LIMKOKWING UNIVERSITY OF CREATIVE TECHNOLOGY
ESWATINI

WEEK 5 : LECTURE 4 - MEMORY MANAGEMENT

Bachelor’s Degree in Information Technology
BITOS4111, OPERATING SYSTEMS
Trimester: 05
From **July 2024** to **October 2024**

OBJECTIVES/TOPICS

- Logical Versus Physical Memory Management.
- Address Space.
- Contiguous Memory Allocation
- Paging
- Virtual Memory.
- Demand Paging.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

INTRODUCTION

- The main purpose of a computer system is to execute programs.

- These programs, together with the data they access, must be at least partially in main memory during execution.

- Modern computer systems maintain several processes in memory during system execution.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

INTRODUCTION

- Many memory-management schemes exist, reflecting various approaches, and the effectiveness of each algorithm varies with the situation.

- Selection of a memory-management scheme for a system depends on many factors, especially on the system's hardware design.

- Most algorithms require some form of hardware support.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

INTRODUCTION

- Memory is central to the operation of a modern computer system.

- Memory consists of a large array of bytes, each with it 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.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

INTRODUCTION

- A typical instruction-execution cycle, for example, first fetches an instruction from memory.
  - The instruction is then decoded and may cause operands to be fetched from memory.
  - After the instruction has been executed on the operands, results may be stored back in memory

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

INTRODUCTION

- The memory unit sees only a stream of memory addresses;
    - it does not know how they are generated (by the instruction counter, indexing, indirection, literal addresses, and so on)
    - or what they are for (instructions or data).

- We can ignore how a program generates a memory address.
    - We are interested only in the sequence of memory addresses generated by the running program.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

INTRODUCTION: Basic Hardware

- Main memory and the registers built into each processing core are the only general-purpose storage that the CPU can access directly.

- There are machine instructions that take memory addresses as arguments, but none that take disk addresses

- Therefore, any instructions in execution, and any data being used by the instructions, must be in one of these direct-access storage devices

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

INTRODUCTION: Basic Hardware

- If the data are not in memory, they must be moved there before the CPU can operate on them.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

INTRODUCTION: Address Binding

- Usually, a program resides on a disk as a binary executable file.

- To run, the program must be brought into memory and placed within the context of a process, where it becomes eligible for execution on an available CPU.

- As the process executes, it accesses instructions and data from memory.
- Eventually, the process terminates, and its memory is reclaimed for use by other processes.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

INTRODUCTION: Address Binding

- Most systems allow a user process to reside in any part of the physical memory.

- Thus, although the address space of the computer may start at 00000, the first address of the user process need not be 00000

- In most cases, a user program goes through several steps—some of which may be optional—before being executed

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

INTRODUCTION: Address Binding

- Most systems allow a user process to reside in any part of the physical memory.

- Thus, although the address space of the computer may start at 00000, the first address of the user process need not be 00000

- In most cases, a user program goes through several steps—some of which may be optional—before being executed

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

INTRODUCTION: Address Binding

- A compiler typically binds these symbolic addresses to relocatable addresses (such as "14 bytes from the beginning of this module").

- The linker or loader (see Section 2.5) in turn binds the relocatable addresses to absolute addresses (such as 74014).

- Each binding is a mapping from one address space to another.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

INTRODUCTION: Address Binding

Classically, the binding of instructions and data to memory addresses can be done at any step along the way:

- **1. Compile time.**
  - If u know at compile time where the process will reside in memory, then absolute code can be generated.
    - E.g., if you know that a user process will reside starting at location R, then the generated compiler code will start at that location and extend up from there.
    - If, at some later time, the starting location changes, then it will be necessary to recompile this code.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

INTRODUCTION: Address Binding

- **2. Load time**.
  - If it is not known at compile time where the process will reside in memory, then the compiler must generate relocatable code.
  - In this case, final binding is delayed until load time.
  - If the starting address changes, we need only reload the user code to incorporate this changed value.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

INTRODUCTION: Address Binding

- **3. Execution time.**
  - If the process can be moved during its execution from one memory segment to another, then binding must be delayed until run time.
  - Special hardware must be available for this scheme to work.
  - Most operating systems use this method.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

LOGICAL VERSUS PHYSICAL MEMORY MANAGEMENT.

- An address generated by the CPU is commonly referred to as a logical address

- Whereas an address seen by the memory unit—that is, the one loaded into the memory-address register of the memory—is commonly referred to as a physical address.

- Binding addresses at either compile or load time generates identical logical and physical addresses.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

LOGICAL VERSUS PHYSICAL MEMORY MANAGEMENT.

- However, the execution-time address-binding scheme results in differing logical and physical addresses.
    - In this case, we usually refer to the logical address as a virtual address.

- We use logical address & virtual address interchangeably in this module

LOGICAL VERSUS PHYSICAL MEMORY MANAGEMENT.

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

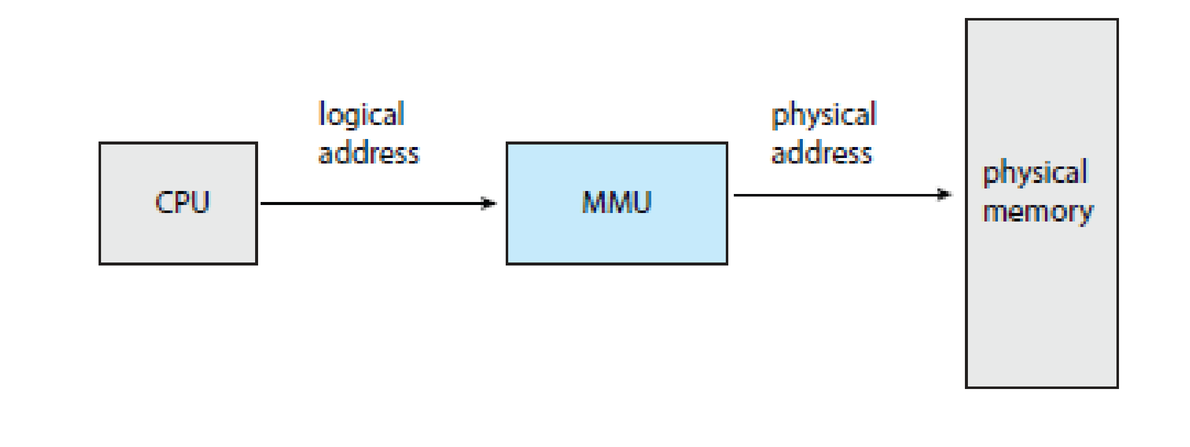
- The set of all physical addresses corresponding to these logical addresses is a physical address space

  - Thus, in the execution-time address-binding scheme, the logical and physical address spaces differ.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

LOGICAL VERSUS PHYSICAL MEMORY MANAGEMENT.

- The run-time mapping from virtual to physical addresses is done by a hardware device called the memory-management unit (MMU) .
- We can choose from many different methods to accomplish such mapping,



Memory management unit (MMU).

LOGICAL VERSUS PHYSICAL MEMORY MANAGEMENT.

- In an MMU, the base register is now called a relocation register.
- The value in the relocation register is added to every address generated by a user process at the time the address is sent to memory
  - E.g. if the base is at 14000, then an attempt by the user to address location 0 is dynamically relocated to location 14000;
  - an access to location 346 is mapped to location 14346

LOGICAL VERSUS PHYSICAL MEMORY MANAGEMENT.

- The user program never accesses the real physical addresses.
  - The program can create a pointer to location 346, store it in memory, manipulate it, and compare it with other addresses—all as the number 346

- Only when it is used as a memory address (in an indirect load or store, perhaps) is it relocated relative to the base register

- The user program deals with logical addresses

LOGICAL VERSUS PHYSICAL MEMORY MANAGEMENT.

- The memory mapping hardware converts logical addresses into physical addresses

- The final location of a referenced memory address is not determined until the reference is made.

- We now have two different types of addresses:
  - logical addresses (in the range 0 to max)
  - physical addresses (in the range R + 0 to R + max for a base value R).

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

LOGICAL VERSUS PHYSICAL MEMORY MANAGEMENT.

- The user program generates only logical addresses and thinks that the process runs in memory locations from 0 to max
- However, these logical addresses must be mapped to physical addresses before they are used
- The concept of a logical address space that is bound to a separate physical address space is central to proper memory management.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

DYNAMIC LOADING

- In our discussion so far, it has been necessary for the entire program and all data of a process to be in physical memory for the process to execute.

- The size of a process has thus been limited to the size of physical memory.

- To obtain better memory-space utilization, we can use dynamic loading.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

DYNAMIC LOADING

- With dynamic loading, a routine is not loaded until it is called.
- All routines are kept on disk in a relocatable load format.
- The main program is loaded into memory and is executed.
- When a routine needs to call another routine, the calling routine first checks to see whether the other routine has been loaded.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

DYNAMIC LOADING

- If it has not, the relocatable linking loader is called to load the desired routine into memory and to update the program's address tables to reflect this change

- Then control is passed to the newly loaded routine.

- The advantage of dynamic loading is that a routine is loaded only when it is needed.
- This method is particularly useful when large amounts of code are needed to handle infrequently occurring cases, such as error routines

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

DYNAMIC LINKING AND SHARED LIBRARIES

- Dynamically linked libraries (DLLs) are system libraries that are linked to use programs when the programs are run

- Some operating systems support only static linking, in which system libraries are treated like any other object module and are combined by the loader into the binary program image.

- Dynamic linking, in contrast, is similar to dynamic loading.
  - Here, though, linking, rather than loading, is postponed until execution time.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

DYNAMIC LINKING AND SHARED LIBRARIES

- A second advantage of DLLs is that these libraries can be shared among multiple processes, so that only one instance of the DLL in main memory.

- For this reason, DLLs are also known as shared libraries, and are used extensively in Windows and Linux systems.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

ADDRESS SPACE : CONTIGUOUS MEMORY ALLOCATION

- The main memory must accommodate both the operating system and the various user processes.
- We therefore need to allocate main memory in the most efficient way possible.
- One early method is Contiguous Memory Allocation.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

ADDRESS SPACE : CONTIGUOUS MEMORY ALLOCATION

- In Contiguous Memory Allocation, the memory is usually divided into two partitions: one for the OS, & one for the user processes.
  - We can place the OS in either low memory addresses or high memory addresses.

- This decision depends on many factors, such as th location of the interrupt vector.
  - Many OSs (including Linux and Windows) place the operating system in high memory, & therefore we discuss only that situation.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

ADDRESS SPACE : CONTIGUOUS MEMORY ALLOCATION

- We usually want several user processes to reside in memory at the same time.

- We therefore need to consider how to allocate available memory to the processes that are waiting to be brought into memory.

- In contiguous memory allocation, each process is contained in a single section of memory that is contiguous to the section containing the next process.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

ADDRESS SPACE : Memory Allocation

- One of the simplest methods of allocating memory is to assign processes to variably sized partitions in memory, where each partition may contain exactly one process.

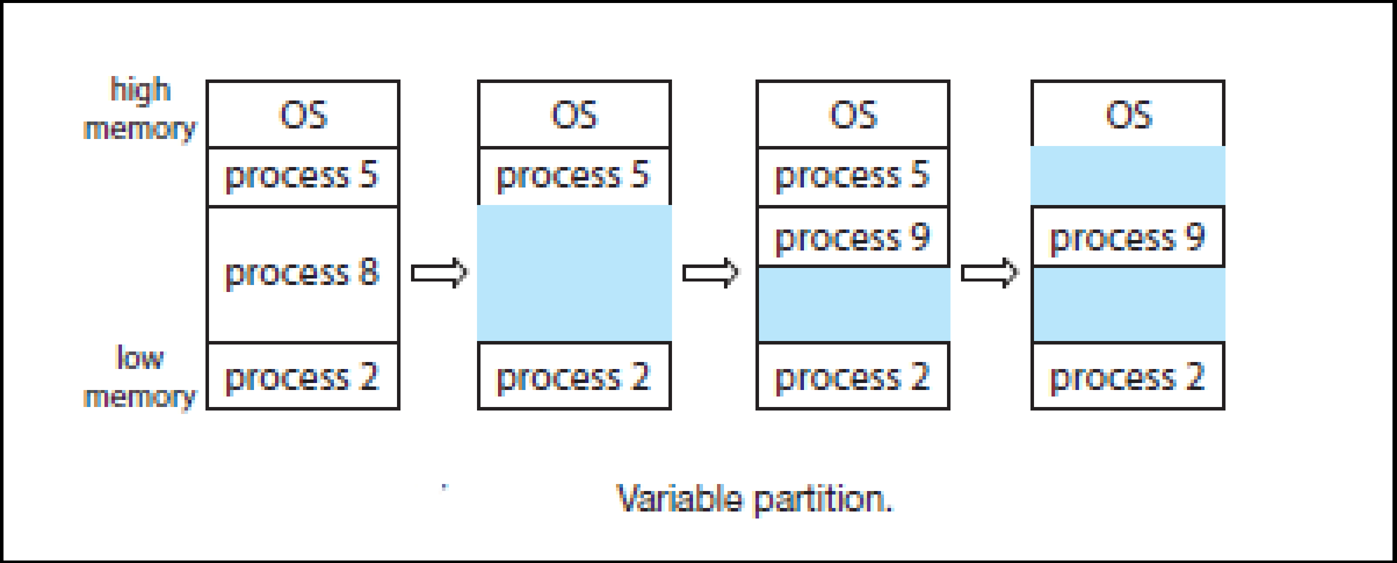
- In this variable-partition scheme, the operating system keeps a table indicating which parts of memory are available and which are occupied.

- Initially, all memory is available for user processes and is considered one large block of available memory, a hole

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

ADDRESS SPACE : Memory Allocation

- Eventually, memory contains a set of holes of various sizes, as show below.



Variable partition.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

ADDRESS SPACE : Memory Allocation

- Initially, the memory is fully utilized, containing processes 5, 8, & 2. After process 8 leaves, there is 1 contiguous hole.

- Later on, process 9 arrives and is allocated memory.

- Then process 5 departs, resulting in two noncontiguous holes.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

ADDRESS SPACE : Memory Allocation

- As processes enter the system, the operating system takes into account the memory requirements of each process and the amount of available memory space in determining which processes are allocated memory.

- When a process is allocated space, it is loaded into memory, where it can then compete for CPU time.

- When a process terminates, it releases its memory, which the operating system may then provide to another process.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

ADDRESS SPACE : Memory Allocation

- What happens when there isn't sufficient memory to satisfy the demands of an arriving process?
  - One option is to simply reject the process and provide an appropriate error message.
  - Alternatively, we can place such processes into a wait queue.
    - When memory is later released, the operating system checks the wait queue to determine if it will satisfy the memory demands of a waiting process

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

ADDRESS SPACE : Memory Allocation

- In general, as mentioned, the memory blocks available comprise a set of holes of various sizes scattered throughout memory.

- When a process arrives and needs memory, the system searches the set for a hole that is large enough for this process.
- If the hole is too large, it is split into two parts.
    - One part is allocated to the arriving process;
    - the other is returned to the set of holes.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

ADDRESS SPACE : Memory Allocation

- When a process terminates, it releases its block of memory, which is then placed back in the set of holes.

- If the new hole is adjacent to other holes, these adjacent holes are merged to form one larger hole

- This procedure is a particular instance of the general dynamic storage-allocation problem, which concerns how to satisfy a request of size n from a list of free holes.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

ADDRESS SPACE : Memory Allocation

- There are many solutions to this problem.
- The first-fit, best-fit, & worst-fit strategies are common

- **1. First fit.**
  - Allocate the first hole that is big enough.
  - Searching can start either at the beginning of the set of holes or at the location where the previous first-fit search ended.
  - We can stop searching as soon as we find a free hole that is large enough.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

ADDRESS SPACE : Memory Allocation

- **2. Best fit**
  - Allocate the smallest hole that is big enough.
  - We must search the entire list, unless the list is ordered by size.
  - This strategy produces the smallest leftover hole
- **3. Worst fit.**
  - Allocate the largest hole.
  - Again, we must search the entire list, unless it is sorted by size.
  - This strategy produces the largest leftover hole, which may be more useful than the smaller leftover hole

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

PAGING

- Memory management discussed thus far has required the physical address space of a process to be contiguous.

- We now introduce paging, a memory management scheme that permits a process's physical address space to be noncontiguous.

- Paging avoids external fragmentation and the associated need for compaction, two problems that plague contiguous memory allocation

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

PAGING

- External fragmentation exists when there is enough total memory space to satisfy a request but the available spaces are not contiguous:
    - storage is fragmented into a large number of small holes.

- Paging is implemented through cooperation between the operating system and the computer hardware.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

PAGING

- The basic method for implementing paging involves breaking physical memory into fixed-sized blocks called frames & breaking logical memory into blocks of the same size called pages

- When a process is to be executed, its pages are loaded into any available memory frames from their source (a file system or the backing store).
  - The backing store is divided into fixed-sized blocks that are the same size as the memory frames or clusters of multiple frames

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

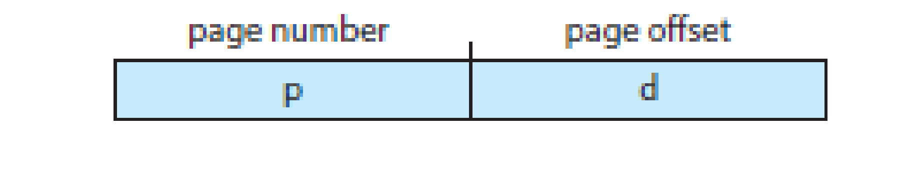
PAGING

- This rather simple idea has great functionality and wide ramifications.
  - For example, the logical address space is now totally separate from the physical address space, so a process can have a logical 64-bit address space even though the system has less than 2^{64} bytes of physical memory.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

PAGING

- Every address generated by the CPU is divided into two parts:
  - a page number (p); and
  - a page offset (d):

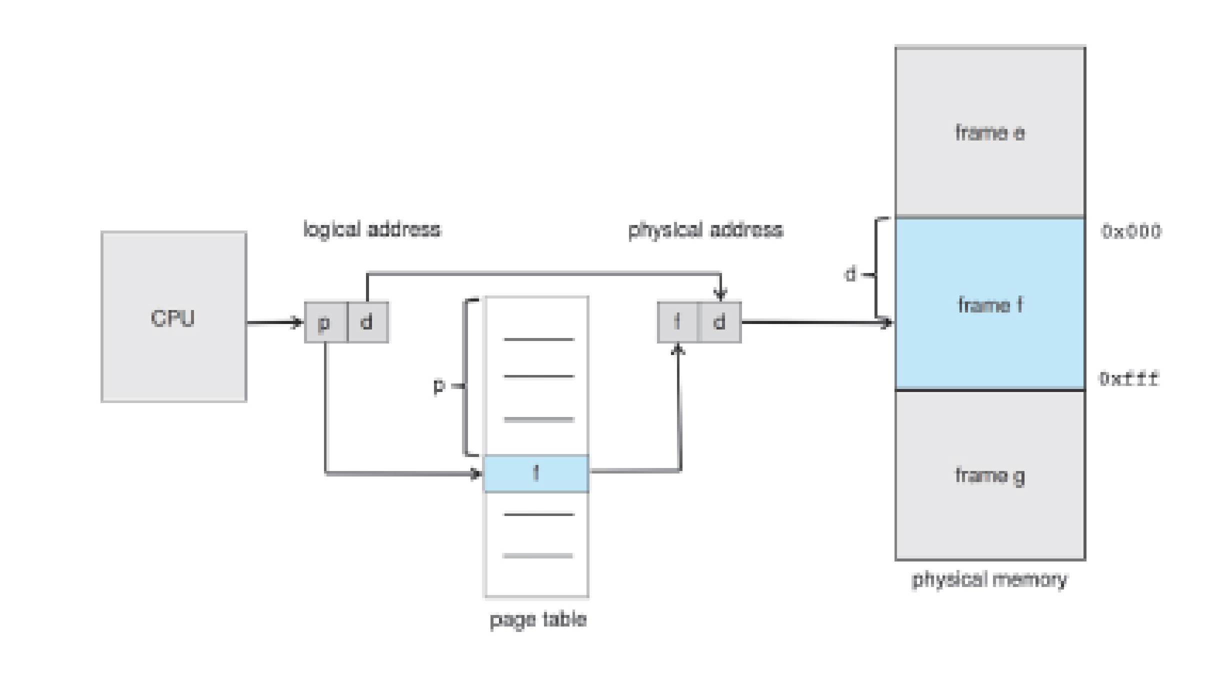


- The page number is used as an index into a per-process page table

- This is illustrated in the next figure

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

PAGING

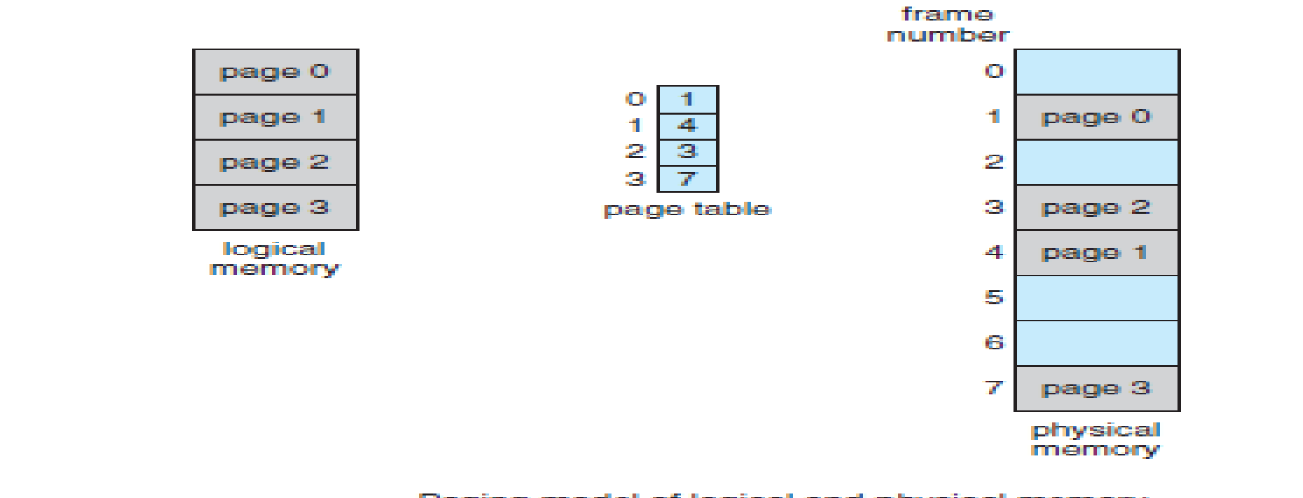


Paging hardware.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

PAGING

- The page table contains the base address of each frame in physical memory, and the offset is the location in the frame being referenced
  - Thus, the base address of the frame is combined with the page offset to define the physical memory address.



Paging model of logical and physical memory.

PAGING

- The following outlines the steps taken by the MMU to translate a logical address generated by the CPU to a physical address:
  - 1. Extract the page number p and use it as an index into the page table.
  - 2. Extract the corresponding frame number f from the page table.
  - 3. Replace the page number p in the logical address with the frame number .

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

PAGING

- As the offset d does not change, it is not replaced, & the frame number & offset now comprise the physical address.

- The page size (like the frame size) is defined by the hardware.

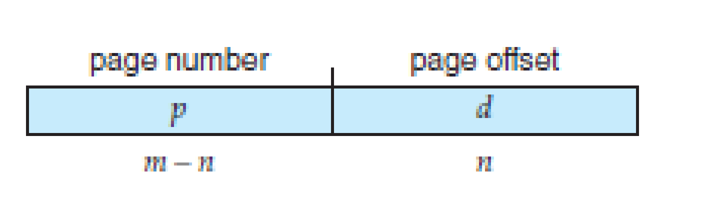
- The size of a page is a power of 2, typically varying between 4 KB and 1 GB per page, depending on the computer architecture.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

PAGING

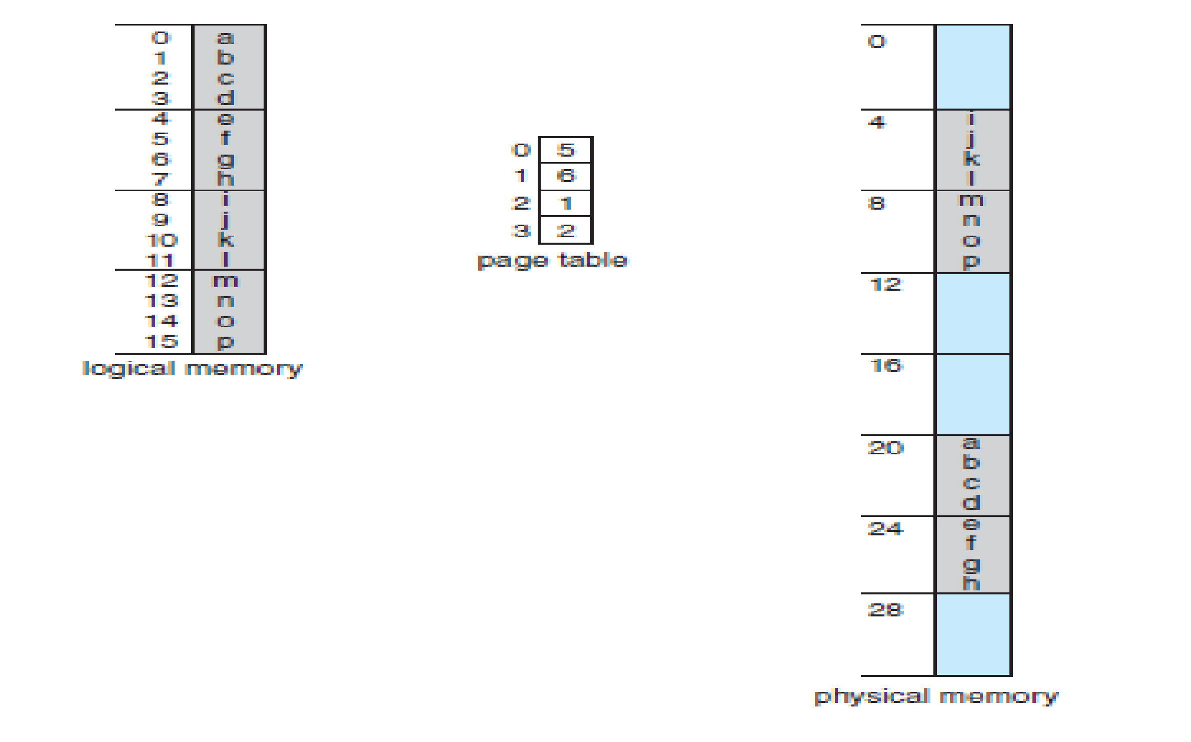
- The selection of a power of 2 as a page size makes the translation of a logical address into a page number & page offset particularly easy.

- If the size of the logical address space is 2^m, and a page size is 2^n bytes, then the high-order *m−n* bits of a logical address designate the page number, and the n low-order bits designate the page offset



PAGING

- As a concrete (although minuscule) example, consider the memory below.



Paging example for a 32-byte memory with 4-byte pages.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

PAGING

- Here, in the logical address, n = 2 and m = 4.

- Using a page size of 4 bytes and a physical memory of 32 bytes (8 pages), we show how the programmer's view of memory can be mapped into physical memory

- Logical address 0 is page 0, offset 0.

- Indexing into the page table, we find that page 0 is in frame 5.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

PAGING

- Thus, logical address 0 maps to physical address 20
  - = (5 × 4) + 0.

- Logical address 3 (page 0, offset 3) maps to physical address 23
  - [= (5 × 4) + 3].

- Logical address 4 is page 1, offset 0; according to the page table, page 1 is mapped to frame 6.
- Thus, logical address 4 maps to physical address 24
  - [= (6 × 4) + 0].

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

VIRTUAL MEMORY

- Memory-management strategies used in computer systems have the same goal:
  - to keep many processes in memory simultaneously to allow multiprogramming.

- However, they tend to require that an entire process be in memory before it can execute.

- Virtual memory is a technique that allows the execution of processes that are not completely in memory.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

VIRTUAL MEMORY

- One major advantage of the virtual memory scheme is that programs can be larger than physical memory.
- Virtual memory also allows processes to share files and libraries, and to implement shared memory.
- In addition, it provides an efficient mechanism for process creation.
- Virtual memory is not easy to implement, however, & may substantially decrease performance if it is used carelessly.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

VIRTUAL MEMORY

- Memory-management algorithms are necessary because of one basic requirement: the instructions being executed must be in physical memory.
    - The approach to meeting this requirement is to place the entire logical address space in physical memory
- The requirement that instructions must be in physical memory to be executed seems both necessary and reasonable; but it is also unfortunate, since it limits the size of a program to the size of physical memory

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

VIRTUAL MEMORY

- In fact, an examination of real programs shows us that, in many cases, the entire program is not needed.

- For instance, consider the following:
  - Programs often have code to handle unusual error conditions. Since these errors seldom, if ever, occur in practice, this code is almost never executed.
  - Arrays, lists, and tables are often allocated more memory than they actually need
  - Certain options and features of a program may be used rarely

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

VIRTUAL MEMORY

- Even in those cases where the entire program is needed, it may not all be needed at the same time.

- The ability to execute a program that is only partially in memory would confer many benefits:
  - 1. A program would no longer be constrained by the amount of physical memory that is available. Users would be able to write programs for an extremely large virtual address space, simplifying the programming task.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

VIRTUAL MEMORY

- 2. Because each program could take less physical memory, more programs could be run at the same time, with a corresponding increase in CPU utilization & throughput but with no increase in response time or turnaround time.

- 3. Less I/O would be needed to load or swap portions of programs into memory, so each program would run faster.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

VIRTUAL MEMORY

- 2. Because each program could take less physical memory, more programs could be run at the same time, with a corresponding increase in CPU utilization & throughput but with no increase in response time or turnaround time.

- 3. Less I/O would be needed to load or swap portions of programs into memory, so each program would run faster.

- Thus, running a program that is not entirely in memory would benefit both the system and its users

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

VIRTUAL MEMORY

- Virtual memory involves the separation of logical memory as perceived by developers from physical memory.

- This separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available

- This makes the task of programming much easier, because the programmer no longer needs to worry about the amount of physical memory available; she can concentrate instead on programming the problem that is to be solved.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

VIRTUAL MEMORY

- This is shown below.

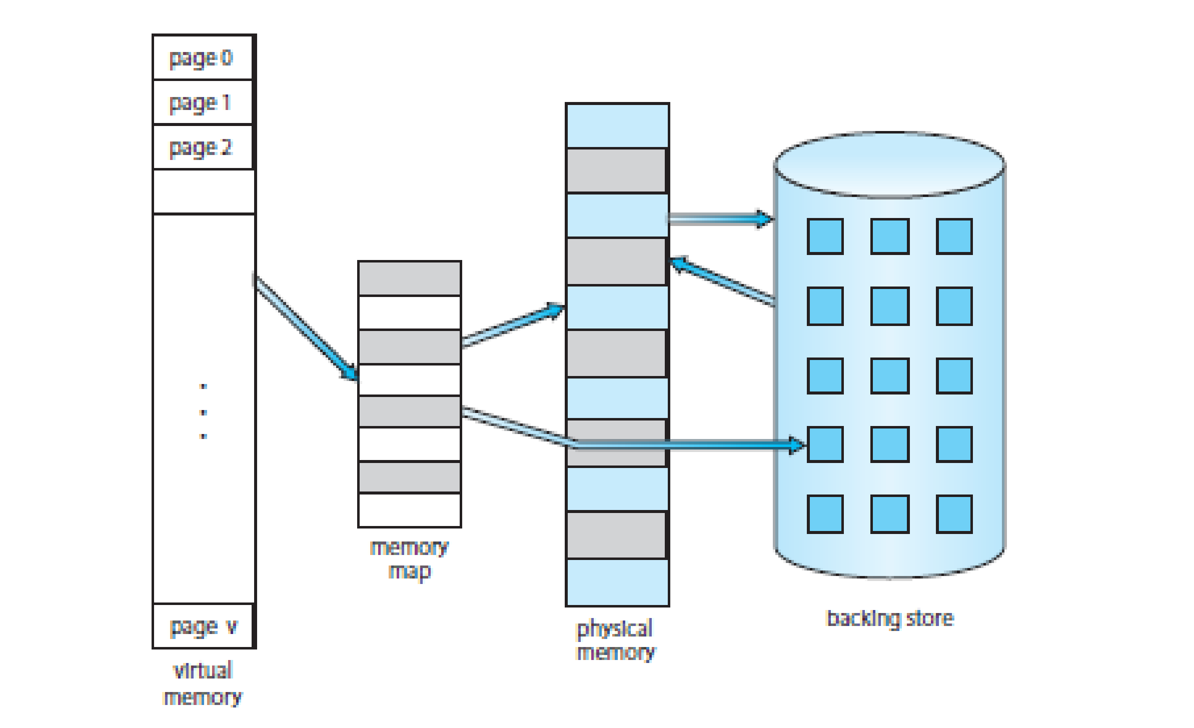


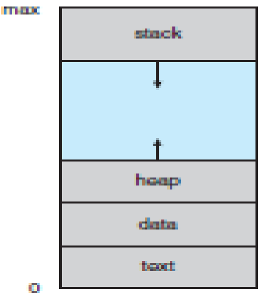
Diagram showing virtual memory that is larger than physical memory.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

VIRTUAL MEMORY

- The virtual address space of a process refers to the logical (or virtual) view of how a process is stored in memory.

- Typically, this view is that a process begins at a certain logical address—say, address 0—and exists in contiguous memory, as shown below



Virtual address space of a process in memory.

VIRTUAL MEMORY

- in fact physical memory is organized in page frames and that the physical page frames assigned to a process may not be contiguous.

- It is up to the memory-management unit (MMU) to map logical pages to physical page frames in memory.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

DEMAND PAGING

- Consider how an executable program might be loaded from secondary storage into memory.

- One option is to load the entire program in physical memory at program execution time.

- However, a problem with this approach is that we may not initially need the entire program in memory

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

DEMAND PAGING

- Suppose a program starts with a list of available options from which the user is to select.
- Loading the entire program into memory results in loading the executable code for all options, regardless of whether or not an option is ultimately selected by the user.
- An alternative strategy is to load pages only as they are needed.
  - This technique is known as demand paging and is commonly used in virtual memory systems.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

DEMAND PAGING

- With demand-paged virtual memory, pages are loaded only when they are demanded during program execution.

- Pages that are never accessed are thus never loaded into physical memory

- Demand paging explains one of the primary benefits of virtual memory—by loading only the portions of programs that are needed, memory is used more efficiently.

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini

THE END

ALL RIGHTS RESERVED
No part of this document may be reproduced without written approval from Limkokwing University Of Creative Technology Eswatini