Chapter: Memory Management

Memory

Memory consists of large array of words or bytes, each with its own address.

CPU fetches instructions from the memory as per the value of program counter.

Memory unit only sees the addresses and is not concerned with how they are generated

Memory

Computer has 2 types of memory:

a) **Main Memory:** temporarily stores data and instructions executed by the computer. CPU retrieve instructions from main memory and executes it.

b) **Secondary Memory:** directly is not accessed by CPU. It is an external storage.

Address Binding

Address Binding: Assigning an address to data or instruction.

Address binding is a process of generating address where the data/instruction is to be stored in memory.

3 types of address binding:

1. Compile time binding
2. Load time binding
3. Run time binding

Address Binding

Address Binding: Fixing a physical address to the logical address of a process address space

Compile time binding: it is known to compiler at compile time where a program will reside in physical memory (Main Mem.)

Load time binding: if program location in memory is unknown until run-time.

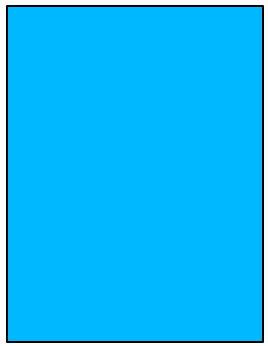
Execution/Run time binding: If the process can be moved during its execution from one memory segment to another, then binding must be delayed until run time. The absolute addresses are generated by hardware.

Address Space of process

•Make sure that each process has a separate memory space.

•Separate per-process memory space protects the processes from each other.

•To separate memory spaces, we need the ability to determine **the range of legal addresses that the process may access** and to ensure that the process can access only these legal addresses.



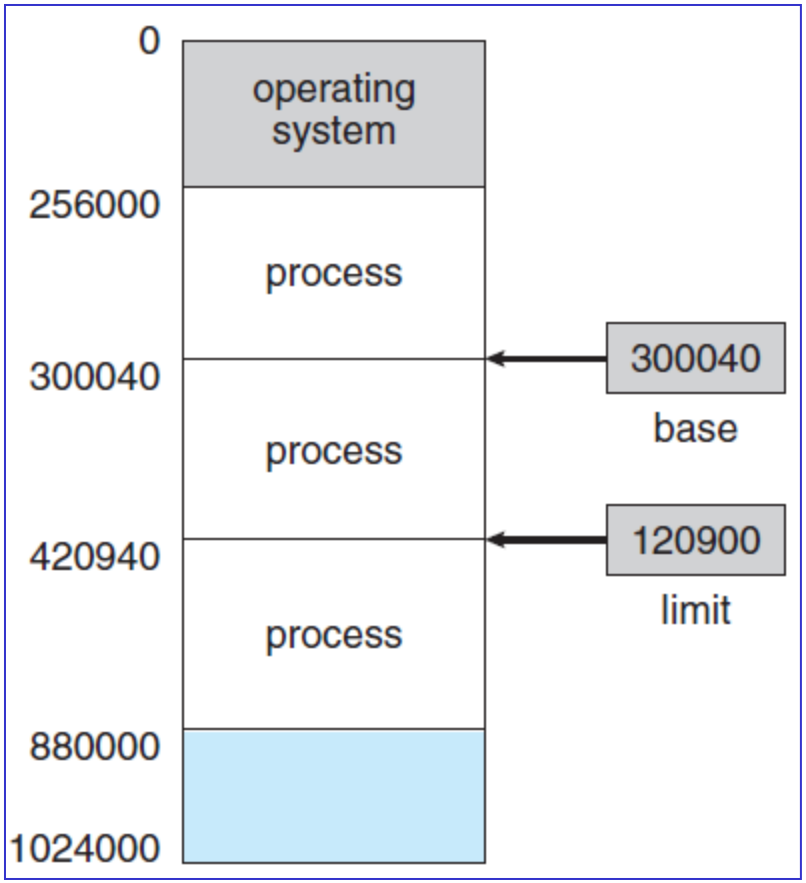
Address Space of process

We can provide this protection by using two registers, usually a base and a limit

Base register holds the smallest legal physical memory address

Limit register specifies the size of the range.

e.g. For example, if the base register holds 300040 and the limit register is 120900, then the program can legally access all addresses from 300040 through 420939 (inclusive)

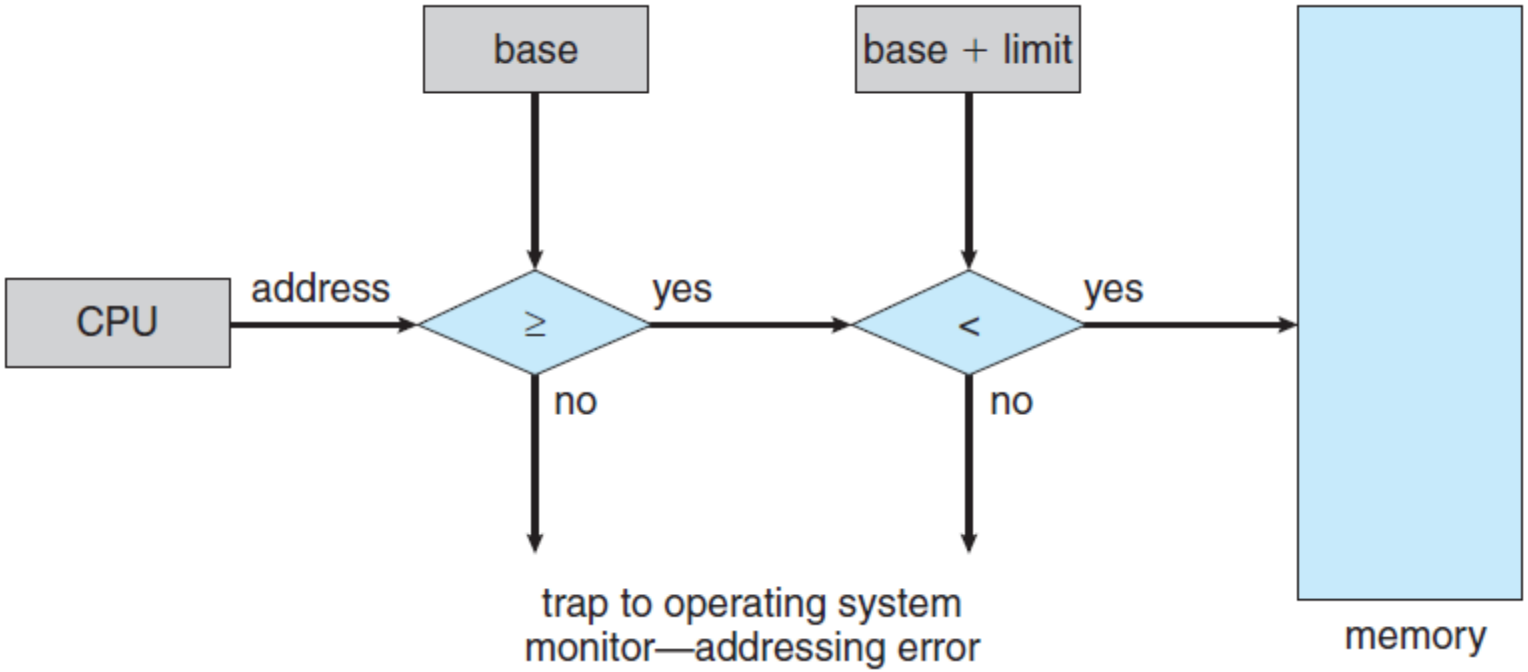


Protection

Protection of memory space is accomplished by having the CPU hardware compare every address generated in user mode with the registers.

Any **attempt by a program executing in user mode to access operating-system memory** or other users' memory **results in a trap to the operating system**

Protection



Base register and limit registers can only be updated by OS in kernel mode.

Logical and Physical Address Space

1. Logical Address or Virtual address:

a. The address generated by CPU.

b. Logical address is address of instruction / data as used by program at some time.

Set of all logical address generated by program is Logical address space

Logical and Physical Address Space

2. Physical Address:

a. It is the address used seen by memory management unit (MMU).

b. It refers to actual location in the main memory.

c. The user can never view physical address of program

Set of all logical address that a process actually occupies is Physical address space

The logical address is used like a reference, to access the physical address

MMU is used to map logical and physical addresses.

Logical and Physical Address Space

- The run-time mapping from logical (virtual) to physical addresses is done by a hardware device called the memory-management unit (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.

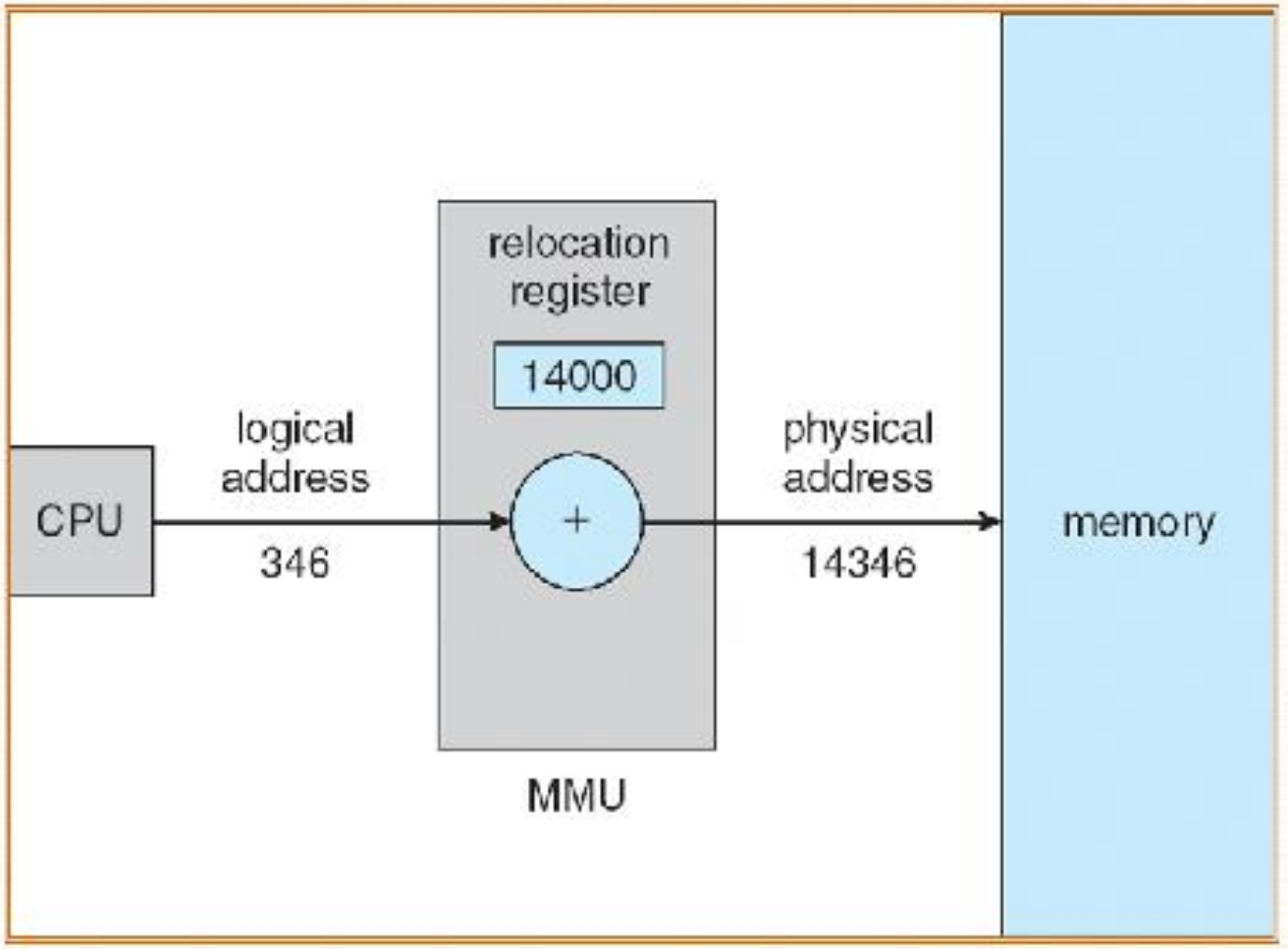
Logical and Physical Address Space

- Assume logical addresses (in the range 0 to max) and physical addresses.

- For User program CPU generates only logical addresses and thinks that the process runs in locations 0 to max.

- These logical addresses must be mapped to physical addresses before they are used by MMU

Logical and Physical Address Space



Dynamic Loading

- Entire program and its entire data should be in memory for process to execute.
- Therefore process size <= memory size
- Sol: Dynamic loading
  - Load a routine only when it is called

Swapping

Technique of removing a process from main memory and storing it into secondary memory, then bringing it back into main memory for continued execution.

It is a technique used in multiprogramming environments which have limited memory capacity.

Action of moving process out of main memory is called **Swap Out**

Action of moving process into main memory is called **Swap In**

Area on the disk where swapped out process are stored is known as Swap Space.

Swapping

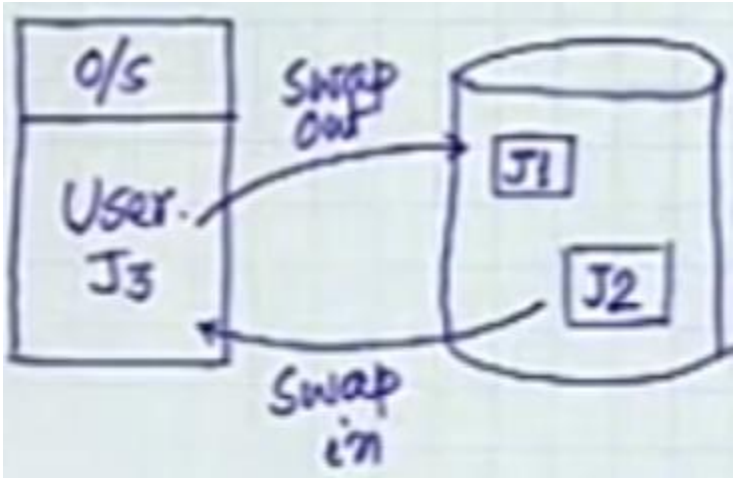
During Swap-Out foll. Things are checked:

While in Main Memory was user program modified?

Yes: write it back to Sec. Memory

No: Sec. Memory already has a copy of it, no need to write.

Overwrite J3 with new program. I.e bring new program

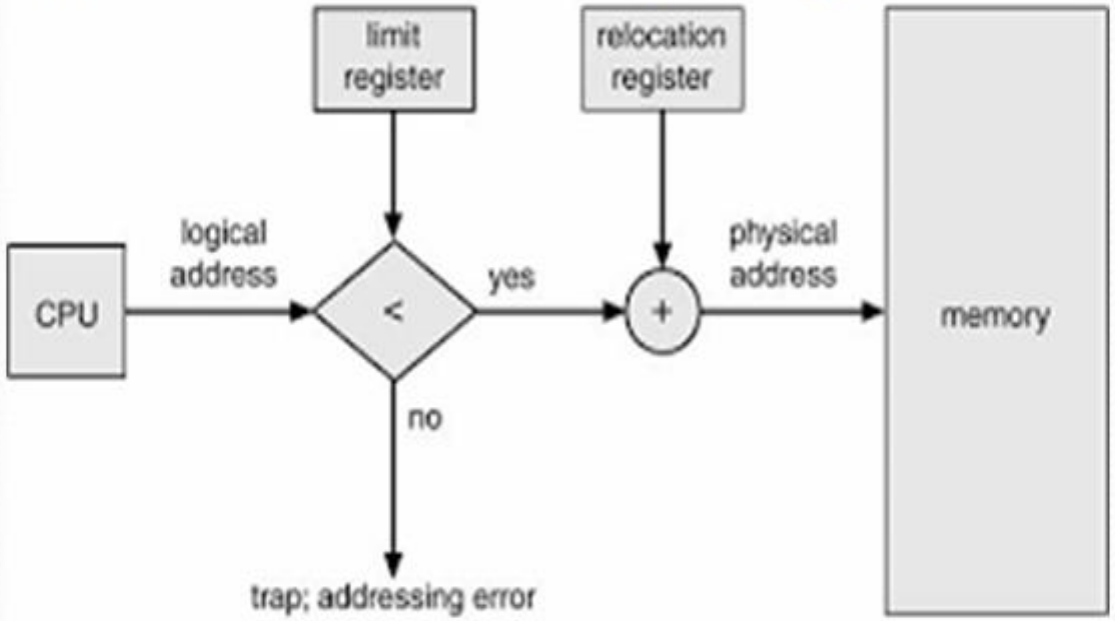


Memory Allocation Schemes

- Contiguous memory allocation
- Paging
- Segmentation

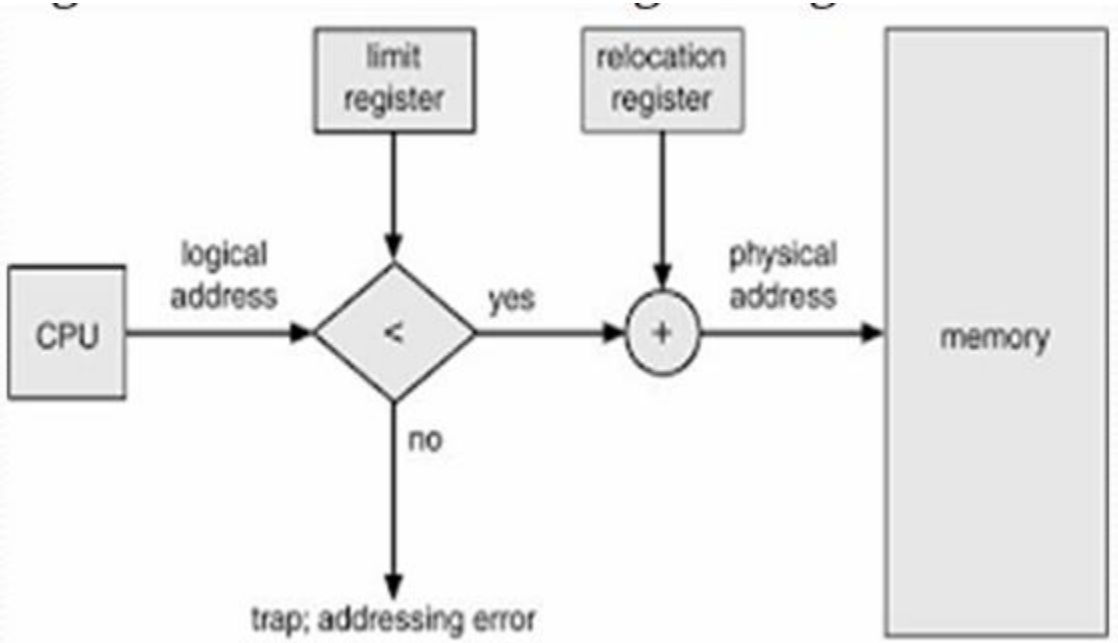
Contiguous File Allocation

- Each process is contained in **a single contiguous section** of memory.
- Memory mapping and protection
  - Relocation register – contains value of smallest physical address
  - Limit register – contains the range of logical addresses



Relocation Register contains base address of the process

Contiguous File Allocation



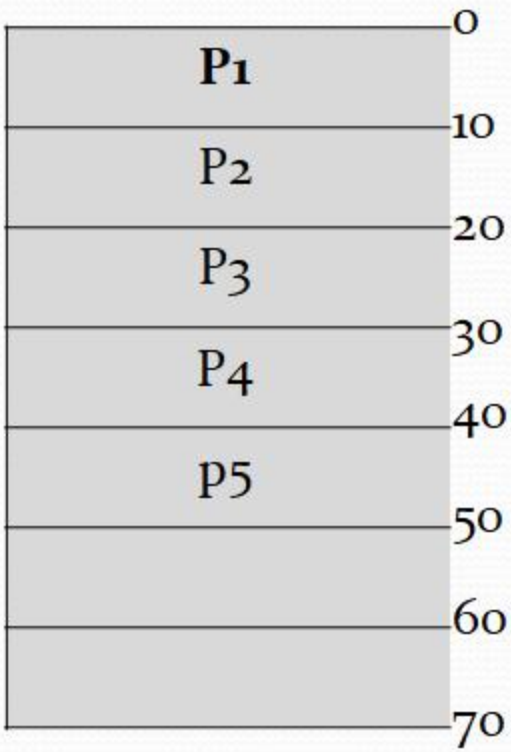
Problem

- Suppose that the value of relocation register is 100040 and limit register is 74600. What are the physical addresses for the following logical addresses?
  1. 70000
  2. 75000

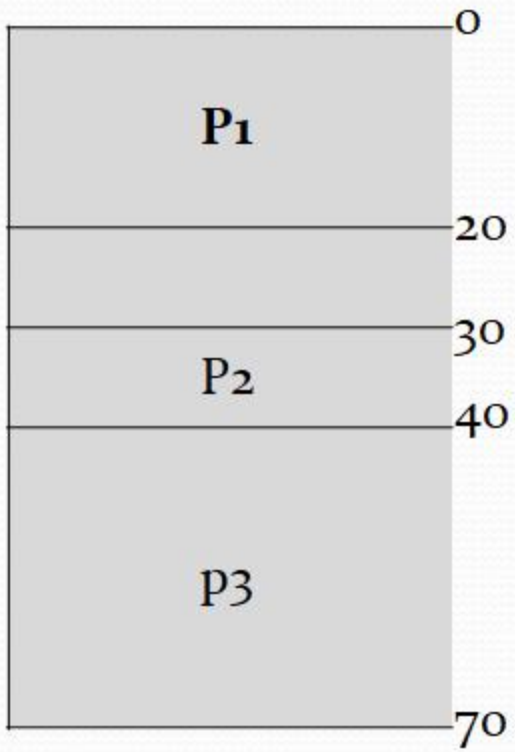
1. Check whether logical address < limit register value.
70000<74600. TRUE.
Physical address = logical address + relocation register value
i.e. Physical address = 70000+100040=170040
2. Check whether logical address < limit register value.
75000<74600. FALSE
Trap: addressing error

Partitions

Fixed partition



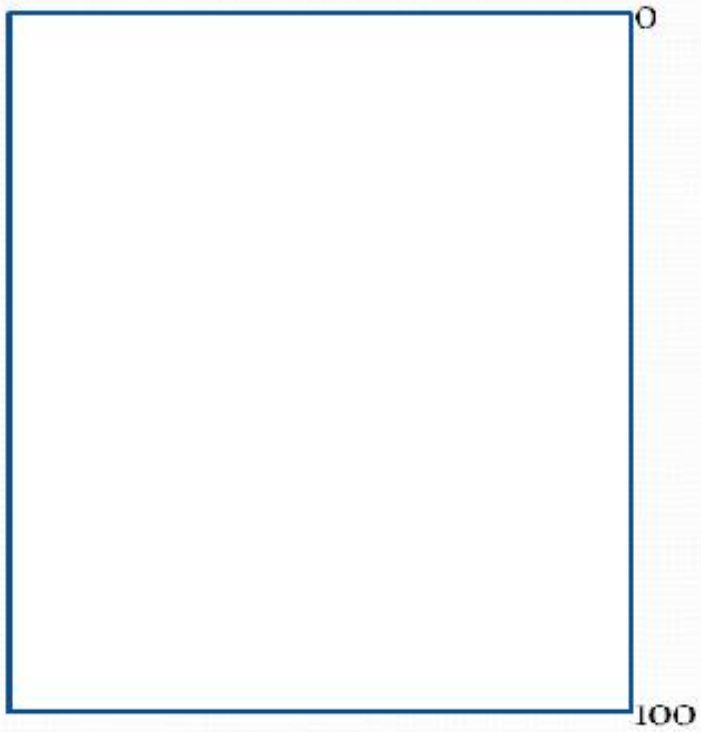
Variable partition



Variable partition working

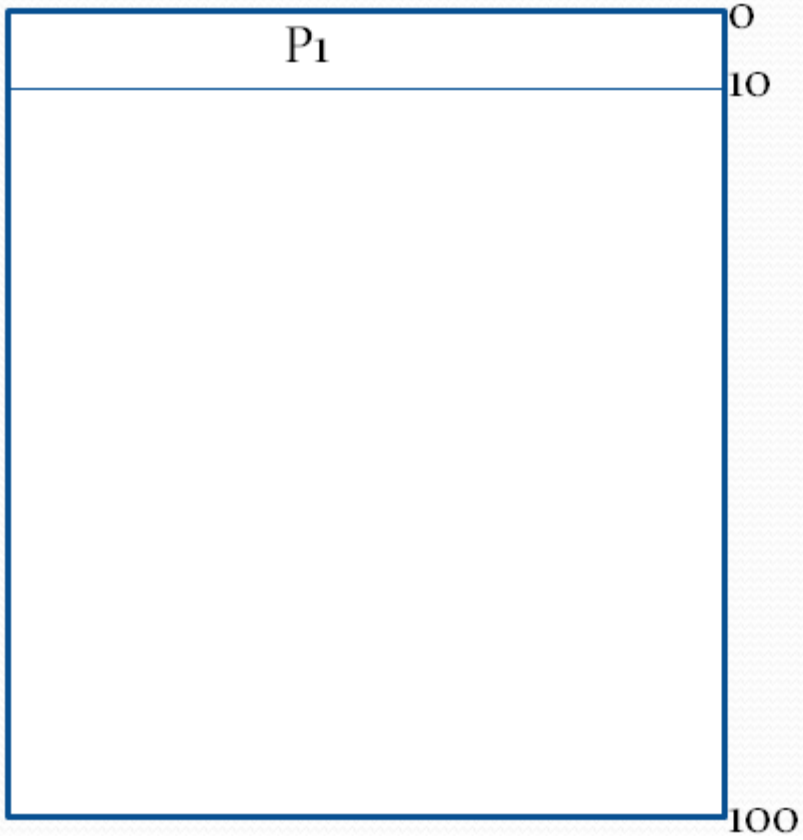
- Suppose total memory is 100Mb

P1 needs 10mb space

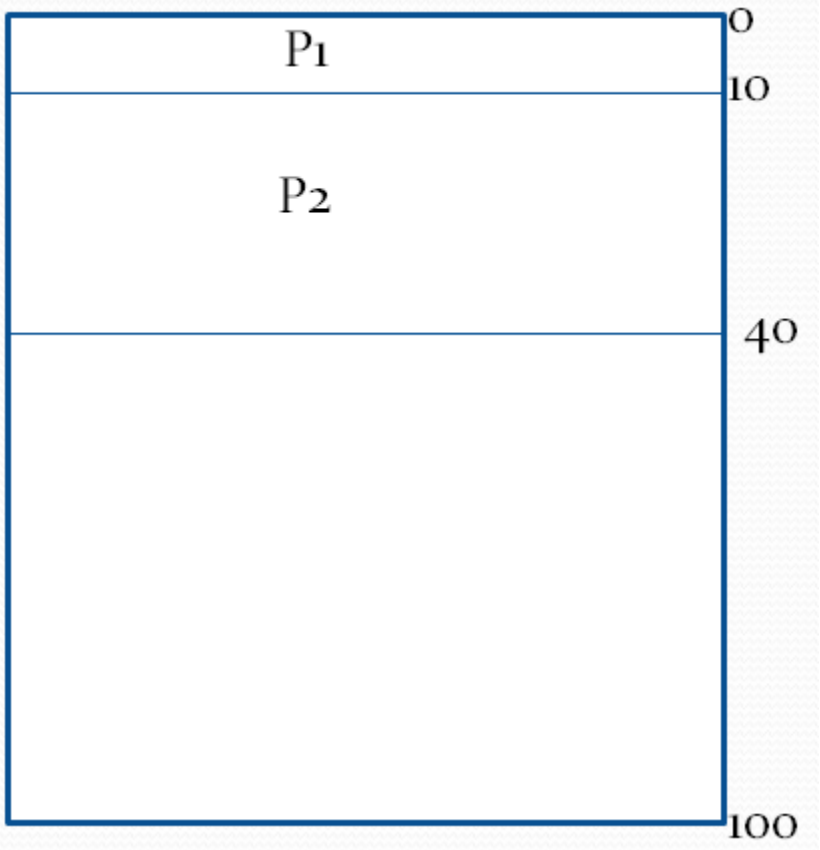


Memory	Status
0-100	Free

- Process P1 requires 10Mb space

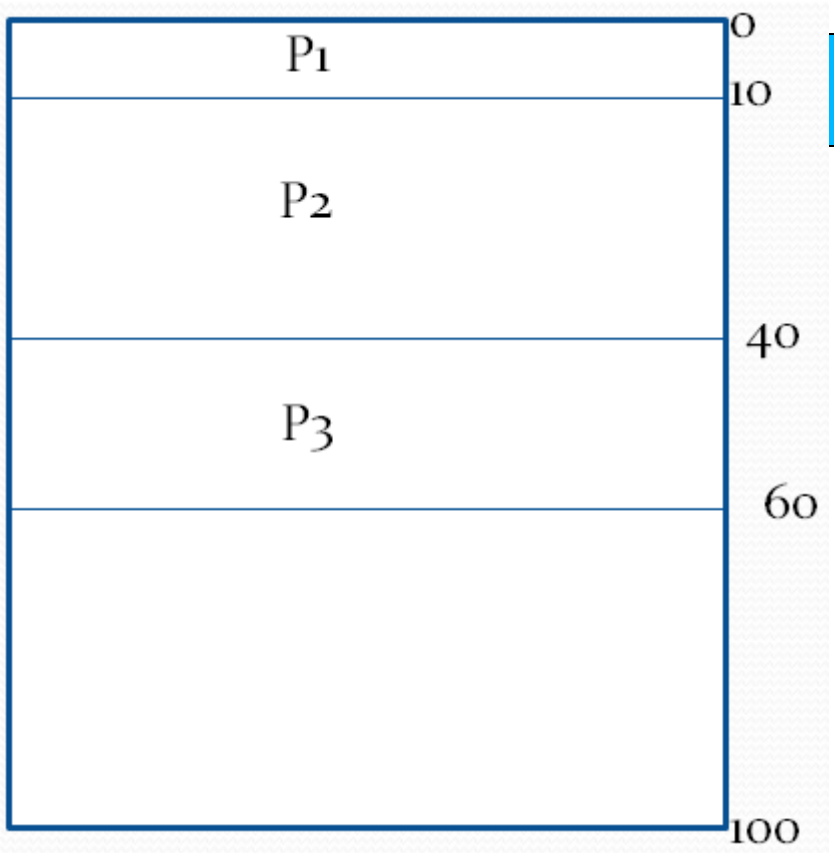


Memory	Status
0-9	P1
10-100	free



P2 needs 30mb space

Memory	Status
0-9	P1
10-39	P2
40-100	Free

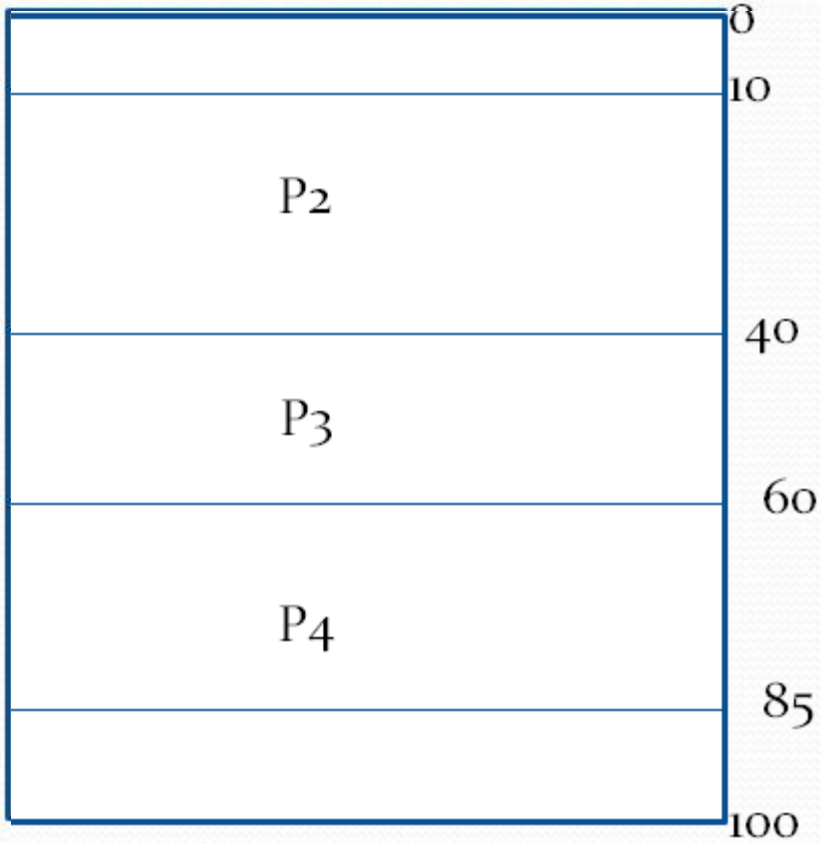


P3 needs 20mb space

Memory	Status
0-9	P1
10-39	P2
40-59	P3
60-100	Free

P1 finishes its execution
So space is free

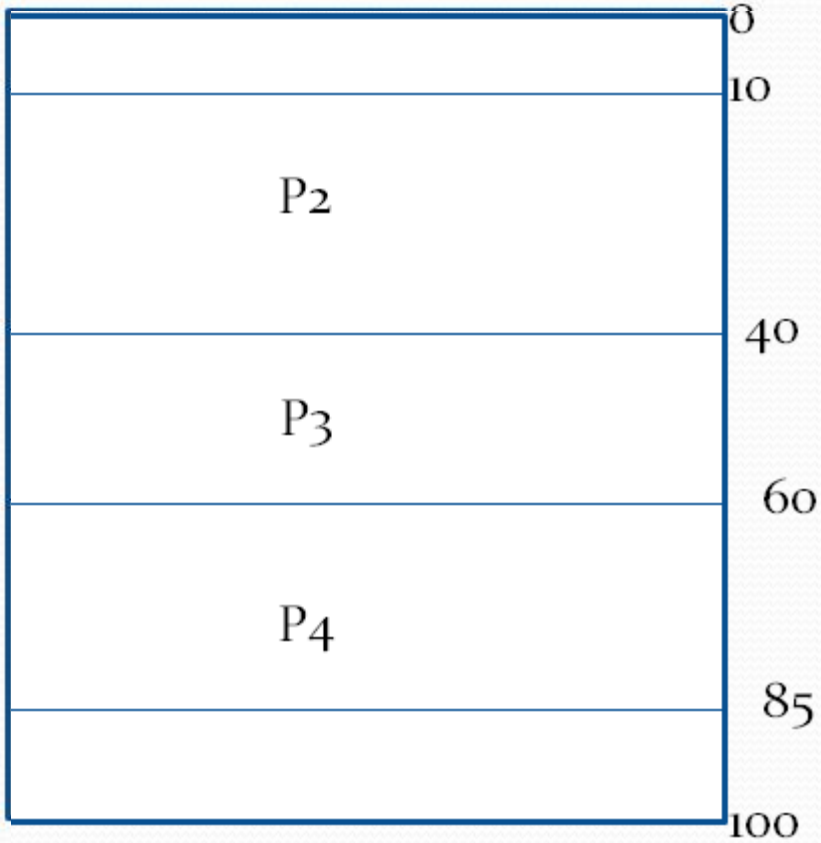
P4 need 25mb



Memory	Status
0-9	Free
10-39	P2
40-59	P3
60-84	P4
85-100	Free

P3 finishes its execution
So space is free

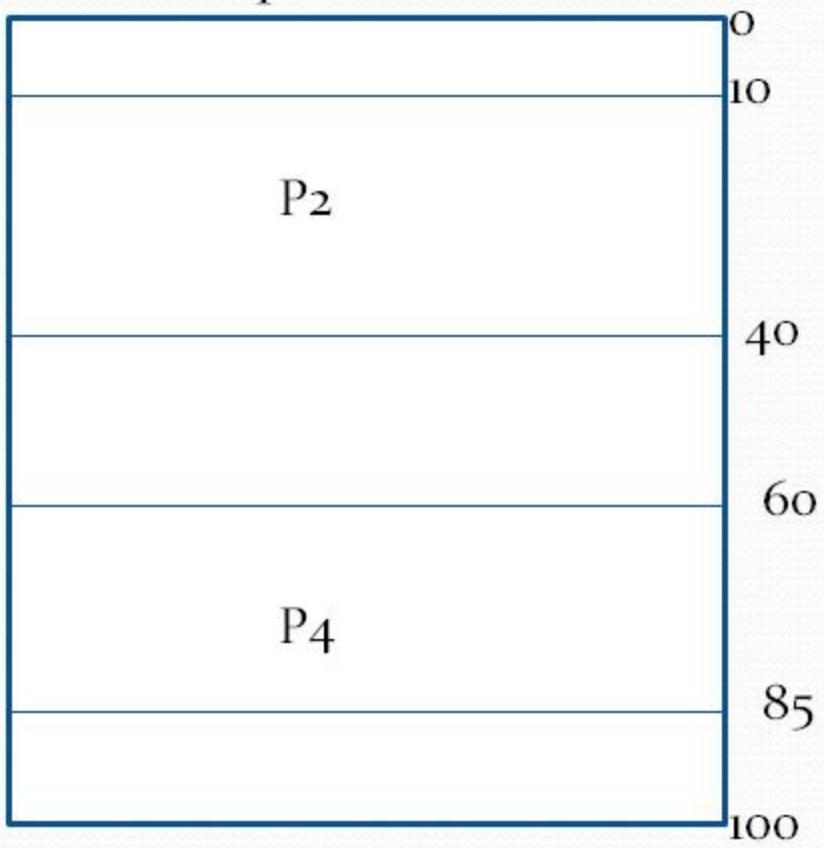
P5 need 25mb



Memory	Status
0-9	Free
10-39	P2
40-59	Free
60-84	P4
85-100	Free

This problem is called "External Fragmentation"

Can it be granted? No, because 25Mb is not free in a single slot(hole)



10 Mb free

20 Mb free

16 Mb free

Memory	Sta
0-9	Free
10-39	P2
40-59	Free
60-84	P4
85-100	Free

Total free space = 10+20+16=46Mb

External Fragmentation

- When there is enough total memory space to satisfy a request, but the available space is not contiguous.
- Solution
  - Compaction – shuffle the memory contents so as to place all free memory together in one large block.

Partition Allocation

One method of allocating contiguous memory is to **divide** all available **memory into equal sized partitions**, and to assign each process to their own partition.

Partition Allocation Algorithms:

a) First Fit:

- Checks all partitions serially
- When partition with size = or > encounters, it is allocated for storage.