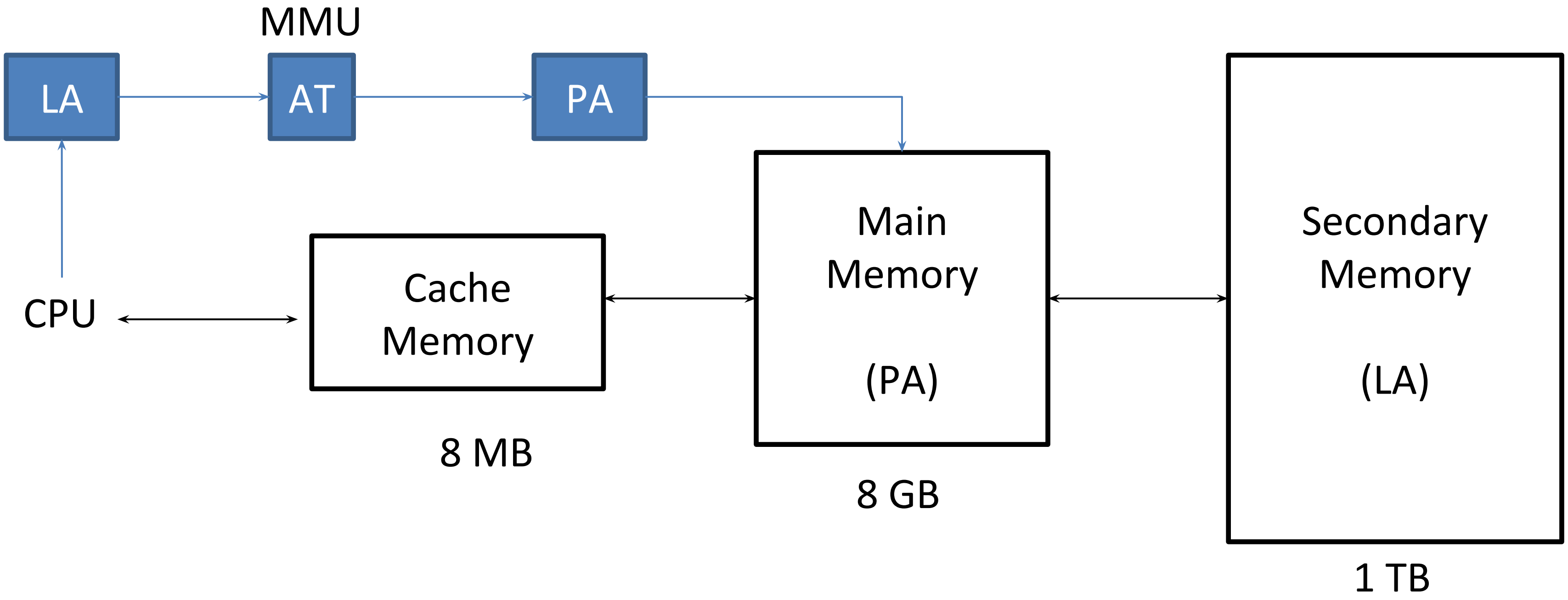
# Memory Management

- By Archana Vyas

# Memory Management

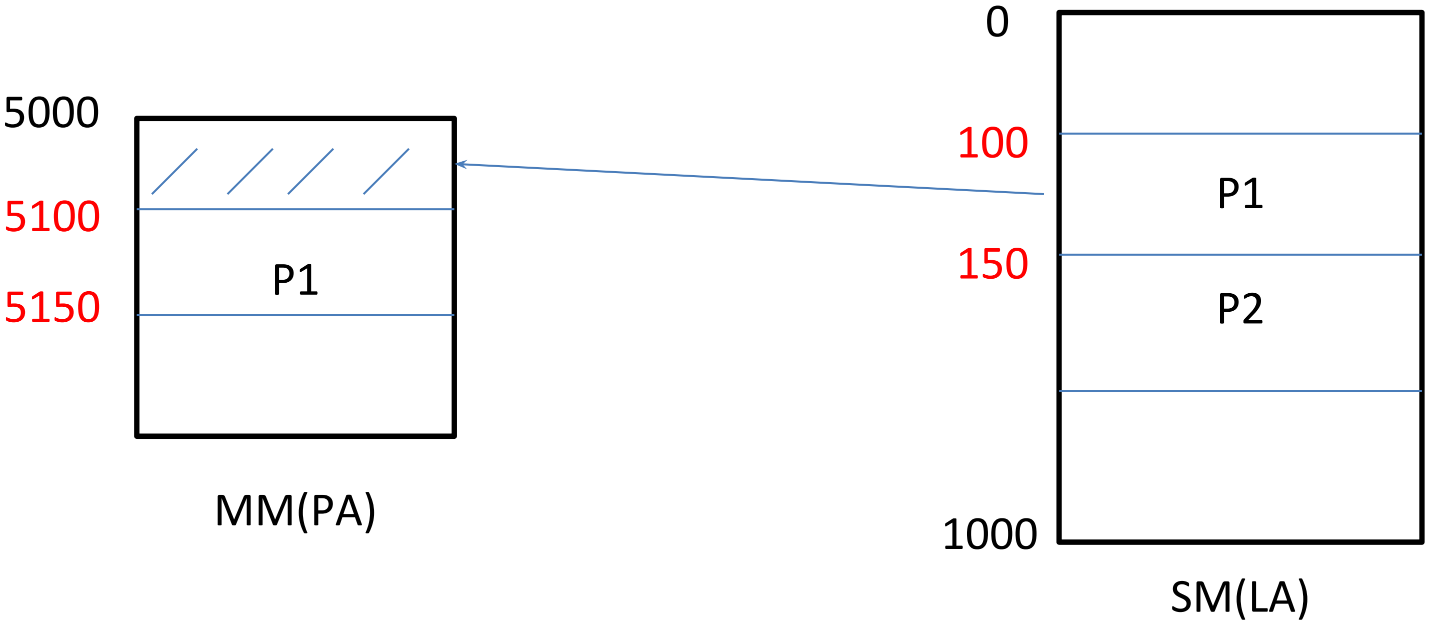
- We want memory’s
  - Size ⍰ large
  - Access time ⍰ low
  - Per unit cost ⍰ less
- But as size increases memory access time also increases



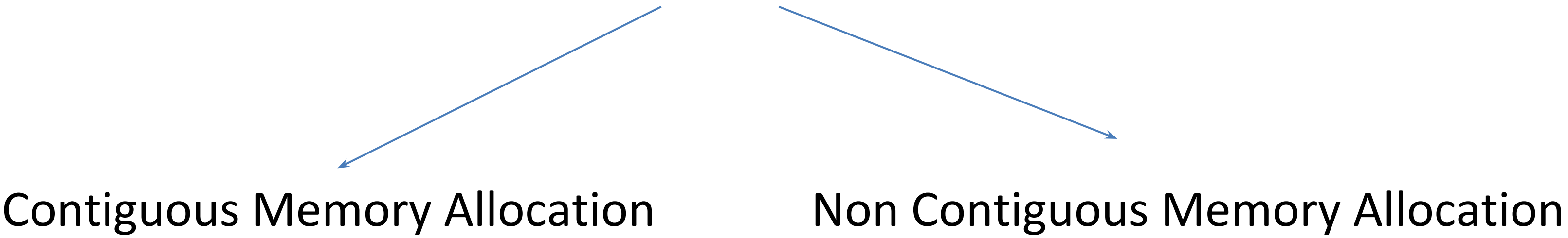
# Memory Management

- It works based on locality of reference.
- The process which is required to get executed is **loaded** from secondary memory to main memory.
- Loading from SM to MM needs **memory allocation**
  - Contiguous Memory Allocation
  - Non Contiguous Memory Allocation

# Memory Management



## Memory Allocation



Contiguous Memory Allocation

Non Contiguous Memory Allocation

# Contiguous Memory Allocation

- Ex:- Array

|  |  |  |  |  |
|---|---|---|---|---|
| 10 | 20 | 30 | 40 | 50 |

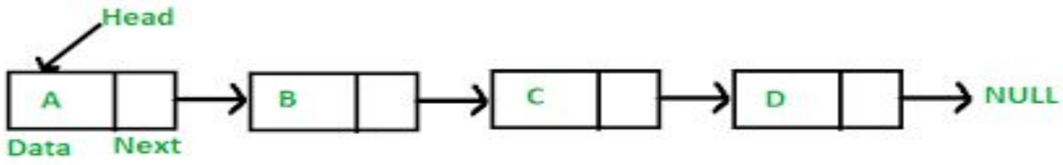
- Assume Memory Available is 10 KB:-

|  |  |  |
|---|---|---|
| 4kb | 2kb | 4kb |

- P1 = 5 Kb
- Even though total available space is 8kb, but it cannot accommodate process p1 because 5kb are not in contiguous fashion.
- This is known as **external fragmentation**.
- Here Address translation is easy, as we only need base address.
- Access time is also fast

# Non - Contiguous Memory Allocation

- Ex:- Linked List



- Assume Memory Available is 10 KB:-

| | | |
|---|---|---|
| 4kb | 2kb | 4kb |

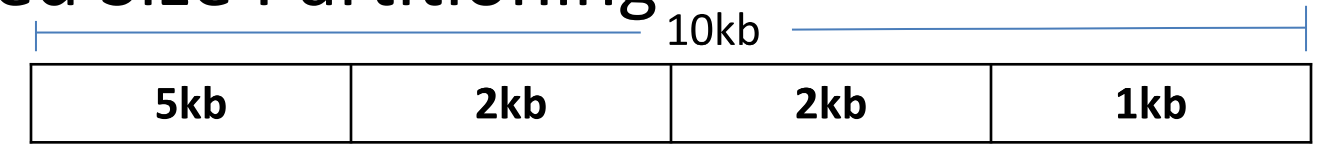
| | | | |
|---|---|---|---|
| P1(4kb) | 2kb | 3kb | P1(1kb) |

- P1 = 5 Kb
- Here total available space is 8kb, so 4kb is allotted in first slot and remaining 1kb in another one.
- As 4th node contains pointer to 5th node, it can be easily fetched.
- Here access time is slow, but it never suffers from external fragmentation

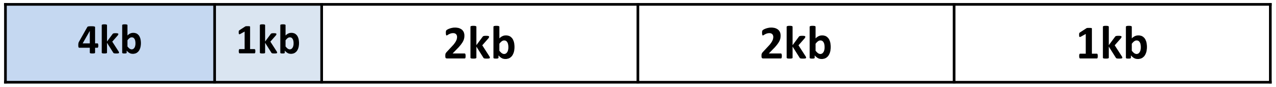
# Contiguous Memory Allocation

**Fixed Size partitioning** **Variable Size partitioning**

- Fixed Size Partitioning



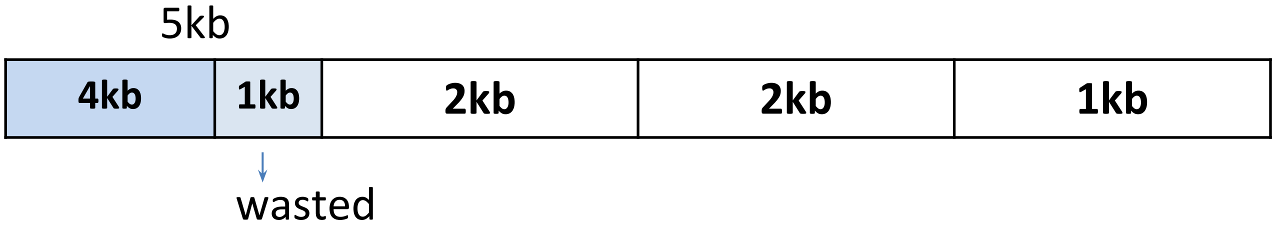
- Total Memory Space = 10kb
- We divided it as per above partitions.
- P1 = 4kb



- It takes the slot of 5kb, here 1kb gets wasted which is known as **internal fragmentation.**

# Contiguous Memory Allocation

- Fixed Size Partitioning :-
  - In this partition remaining space can never be reused
  - In one partition only one process can reside
  - Suppose p2 = 3kb, then it cannot be accommodated.



5kb

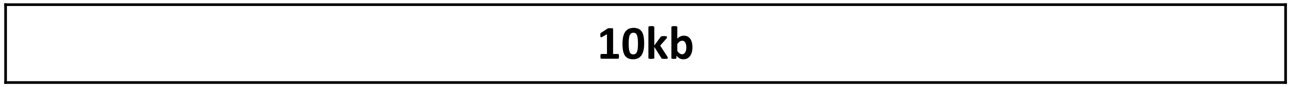
4kb | 1kb | 2kb | 2kb | 1kb

wasted

- Advantage – Easy to manage
- Disadvantage – Suffers from internal fragmentation

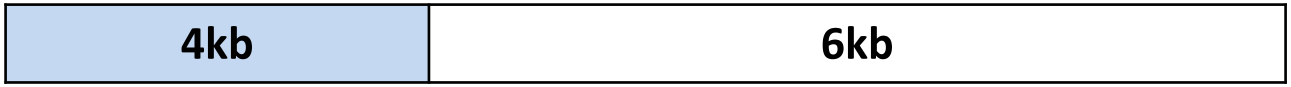
# Contiguous Memory Allocation

- Variable Size Partitioning :-
  - Memory size = 10kb



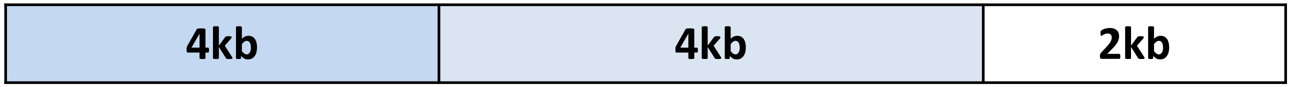
10kb

  - P1 = 4kb



4kb 6kb

  - P2 = 4kb



4kb 4kb 2kb

  - Here internal fragmentation never occurs.

# Methods of Contiguous Memory Allocation

- Irrespective of type of partitioning(fixed size or variable size) following methods are used:-
  - First fit
  - Best fit
  - Worst fit
- Ex:-

P1=90k

P2=20k

P3=50k

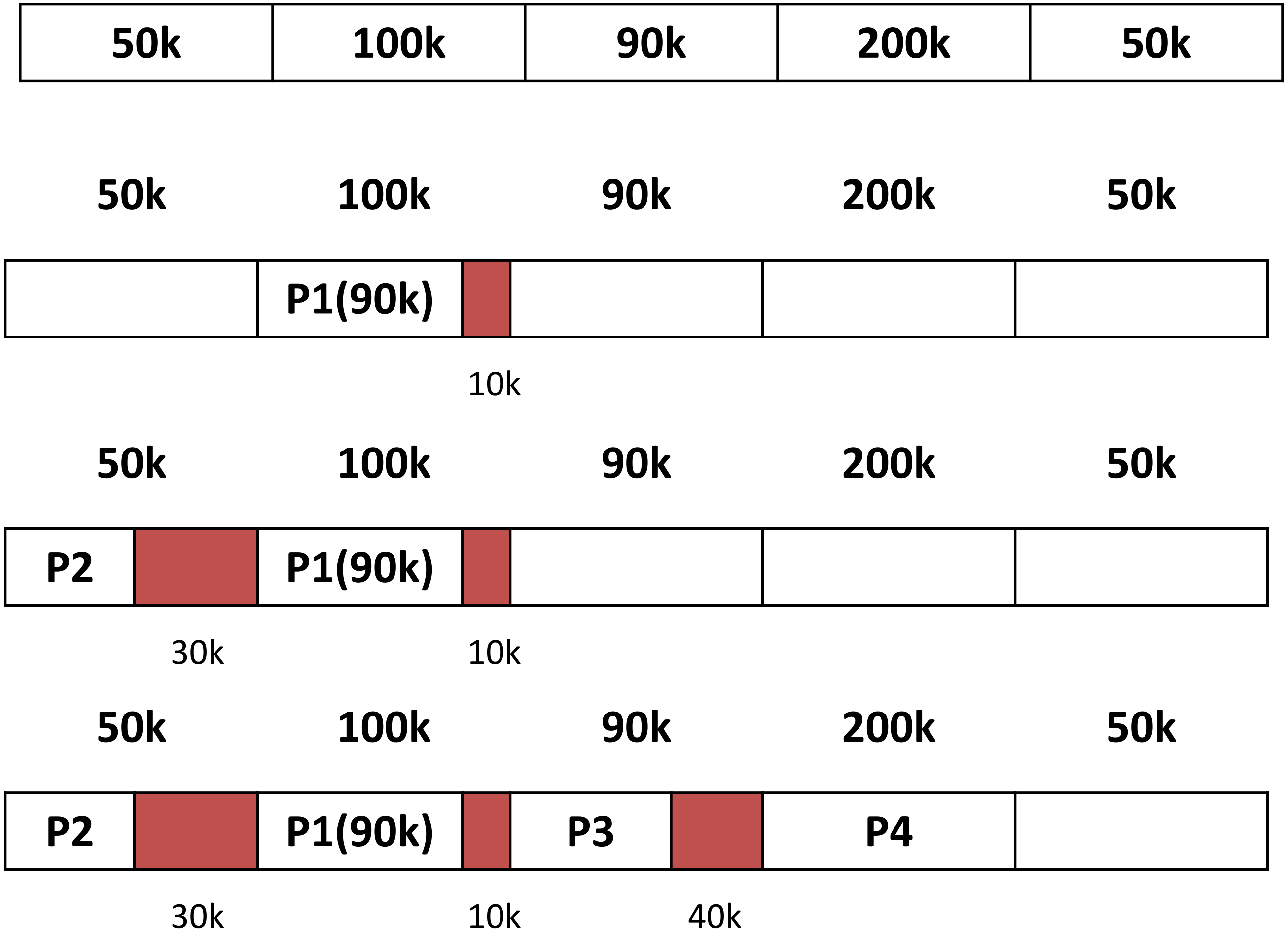
P4=200k

| | | | | |
|---|---|---|---|---|
| **50k** | **100k** | **90k** | **200k** | **50k** |

# Fixed Size Partitioning

P1=90k , P2=20k , P3=50k , P4=200k

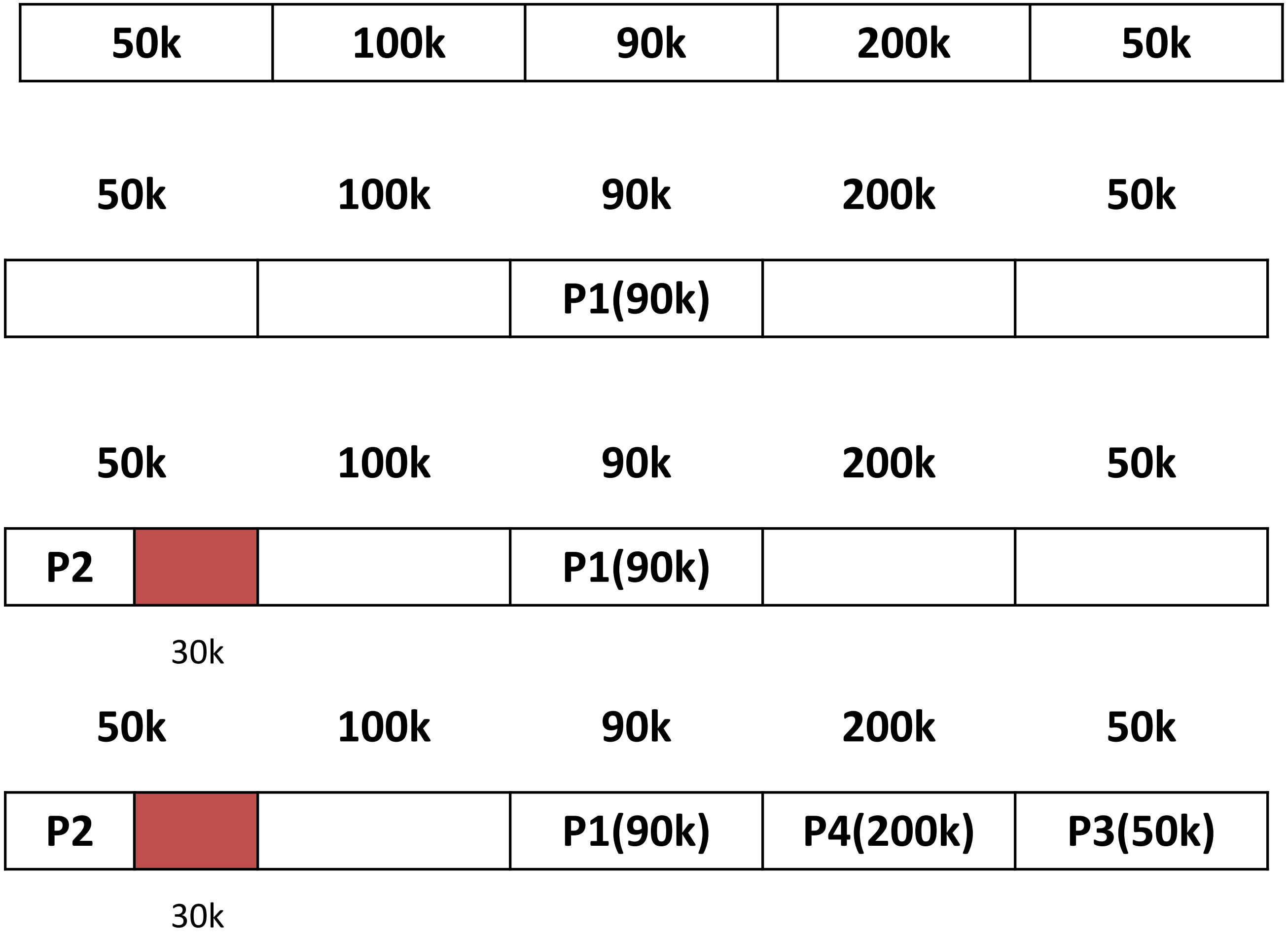
- First Fit



# Fixed Size Partitioning

P1=90k , P2=20k , P3=50k , P4=200k

- <u>Best Fit</u>



| 50k | 100k | 90k | 200k | 50k |
|---|---|---|---|---|

| 50k | 100k | 90k | 200k | 50k |
|---|---|---|---|---|
|  |  | P1(90k) |  |  |

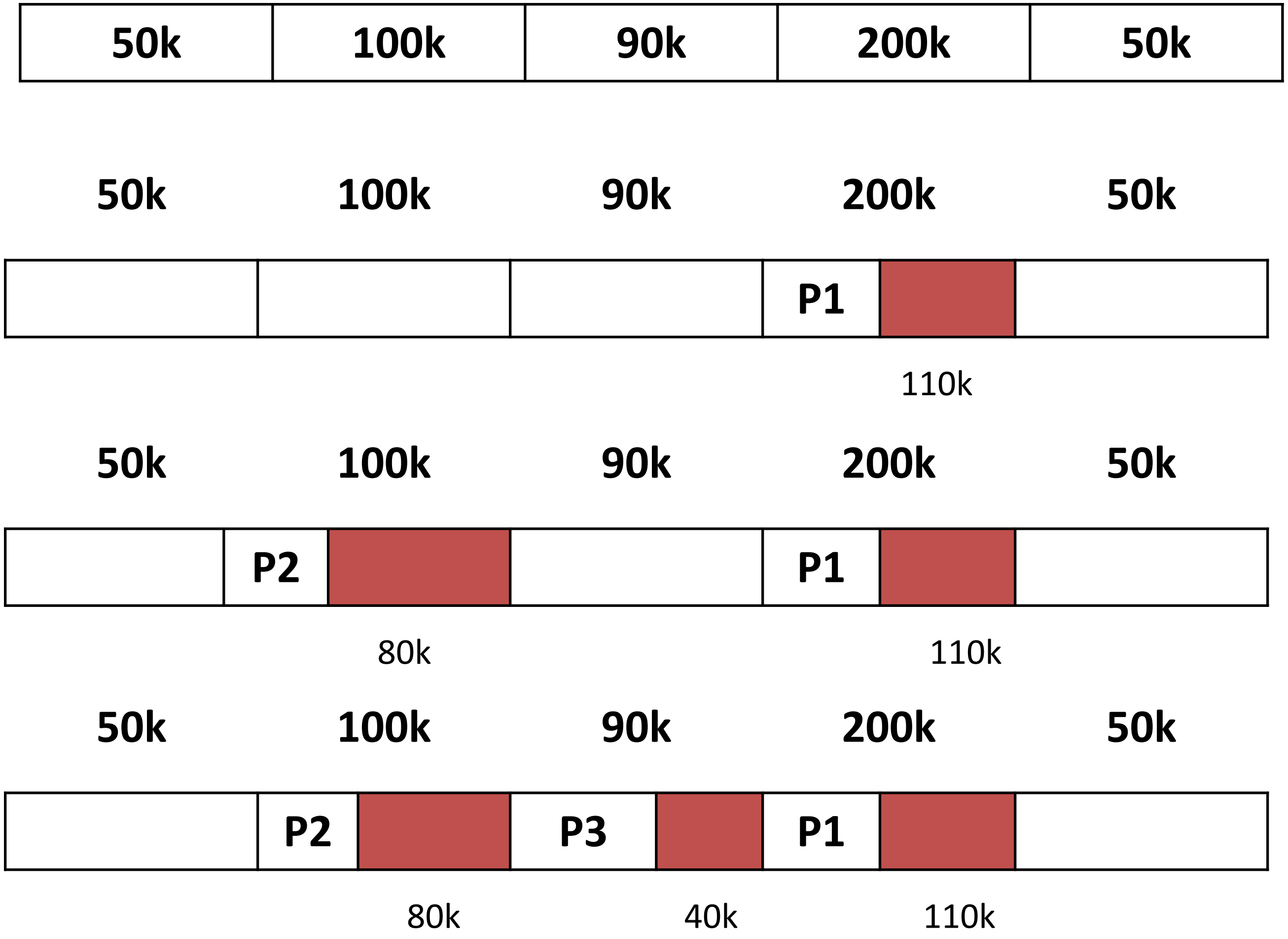
| 50k | 100k | 90k | 200k | 50k |
|---|---|---|---|---|
| P2 🟥<br>30k |  | P1(90k) |  |  |

| 50k | 100k | 90k | 200k | 50k |
|---|---|---|---|---|
| P2 🟥<br>30k |  | P1(90k) | P4(200k) | P3(50k) |

# Fixed Size Partitioning

P1=90k , P2=20k , P3=50k , P4=200k

- <u>Worst Fit</u>



50k | 100k | 90k | 200k | 50k

50k 100k 90k 200k 50k

P1

110k

50k 100k 90k 200k 50k

P2 P1

80k 110k

50k 100k 90k 200k 50k

P2 P3 P1

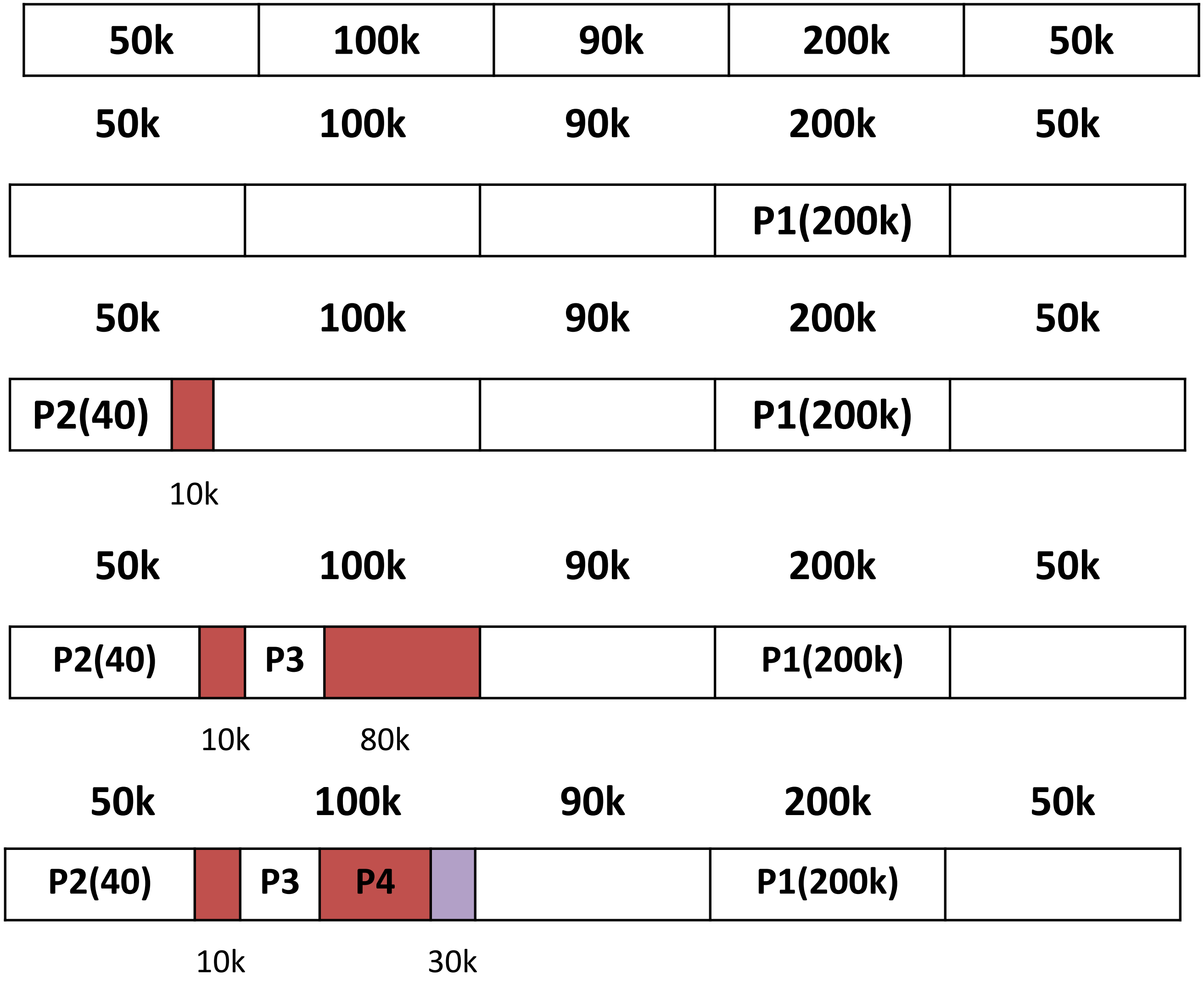
80k 40k 110k

Here **P4** cannot be accommodated

# Variable Size Partitioning

P1=200k , P2=40k , P3=20k , P4=50k

- <u>First Fit</u>



50k 100k 90k 200k 50k

50k 100k 90k 200k 50k

P1(200k)

50k 100k 90k 200k 50k

P2(40) P1(200k)

10k

50k 100k 90k 200k 50k

P2(40) P3 P1(200k)

10k 80k

50k 100k 90k 200k 50k

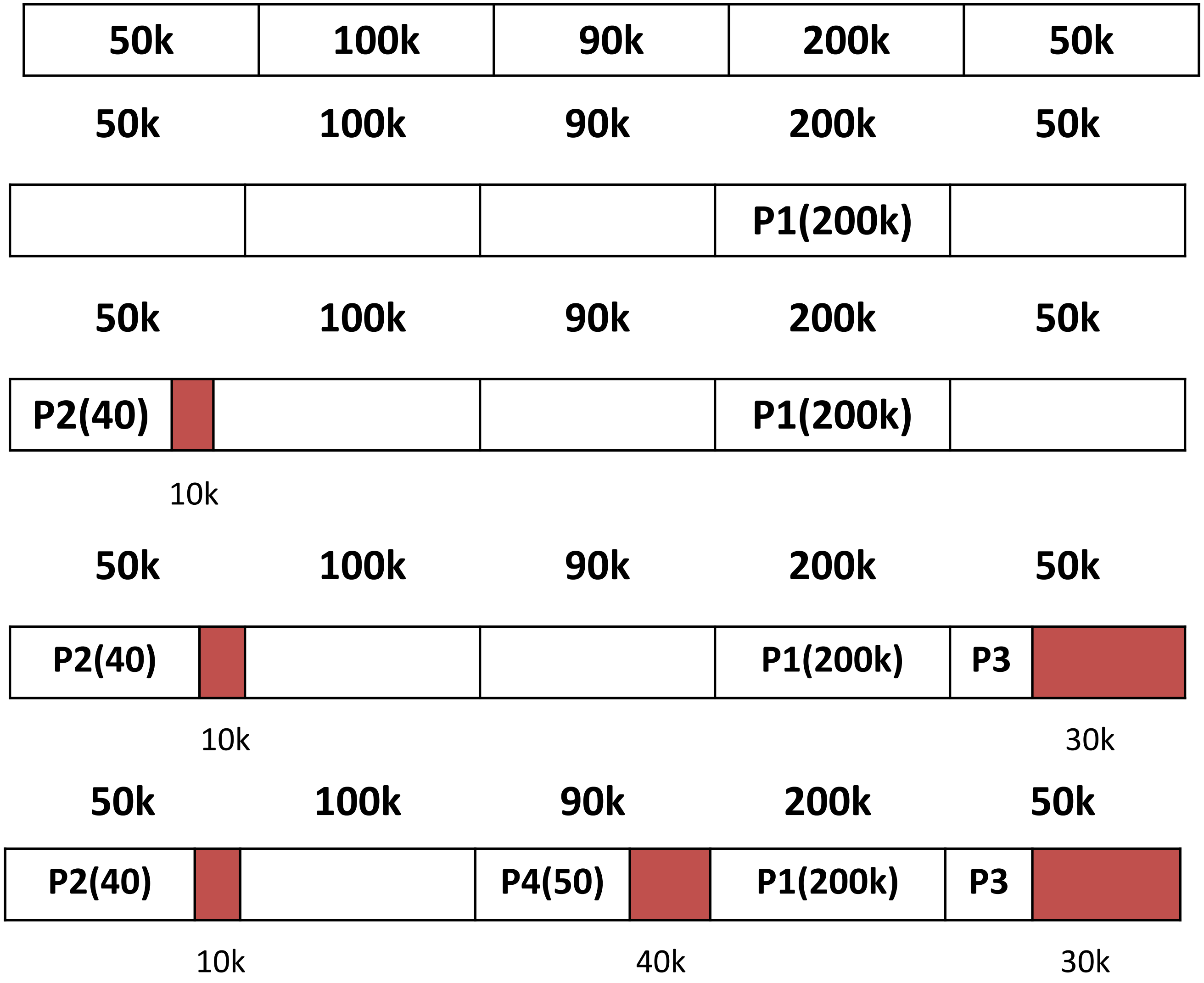
P2(40) P3 P4 P1(200k)

10k 30k

# Variable Size Partitioning

P1=200k , P2=40k , P3=20k , P4=50k

- <u>Best Fit</u>



| 50k | 100k | 90k | 200k | 50k |
|---|---|---|---|---|

| 50k | 100k | 90k | 200k | 50k |
|---|---|---|---|---|
| | | | P1(200k) | |

| 50k | 100k | 90k | 200k | 50k |
|---|---|---|---|---|
| P2(40) 🟥 10k | | | P1(200k) | |

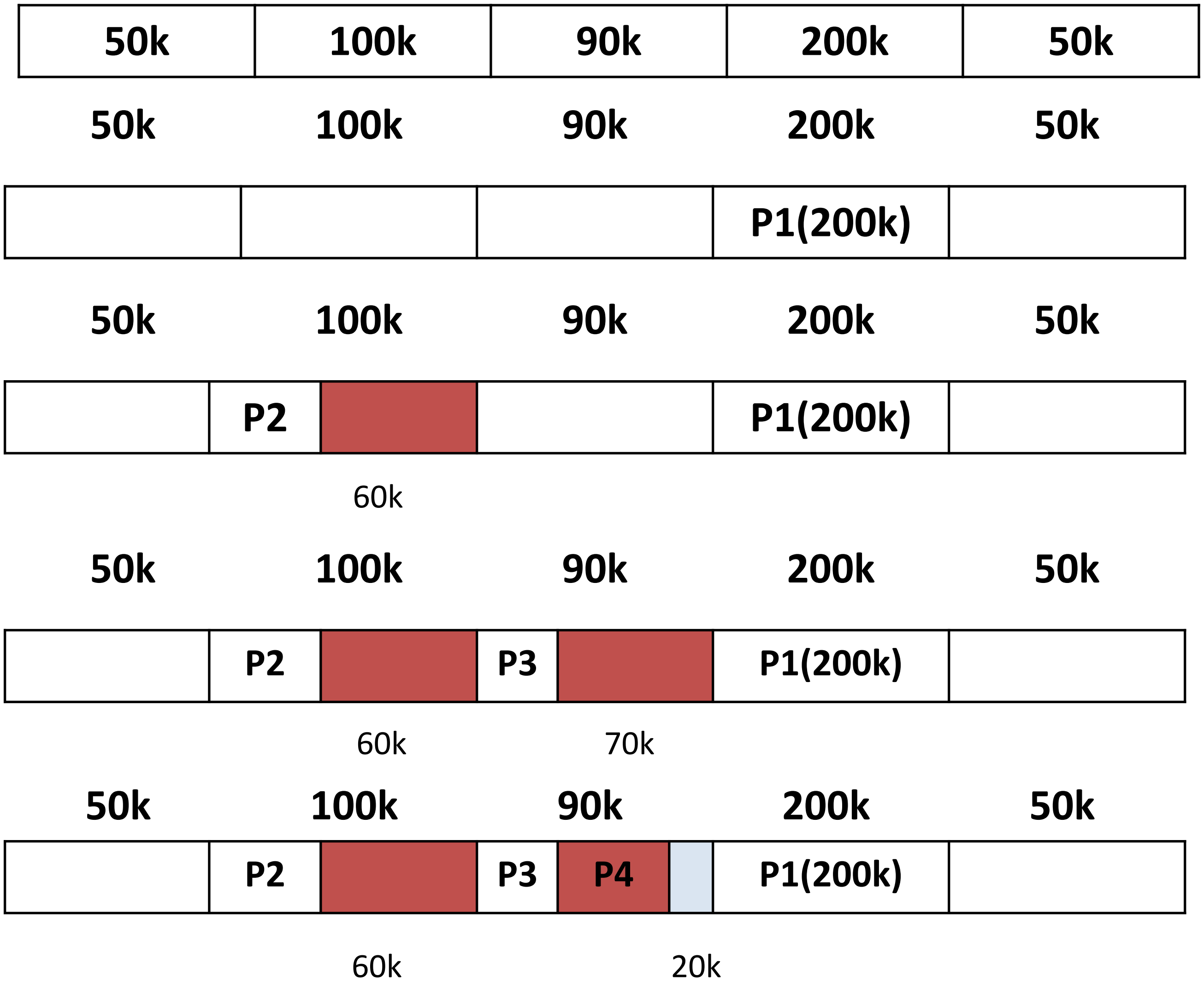
| 50k | 100k | 90k | 200k | 50k |
|---|---|---|---|---|
| P2(40) 🟥 10k | | | P1(200k) | P3 🟥 30k |

| 50k | 100k | 90k | 200k | 50k |
|---|---|---|---|---|
| P2(40) 🟥 10k | | P4(50) 🟥 40k | P1(200k) | P3 🟥 30k |

# Variable Size Partitioning

P1=200k , P2=40k , P3=20k , P4=50k

- Worst Fit



50k 100k 90k 200k 50k

50k 100k 90k 200k 50k

P1(200k)

50k 100k 90k 200k 50k

P2 P1(200k)

60k

50k 100k 90k 200k 50k

P2 P3 P1(200k)

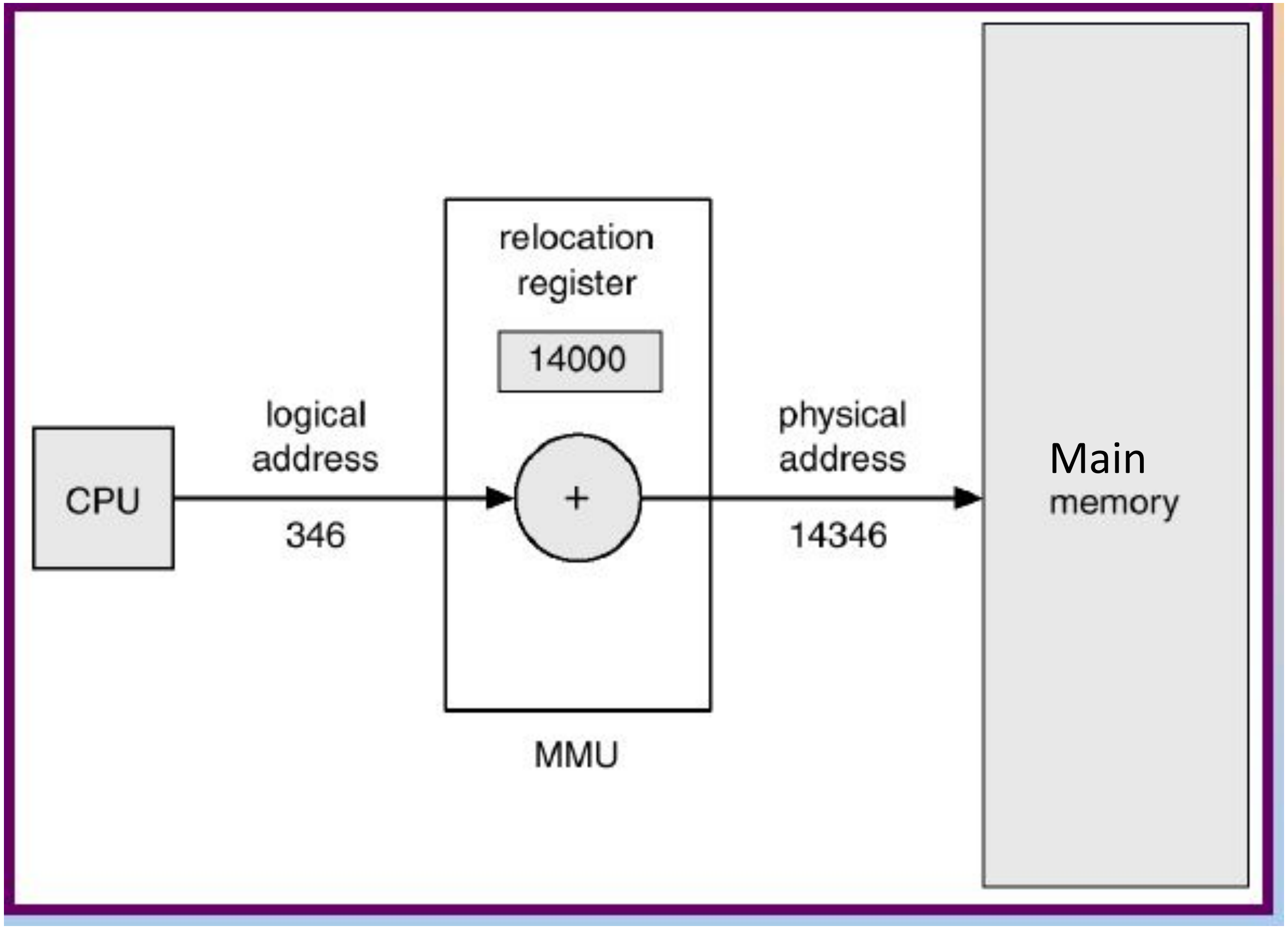
60k 70k

50k 100k 90k 200k 50k

P2 P3 P4 P1(200k)

60k 20k

# Contiguous Memory Allocation

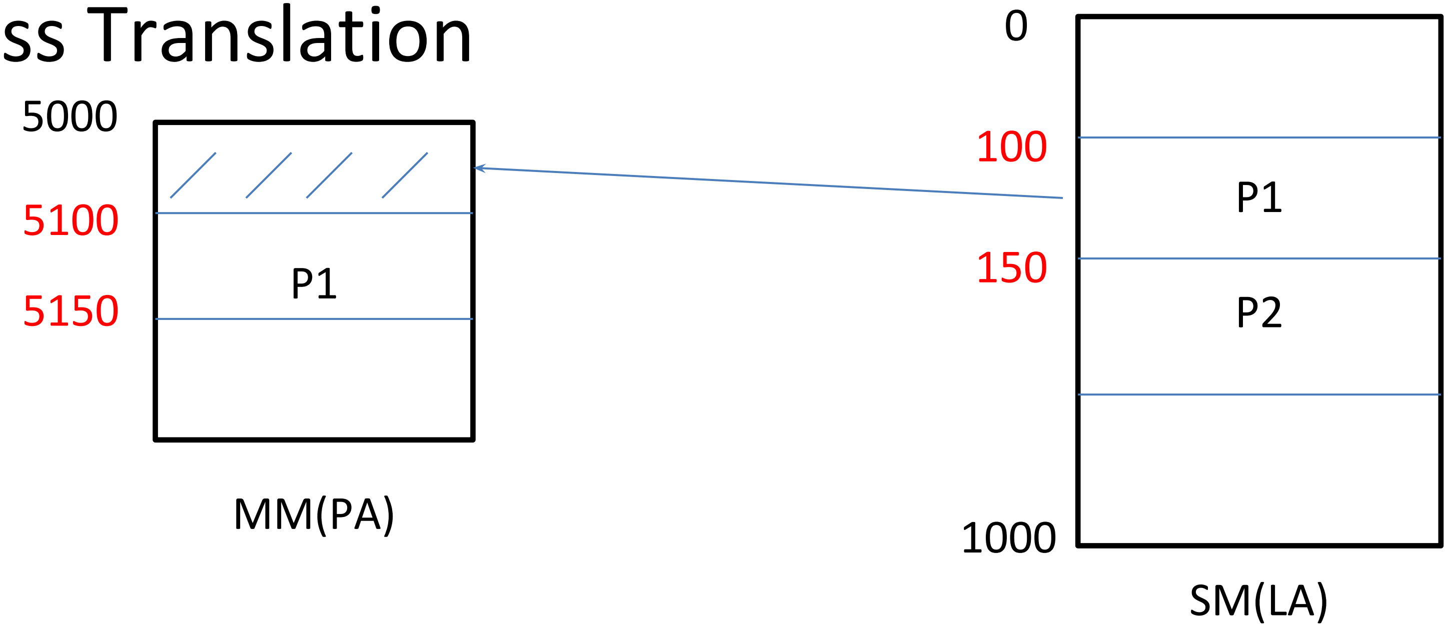


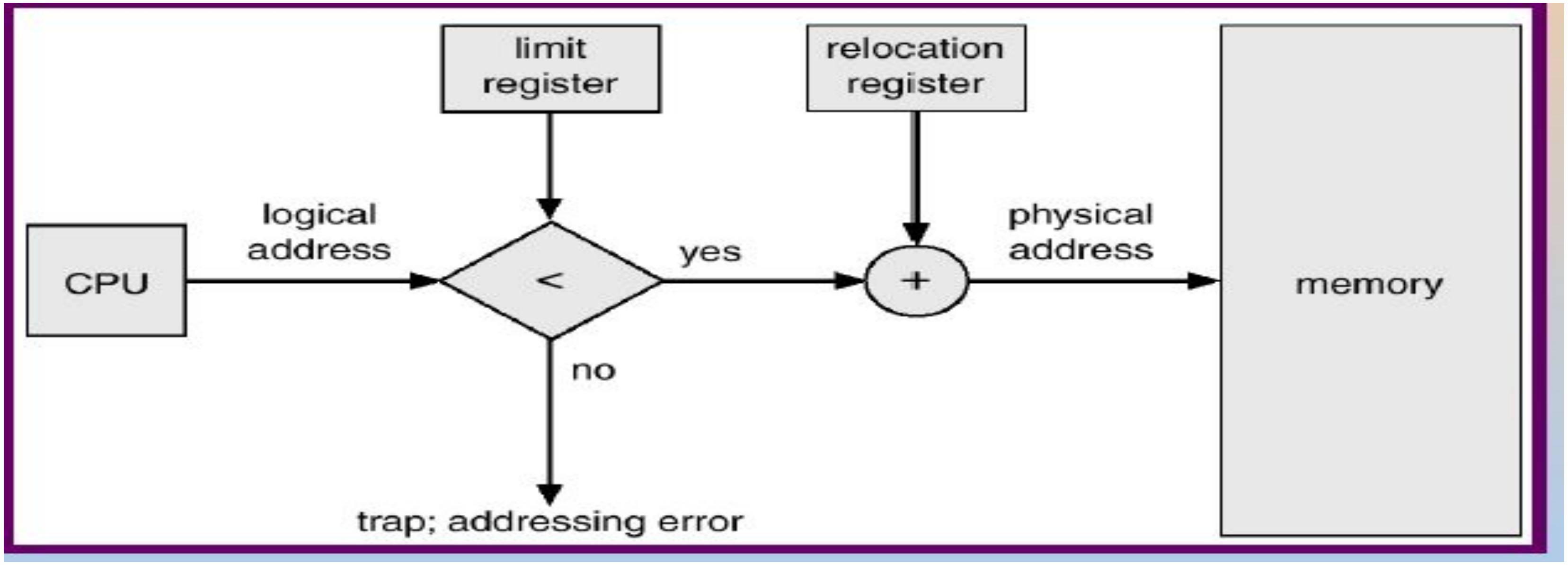
**Secondary Memory**

**[Logical Address]**

# Contiguous Memory Allocation

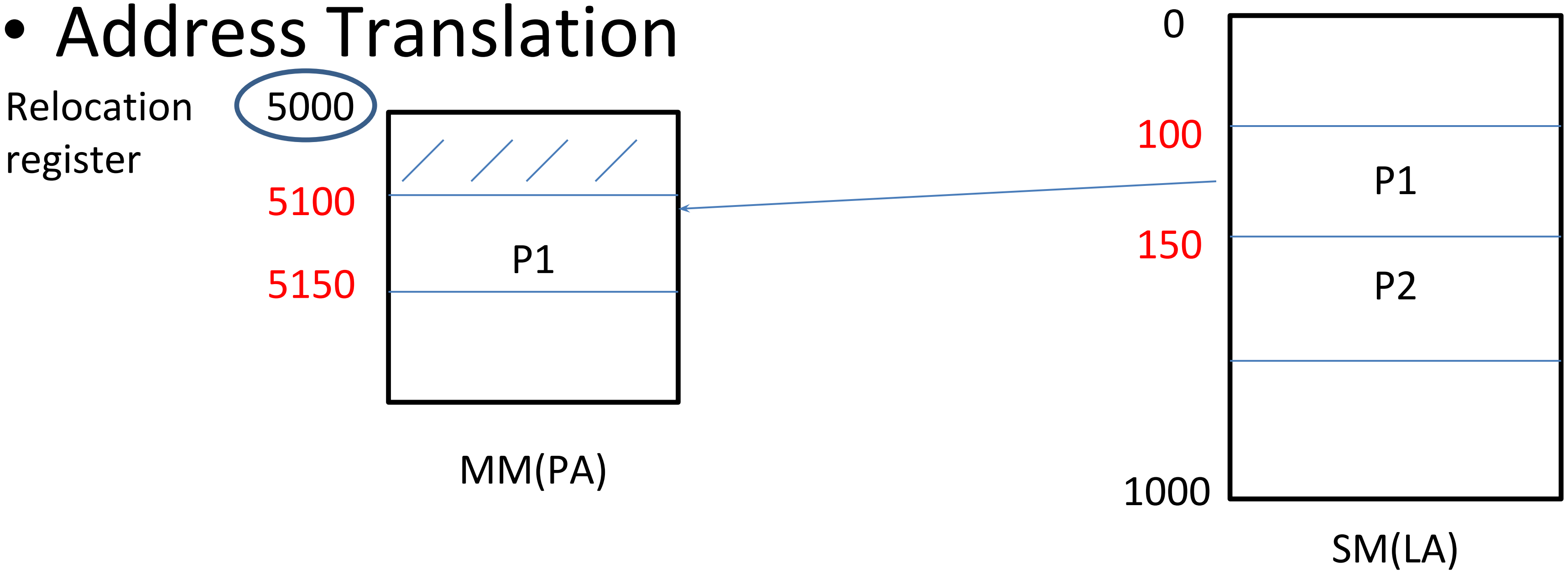
- Address Translation





# Contiguous Memory Allocation

- Address Translation



Relocation register 5000

5100

5150

P1

MM(PA)

0

100

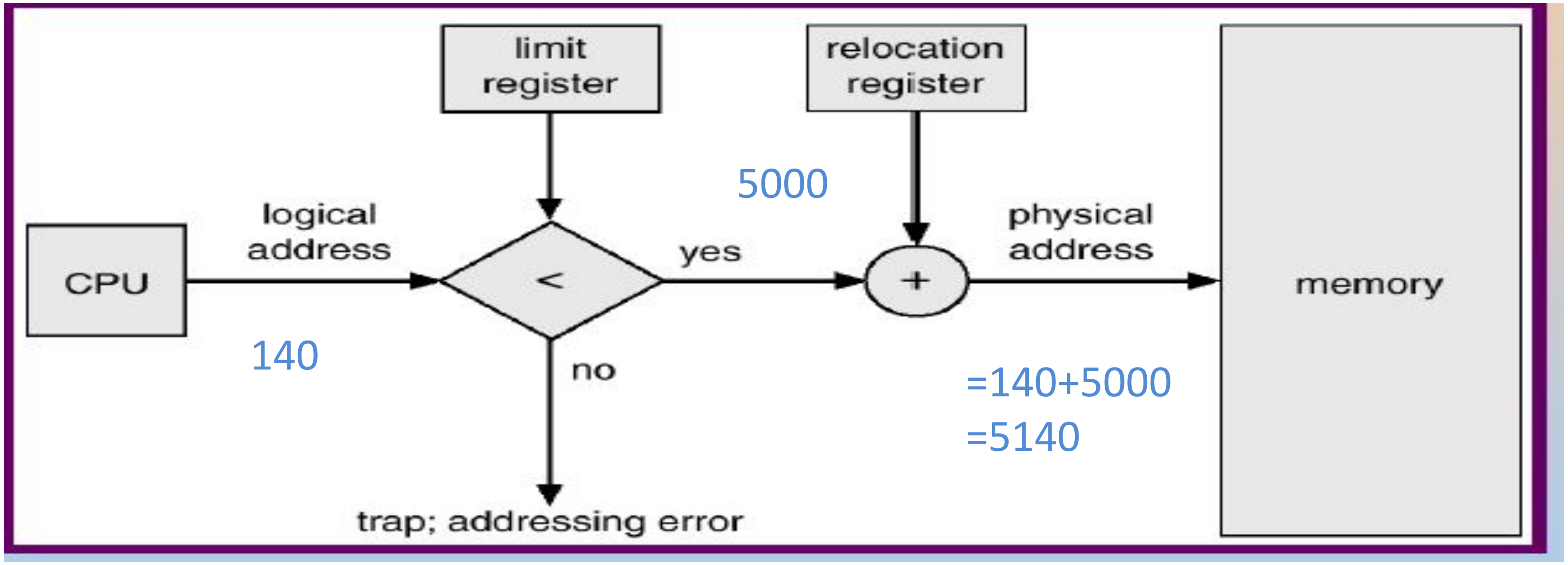
P1

150

P2

1000

SM(LA)



limit register

relocation register

5000

CPU

logical address

<

yes

+

physical address

memory

140

no

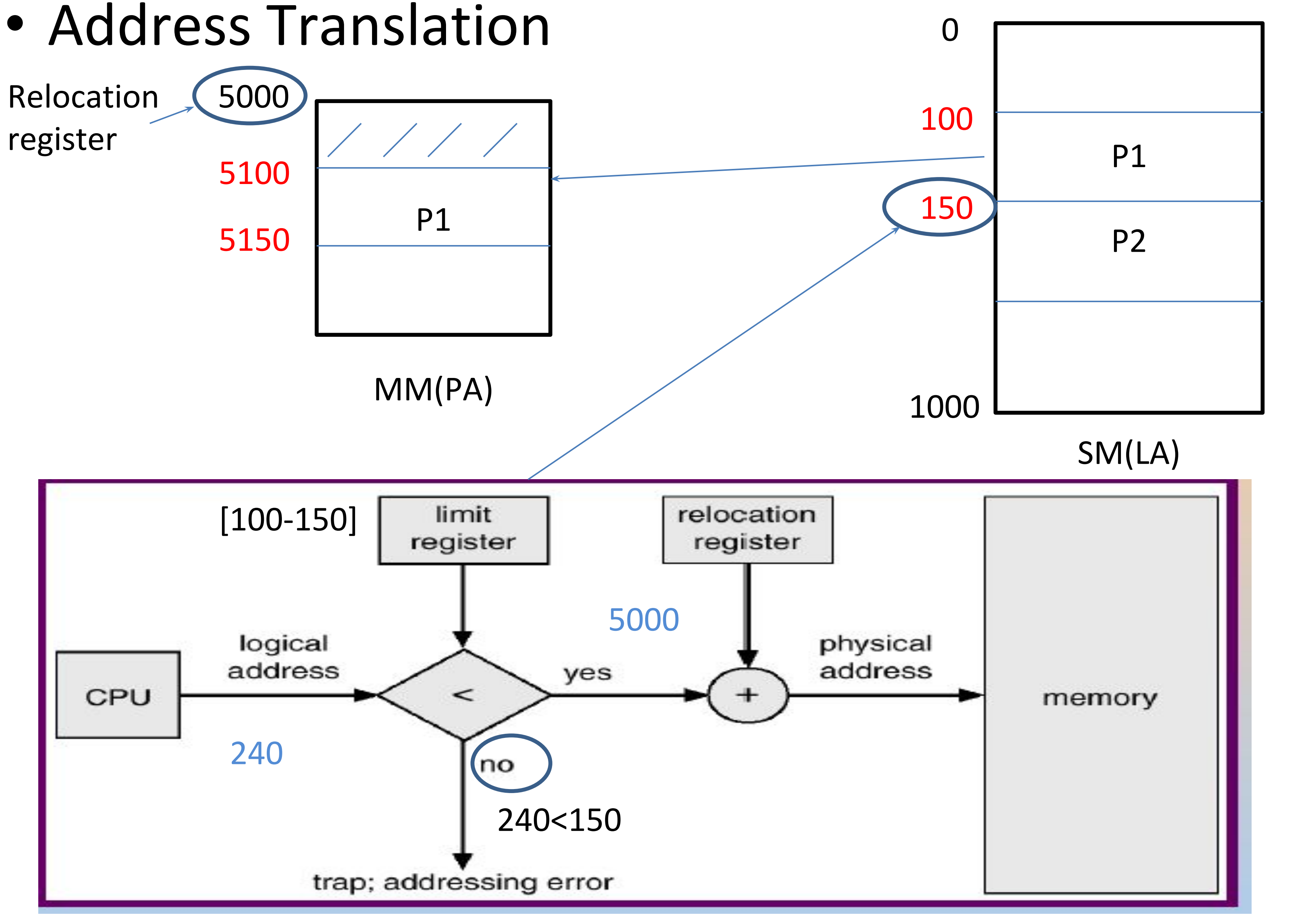
=140+5000

=5140

trap; addressing error

# Contiguous Memory Allocation

- Address Translation



Relocation register

5000

5100

5150

P1

MM(PA)

0

100

P1

150

P2

1000

SM(LA)

[100-150]

5000

240

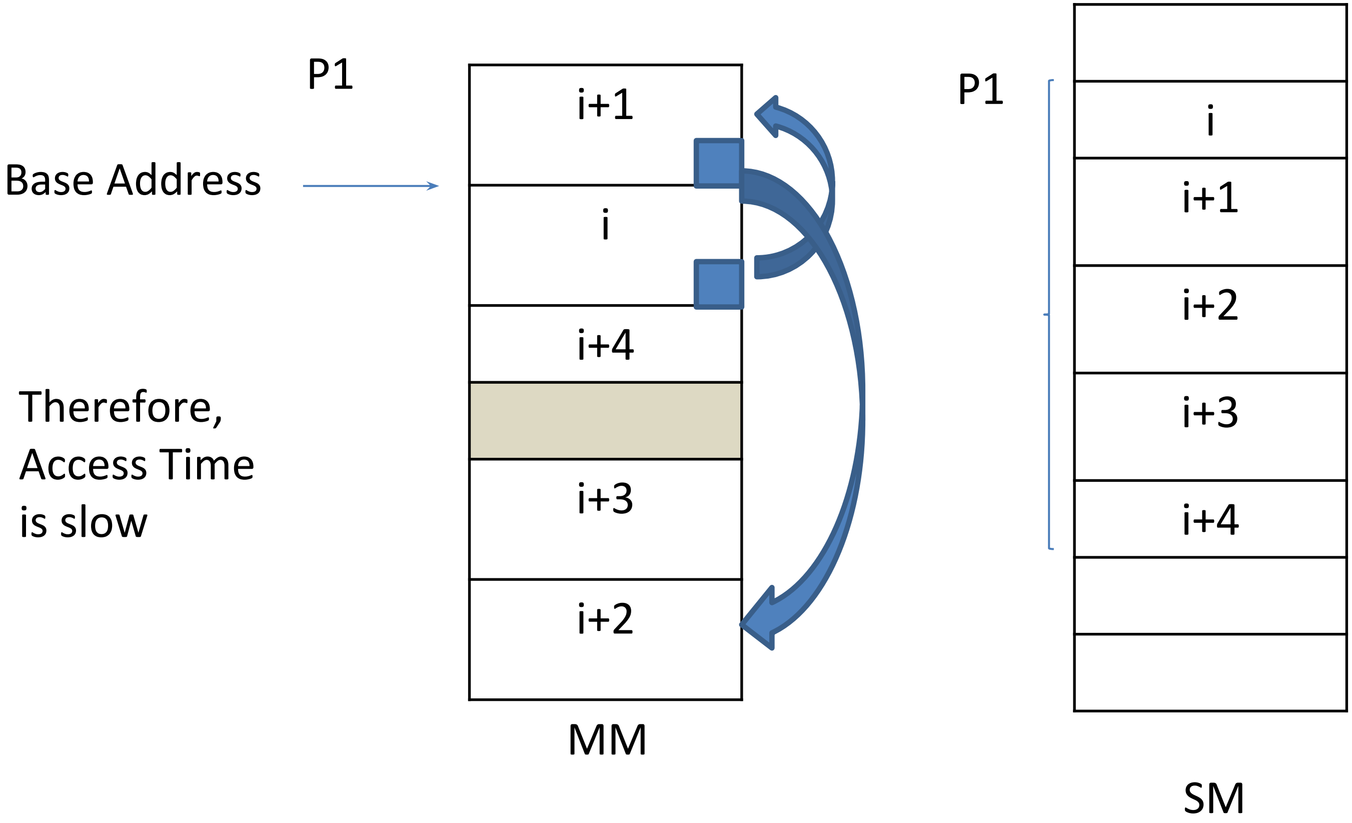
240<150

# Contiguous Memory Allocation

- Data is contiguously or sequentially present in memory.
- So access time is very fast
- But it creates external as well as internal fragmentation
- External fragmentation is more severe problem than internal fragmentation.

# Non Contiguous Memory Allocation

- Not necessary to allocate memory in sequence

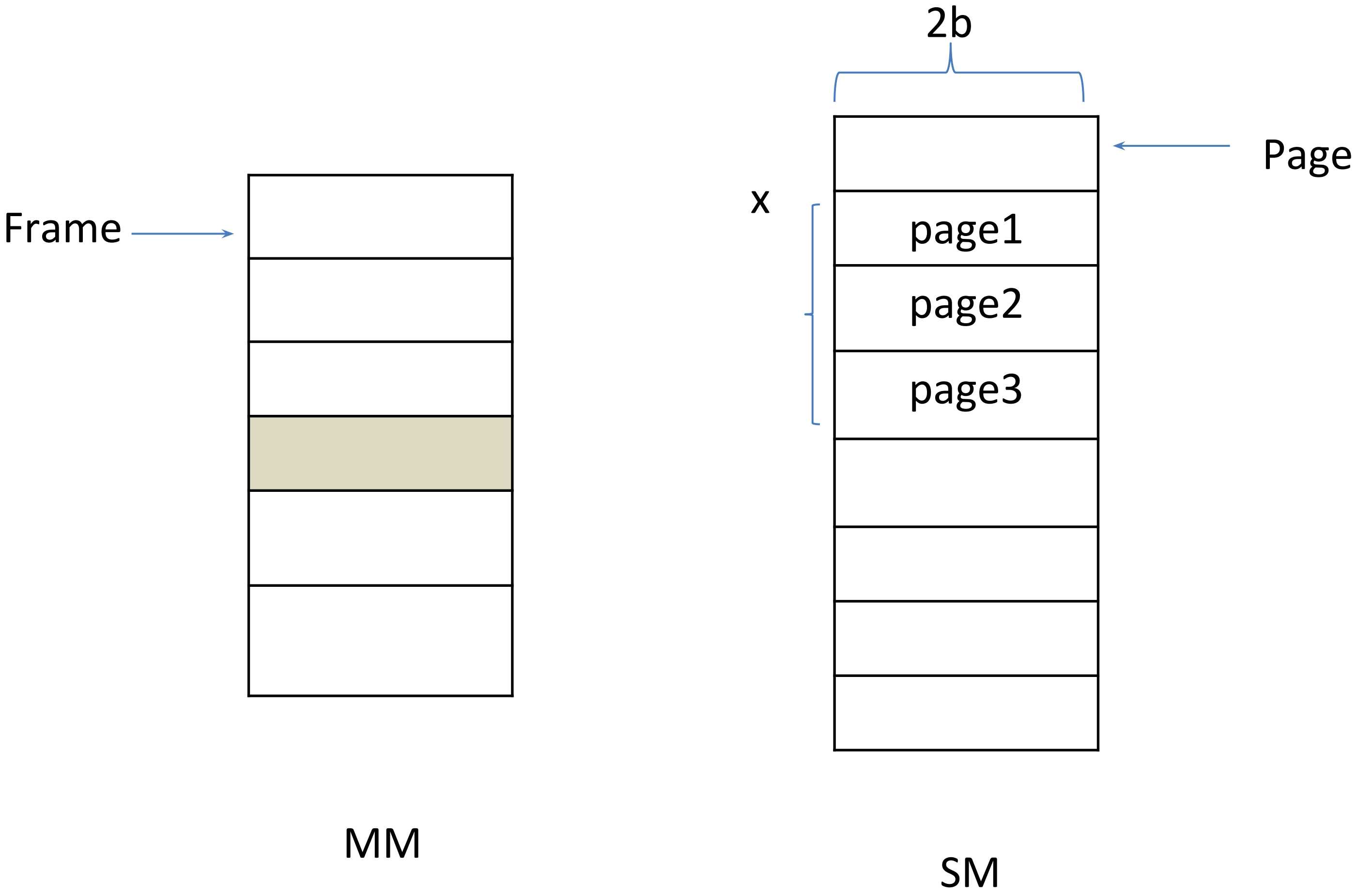


Therefore, Access Time is slow

# Non Contiguous Memory Allocation

- Partition Secondary Memory as well as Main Memory
- Size of each and every partition should be fixed and same
- It follows fixed size partitioning , so also suffers with internal fragmentation, but it is not a major issue.
- Equal size partition in SM is called **Page**
- Equal size partition in MM is called **Frame**

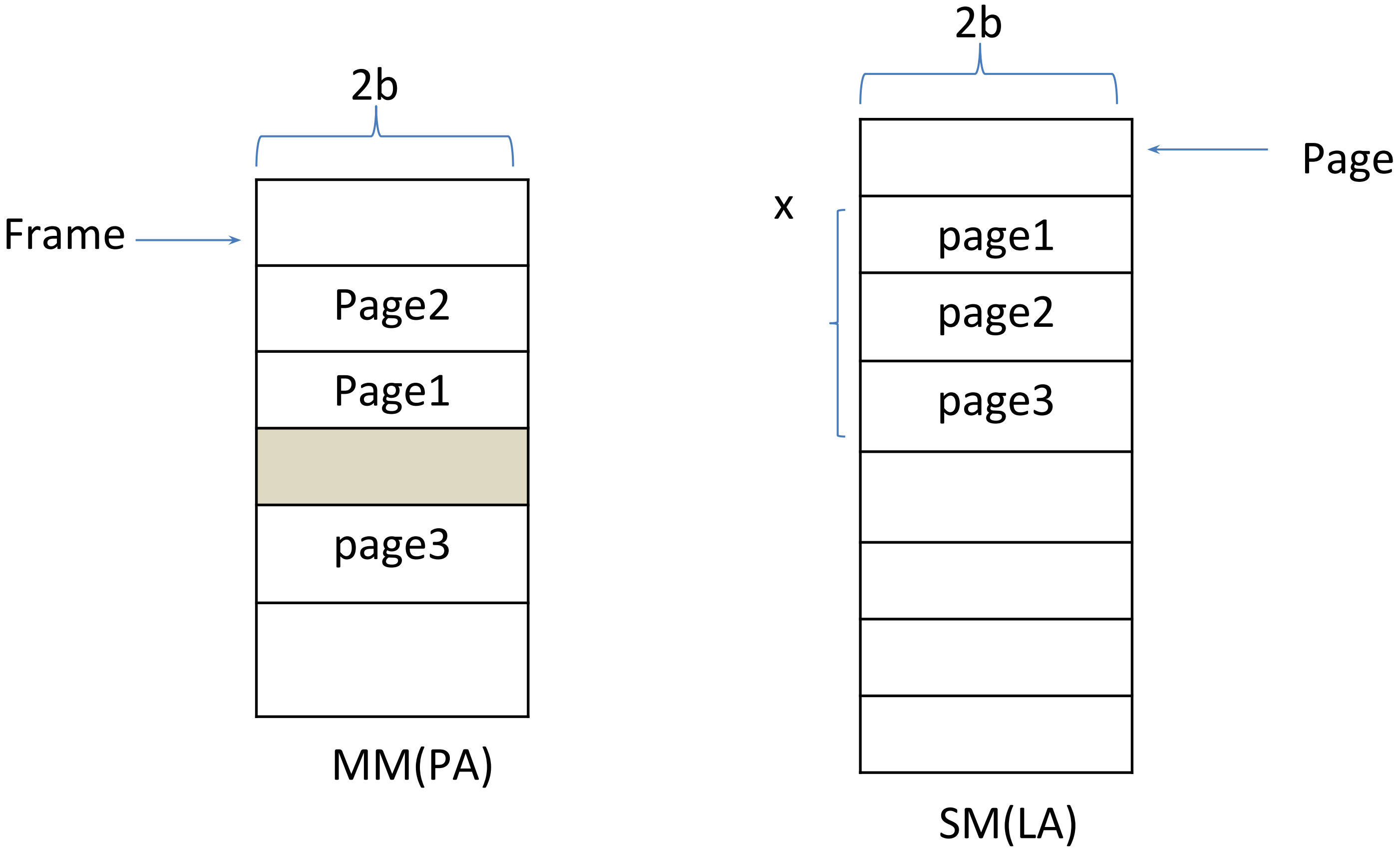
# Paging



- Assume Page size = 2b, process x requires 5b, then also it needs 3 pages
- Here, 1b is wasted due to internal fragmentation

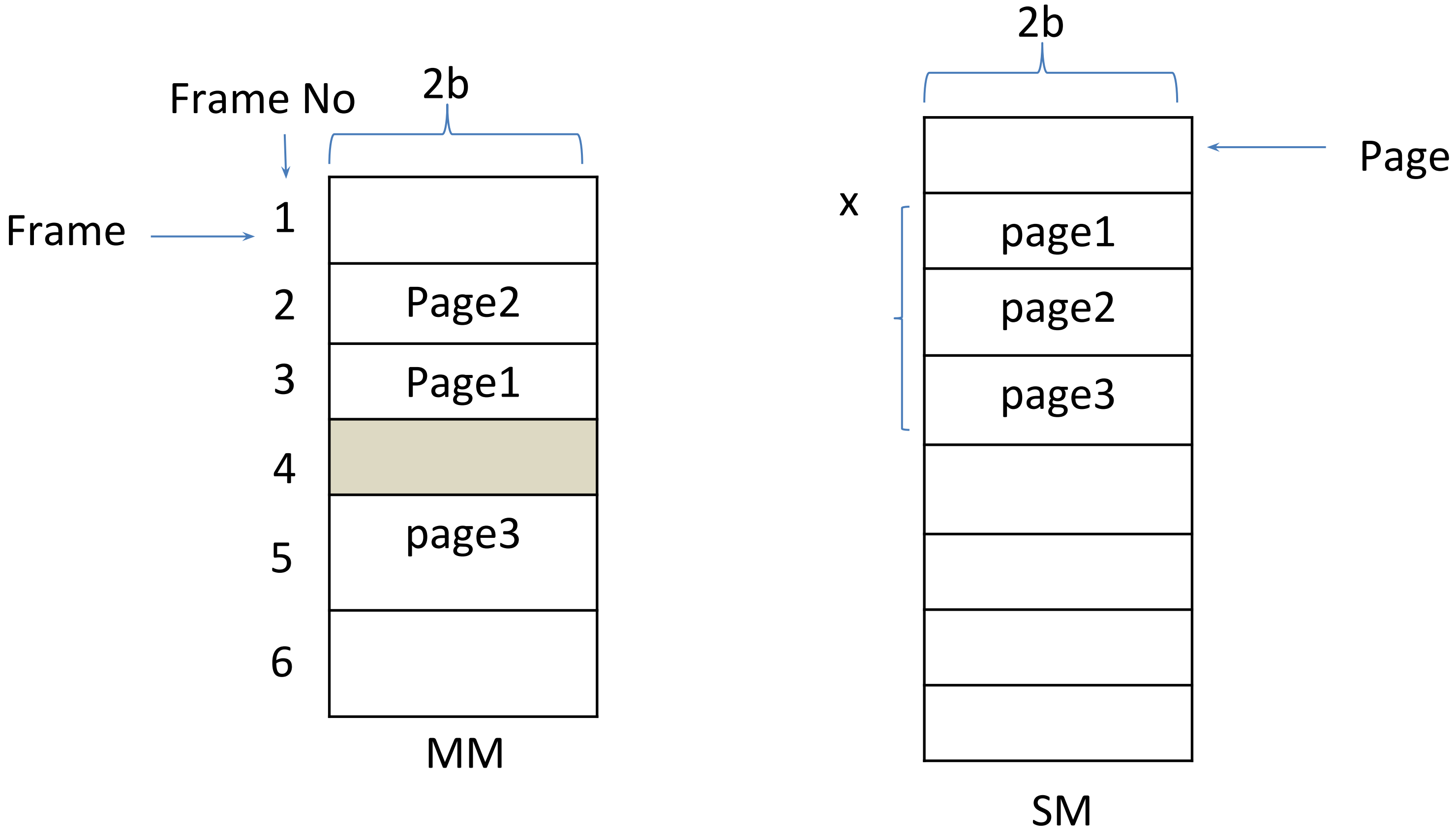
# Paging

- We want to load process x from SM to MM
- It is possible only if size of page is same as size of frame
- CPU knows only SM, so it generates only Logical Address



# Paging

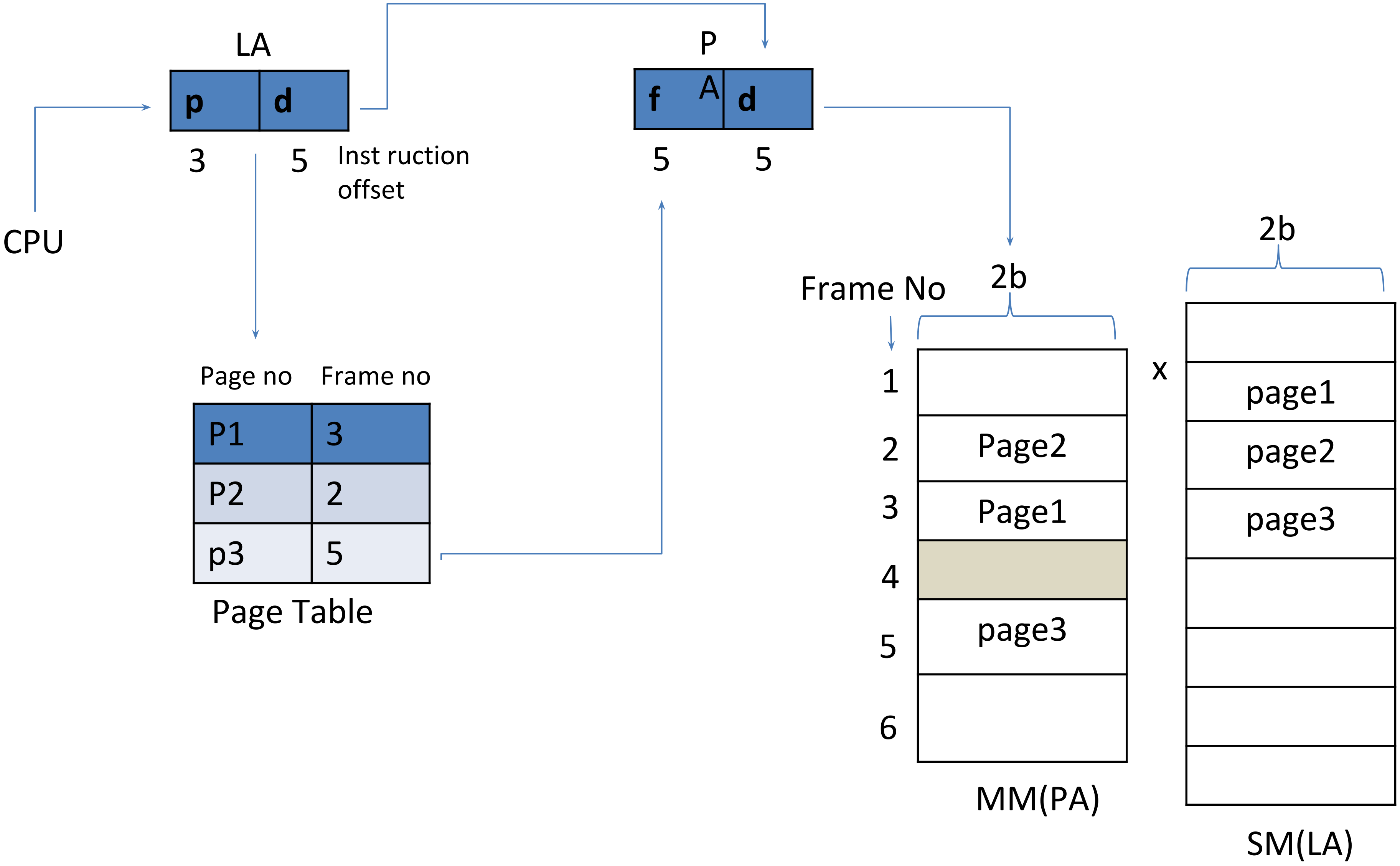
- We want to load process x from SM to MM
- It is possible only if size of page is same as size of frame
- CPU knows only SM, so it generates only Logical Address



# Page Table

- Page Table is a data structure
- It maps page number to frame number
- It is stored in Main Memory
- Every process contains its own page table
- Page table information cannot be present in PCB, as it becomes very heavy.
- PCB contains PTBR(Page table base register), in turn PTBR has page table

# Paging



| Page no | Frame no |
|---|---|
| P1 | 3 |
| P2 | 2 |
| p3 | 5 |

Page Table