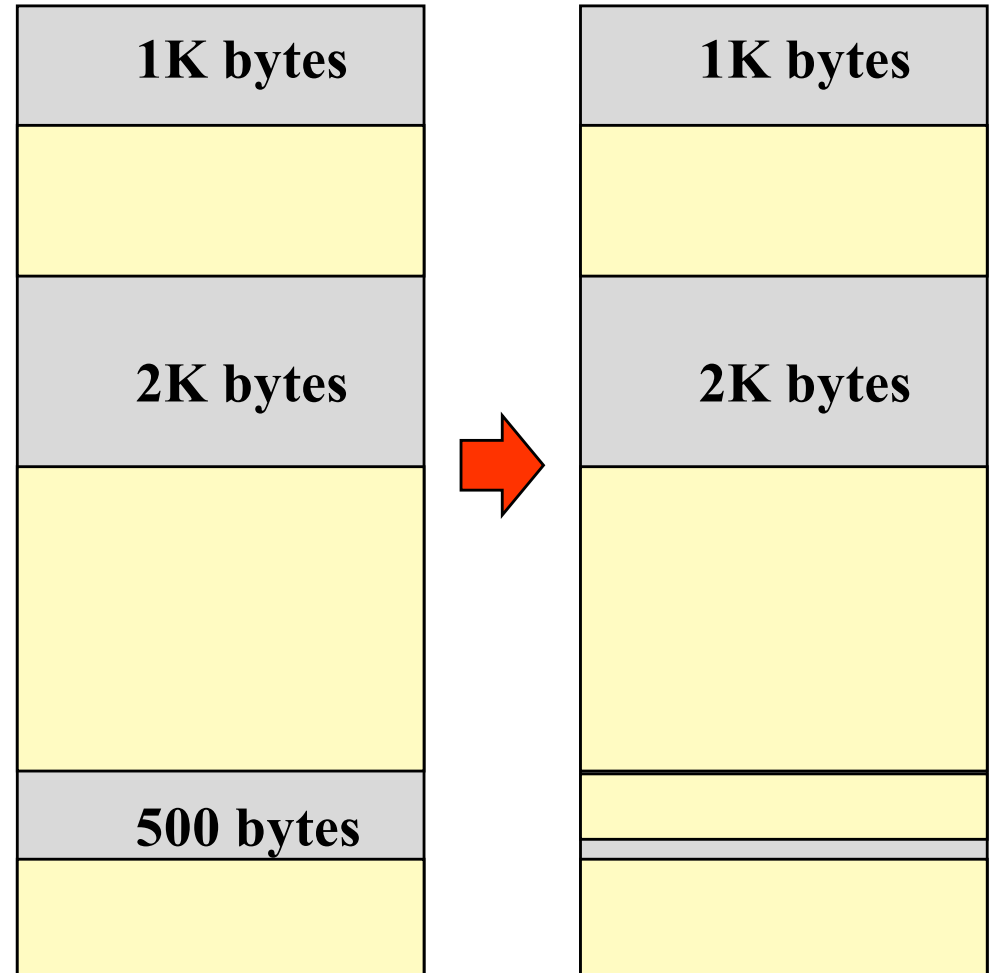
Disadvantages?

Memory Allocation Policies:

Best-Fit

To allocate n bytes, use the *smallest* available free block such that the block size is larger than or equal to n .

To allocate 400 bytes, we use the 3rd free block



Memory Allocation Policies:

Best-Fit

Goals:

- To avoid fragmenting big free blocks
- To minimize the size of resulting external fragments

Requirements:

- Free block list sorted by size
- Allocation requires a search for a suitable position
- De-allocation requires a check to see if the freed partition could be merged with any adjacent free partitions

Advantages?

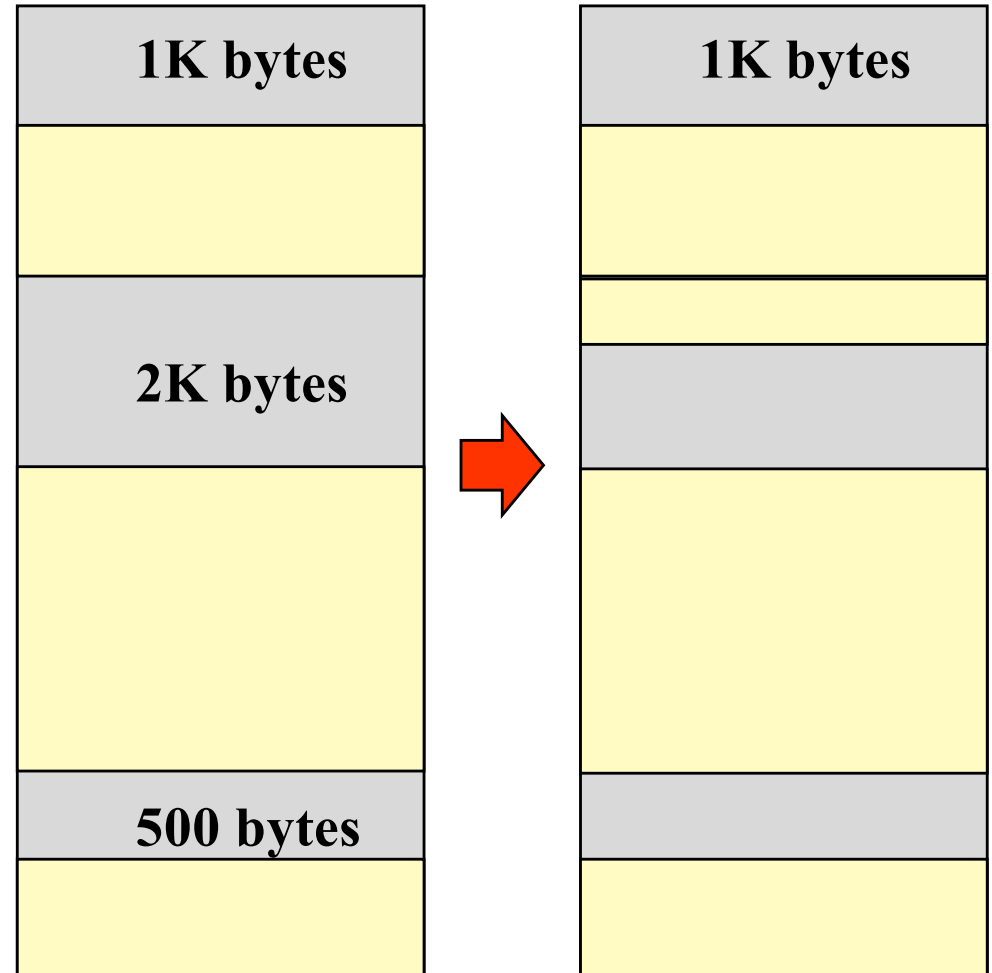
Disadvantages?

Memory Allocation Policies:

Worst-Fit

To allocate n bytes, use the *largest* available free block such that the block size is larger than or equal to n .

To allocate 400 bytes, we use the 2nd free block



Memory Allocation Policies:

Worst-Fit

Goals:

- To avoid having too many tiny fragments

Requirements:

- Free block list sorted by size
- Allocation is fast (get the largest)
- De-allocation requires a check to see if the freed partition could be merged with any adjacent free partitions

Advantages?

Disadvantages?

iClicker Question

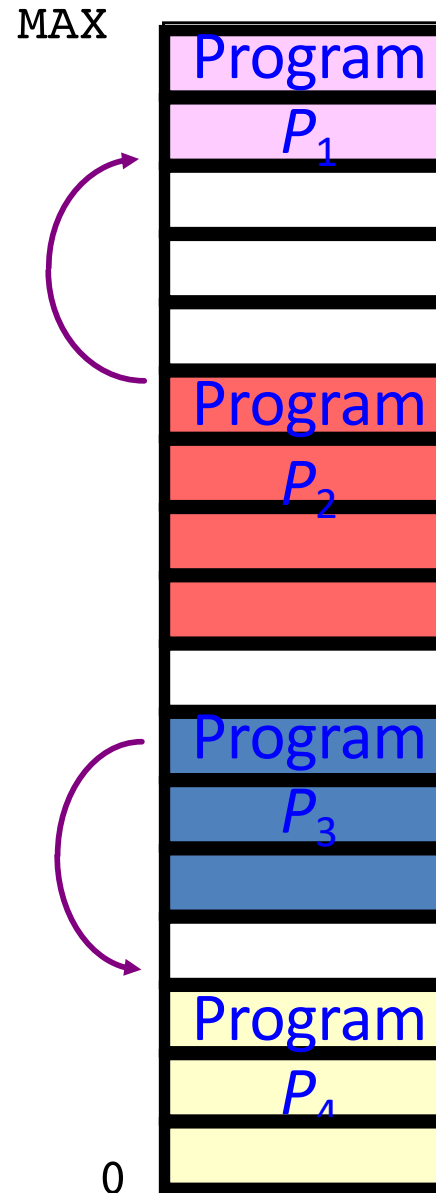
First-fit, Best-fit, and Worst-fit all suffer from external fragmentation.

- A. True
- B. False

Eliminating Fragmentation

Compaction

Relocate
processes to
coalesce
holes



Swapping

- Suspend the process
- Put process address spaces on disk
- Reassign the memory that they were using

