CS & IT ENGINEERING

Operating System

Memory Management (Part - 01)

Revision

Lecture No.- 09

By- Dr. Khaleel Khan Sir

Recap of Previous Lecture

Topic Deadlocks

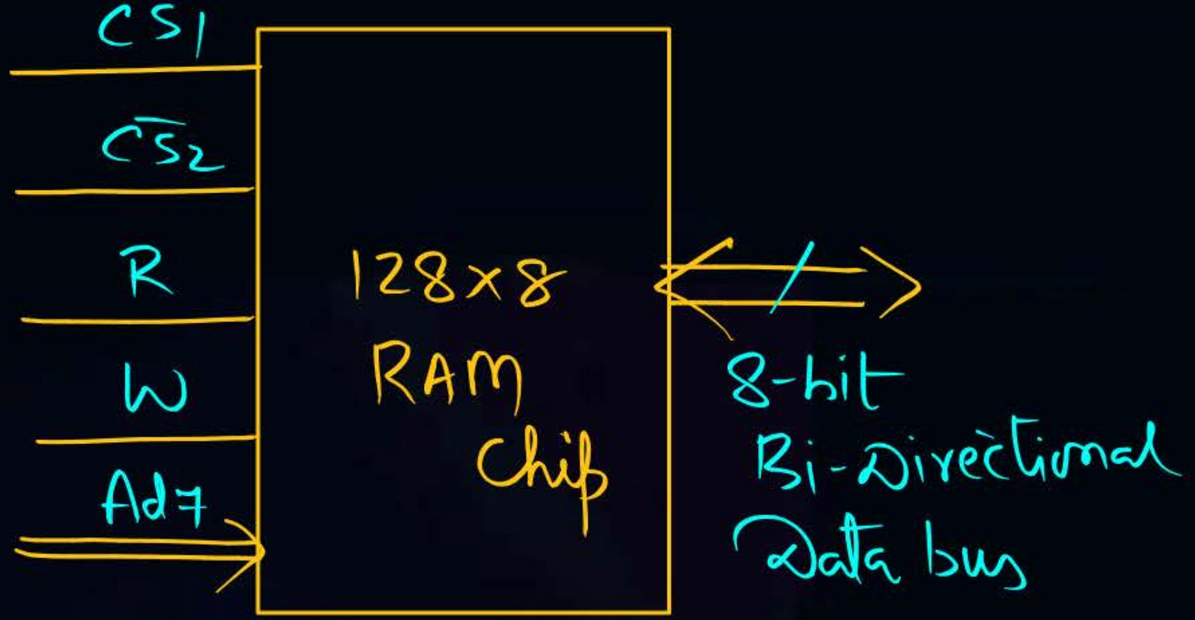
Topics to be Covered

- Topic: Background
- Topic: Protection
- Topic: Hardware Address Protection
- Topic: Paging Hardware

Topic : 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 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

Mem ⟹ RAM chips



Mem. Specification :

No. of words (N) × width (Size) of word (m)

$N \times m$

⟹ Address bits (n)

$N \propto n \propto m$

$N = 2^n$ words/Bytes

$n = \lceil \log_2 N \rceil$ bits

$\Rightarrow$ $n = 22$ bits ; $m = 32$ bits $(4B)$

$N_W = 2^{22} = 4MW$ (Word-view)

$1W = 4B$

$N_{Bytes} = 4M \times 4B = 16MB$ (Byte View) : In practice

$N_{Bits} = 16M \times 8$ bits (Bit-view)

$= 128M$ bits

$n = 45$ bits

Memory has $32T$ bits : If it is divided into words of size 128 bits

$$n_W = 38 \text{ bits} \Leftarrow N_W = \frac{32T \text{ bits}}{128 \text{ bits}} = \frac{2^{45}}{2^7} = 2^{38} = 256GW \checkmark$$

$$n_{By} = 42 \text{ bits} \Leftarrow N_B = \frac{32T \text{ bits}}{8 \text{ bits}} = 4TB \checkmark$$

8 bits

1W = 2 bits

$N_W = \frac{8}{2} = 4W \checkmark$

Memory Interconnection to CPU (Memory Interfacing)

< organization of Higher Capacity Memory, using Lower Denom. chips >

128 B

$\Rightarrow$ using (128×8) RAM chip, organize Mem of Capacity (512 Bytes);

Q1) No. of chips :
Q2) Decoder Size

$$\frac{\text{Mem desired}}{\text{chip Size}} : \frac{512\,B}{128} = 4$$

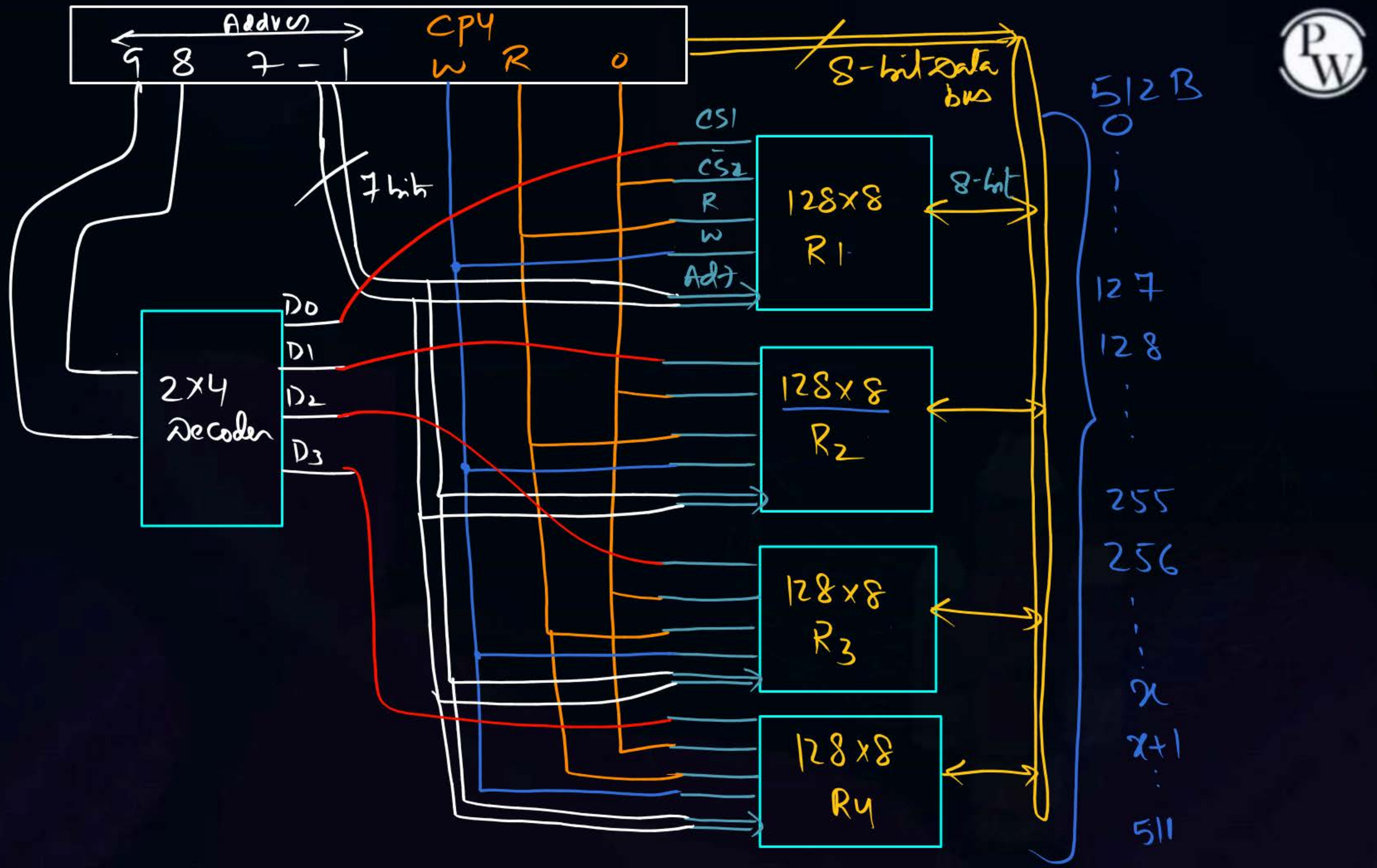
(i) Vertical organization
(word size is same)

Addressing

- Chip: 7 bits of Address
- Mem (512B): 9 bits (CPU)
  - 2 bits (MSB) (Chip Select)
  - 7 bits (LSB) (word_Select)

RAM : 1 GB : (4 GB) RAM

- 1 Single chip of Size 4GB
- 

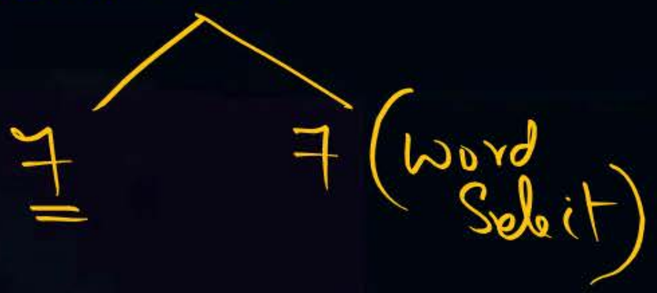

Addres
9 8 7 - 1
CPU
W R 0
8-bit data bus
512 B
0
1
127
128
255
256
x
x+1
511
7 bit
CS1
CS2
R
W
Adr
8-bit
128x8
R1
128x8
R2
128x8
R3
128x8
R4
2x4
Decoder
D0
D1
D2
D3

Q2) Using 128 × 8 RAM chip ⟹ 16 KB;

$$\text{chips} = \frac{16\,KB}{128\,B} = 2^{14}/2^{7} = 2^{7} = 128 \text{ chips}$$

$$\text{decoder} = 7 \times 128$$

Address: 14 bit

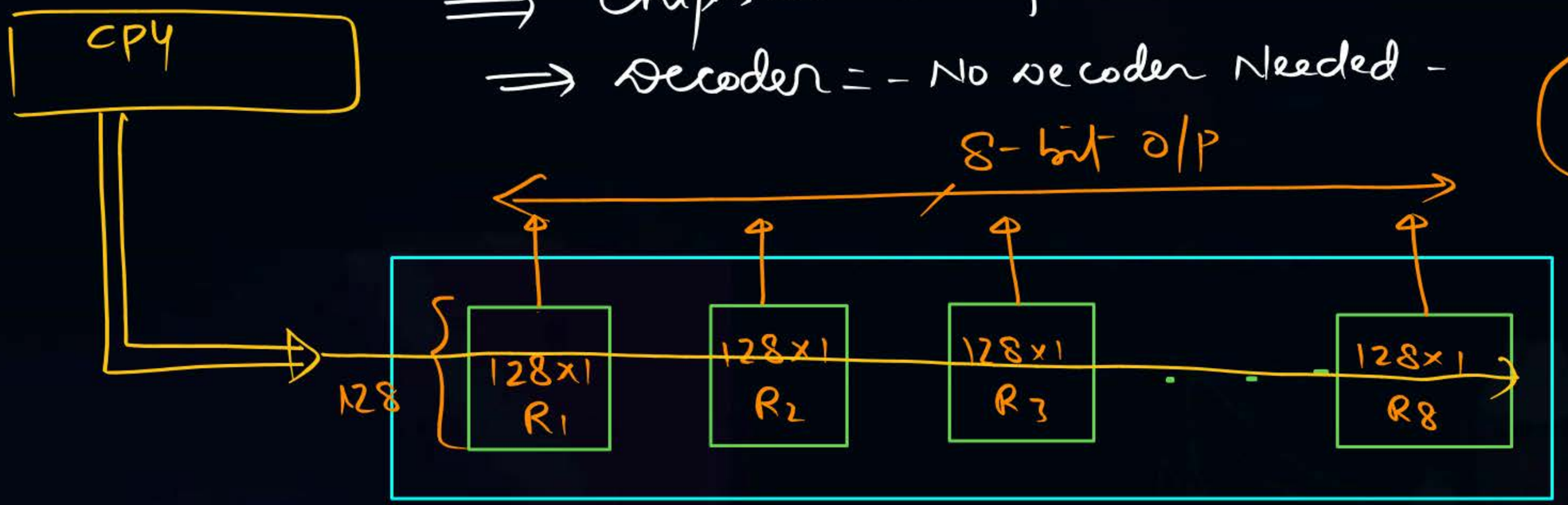


II: Using $\underline{\underline{128}} \times \underline{1}$ RAM chip, organize Mem of Capacity 128 B (128×8)

$\Rightarrow$ chips = 8 chips

$\Rightarrow$ Decoder = - No decoder Needed -

Horizontal orgnz.



III : Matrix organization :

Q) using 128×1 RAM chip, organize Mem of Capacity 16KB ;

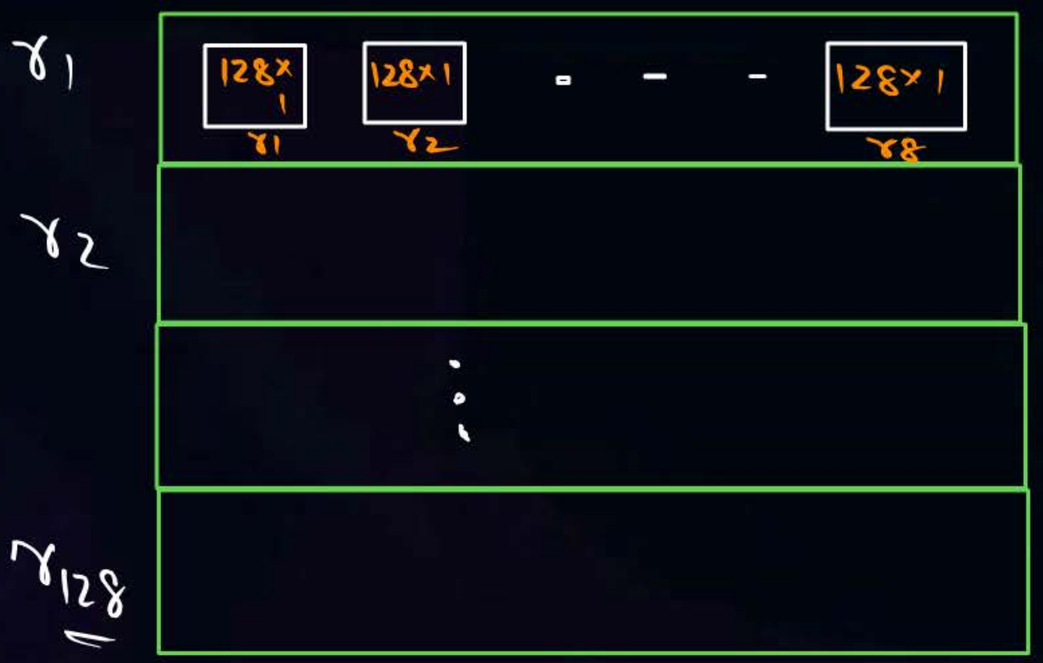
(i) chips =

(ii) decoder =

16KB
↓
14 bits

ch / Row	ws
7	7

7×128



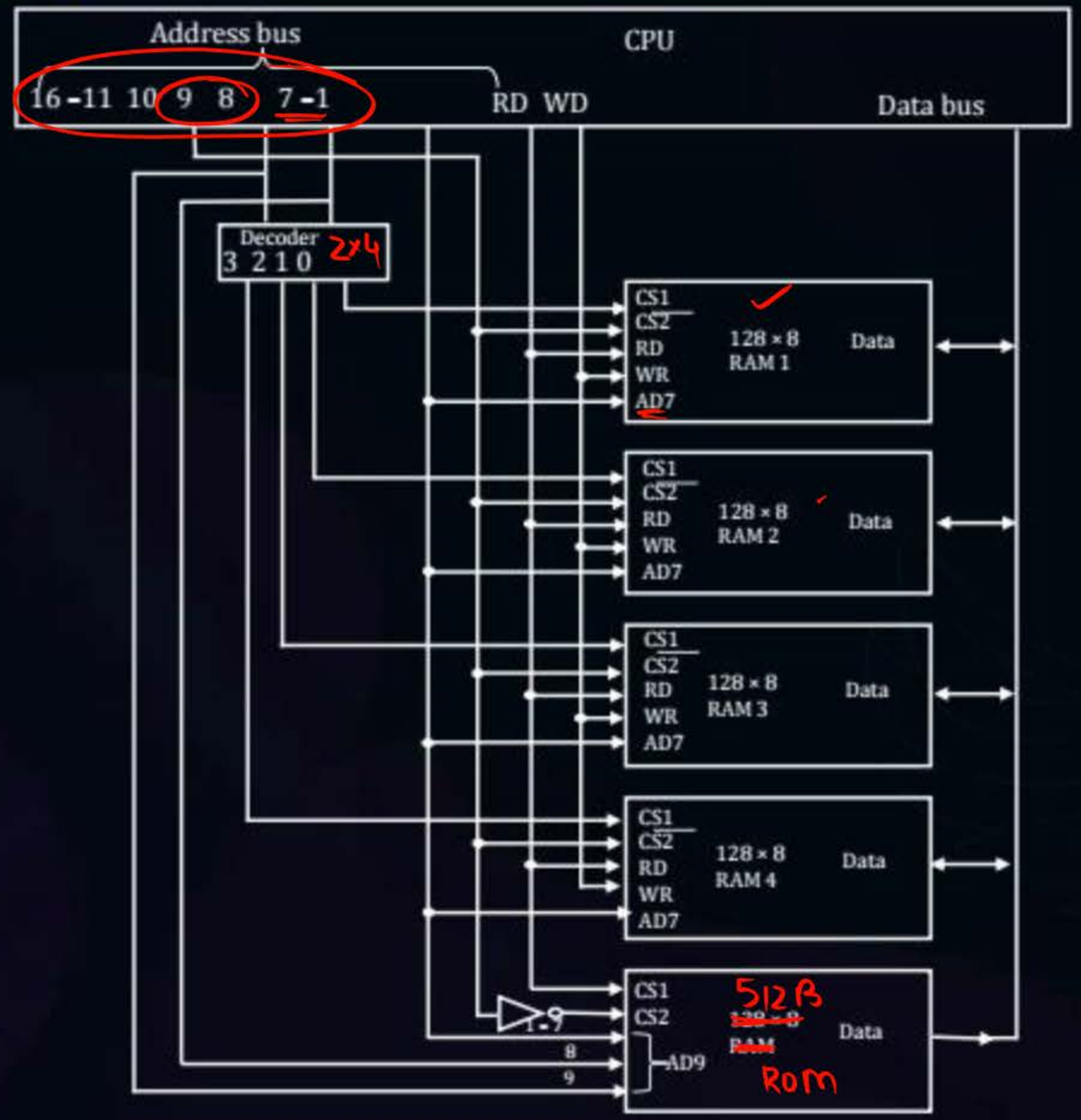
1 Row (128×1 / 8 chips) ⟹ 128 B

? ⟸ 16KB

No. of Rows = 16KB / 128B = 128 rows (8 chips)

Total chips = 128×8 = 1024
= 1K

Topic : Interconnection to CPU



5121 3 q RAM

[MCQ]

[GATE : 2021]

#Q. A RAM chip has a capacity of 1024 words of 8 bits each (1K × 8). The number of 2 × 4 decoders with enable line needed to construct a 16K × 16 RAM from 1K × 8 RAM is

(A) 4

(B) 5

(C) 6

(D) 7

(1024×8) — 10

No. of chips $= 16 \times 2 = 32$;

decoder : 4×16

Using $(1K \times 8)$ RAM chip

$(16K \times 16)$: Memory size — 14

14 bits

4 — 10

Chips =

1 Row ? $\left(\frac{2 \text{ chips}}{1K \times 8}\right) \Longrightarrow 1K \times 16$

$16K \times 16 \Longleftarrow$

No. of rows $= 16K \times 16 / 1K \times 16 = 16$

[MCQ]

[GATE : ~~2021~~]

#Q. How many ~~32K × 1~~ RAM chips are needed to provide a memory capacity of 256K bytes?

32K × 2

Matrix

A 8

B 32

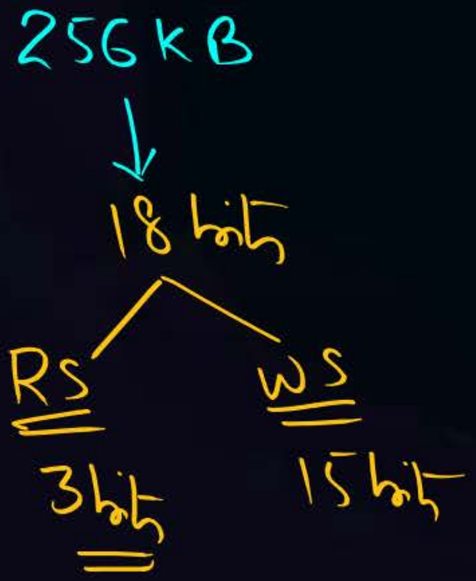
C 64 ✓

D 128

Decoder : 3 × 8

1 Row (8 chips, 32K × 1) ⟹ 32KB

⟸ 256KB

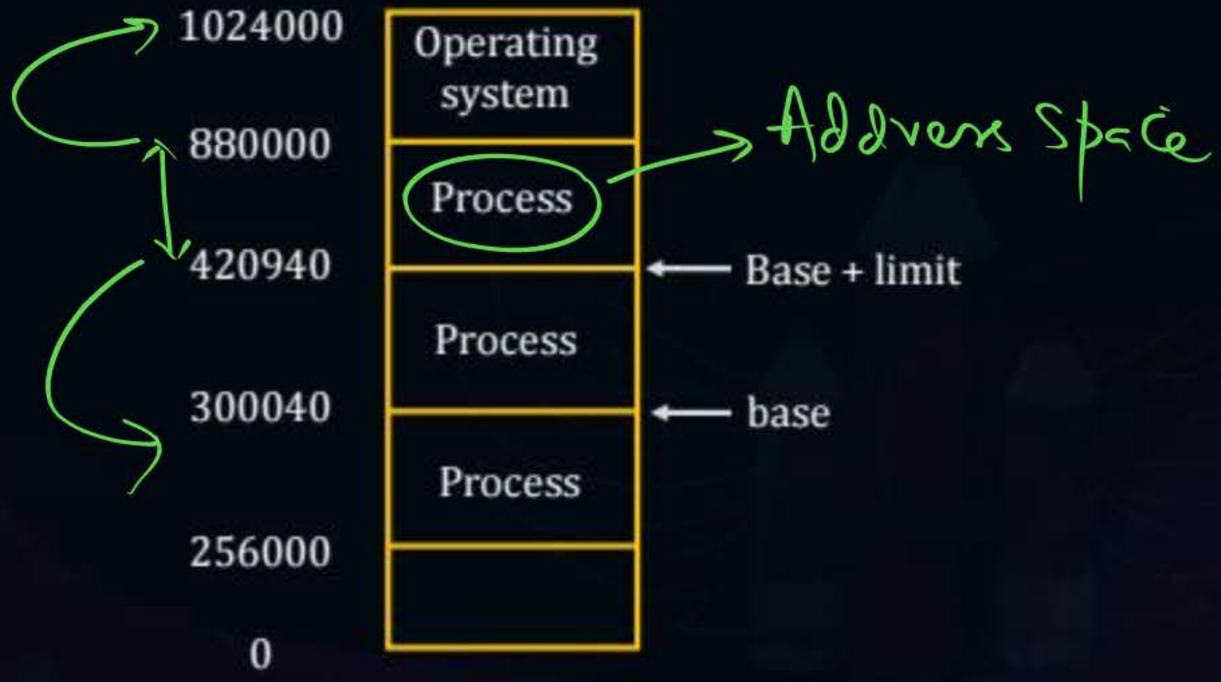


No. of rows = $\frac{256KB}{32KB}$

$= \frac{2^{18}}{2^{15}} = 2^3 = 8$

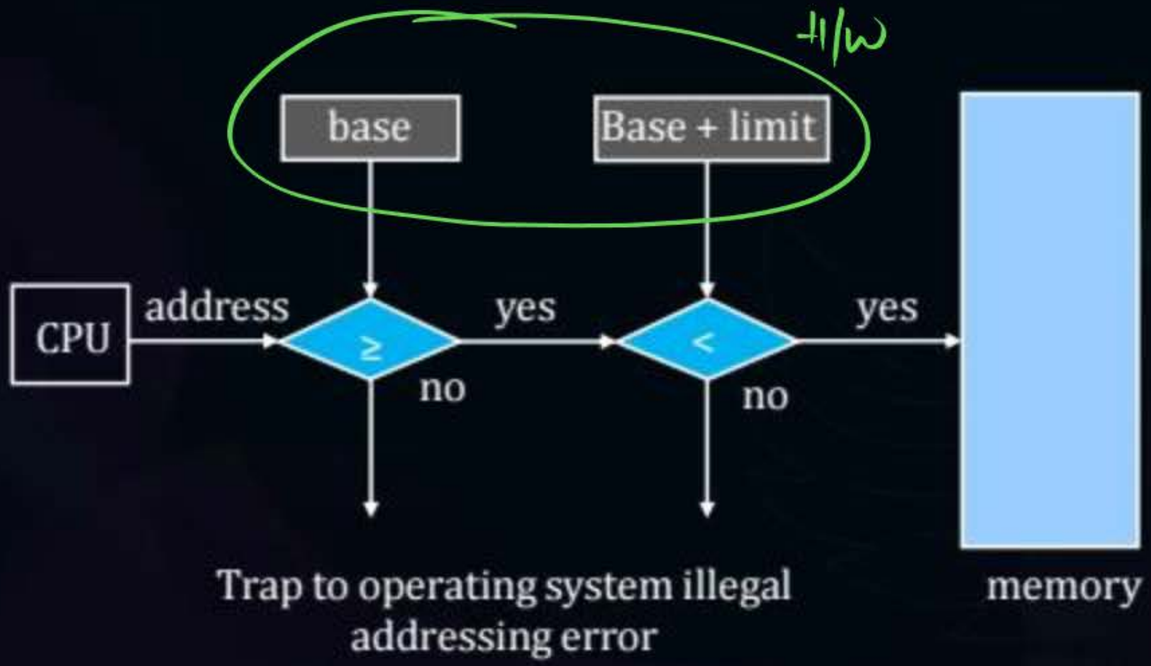
Topic : Protection

- 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** define the logical address space of a process



Topic : 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



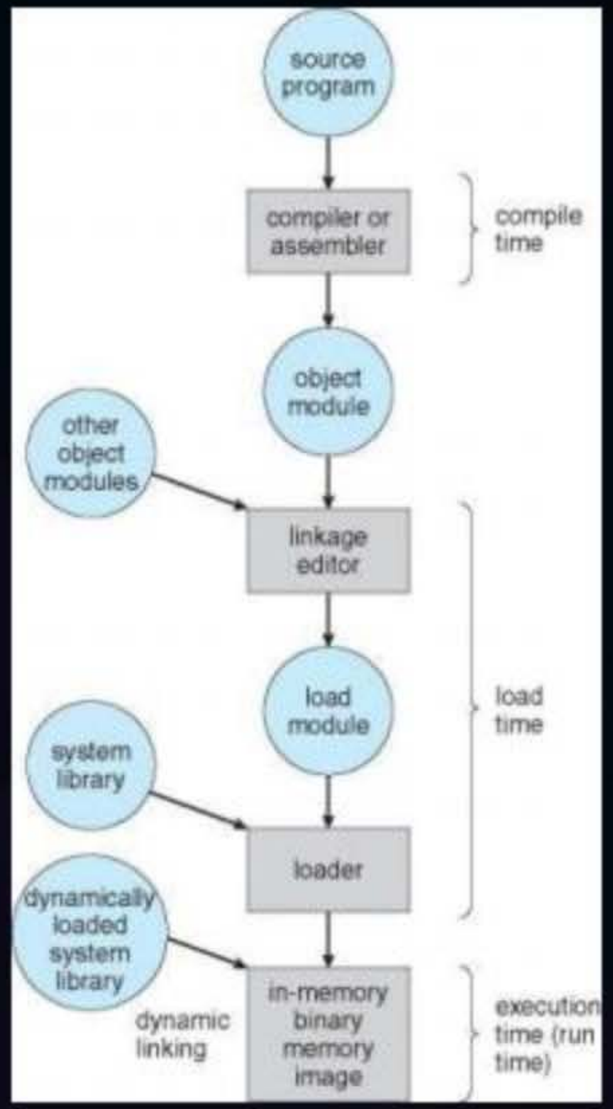
Topic : Address Binding

- 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
  - 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

Topic : Binding of Instructions and Data to Memory

- Address binding of instructions and data to memory addresses can happen at three different stages
  - **Compile time:** If memory location known a priori, **absolute code** can be generated; must recompile code if starting location changes
  - **Load time:** Must generate **relocatable code** if memory location is not known at compile time
  - **Execution time:** Binding delayed until run time if the process can be moved during its execution from one memory segment to another
    - ✓ Need hardware support for address maps (e.g., base and limit registers)

Topic : Multistep Processing of a User Program



CPU

L·A / V·A

P·A (RAM)

(i) C·T / L·T ~ L·A ≃ P·A

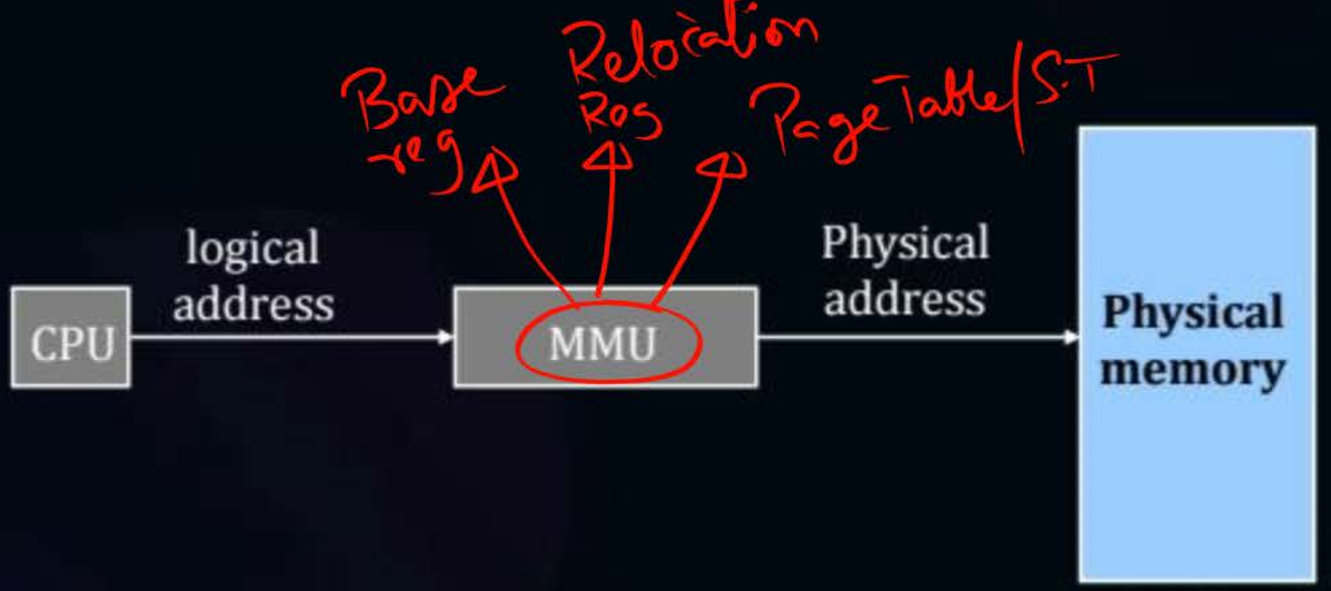
(ii) R·T ~ L·A ≠ P·A

Topic : Logical Vs. Physical Address Space

- 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 and physical addresses are the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme
- **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

Topic : Memory – Management Unit (MMU)

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



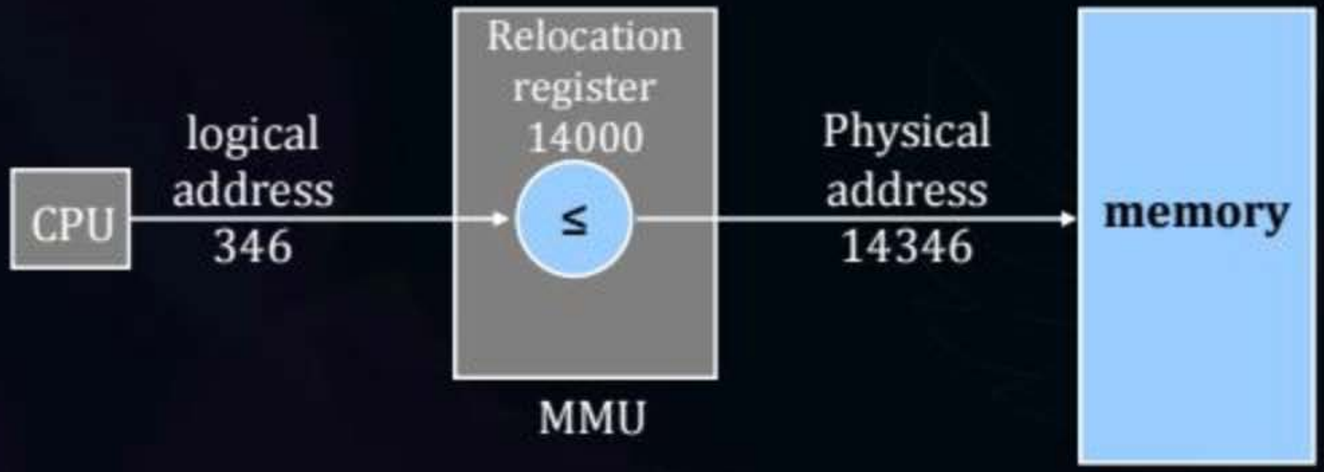
- Many methods possible, covered in the rest of this chapter

Topic : Memory – Management Unit (Cont.)

- 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

Topic : Memory – Management Unit (Cont.)

- 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



Topic : Dynamic Loading

- The entire program does need to be in memory to execute
- 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

Topic : Dynamic Linking

- **Static linking** – system libraries and program code combined by the loader into the binary program image
- Dynamic linking –linking postponed until execution time
- Small piece of code, **stub**, used to locate the appropriate memory-resident library routine
- Stub replaces itself with the address of the routine, and executes the routine
- Operating system checks if routine is in processes' memory address
  - If not in address space, add to address space
- Dynamic linking is particularly useful for libraries
- System also known as **shared libraries**
- Consider applicability to patching system libraries
  - Versioning may be needed

Topic : Contiguous Allocation

- 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

Mem. Mgmnt Techniques

- Contiguous <Centralized>
  - (i) Partitions
    - Fixed
    - Variable → (Buddy System) (Power of 2 Allocator)
  - Old / Primitive
    - (ii) overlays
    - (iii) Swapping
- Non-Contiguous <Decentralized / Distributed>
  - Paging
    - Demand Paging <V.M>
  - Segmentation
  - Seg-Paging

Functions

1) Allocation
2) Protection
3) Address Transl.
4) Free Space Mgmnt
5) Deallocation

Goals

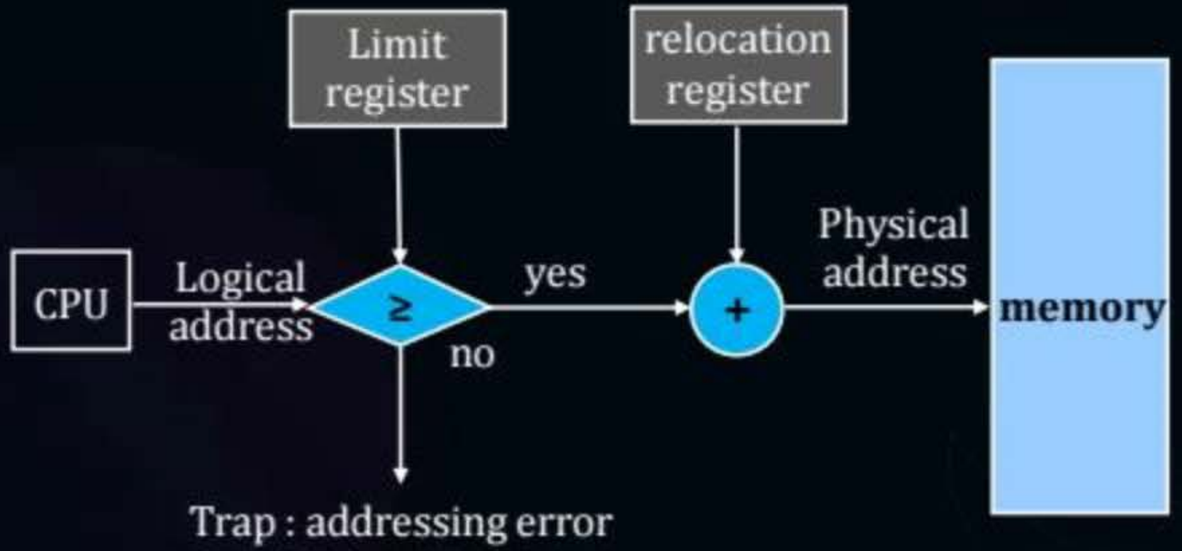
⟹ efficient utiliz. of Mem
(less wastage)
↳ Fragmentation
I.F E.F

⟹ Manage Exec. of
larger Programs in Small
Mem. Areas (V.M)

Topic : Contiguous Allocation (cont.)

- 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

Topic : Hardware Support for Relocation and Limit Registers



Partitions

- Fixed (M.F.T)
- Variable

(i) Ltd. degree of M.Pr. ✓

(ii) Int. Frag ✓

(iii) No Ext. Frag.

(iv) Max. Process Size is ltd ✓

(v) Best Fit is Superior (Less I.F)

Alloc. Policy

→ F.F

→ B.F

→ W.F

→ N.F

2 mins Summary

Topic One : RAM chips

Topic Two : Addressing vs Capacity

Topic Three : Memory Interfacing

Topic Four : Techniques Taxonomy

55

Topic Five : Fixed Partitions

THANK - YOU