Tishk International University
Science Faculty
IT Department

Operating Systems

Lecture 4: Memory Management

3rd Grade - Fall Semester

Instructor: Alaa Ghazi

Lecture 4: Memory Management - Agenda

- Background
  - Main Memory
  - Cache Memory
  - Address Binding
- Memory Management Approaches
  1. Single Contiguous Model
  2. Partition with Contiguous Allocation
  3. Swapping
  4. Segmentation
  5. Paging
- Virtual Memory Basics
- Optimizing Applications Performance

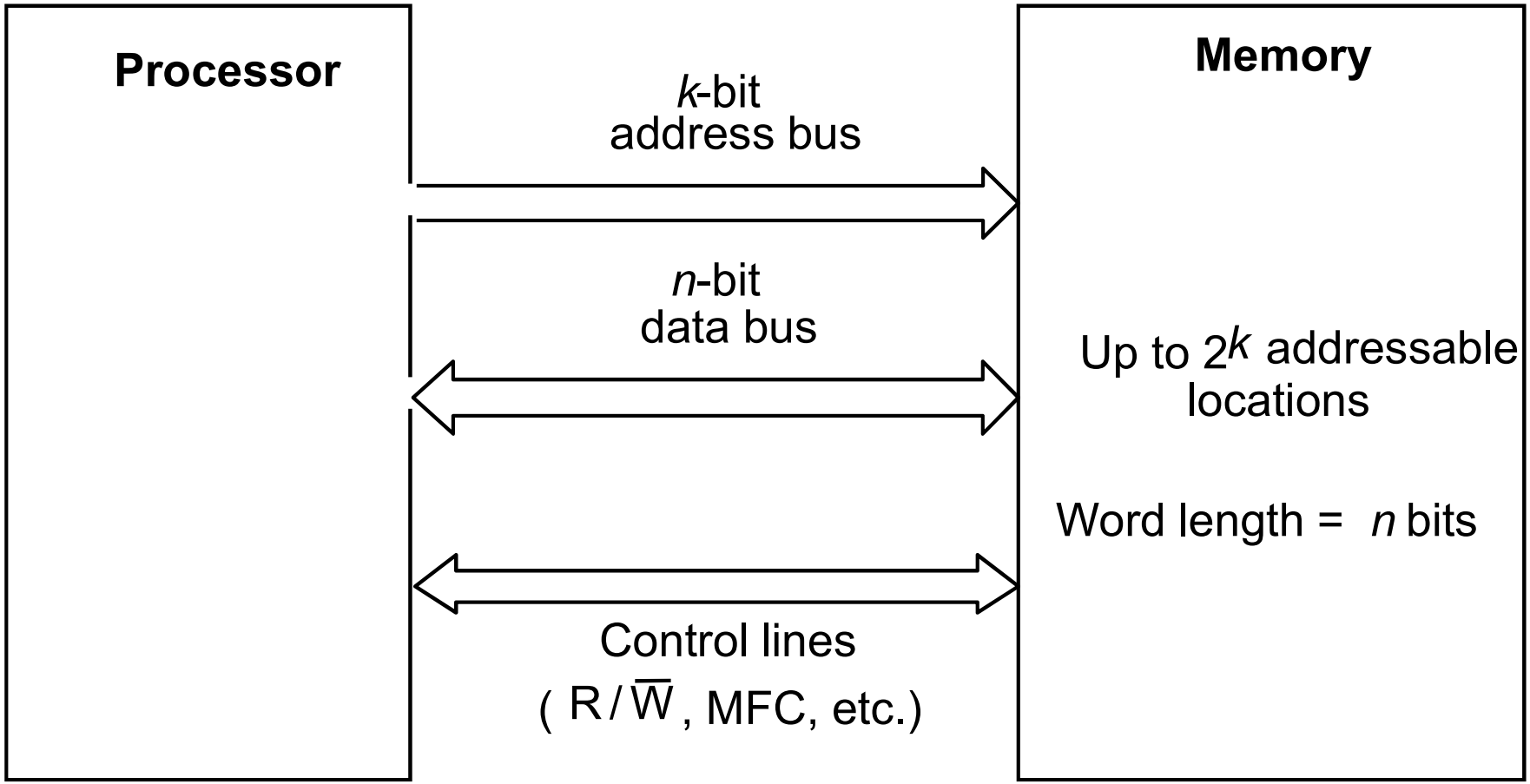
Background

- The second most important hardware resource in the computer system after the CPU is the Memory.
- Memory accesses and memory management are a very important part of modern computer operation. Every instruction has to be fetched from memory before it can be executed, and most instructions involve retrieving data from memory or storing data in memory or both.
- The introduction of multi-tasking OSes increases the need for complex memory management, ***because*** as processes are swapped in and out of the CPU, so must their code and data be swapped in and out of memory, all at high speeds and without interfering with any other processes.

What is Main Memory

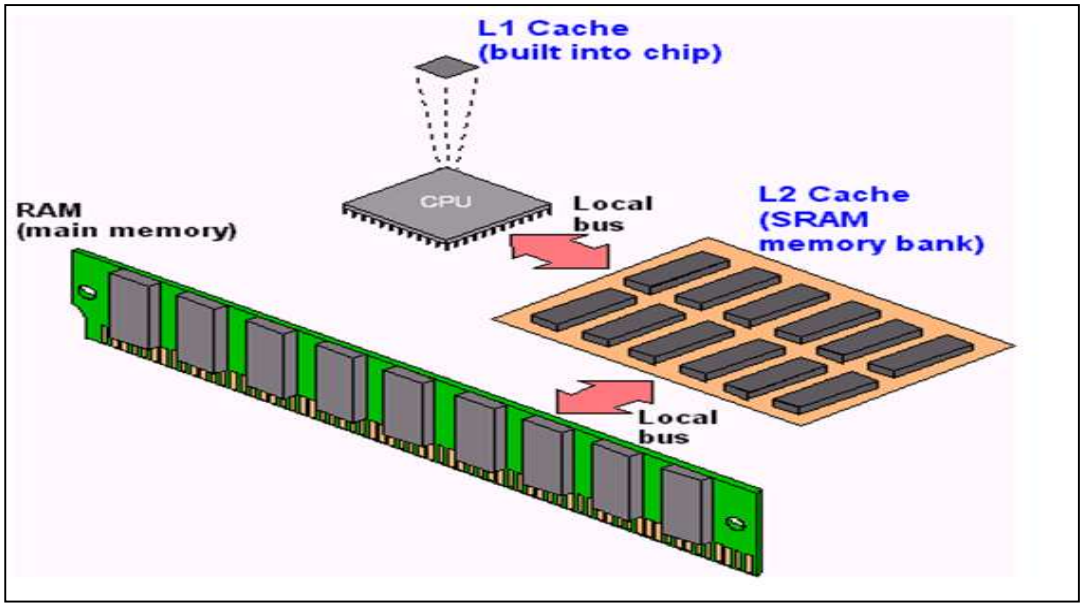
- **Main Memory or (Random Access Memory - RAM):** is the area in a computer in which data is stored for quick access by the computer's processor.
- RAM speed is measured in nanoseconds (billionths of a second), while magnetic and SSD storage is measured in milliseconds (thousandths of a second).
- Program must be brought (from disk) into memory and placed within a process for it to be run
- Main memory and registers are only storage CPU can access directly

CPU-Main Memory Connection (not required in the exam)



Cache Memory

- **Cache Memory**: is a small-sized type of volatile computer memory that provides high-speed data access to a processor and stores frequently used computer code and data.
- It is the fastest memory in a computer, and is typically integrated onto the motherboard and/or directly embedded in the processor.

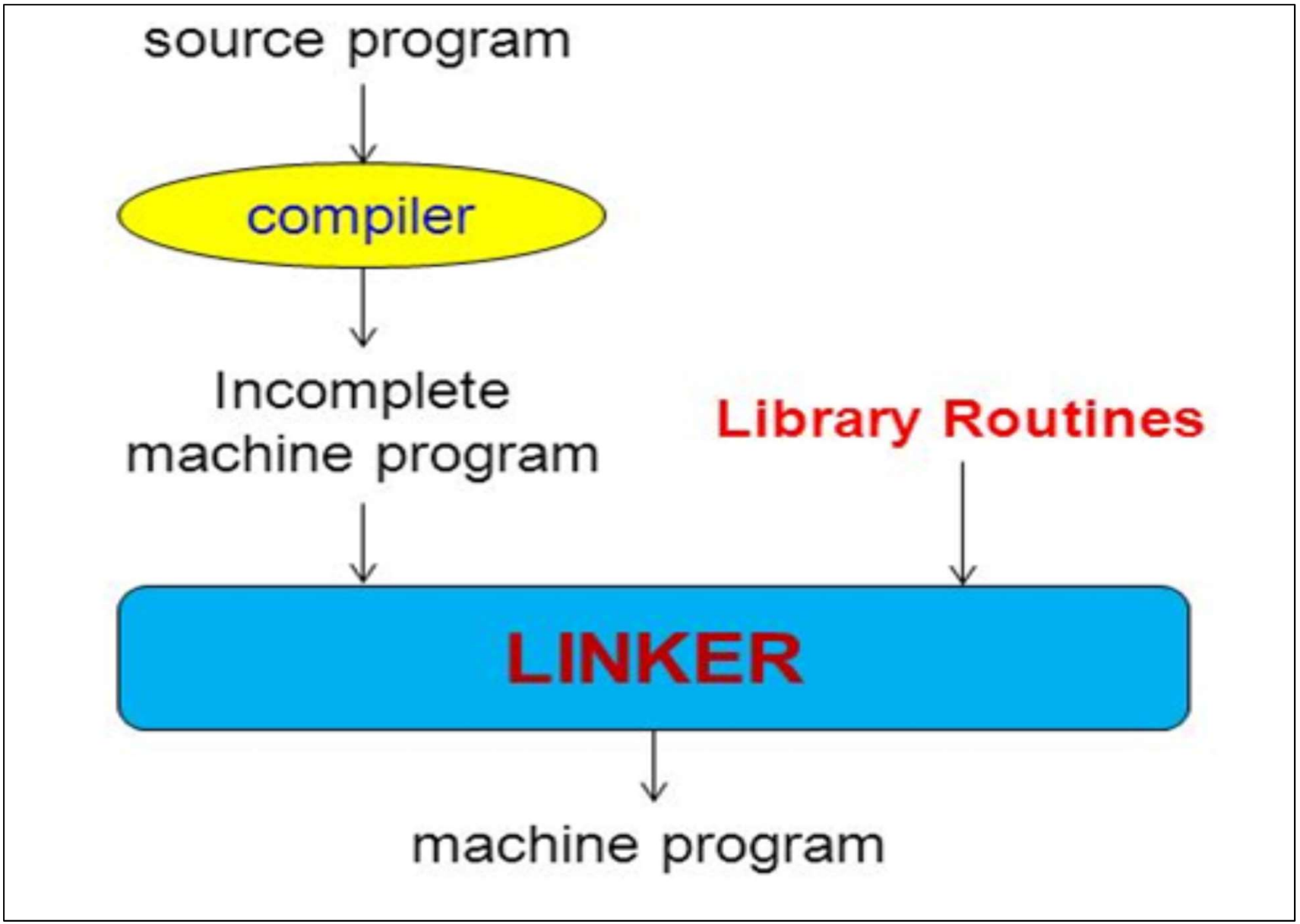


Libraries Linking

- **Static linking** – It is the case when system libraries and program code are combined by the loader into the binary program image.
- **Dynamic linking** – It is the case when linking of the routines to the main program is postponed until execution time.
- **Stub** is a small piece of code, used to locate the appropriate memory-resident library routine and replaces itself with the address of the routine, and executes the routine
- **Advantages of Dynamic Linking** and Shared Libraries:

1. Less program loading time
2. Less memory space
3. Less disk space to store binaries

Static Linking Diagram



Address Binding Schemes

Address binding of instructions and data to memory addresses can happen at three different schemes:

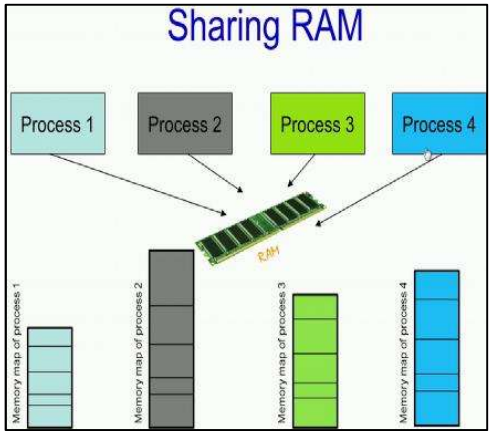
- **Compile time binding**: when the ***absolute code*** will be generated by the compiler, containing actual physical addresses (like in MSDOS .com programs).
- **Load time binding**: the compiler must generate ***relocatable code***, which references addresses relative to the start of the program.
- **Execution time binding**: is when binding must be delayed until execution time, so the program can be moved around in memory during execution. This is the method which is implemented by most **modern OSes.**

Memory Management Approaches

Memory Management is allocating, freeing, and re-organizing memory in a computer system to optimize the available memory or to make more memory available. It keeps track of every memory location(if it is free or occupied).

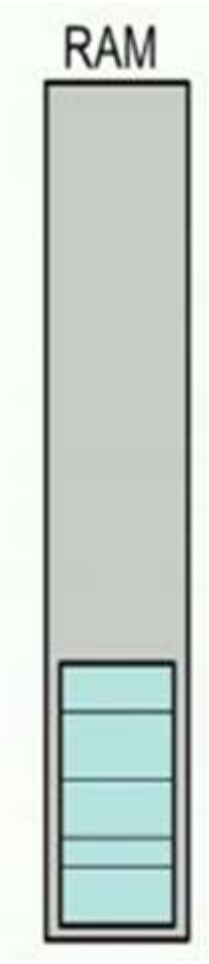
Historically, the below memory management approaches have been used

1. **Single Contiguous Model**
2. **Partition with Contiguous Allocation**
3. **Swapping**
4. **Segmentation**
5. **Paging**



1. Single Contiguous Model

- Is one of the most primitive ways of managing memory especially done for the older on a operating systems.
- So RAM is occupied by one process at a time. Essentially at any particular instant there is only one process and its memory size is restricted by the RAM size.
- After this process completes executing, the next process will be loaded into the RAM (no sharing).
- Process memory size is restricted by RAM size.



2. Partition with Contiguous Allocation

- In this approach at any instant of time, we could have multiple processes that occupy the RAM simultaneously but each process should be contained in a single contiguous partition of memory.
- As long sufficient contiguous space is available, new processes are allocated memory. When a process completes execution, the area in RAM that it holds will be de-allocated. Consequently the entry corresponding to the partition table will be free.
- **This approach is a slight improvement over the single contiguous model.**
- Main memory is usually partitioned into two **parts**:
  - Resident operating system area.
  - User processes area.
- The Partition Table would have the base address of a process, the size of the process and a process identifier.

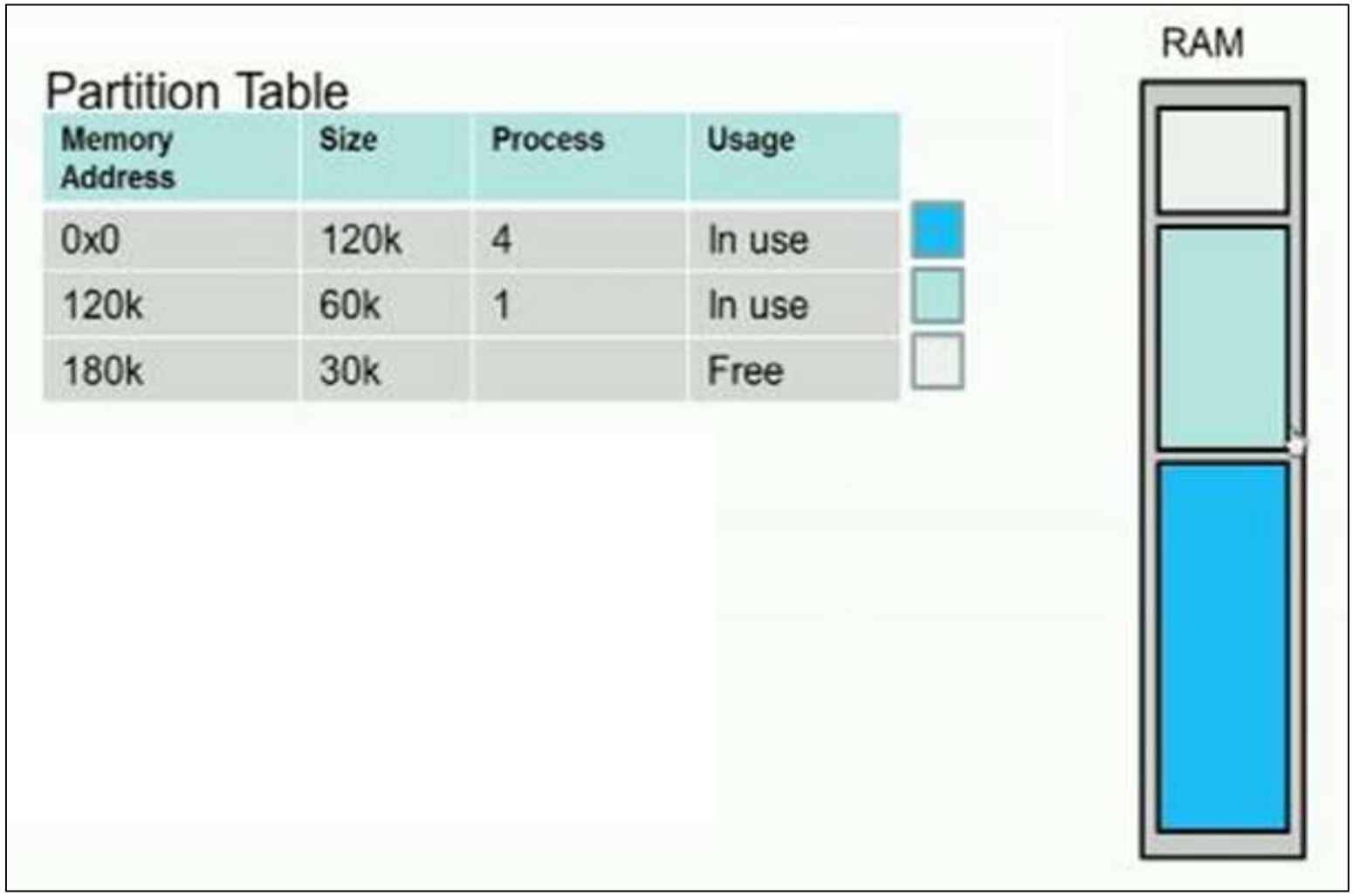


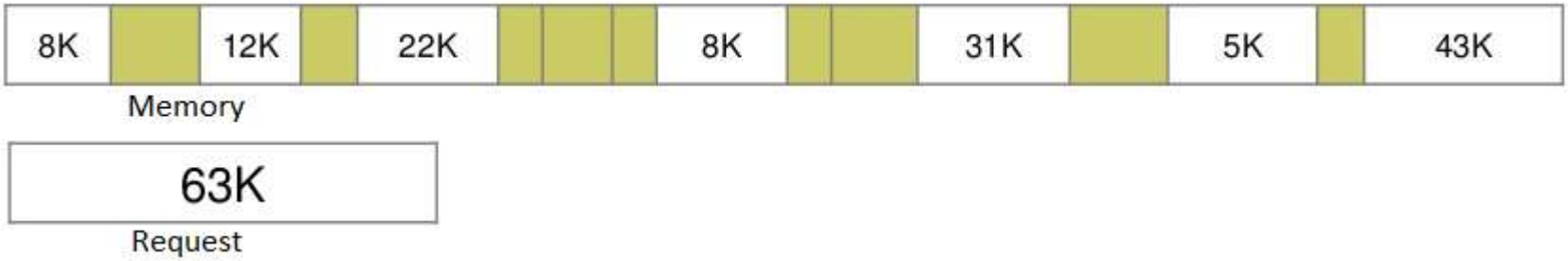
Diagram of Partition with Contiguous Allocation

Limitations of Partition with Contiguous Allocation

- Each Process needs to be entirely in RAM.
- Allocation needs to be in contiguous memory
- External Fragmentation
- Limit the size of the process by RAM size
- Performance Degradation

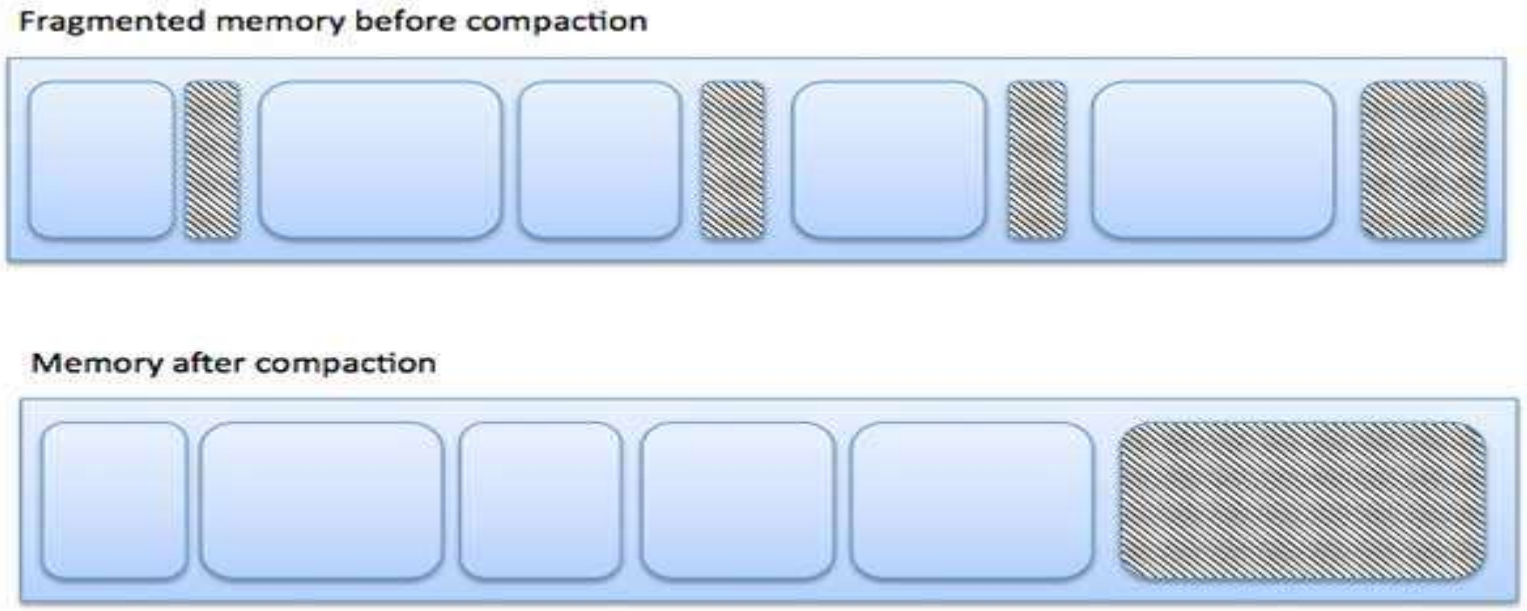
External Fragmentation

- **External Fragmentation** – It is the case when total memory space exists to satisfy a request, but it is not contiguous. This is a problem with Partition with Contiguous Allocation memory management approach.



Compaction

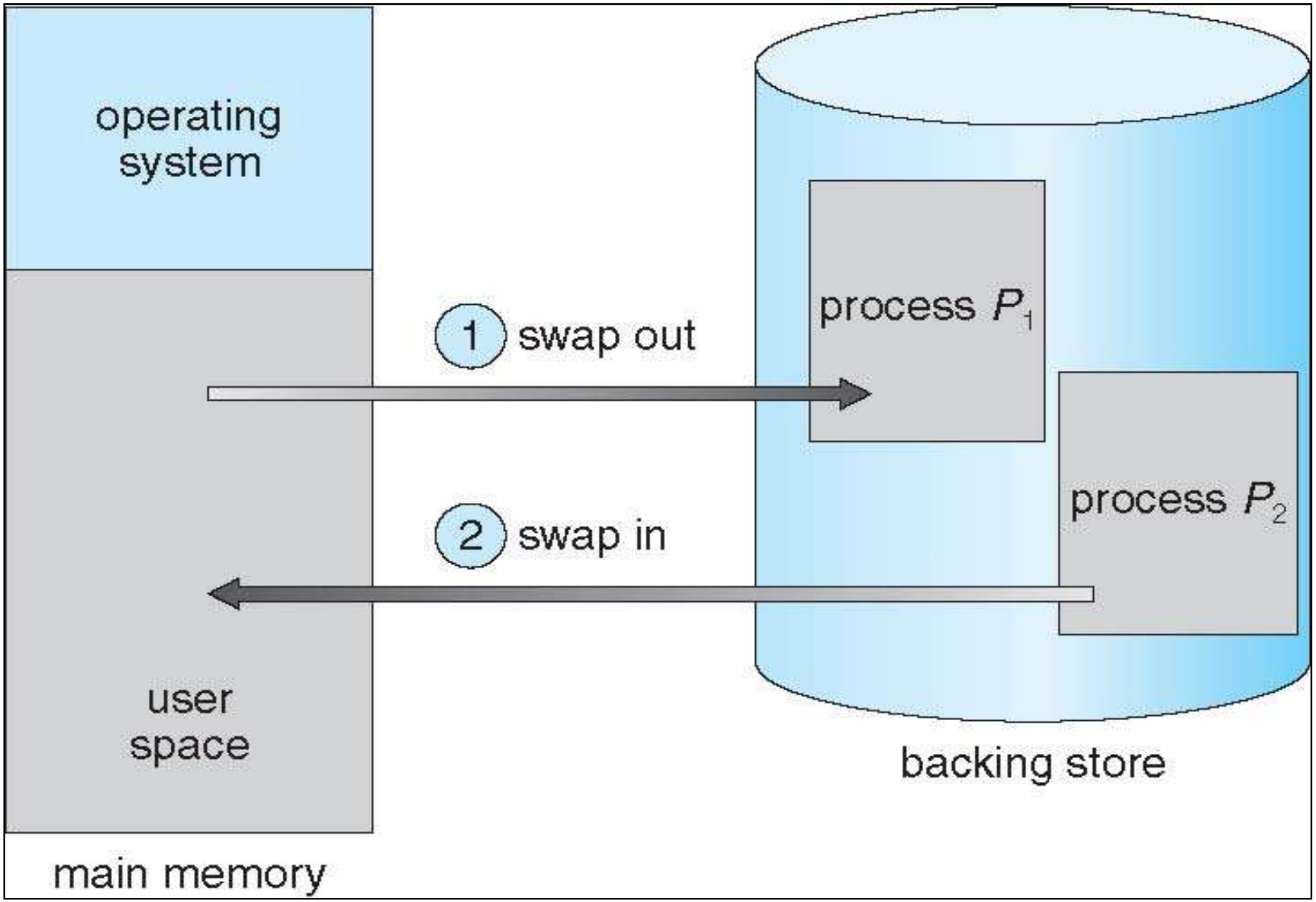
- **Compaction:** is shuffling memory contents to place all free memory together in one large block
- Compaction is a solution to Reduce external fragmentation.
  - Compaction is possible *only* if relocation is dynamic, and is done at execution time



3. Swapping

- **Swapping** is a technique in which a process can be moved temporarily out of memory to Backing Store (HD or SSD), and then brought back into memory for continued execution.
- **Backing store** – is a fast disk large enough to accommodate copies of all memory images for all processes; and must provide direct access to these memory images.
- The swapping procedures that are found on modern operating systems (i.e., UNIX, Linux, and Windows) are:
  - Swapping is normally disabled
  - Swapping will be started if memory demand is more than certain threshold amount of memory.
  - Swapping will be disabled again once memory demand is reduced below certain threshold.

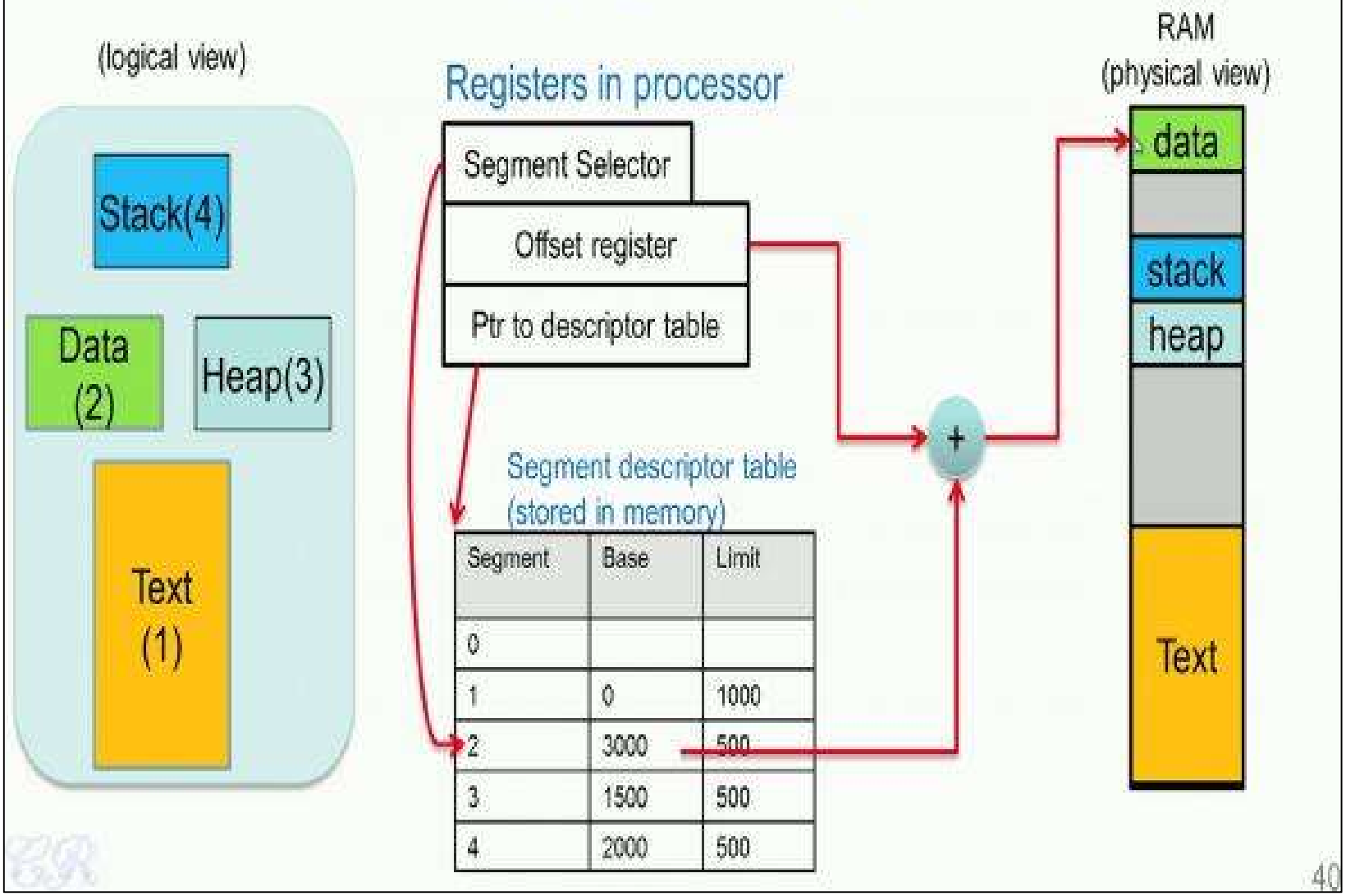
Schematic View of Swapping



4. Segmentation

- **Segmentation**: is a memory-management approach that is implemented by segmenting processes and loading them into different non-contiguous addressed spaces in memory with different segment address.
- In Segmentation we have a **Segment Descriptor Table** which is stored in memory. Each row in the segment descriptor table refers to one particular segment. For instance, the Data segment 2 is at an offset 2 in the Segment descriptor table, and the offset would specify the Base address in RAM and the Limit of the segment.

Address Mapping with Segmentation



5. Paging

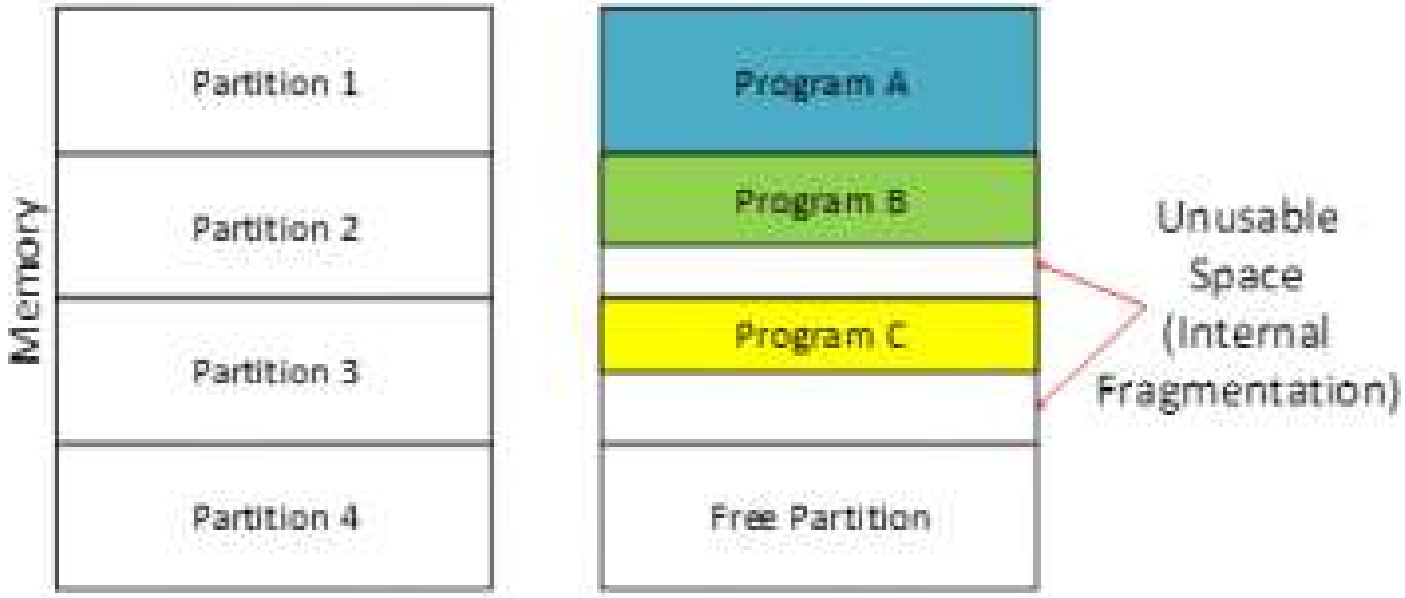
- **Paging** is to divide physical memory into fixed-sized blocks called **frames** and divide logical memory into blocks of same size called **pages**
- Physical address space of a process can be noncontiguous.
- Paging eliminates external fragmentation , but it still suffers from Internal fragmentation.
- Page Size is a power of 2, usually between 512 bytes and 16 Mbytes
- To load a process of size S where

(N-1) pages < S < (N) pages,

it is required to find ***N*** free frames to load the process.

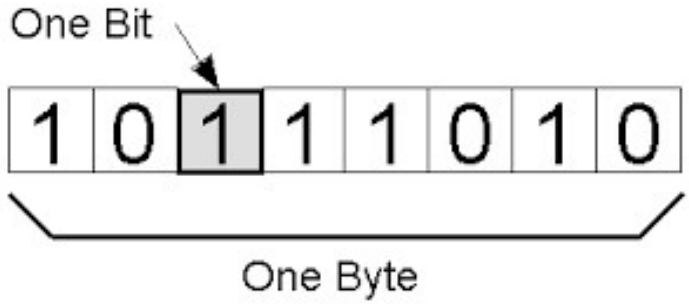
Internal Fragmentation

- **Internal Fragmentation** – is this size difference in memory that happens when we divide memory to fixed partitions and then allocated memory may be slightly larger than requested memory.



What is Kilobyte?

Common prefix			
Name	Symbol	Decimal SI	Binary JEDEC
kilobyte	KB/kB	10^3	2^{10}
megabyte	MB	10^6	2^{20}
gigabyte	GB	10^9	2^{30}
terabyte	TB	10^{12}	2^{40}
petabyte	PB	10^{15}	2^{50}
exabyte	EB	10^{18}	2^{60}
zettabyte	ZB	10^{21}	2^{70}
yottabyte	YB	10^{24}	2^{80}



1 kilobyte = 1024 bytes

Page Table

The **page table** is the table used to look up what frame a particular page is stored in at the moment. It translates logical to physical addresses.

Associative Memory is a special fast-lookup hardware cache that can solve the two memory access problem if page table is stored in main memory.

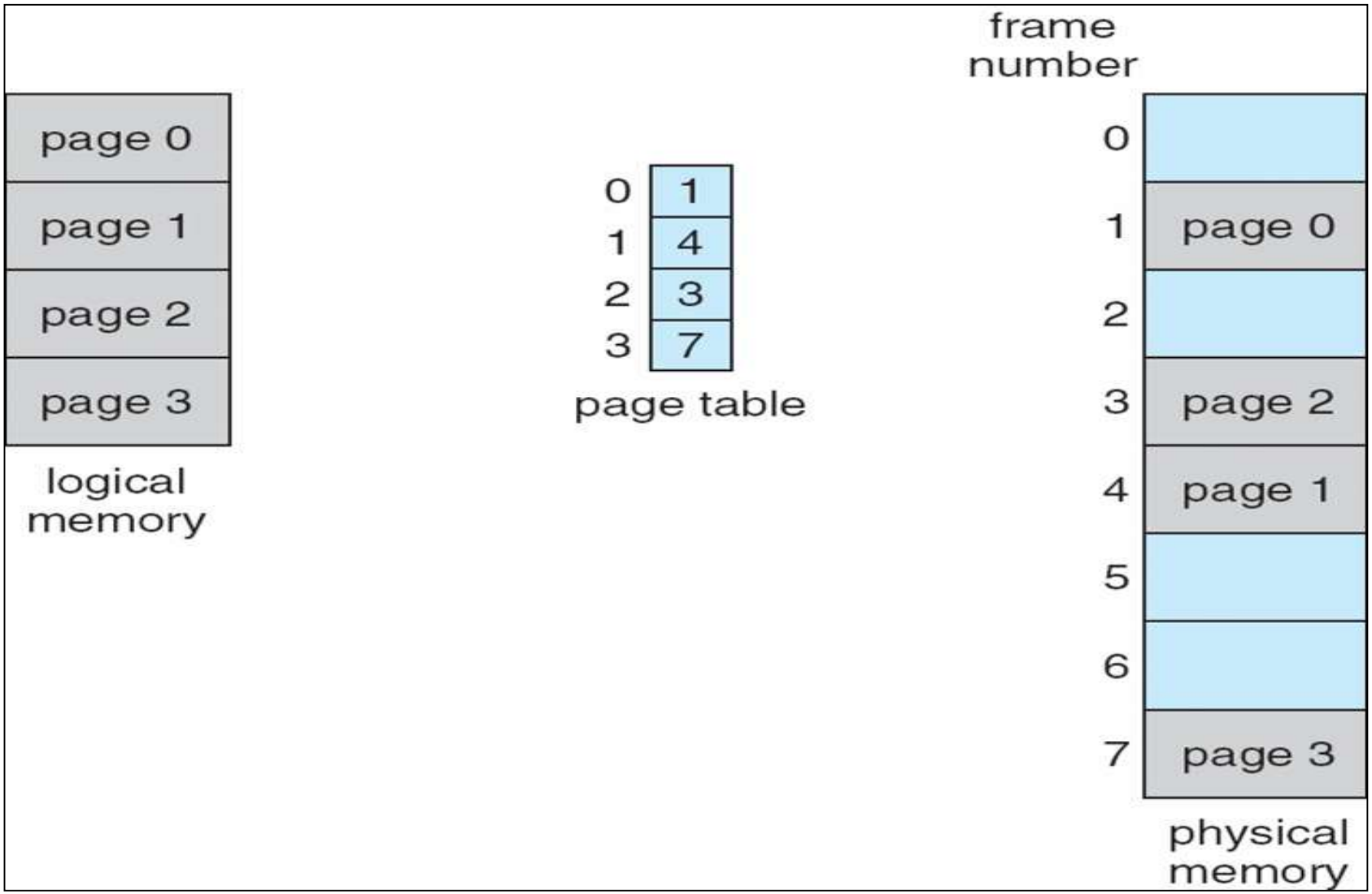
Q\ Why Page size selection is critical?

- A large page will result in increase of internal fragmentation
- A small page size will increase the size of the page table

The **Page Table Structure** can be of two types:

- **Basic Paging**: A single page table which stores the page number and the offset
- **Hierarchical Paging**: A multi-level table which breaks up the virtual address into multiple page tables.

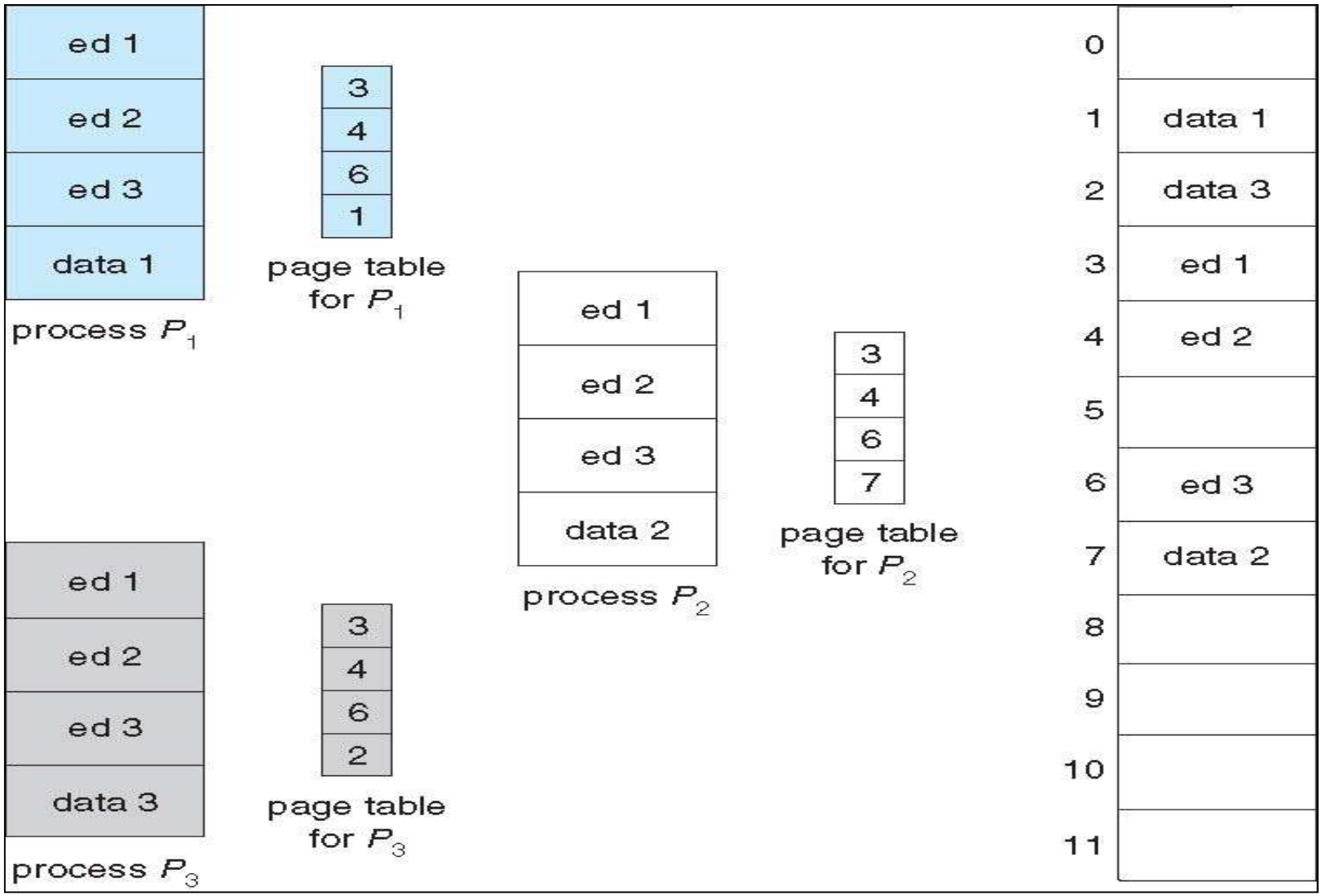
Basic Paging Diagram



Shared Pages

- **Shared code**
  - One copy of read-only code can be shared among processes (i.e., text editors, compilers, window systems)
  - It is also useful for inter-process communication if sharing of read-write pages is allowed
- **Private code and data**
  - Each process keeps a separate copy of the code and data
  - The pages for the private code and data can appear anywhere in the logical address space

Shared Pages Diagram



Virtual Memory Basics

- **Virtual Memory** increases the available memory of the computer by enlarging the "address space," or places in memory where data can be stored. It does this by using hard disk space for additional memory allocation.
- However, since the hard drive is much slower than the RAM, data stored in virtual memory must be mapped back to real memory in order to be used.
- Q\ Why most real processes do not need all their pages?
  - Error handling code is not needed unless that error occur.
  - Only a small fraction of the arrays are actually used.
  - Certain routines of programs are rarely used.

Logical vs. Physical Address Space

- The concept of a logical address space that is bound to a separate **physical address space** is central to virtual memory
  - **Logical address (virtual address)** – it is the address generated by the process executing currently on the CPU.
  - **Physical address** – it is the address seen by the memory management unit where actual code and data are loaded.
- The user process deals with logical addresses; it never sees the real physical addresses
- **Memory-Management Unit**: is the Hardware device that at run time maps virtual to physical address.

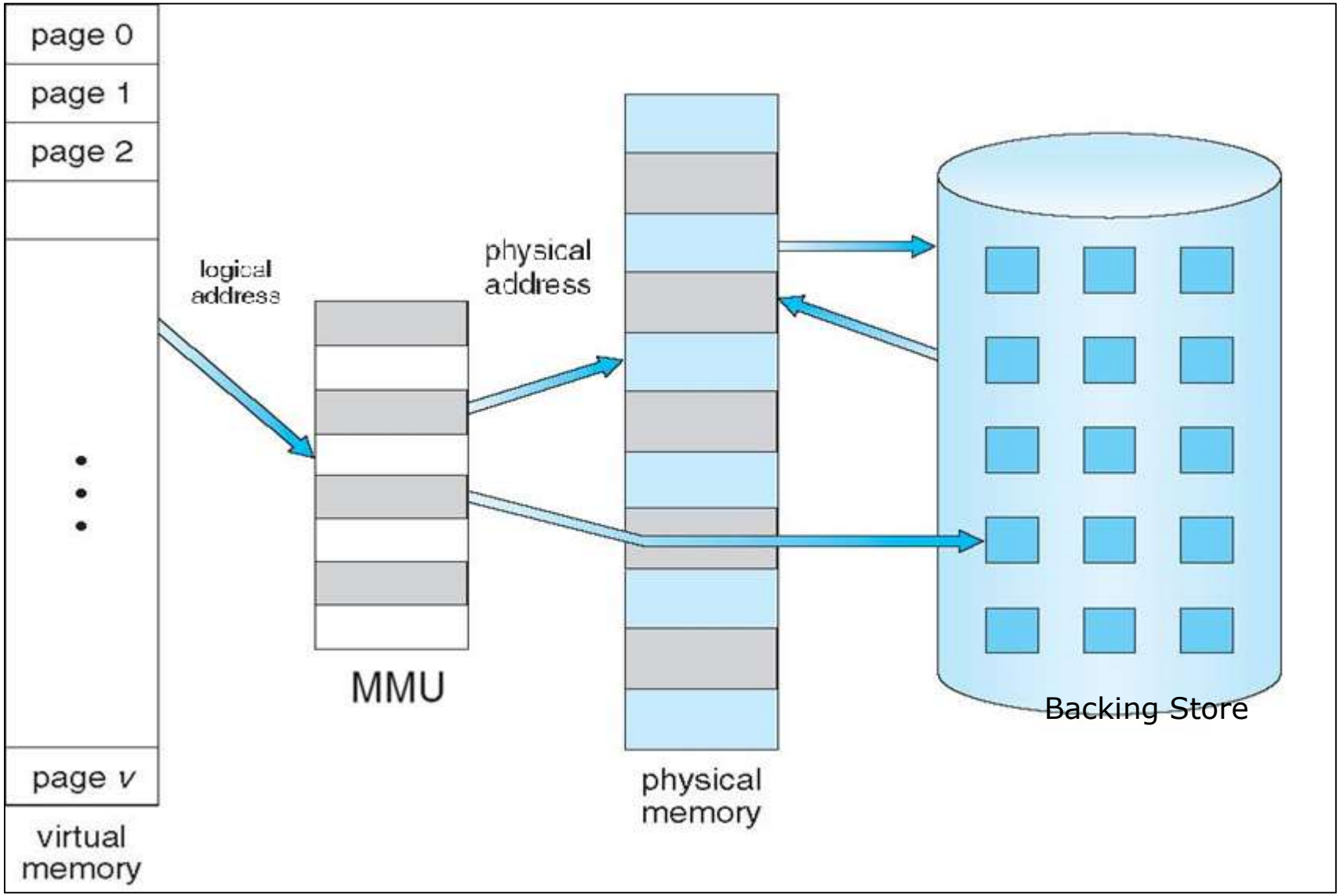
Benefits of Virtual Memory

- In virtual memory, Logical address space can therefore be much larger than physical address space
- Every process executing in the system would have its own process page table.
- **Benefits of Virtual Memory** are:
  - Only part of the program needs to be in memory for execution
  - Allows address spaces to be shared by several processes
  - More programs can run concurrently

Page Fault: is a type of trap raised when a running process accesses a memory **page** that is not currently in the physical RAM

Page replacement is finding some page in memory, which is not really in use, in order to page it out.

The General Layout of Virtual Memory



Thrashing and Memory Leaks

- **Thrashing**: it is the case when a process does not have "enough" pages, then page-fault rate will become very high and the process will be busy swapping pages in and out
- **Memory Leak:** occurs when an application is using more RAM than it normally does which in turn slows down the system, causing it to struggle with performing even the basic tasks.

Optimizing Applications Performance

- Optimizing Applications depends heavily on memory organization in the computer system. The next few points offer some pointers for **improving the performance of applications under Windows :**

1. Adding More Physical Memory
2. Defragment the Hard Drive containing the paging file
3. Installing Applications to the Fastest Hard Drive
4. Getting the Latest Device Drivers
5. Move Extra workload to Another Server

1. Adding More Physical Memory

- All applications run in RAM, of course, so the more RAM you have, the less likely it is that Windows will have to store excess program or document data in the page file on the hard disk, which is a real performance killer.

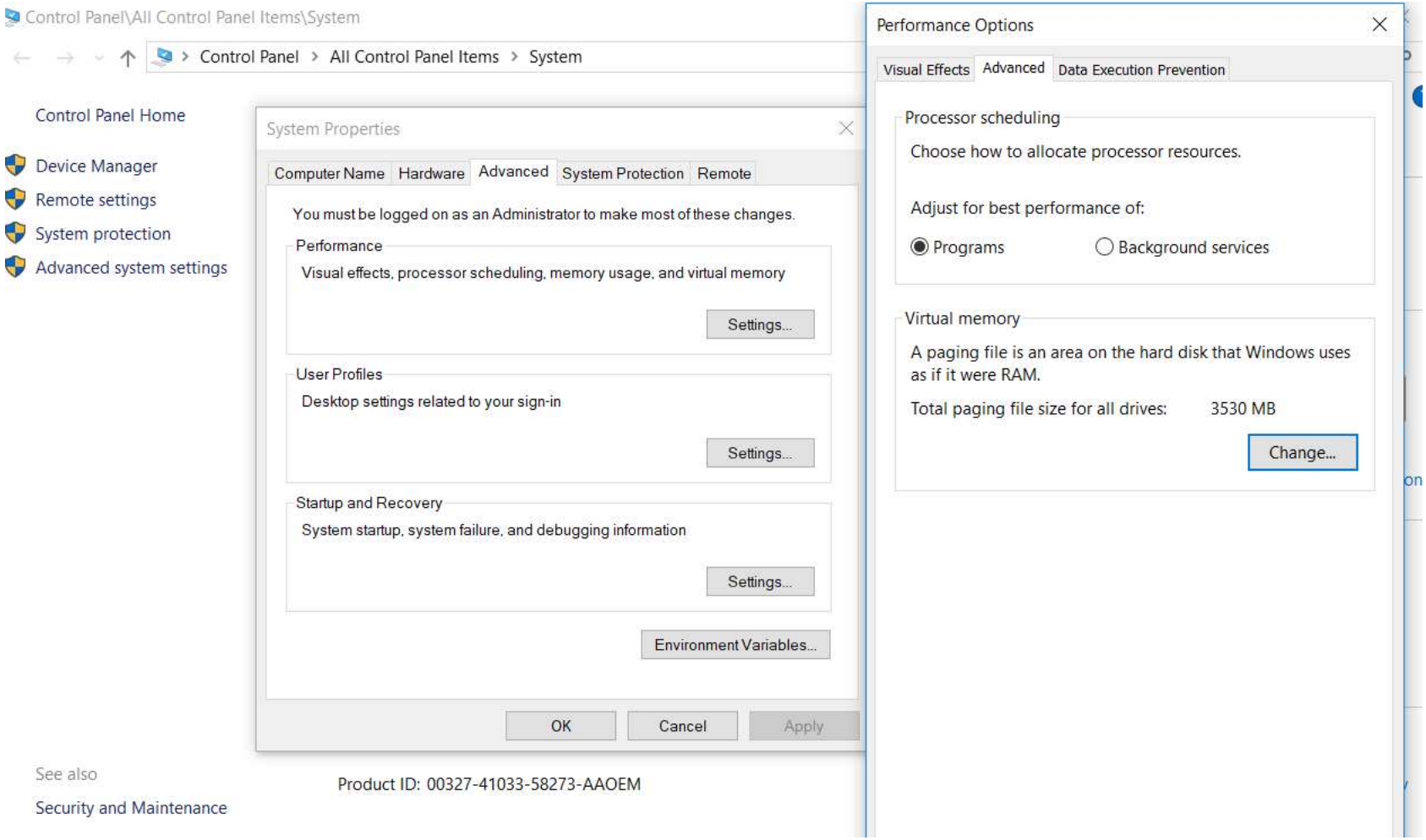
Use one of the following **Windows monitoring tools** to watch the available memory:

- **Task Manager** —Display the Performance tab and watch the Physical Memory: Available value. .
- **Resource Monitor** —Display the Memory tab and watch the Available to Programs value. .
- **Performance Monitor** —Start a new counter, open the Memory category, and then select the Available Mbytes counter.

2. Defragment the Hard Drive containing the Paging File

- This is needed when the page file is located on a disk that is heavily used by other applications.
- **Hard Disk Defragmentation** is to organize the files parts in contiguous sectors on the disk, thereby improving computer performance and maximizing disk space.
- The **solution steps** are:
  1. move the page file to another drive temporarily,
  2. set the paging file on the original drive to be fragmented to 0 MB.
  3. reboot the system to enable the other paging file to be used.
  4. perform the disk defragmentation on the original drive.
  5. set the paging file on the original drive to the necessary values, and reboot.

Paging file in Windows 10 (not required in the exam)



3. Installing Applications to the Fastest Hard Drive

- If your system has multiple hard drives that have different performance ratings, install your applications on the fastest drive. This enables Windows to access the application's data and documents faster.

4. Getting the Latest Disk Drivers

- If your application works with a device, check with the manufacturer or Windows Update to see whether a newer version of the device driver is available. In general, the newer the driver, the faster its performance.

5. Move Extra workload to Another Server

- For Server Machine, you may also elect to offload some of the workload to another system.