

Operating Systems

Prepared by Dr. Alaa Mohamed

Module 5-part1

جامعة الجميع الذكية

تسعى لتكون الكيان الأول الذي يتبنى مبدأ المجتمع التعليمي
التعاوني ويقدم العلم مقابل العمل لتنمية وبناء المجتمعات.

Memory Management

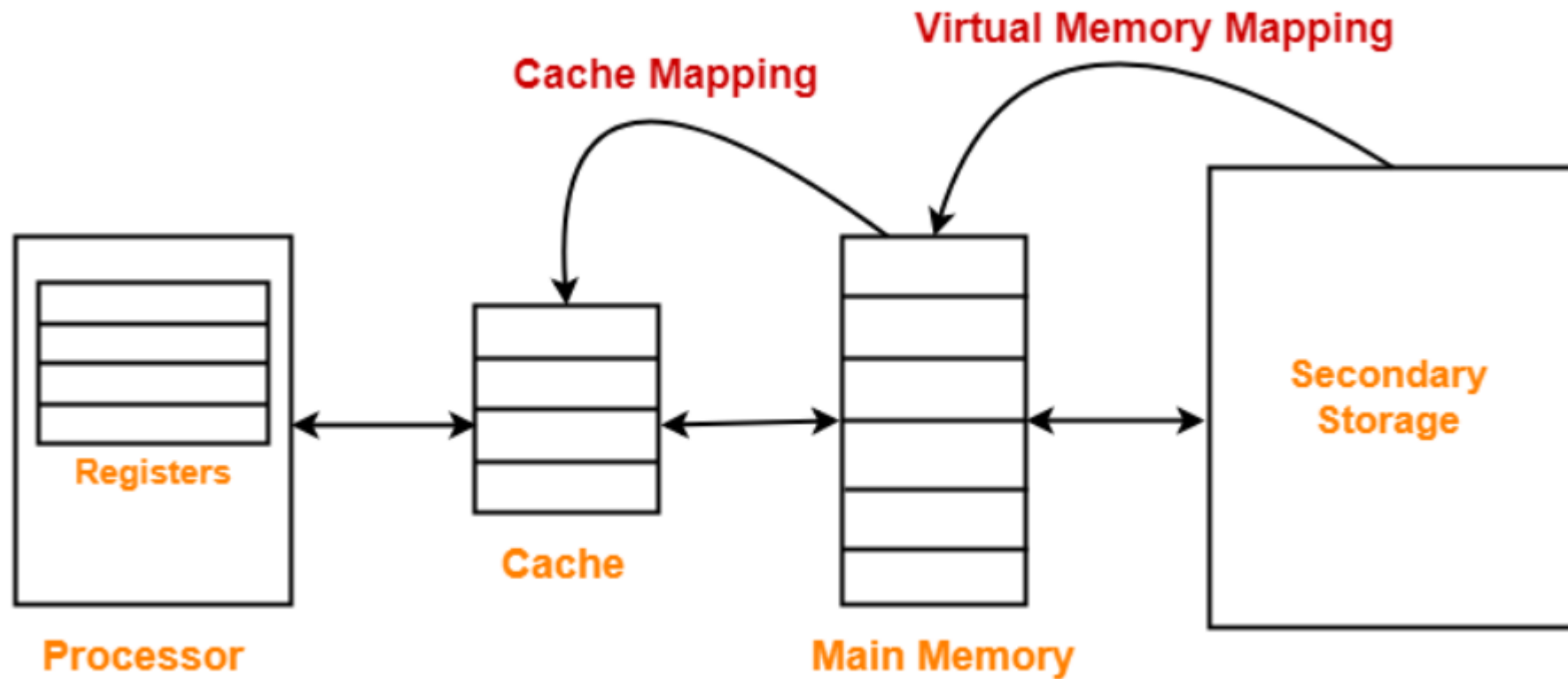


جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY

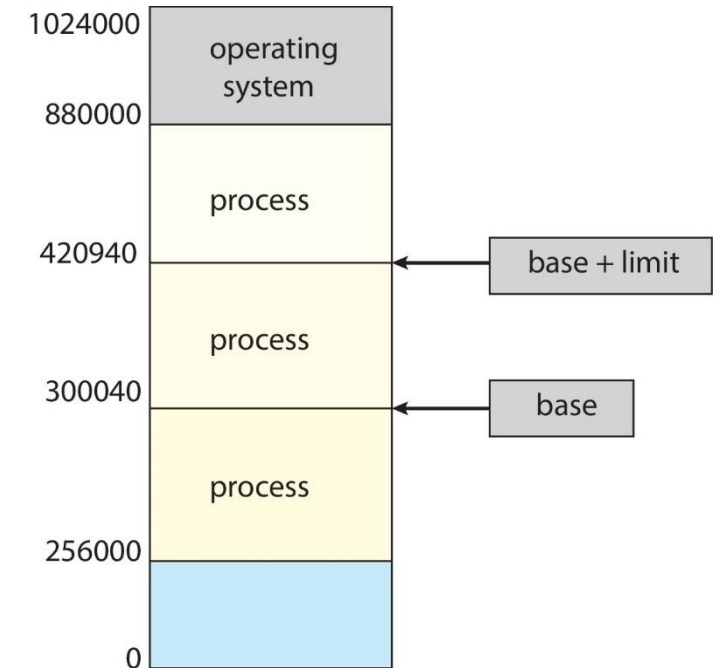
- Objectives:
 - Background
 - Contiguous Memory Allocation
 - Paging
 - Structure of the Page Table

- Topics To Be Covered:
 - To provide a detailed description of various ways of organizing memory hardware
 - To discuss various memory-management techniques (Paging).

- 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
- Memory unit only sees a stream of:
 - addresses + read requests, or
 - address + data and write requests
- Register access is done in one CPU clock (or less)
- Main memory can take many cycles, causing a **stall**
- **Cache** sits between main memory and CPU registers
- Protection of memory required to ensure correct operation

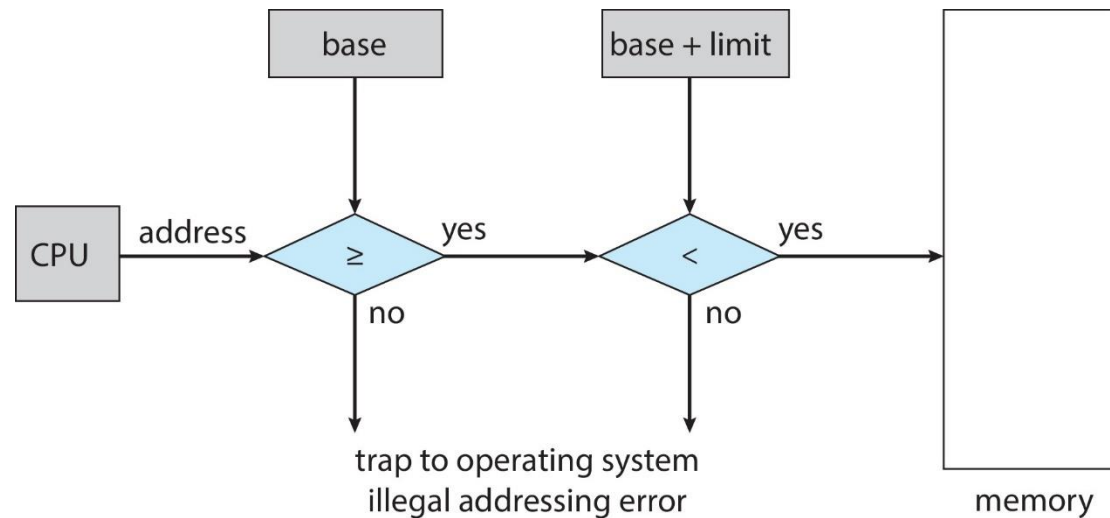


- Need to ensure that a process can access only those addresses in its address space.
- We can provide this protection by using a pair of **base** and **limit registers** to define the logical address space of a process



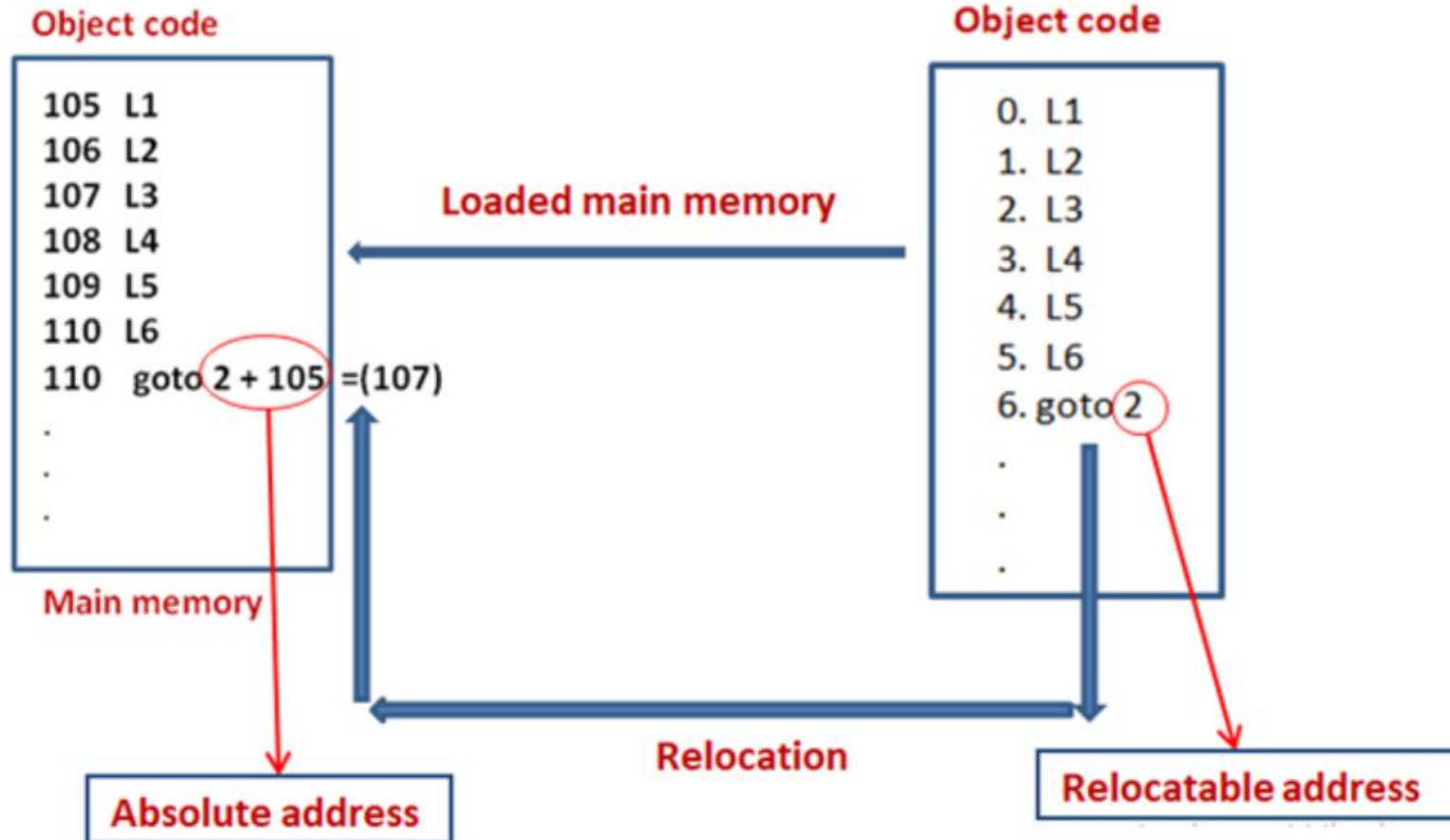
Hardware Address Protection

- CPU must check every memory access generated in user mode to be sure it is between base and limit for that user



- the instructions to loading the base and limit registers are privileged

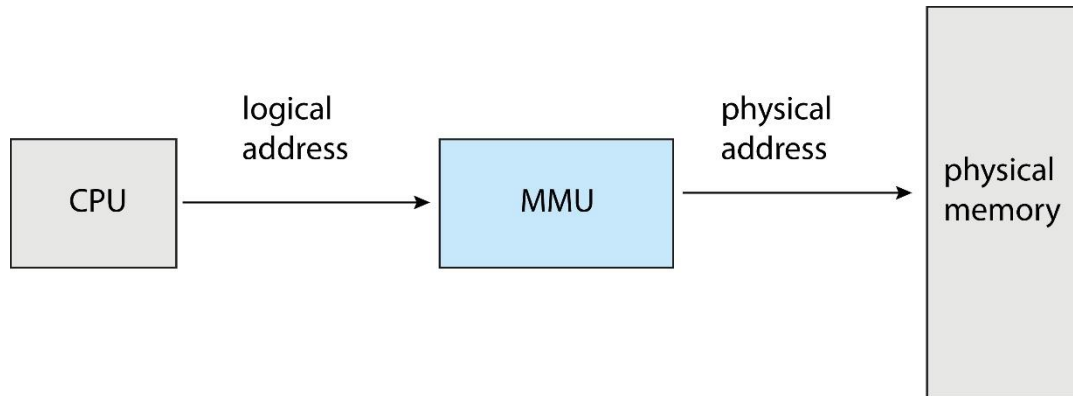
- Programs on disk, ready to be brought into memory to execute form an **input queue**
 - Without support, must be loaded into address 0000
- Inconvenient to have first user process physical address always at 0000
 - How can it not be?
- Addresses represented in different ways at different stages of a program's life
 - Source code addresses usually symbolic
 - Compiled code addresses **bind** to relocatable addresses
 - i.e. "14 bytes from beginning of this module"
 - Linker or loader will bind relocatable addresses to absolute addresses
 - i.e. 74014
- Each binding maps one address space to another



- The concept of a logical address space that is bound to a separate
- **physical address space** is central to proper memory management
 - **Logical address** – generated by the CPU; also referred to as **virtual address**
 - **Physical address** – address seen by the memory unit
- **Logical address space** is the set of all logical addresses generated by a program
- **Physical address space** is the set of all physical addresses generated by a program

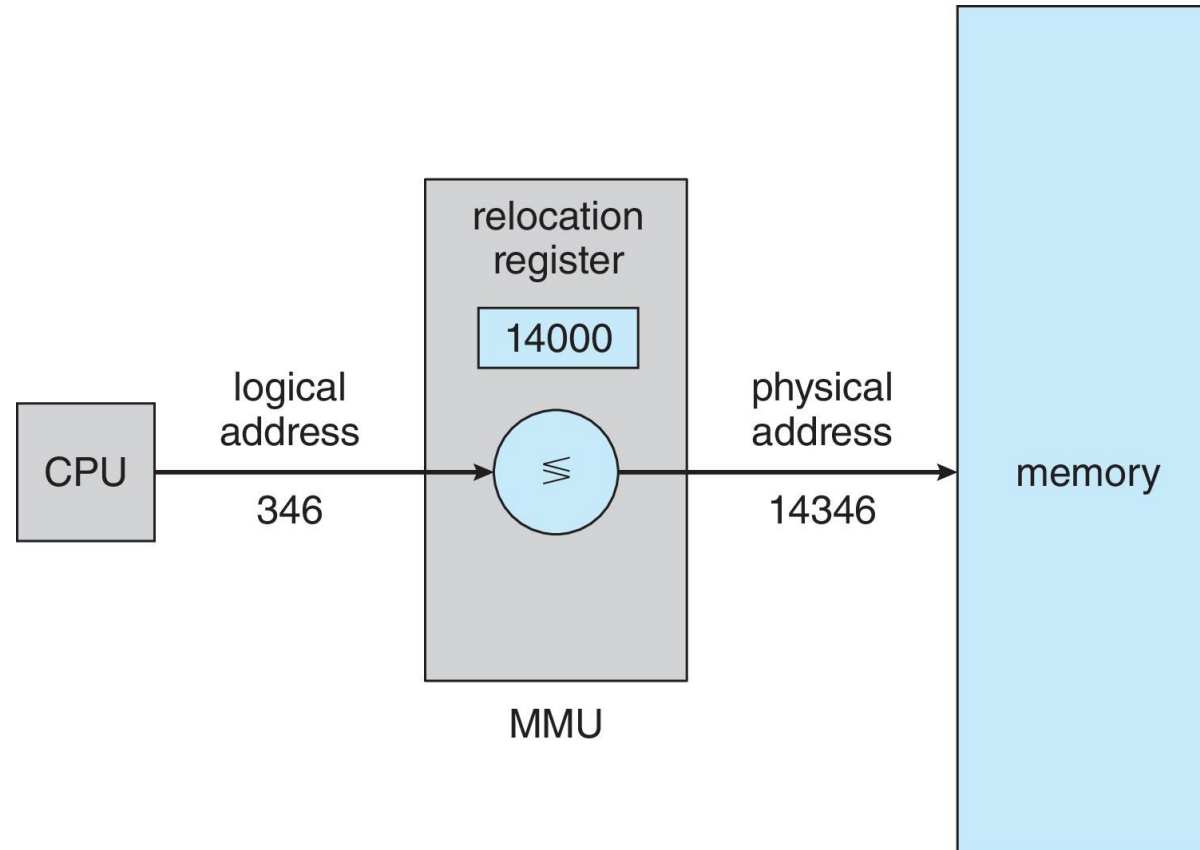
Memory-Management Unit (MMU)

- Hardware device that at run time maps virtual to physical address



- Many methods possible, covered in the rest of this chapter

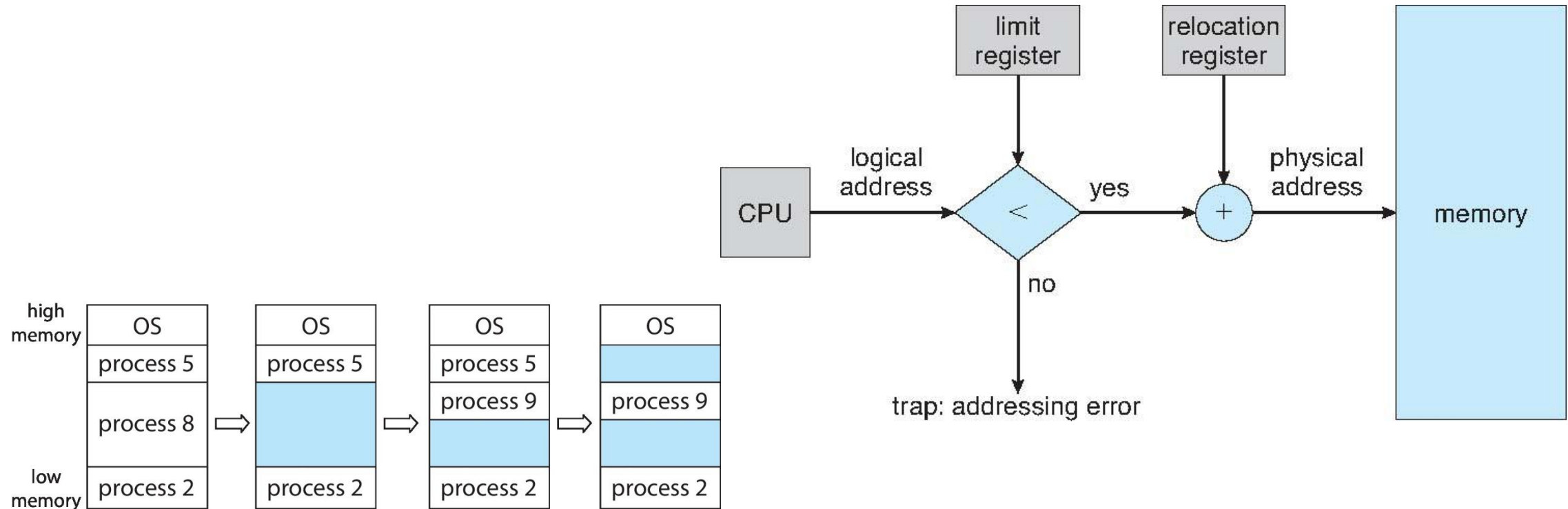
- Consider simple scheme. which is a generalization of the base-register scheme.
 - The base register now called **relocation register**
- The value in the relocation register is added to every address generated by a user process at the time it is sent to memory
- The user program deals with *logical* addresses; it never sees the *real*/physical addresses
 - Execution-time binding occurs when reference is made to location in memory
 - Logical address bound to physical addresses



- Main memory must support both OS and user processes
- Limited resource, must allocate efficiently
- Contiguous allocation is one early method
- Main memory usually into two **partitions**:
 - Resident operating system, usually held in low memory with interrupt vector
 - User processes then held in high memory
 - Each process contained in single contiguous section of memory

- Relocation registers used to protect user processes from each other, and from changing operating-system code and data
 - Base register contains value of smallest physical address
 - Limit register contains range of logical addresses – each logical address must be less than the limit register
 - MMU maps logical address *dynamically*
 - Can then allow actions such as kernel code being **transient** and kernel changing size

Hardware Support for Relocation and Limit Registers



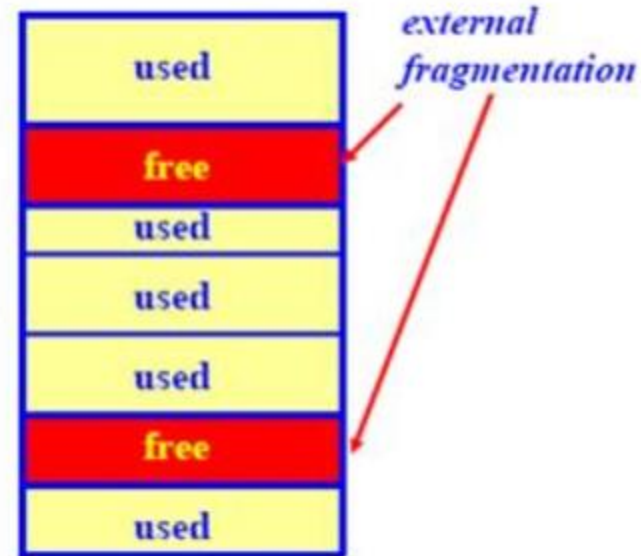
Dynamic Storage-Allocation Problem

How to satisfy a request of size n from a list of free holes?

- **First-fit**: Allocate the *first* hole that is big enough
- **Best-fit**: Allocate the *smallest* hole that is big enough; must search entire list, unless ordered by size
 - Produces the smallest leftover hole
- **Worst-fit**: Allocate the *largest* hole; must also search entire list
 - Produces the largest leftover hole

First-fit and best-fit better than worst-fit in terms of speed and storage utilization

- **External Fragmentation** – total memory space exists to satisfy a request, but it is not contiguous
- First fit analysis reveals that given N blocks allocated, $0.5 N$ blocks lost to fragmentation
 - $1/3$ may be unusable -> **50-percent rule**



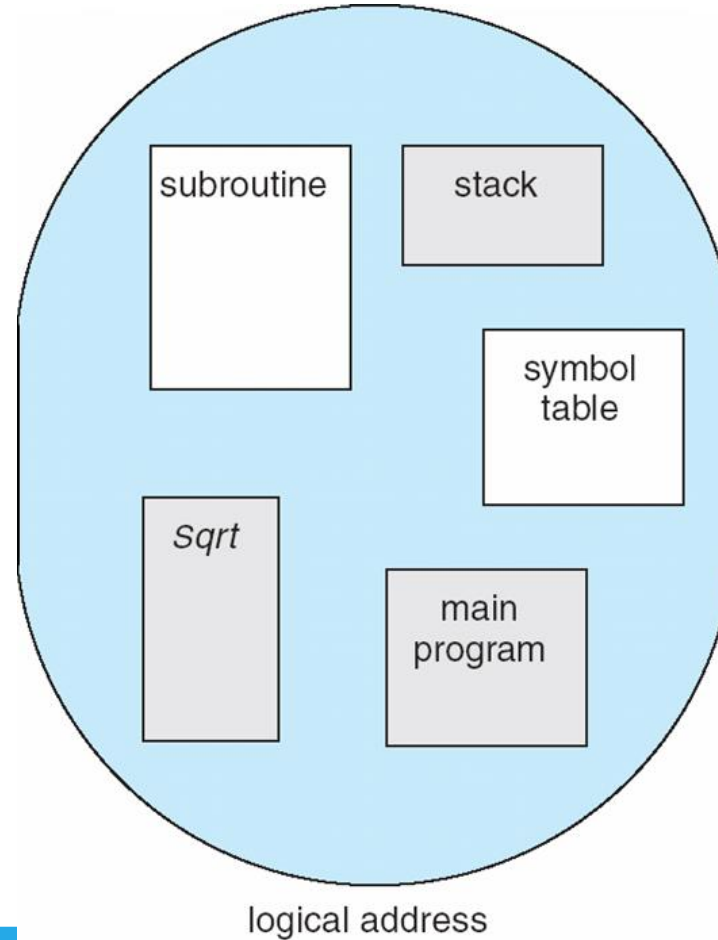
Fragmentation

- Reduce external fragmentation by **compaction**
 - Shuffle memory contents to place all free memory together in one large block
 - Compaction is possible *only* if relocation is dynamic, and is done at execution time
 - I/O problem
 - Latch job in memory while it is involved in I/O
 - Do I/O only into OS buffers

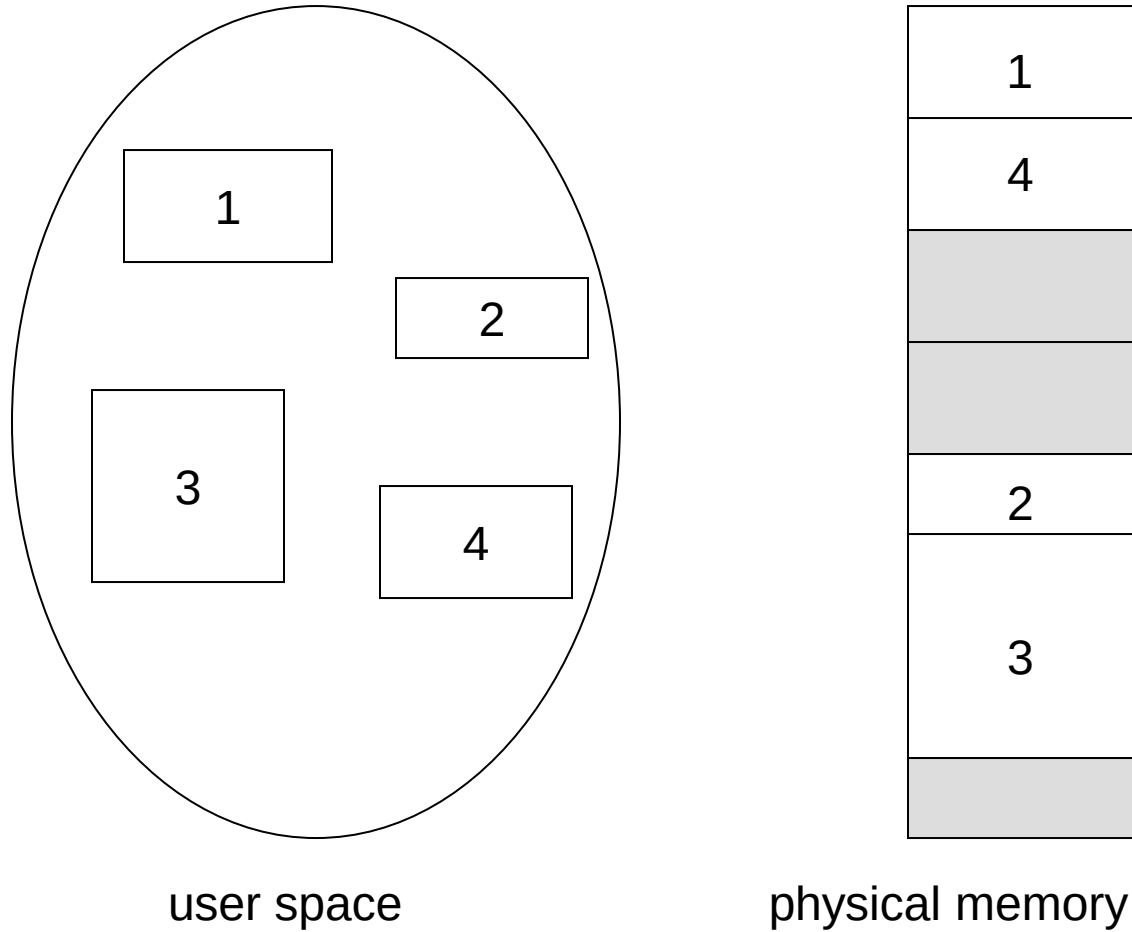
- Memory-management scheme that supports user view of memory
- A program is a collection of segments
 - A segment is a logical unit such as:
 - main program
 - procedure
 - function
 - method
 - object
 - arrays

Non- Contiguous Allocation. Segmentation

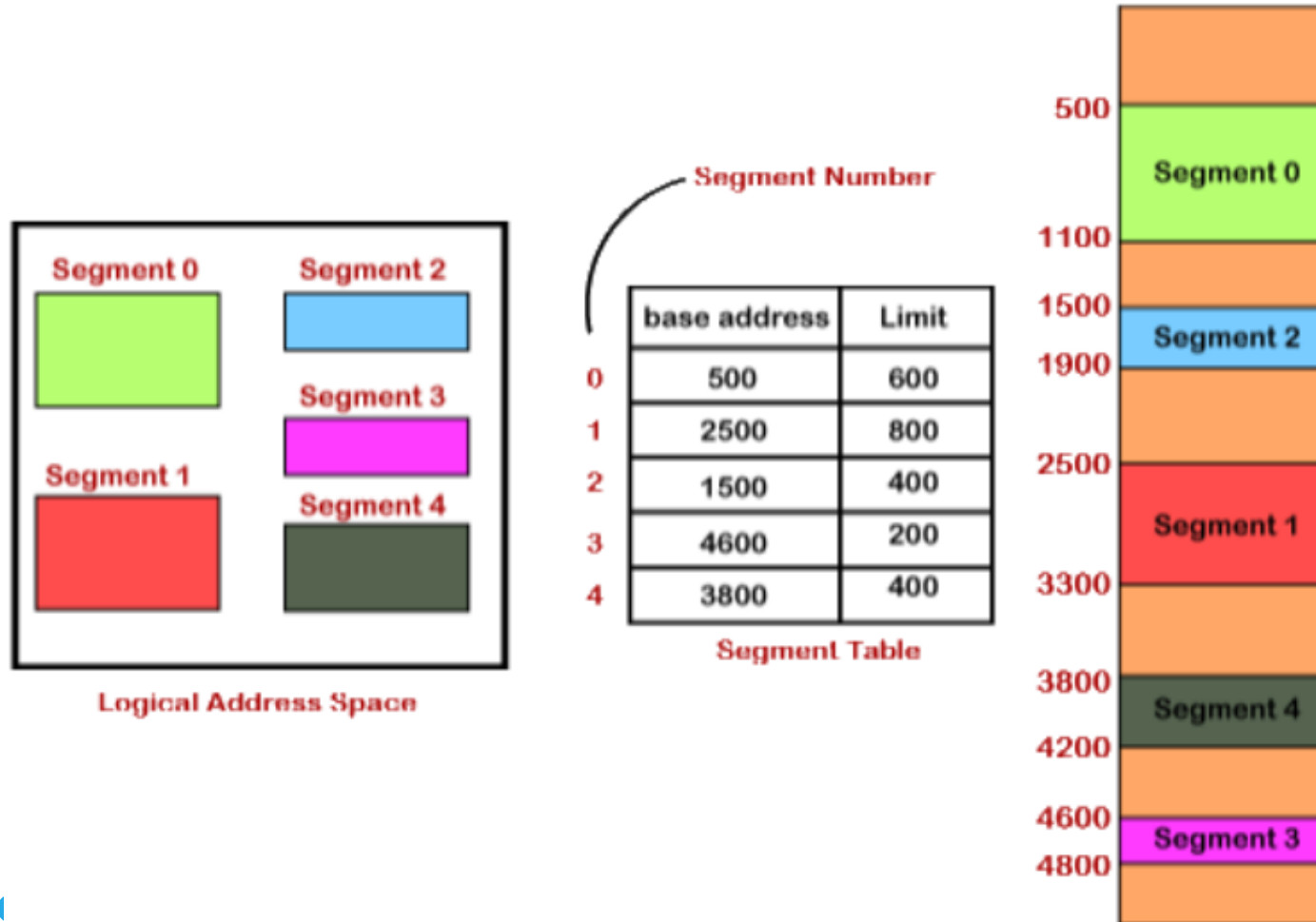
- Memory-management scheme that supports user view of memory
- A program is a collection of segments
 - A segment is a logical unit such as:
 - main program
 - procedure
 - function
 - method
 - object
 - arrays

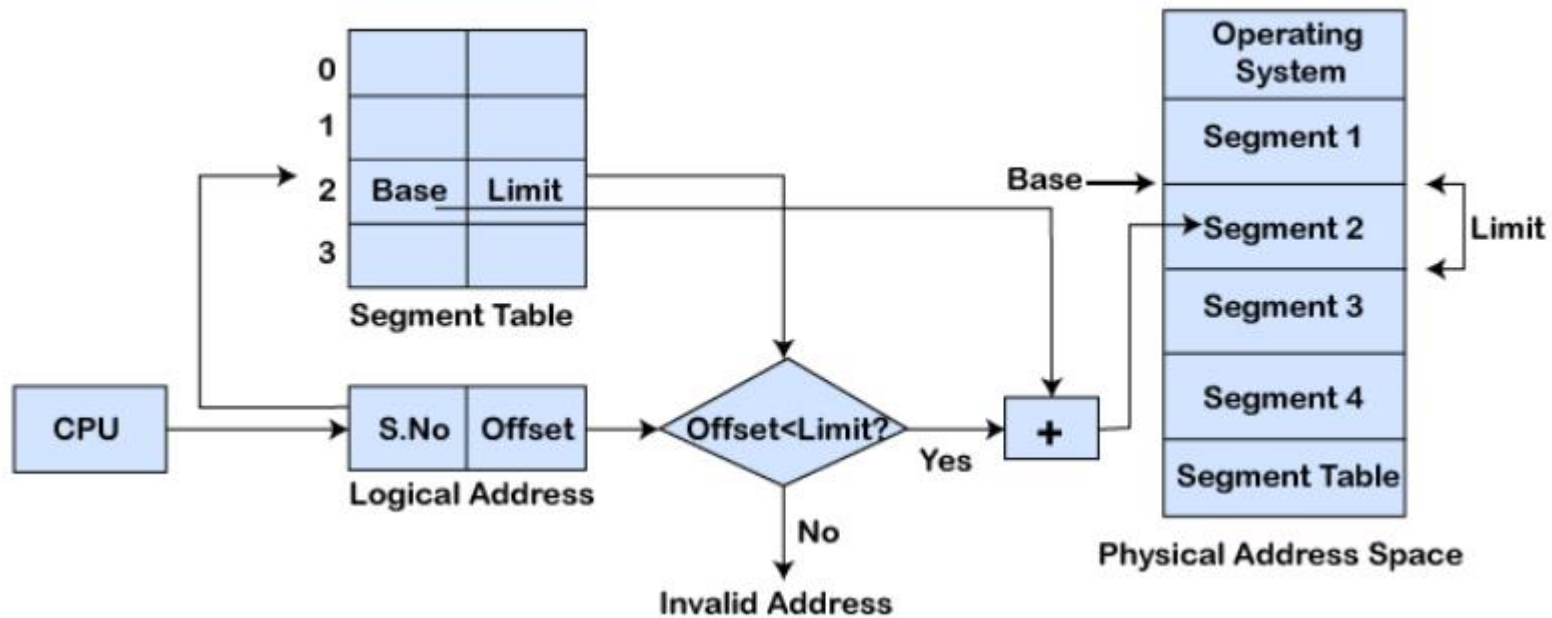


Non- Contiguous Allocation. Segmentation



- **Segment table** – maps two-dimensional physical addresses; each table entry has:
 - **base** – contains the starting physical address where the segments reside in memory
 - **limit** – specifies the length of the segment

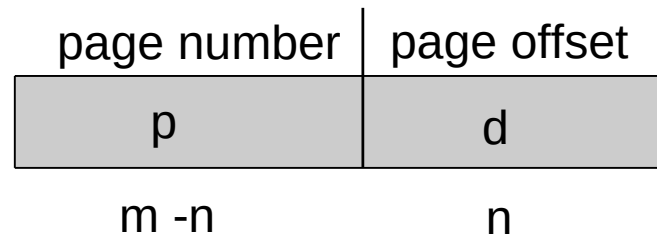




- Physical address space of a process can be noncontiguous;
 - Avoids external fragmentation
 - Avoids problem of varying sized memory chunks
- Divide physical memory into fixed-sized blocks called **frames**
 - Size is power of 2, between 512 bytes and 16 Mbytes
- Divide logical memory into blocks of same size called **pages**

- Keep track of all free frames
- To run a program of size N pages, need to find N free frames and load program
- Set up a **page table** to translate logical to physical addresses
- Backing store likewise split into pages
- Still have Internal fragmentation

- Operating system maintains a page table for each process
 - Contains the frame location for each page in the process
 - Memory address consist of a page number and offset within the page



Frame number	Main memory
0	A.0
1	A.1
2	A.2
3	A.3
4	D.0
5	D.1
6	D.2
7	C.0
8	C.1
9	C.2
10	C.3
11	D.3
12	D.4
13	
14	

0	0
1	1
2	2
3	3

Process A
page table

0	—
1	—
2	—

Process B
page table

0	7
1	8
2	9
3	10

Process C
page table

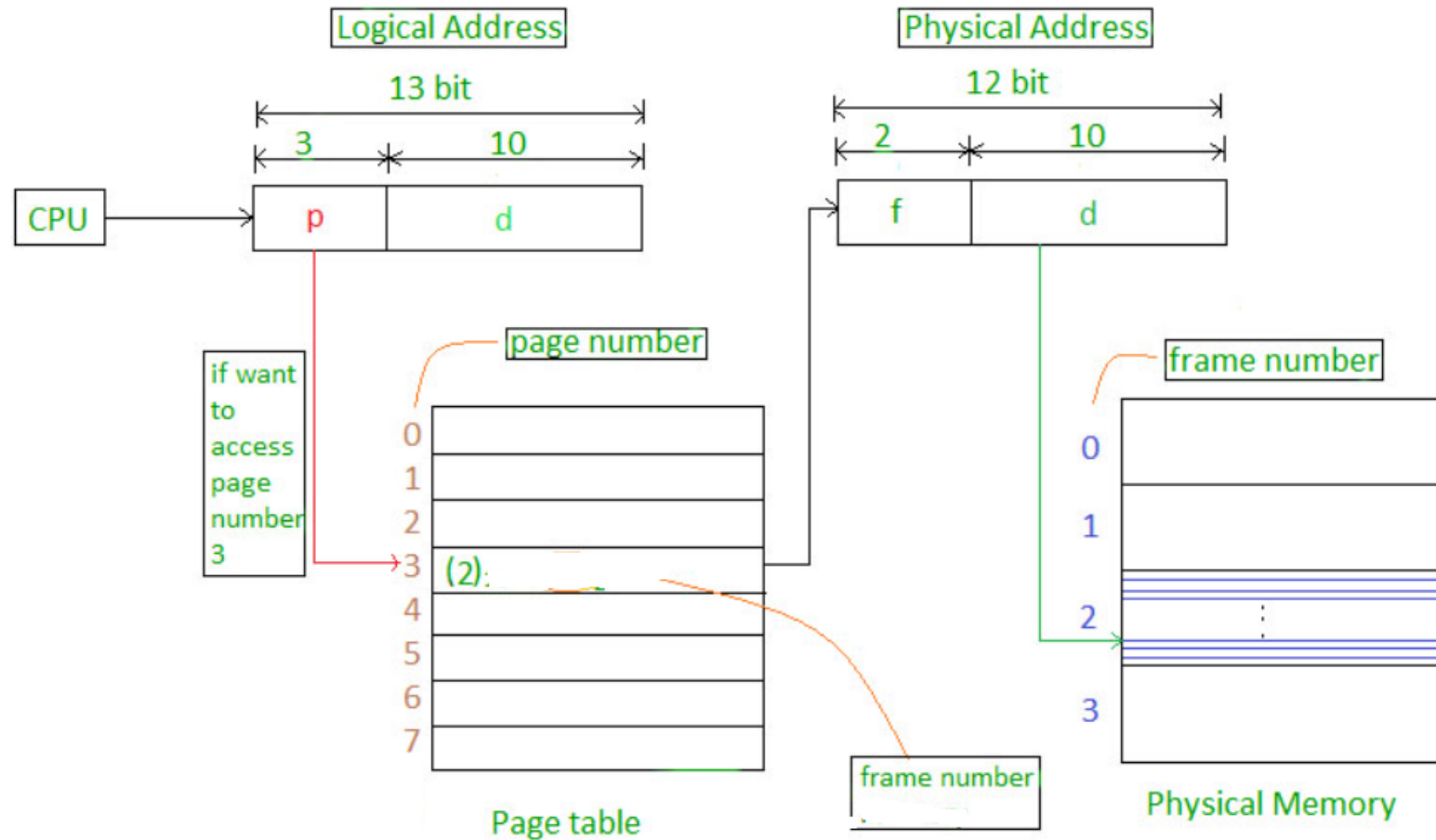
0	4
1	5
2	6
3	11
4	12

Process D
page table

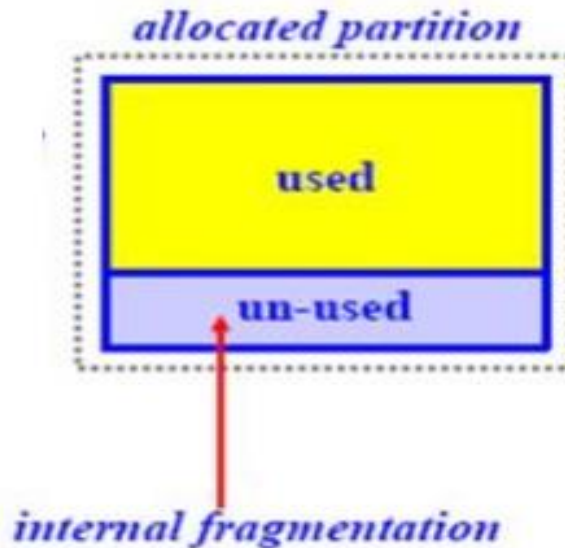
13
14

Free frame
list





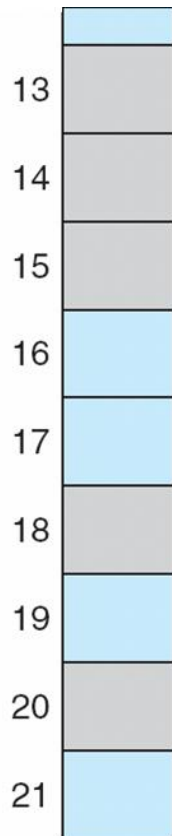
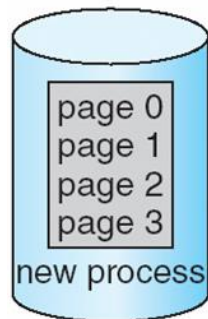
- **Internal Fragmentation** – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used



Paging -- Calculating internal fragmentation

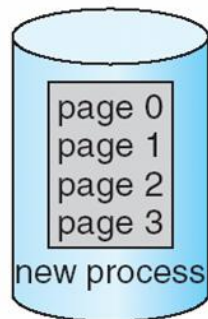
- Page size = 2,048 bytes
- Process size = 72,766 bytes
- 35 pages + 1,086 bytes
- Internal fragmentation of $2,048 - 1,086 = 962$ bytes
- Worst case fragmentation = 1 frame – 1 byte
- On average fragmentation = $1 / 2$ frame size
- So small frame sizes desirable
- But each page table entry takes memory to track

free-frame list
14
13
18
20
15



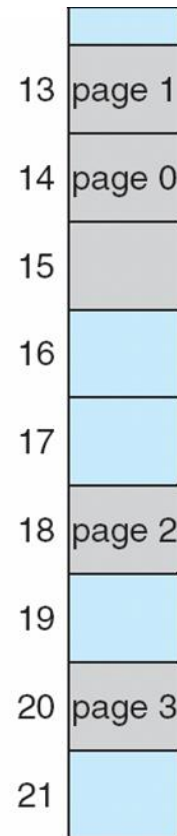
(a)

free-frame list
15



0	14
1	13
2	18
3	20

new-process page table



(b)

Paging VS Segmentation

Paging	Segmentation
Paging is a memory management technique. In paging, memory is partitioned into fixed-size pages.	Segmentation is also a memory management technique. In segmentation memory is partitioned into the variable size segments.
In paging, with the help of hardware page size is determine.	In segmentation, the user gives the size of the segments.
Paging is faster than the segmentation.	Segmentation is slower than the paging

Non- Contiguous Allocation. Segmentation

Paging	Segmentation
In paging, there may be a chance of internal fragmentation.	In segmentation, there may be a chance of external fragmentation.
In This technique, the logical address is partitioned into the page number and the page offset.	In this technique, the logical address is partitioned into the Segment number and the Segment offset.

Non- Contiguous Allocation. Segmentation

Paging	Segmentation
The user cannot see the paging	The user can see the paging
In this technique, the list of free frames is maintained by the operating system.	In this technique, the list of holes is maintained by the operating system.

End of Module 5 part1

