OPERATING SYSTEMS

Memory Management -1

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

OPERATING SYSTEMS

Course Syllabus - Unit 3

Unit-3:Unit 3: Memory Management: Main Memory

Hardware and control structures, OS support, Address translation, Swapping, Memory Allocation (Partitioning, relocation), Fragmentation, Segmentation, Paging, TLBs context switches Virtual Memory - Demand Paging, Copy-on-Write, Page replacement policy - LRU (in comparison with FIFO & Optimal), Thrashing, design alternatives - inverted page tables, bigger pages. Case Study: Linux/Windows Memory

OPERATING SYSTEMS

Course Outline

25	Main Memory: Hardware and control structures, OS support, Address translation	8.1	64.2
26	Dynamic linking, Swapping	8.2	
27	Memory Allocation (Partitioning, relocation), Fragmentation	8.3	
28	Segmentation	8.4	
29	Paging: OS Support, TLBs, Address Translation	8.5	
30	Structure of the Page Table	8.6	
31	Design Alternatives - Inverted Page Tables, Bigger Pages	8.7-8.8	
32	Virtual Memory: Demand Paging, Copy-OnWrite	9.1-9.3	
33	Page replacement policy - LRU	9.4	
34	FIFO & Optimal	9.5	
35	Thrashing	9.6	
36	Case Study: Linux/ Windows Memory Management	9.10	

Topic Outline

- Background
- Base, Limit Register and Address Translation
- Address Binding
- Binding of Instruction and Data in Memory
- Multistep Processing of User Program
- Logical and Physical Address Space
- Memory management Unit (MMU)
- Dynamic Allocation using Relocation Register

Background

- 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 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

Background

- Memory management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution.
- Memory management keeps track of each and every memory location, regardless of either it is allocated to some process or it is free.
- It checks how much memory is to be allocated to processes.
- It decides which process will get memory at what time.
- It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

Background

- The process address space is the set of logical addresses that a process references in its code.
- For example, when 32-bit addressing is in use,
    - Addresses can range from 0 to 0x7fffffff; that is, 2**31 possible numbers, for a total theoretical size of 2 gigabytes.
- The operating system takes care of mapping the logical addresses to physical addresses at the time of memory allocation to the program.

Background

- The runtime mapping from virtual to physical address is done by the **Memory Management Unit** (**MMU**) which is a hardware device. **MMU** uses following mechanism to convert virtual address to physical address.

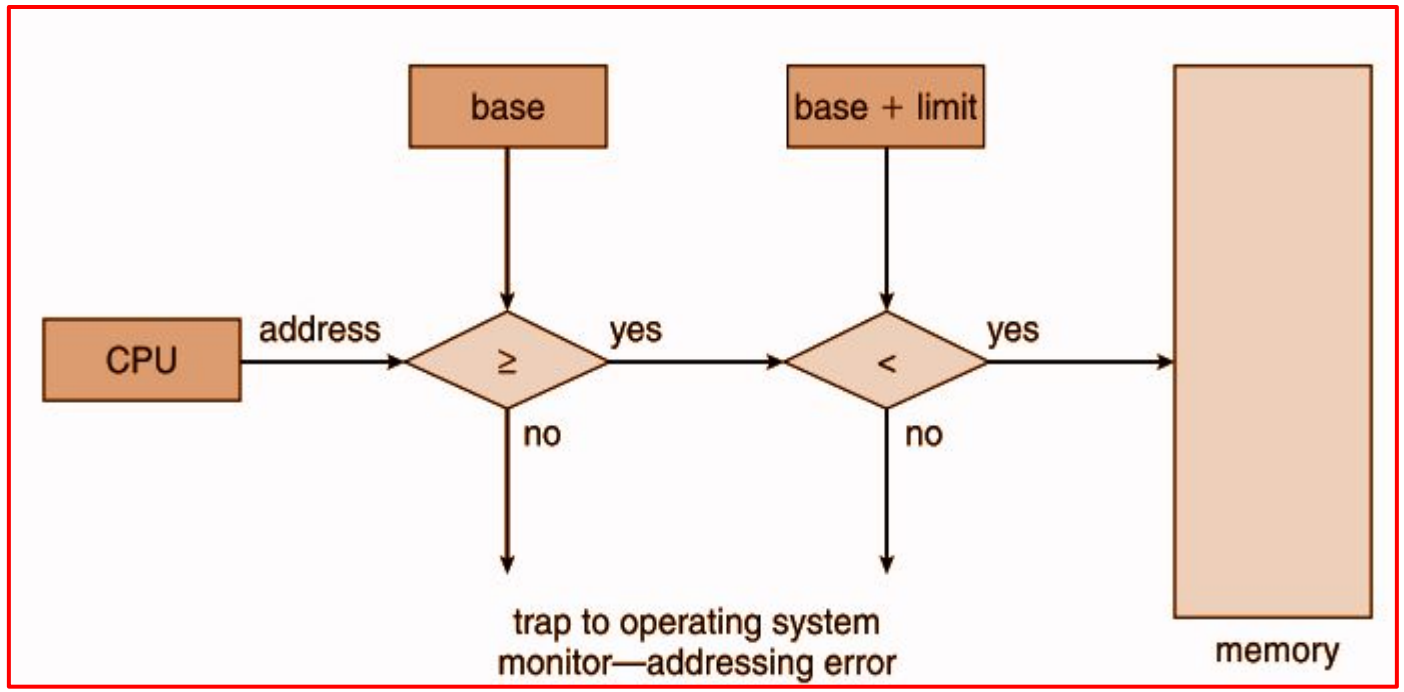
- The value in the base register is added to every address generated by a user process, which is treated as offset at the time it is sent to memory. For example, if the base register value is 10000, then an attempt by the user to use address location 100 will be dynamically re-allocated to location 10100.

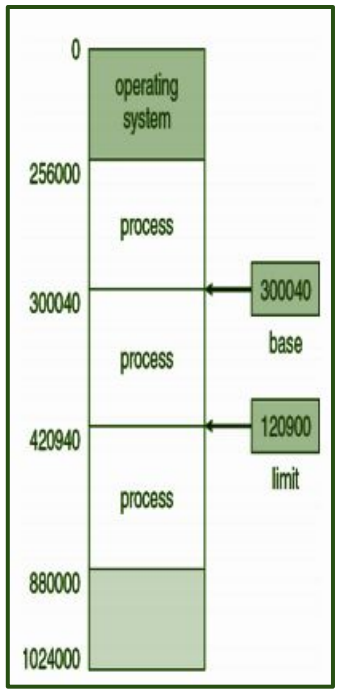
- **The user program deals with virtual addresses; it never sees the real physical addresses.**

Base, Limit Register and Address Translation

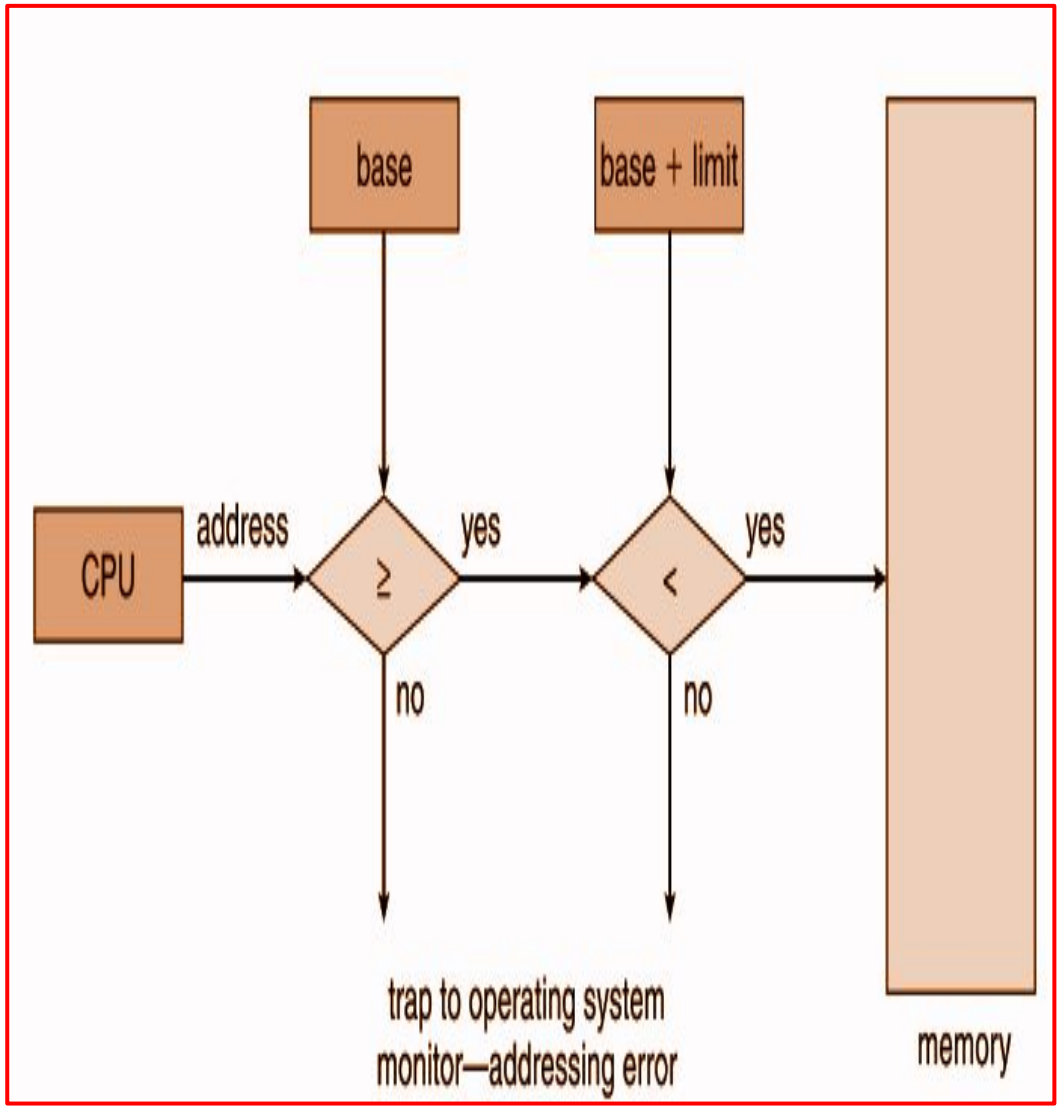
- A pair of base and limit registers define the logical address space

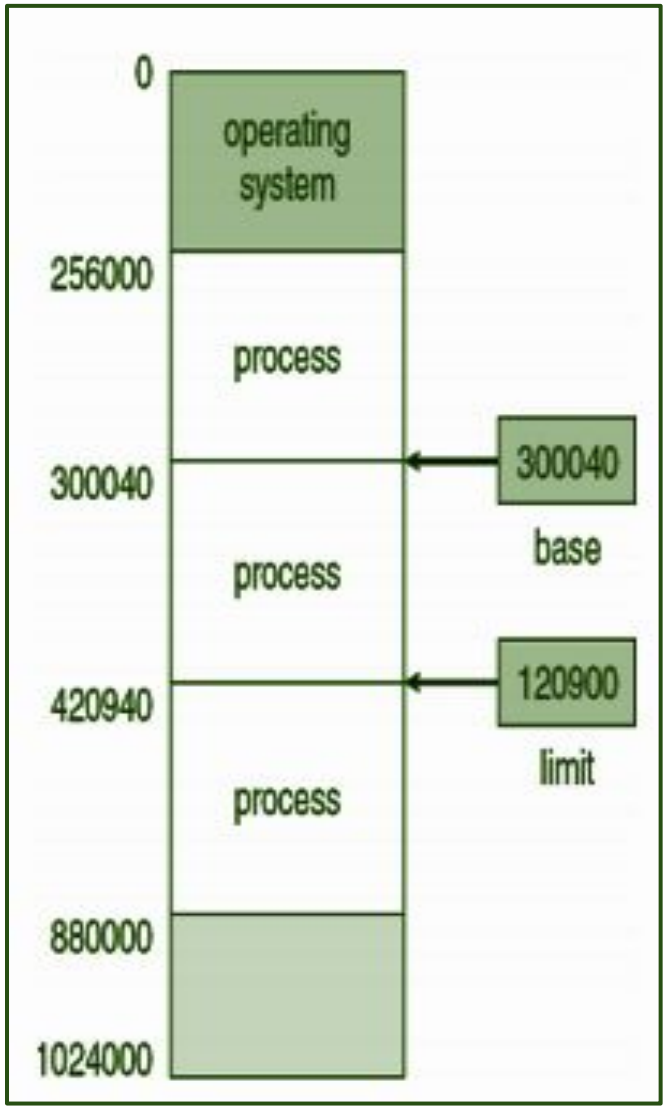
- CPU must check every memory access generated in user mode to be sure it is between base and limit for that user





Base, Limit Register and Address Translation





Background

There are three types of addresses used in a program before and after memory is allocated

Sl#	Memory Addresses	Description
1	**Symbolic addresses**	The addresses used in a source code. The variable names, constants, and instruction labels are the basic elements of the symbolic address space.
2	**Relative addresses**	At the time of compilation, a compiler converts symbolic addresses into relative addresses.
3	**Physical addresses**	The loader generates these addresses at the time when a program is loaded into main memory.

Mapping Virtual Addresses to Physical Addresses

- Memory consists of large array of words or arrays, each of which has address associated with it.

- The work of CPU is to fetch instructions from the memory based program counter. Further these instruction may cause loading or storing to specific memory address.

- Address binding is the process of mapping from one address space to another address space. Logical address is address generated by CPU during execution whereas Physical Address refers to location in memory unit - the one that is loaded into memory

- Note that user deals with only logical address (Virtual address). The logical address undergoes translation by the MMU or address translation unit in particular. The output of this process is the appropriate physical address or the location of code/data in RAM.

Address Binding

- 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 Why is it convenient to have the user process at physical address 0 ?
- Further, 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

Address Binding of Data and Instructions in the memory

- Virtual and physical addresses are the same in compile-time and load-time address-binding schemes. Virtual and physical addresses differ in execution-time address-binding scheme.

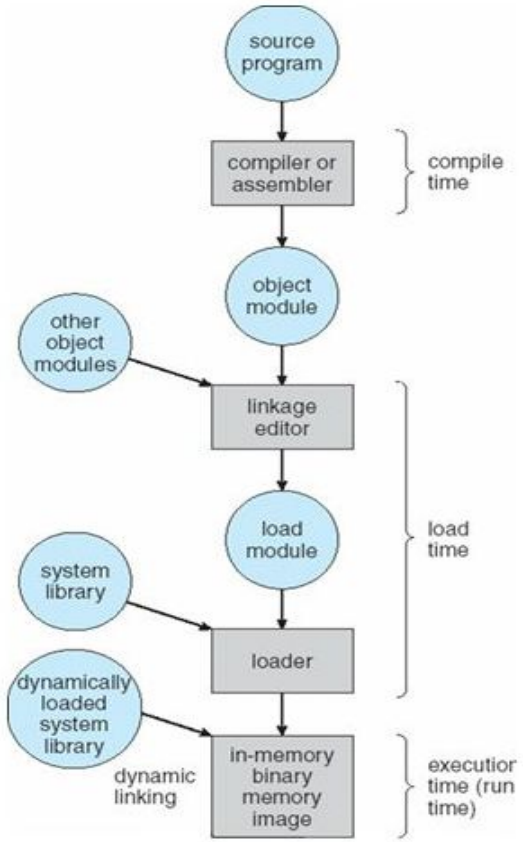
- The set of all logical addresses generated by a program is referred to as a **Logical Address Space (LAS).** The set of all physical addresses corresponding to these logical addresses is referred to as a **Physical Address Space (PAS)**

Address Binding of Data and Instructions in the memory

Address Binding

Address binding involves converting a variable in the source program to a physical address in memory. This happens in multiple stages:

1. **Compile time**: If the memory location of the program is known when it is compiled, then **absolute code** can be generated. If the location changes, the code must be recompiled.
2. **Load time**: If the memory location is not known at compile time, the compiler must generate **relocatable code**. Binding may happen when the program is loaded into memory.
3. **Execution time**: If the process can be moved during its execution from one memory segment to another, then binding may be delayed until run time. This is the most common method, though it requires hardware support.



Logical Address versus Physical Address

Basis for Comparison	Logical Address	Physical Address
Description	Logical address is the address that is generated by the central processing unit (CPU) in perspective of a program. Logical address can also be referred to as a virtual address.	Physical address is a location that exists in the memory; it allows accessing a particular storage cell in the main memory.
Address Space	Logical address space is the set of all logical addresses generated by CPU for a program.	Physical address space is the set of all physical address mapped to corresponding logical addresses.

Logical Address versus Physical Address

Basis for Comparison	Logical Address	Physical Address
Visibility	The logical address exists virtually and does not have a specific location to exist physically in memory unit hence it is also known as virtual address.	The physical address is an accessible physical location existing within the memory.
Generation	The logical address is generated by the central processing unit (CPU).	Physical address is computed and mapped by Memory Management Unit (MMU).

Logical Address versus Physical Address

Basis for Comparison	Logical Address	Physical Address
Use	The logical address helps to obtain the physical address.	The physical address helps to identify a location in the main memory.
Flexibility	The logical address is flexible hence will keep changing with the system from time to time when mapped to physical address	Physical address of the object always remains constant.

Logical Address versus Physical Address

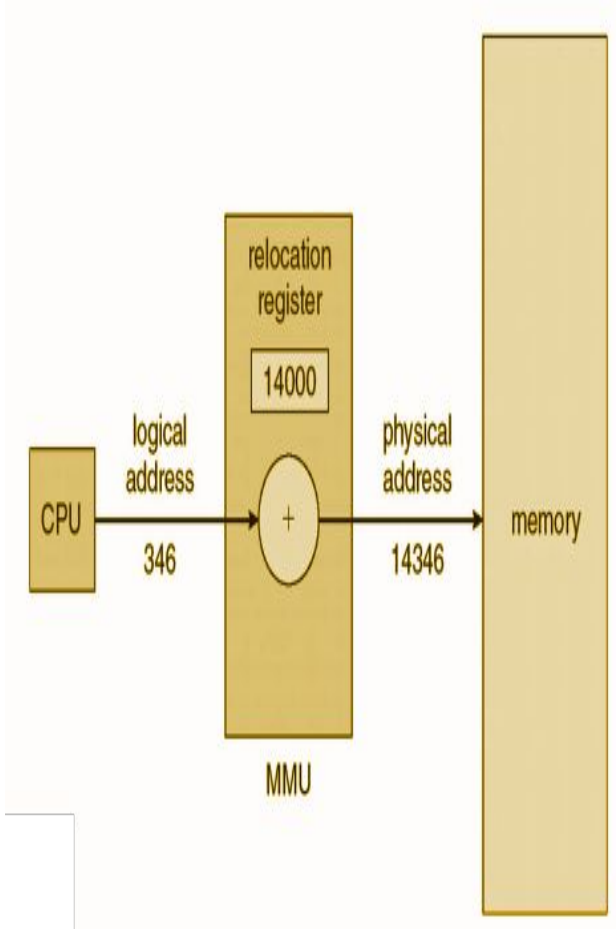
Basis for Comparison	Logical Address	Physical Address
User	The user program can use the logical address to access the physical address.	The user program does not have the ability to view the physical address directly.
Rebooting	Logic address mapped to physical address is erased when the system is rebooted.	Physical address is not affected when the system is rebooted.

Memory Management Unit (MMU)

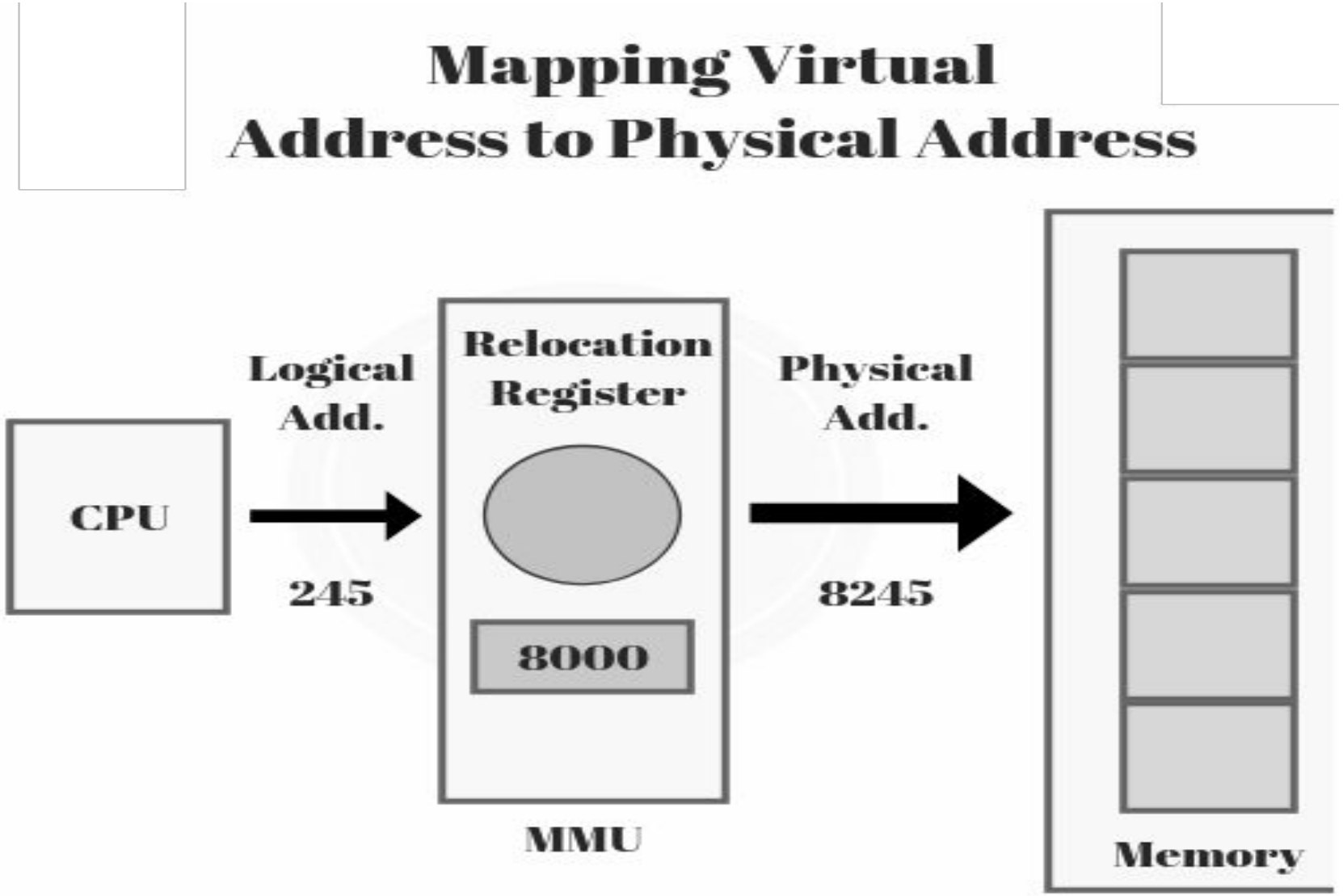
- Hardware device that at run time maps virtual to physical address
- Many methods possible, covered in the rest of this chapter
- To start, consider simple scheme where the value in the relocation register is added to every address generated by a user process at the time it is sent to memory
- Base register is now called relocation register
- MS-DOS on Intel 80x86 used 4 relocation registers
- 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

Dynamic Allocation using Relocation Register

- Routine is not loaded until it is called
- Better memory-space utilization; unused routine is never loaded
- All routines kept on disk in relocatable load format
- Useful when large amounts of code are needed to handle infrequently occurring cases
- No special support from the operating system is required
- Implemented through program design
- OS can help by providing libraries to implement dynamic loading



Dynamic Allocation using Relocation Register



THANK YOU

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

nitin.pujari@pes.edu

For Course Deliverables by the Anchor Faculty click on www.pesuacademy.com