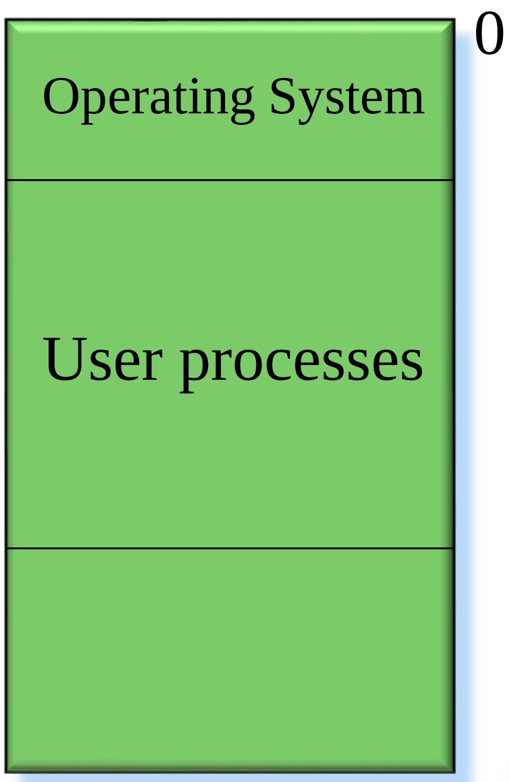
# Memory Management

Contiguous Memory Allocation: Fixed-partition and Variable-Partition

Non-contiguous Memory Allocation: Paging

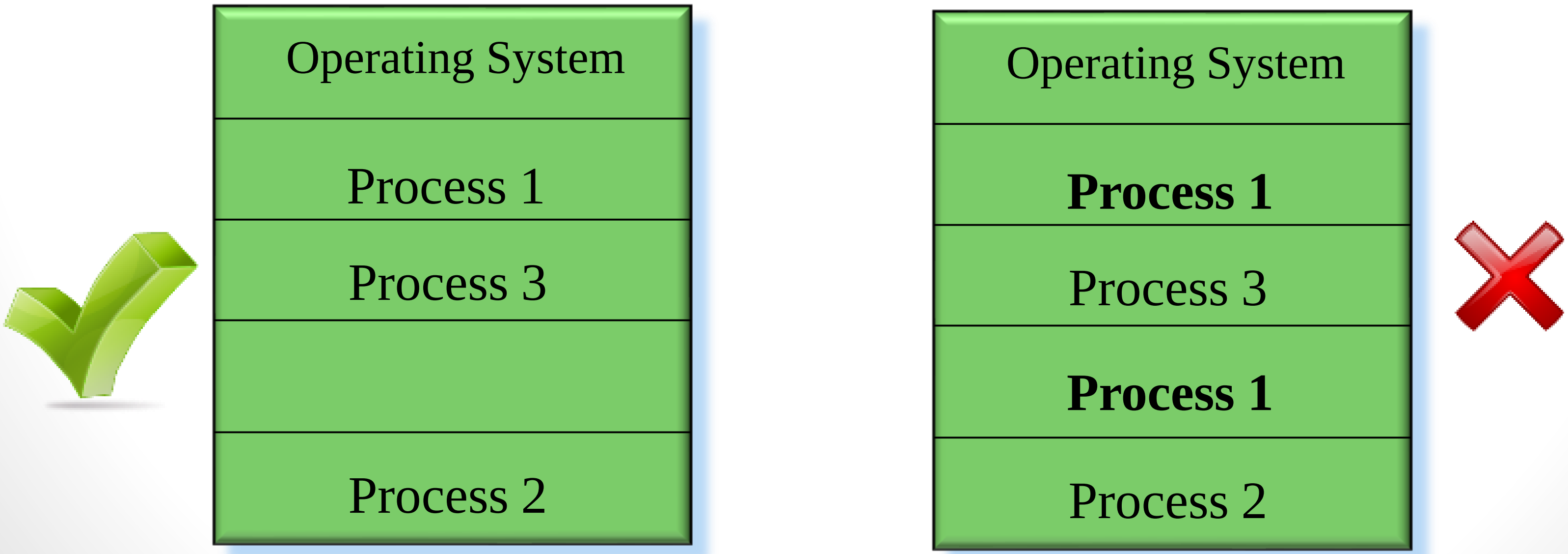
# Memory Allocation

- Memory is usually divided into two partitions, one for the resident operating system, and one for the user processes. It is common for the operating system to occupy low memory.
- Memory Allocation for User Processes:
  - Contiguous
  - Non-contiguous



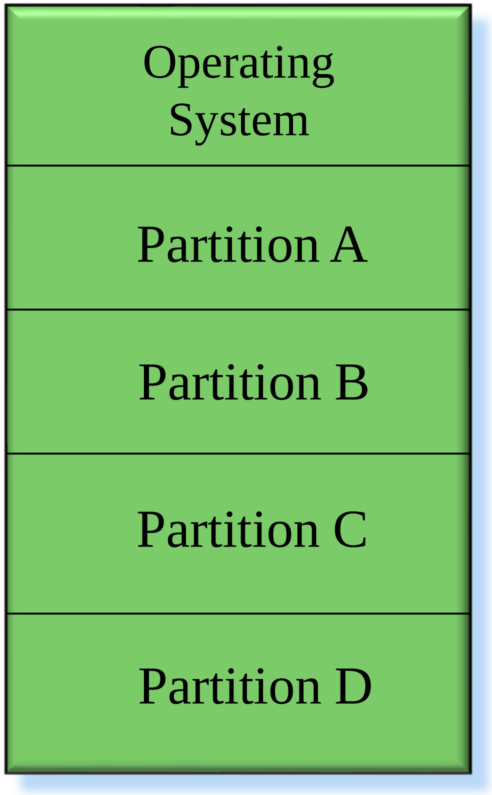
# Contiguous Memory Allocation

- Each process is contained in a single contiguous section of memory
  - To run a process, the system has to find enough contiguous memory to accommodate the entire process.



# Fixed-partition Memory Allocation

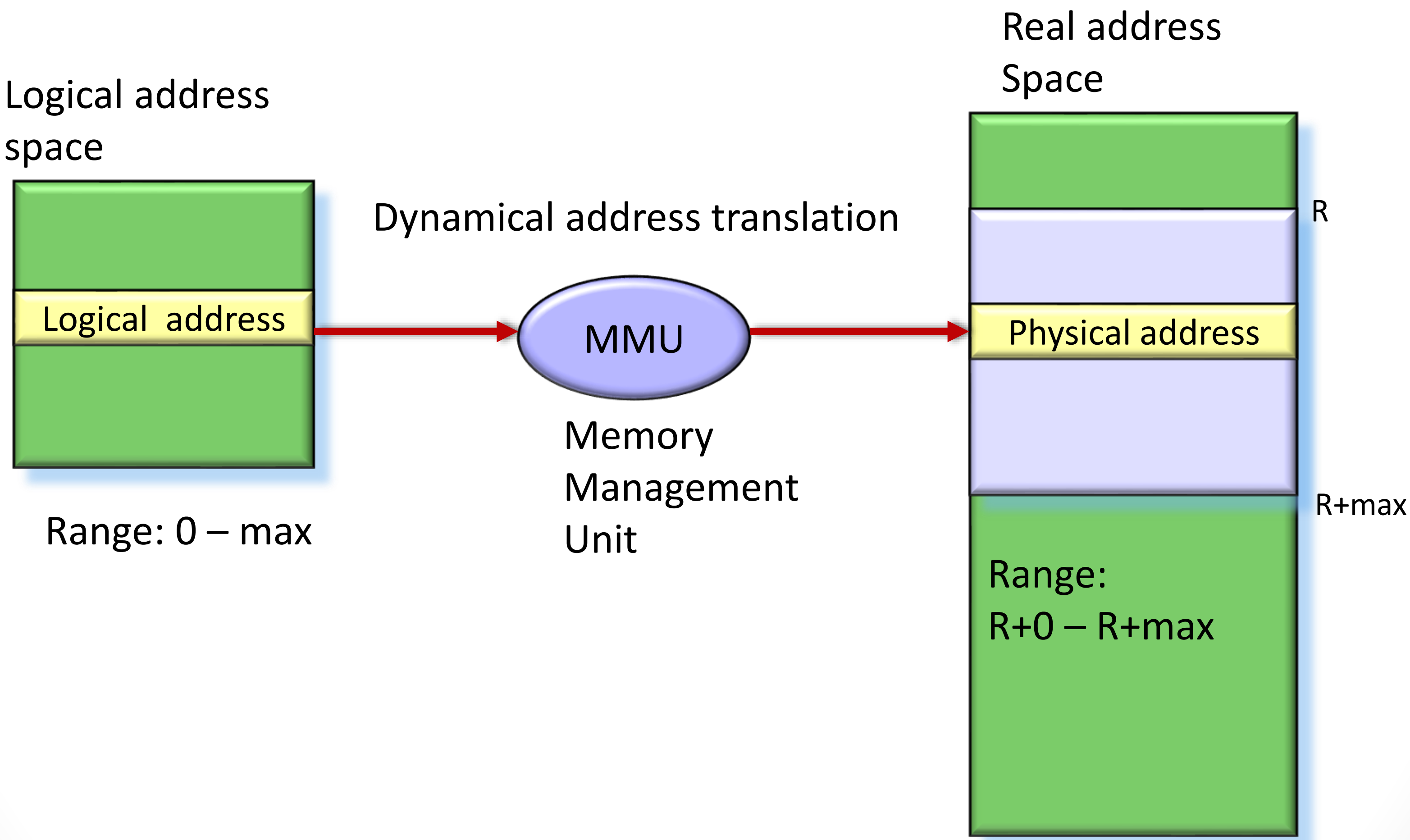
- Divide memory into partitions of fixed and equal size.
- Each partition contains at most one process.



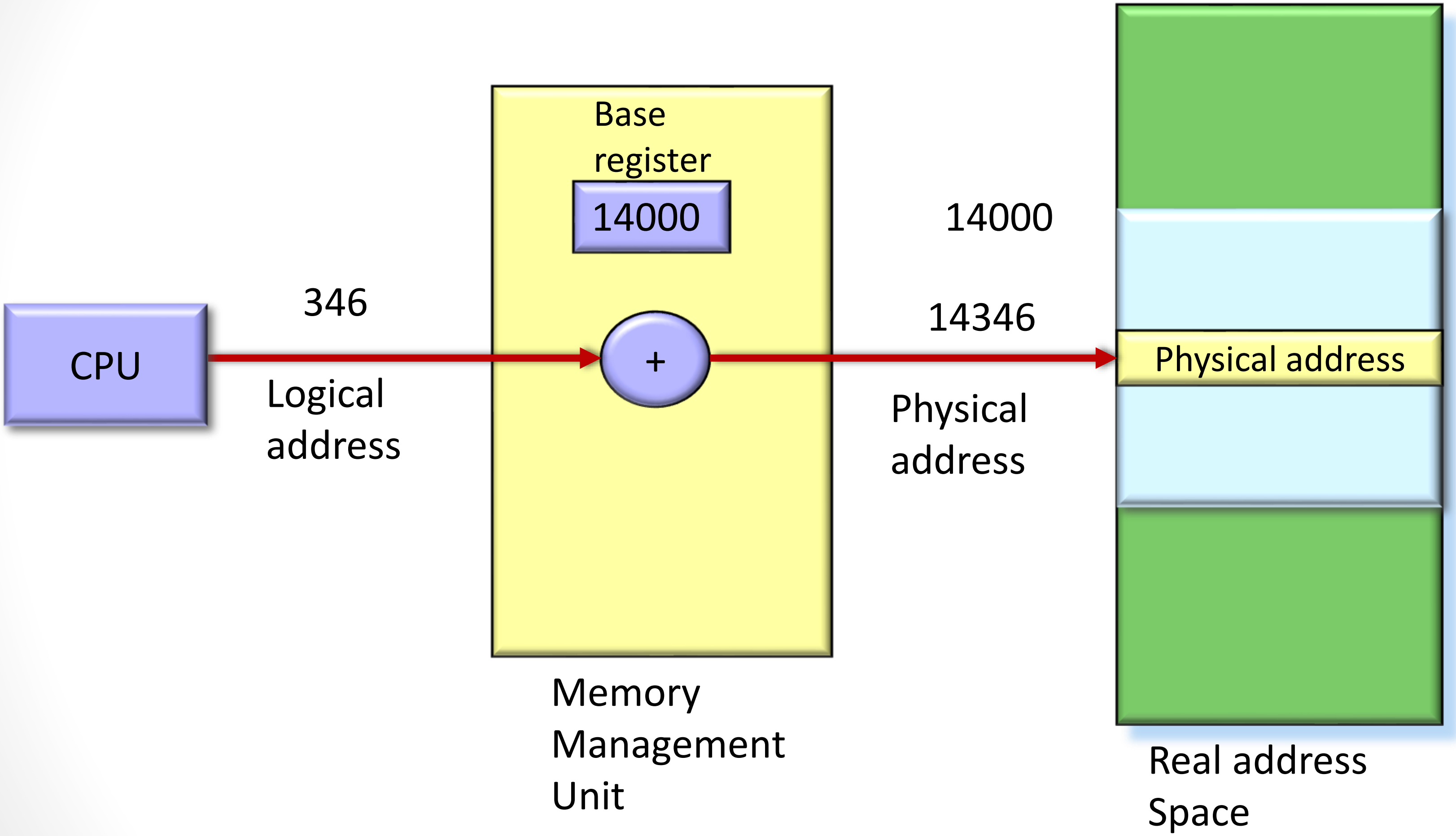
# Fixed-partition Memory Allocation

- Advantage
  - Simple
- Disadvantage
  - The degree of multiprogramming is constrained.
  - The size of each process is bounded.
  - Suffers **internal fragmentation**
    - Memory that is internal to a partition but is not being used

# Logical and Physical Addresses



# Logical and Physical Addresses




Dynamical address translation

# Variable-partition Memory Allocation

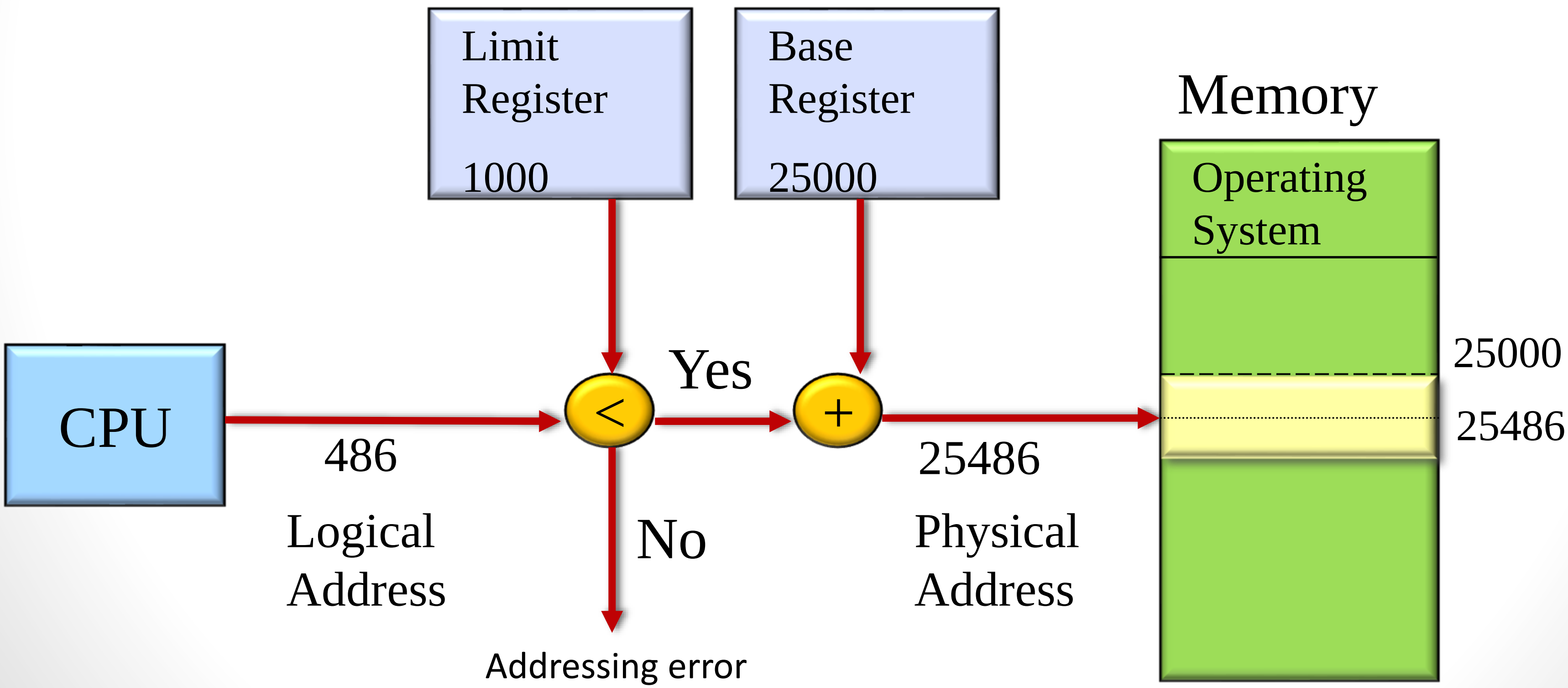
- Initially, all memory is considered as one large block of available memory, a **hole**.
- When a process needs memory, a hole large enough for the process is allocated for it.
- A free-memory list is used to track available memory.

# Memory Protection

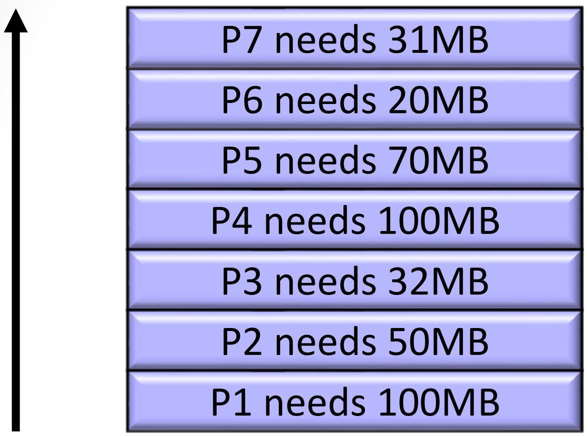
- **Base Register:**

  Start of (physical) memory allocated to a process.
- **Limit Register:**

  Amount of memory allocated to the process.

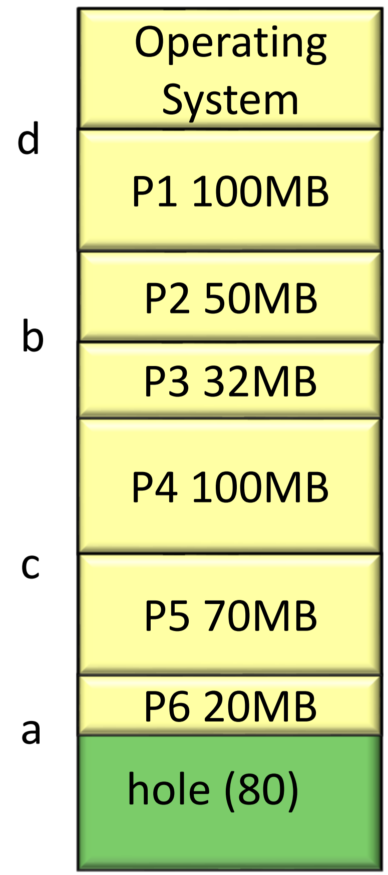


# Variable-Partition Memory Allocation




Which hole should P7 be put into?

Physical memory




Free-Memory List

| Starting address | length |
|---|---|
| a | 80MB |
| b | 32MB |
| c | 70MB |
| d | 100MB |

# Placement Strategies

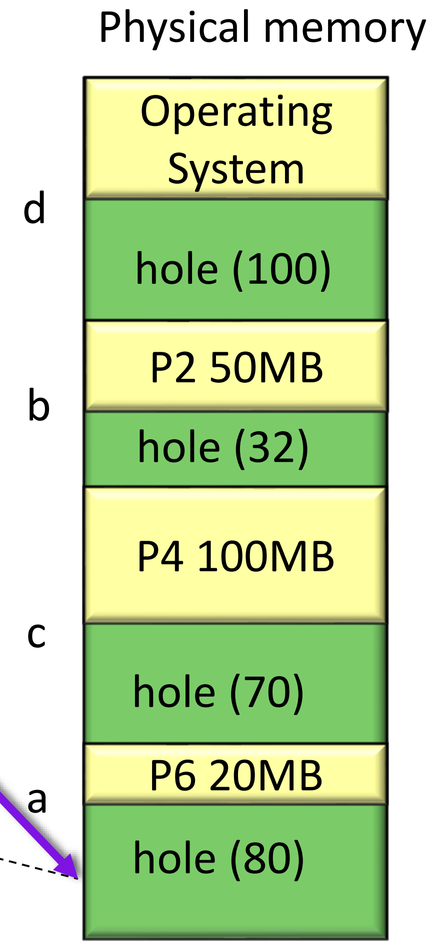
- **First-fit:** allocate the first hole large enough.
- **Best-fit:** allocate hole with the smallest leftover.
- **Worst-fit:** allocate the largest hole.

# First-Fit Strategy

P7 requests 31MB

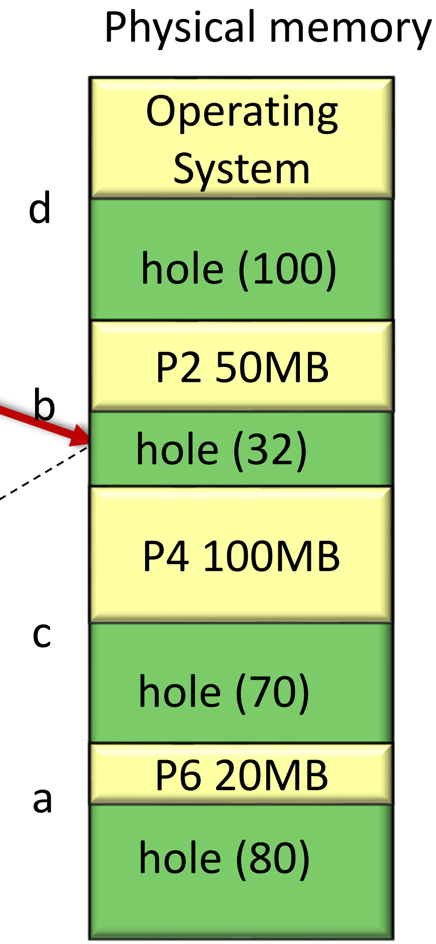
| Starting address | length |
|---|---|
| a | 80MB |
| b | 32MB |
| c | 70MB |
| d | 100MB |

Free-Memory List



# Best-Fit Strategy

P7 requests 31MB

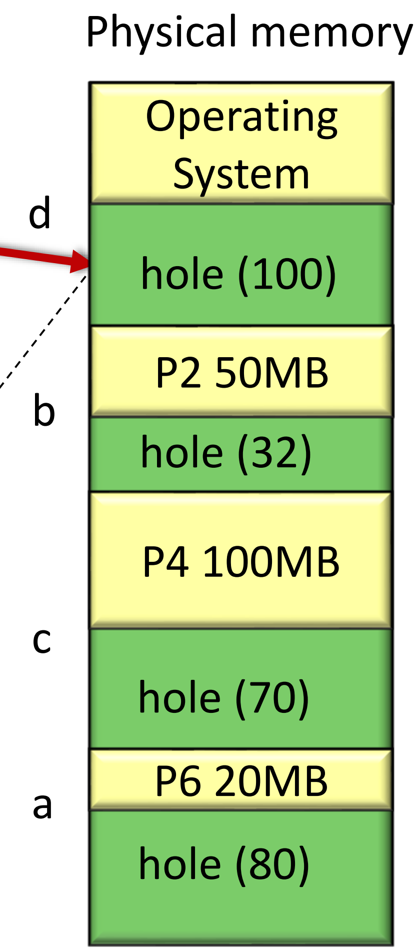



| Starting address | length |
|---|---|
| a | 80MB |
| b | 32MB |
| c | 70MB |
| d | 100MB |

Free-Memory List

# Worst-Fit Strategy

P7 requests 31MB




| Start address | length |
|---|---|
| a | 80MB |
| b | 32MB |
| c | 70MB |
| d | 100MB |

Free-Memory List

# An Example

Assume that we have 2560k of memory available and a resident OS of 400k.

Five processes arrive in the order given below. If they are CPU-scheduled in SJF order, how would the First-Fit algorithm allocate memory to them?
(The newly-freed memory hole is appended to the end of the free-memory list)

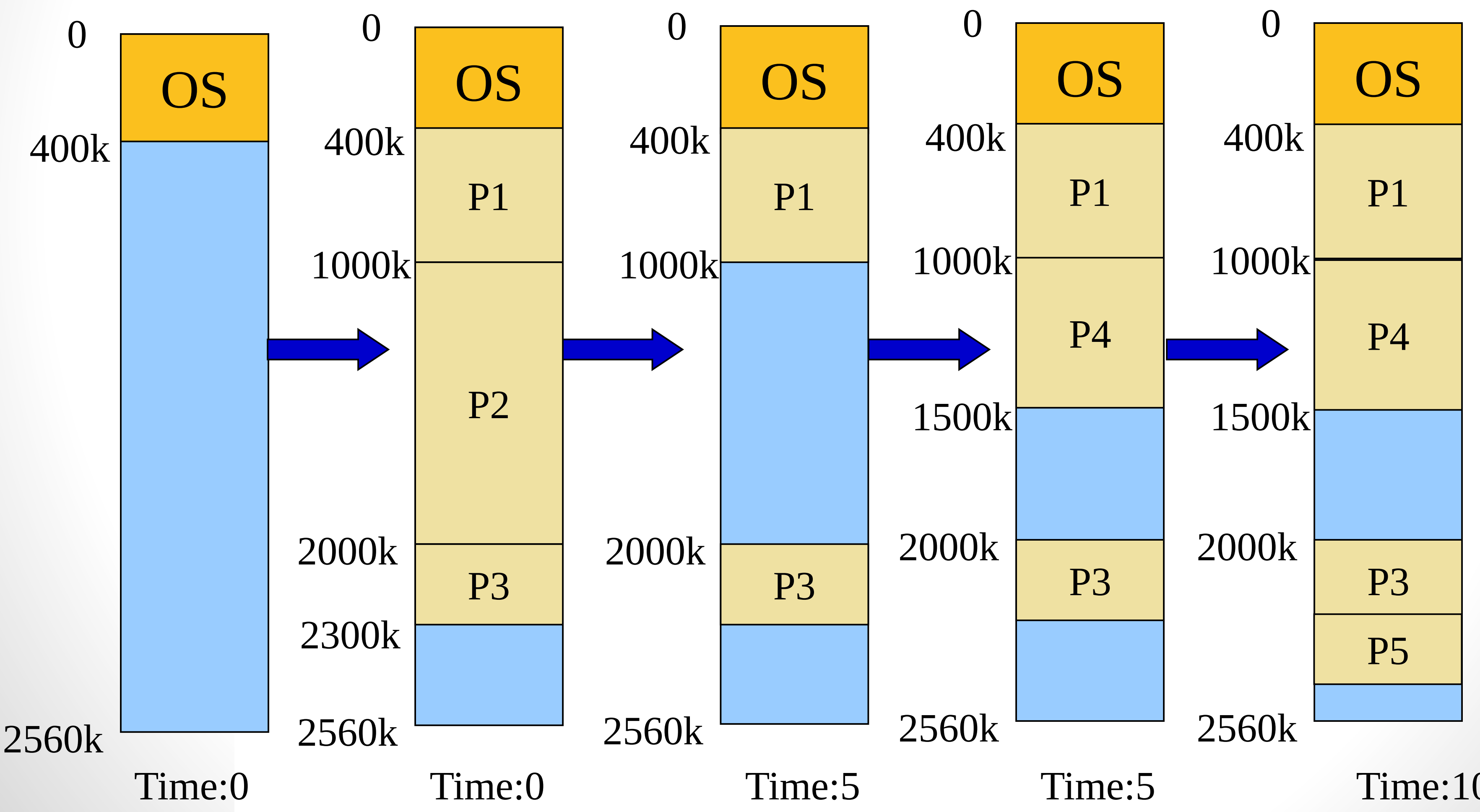
front →

| Process | Memory | Burst Time | Arrival Time |
|---|---|---|---|
| P1 | 600K | 10 | 0 |
| P2 | 1000K | 5 | 0 |
| P3 | 300K | 40 | 0 |
| P4 | 500K | 35 | 0 |
| P5 | 200K | 20 | 10 |

## Free-Memory List

| Starting address | length |
|---|---|
| 2500k | 60k |
| 1500k | 500k |

| Process | Memory | Burst Time | Arrival Time |
|---|---|---|---|
| P1 | 600K | 10 | 0 |
| P2 | 1000K | 5 | 0 |
| P3 | 300K | 40 | 0 |
| P4 | 500K | 35 | 0 |
| P5 | 200K | 20 | 10 |



# Variable-Partition Memory Allocation

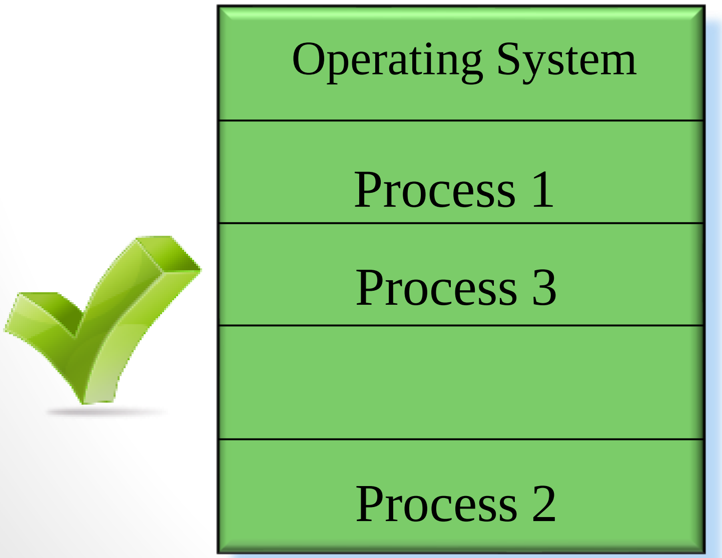
- No internal fragmentation
  - Allocated memory is just as much as the process requests
- Suffers from **external fragmentation**
  - After inserting/removing many processes, available memory space is broken into chunks
    ➔ Largest contiguous chunk is insufficient for a request, although total free memory is sufficient.

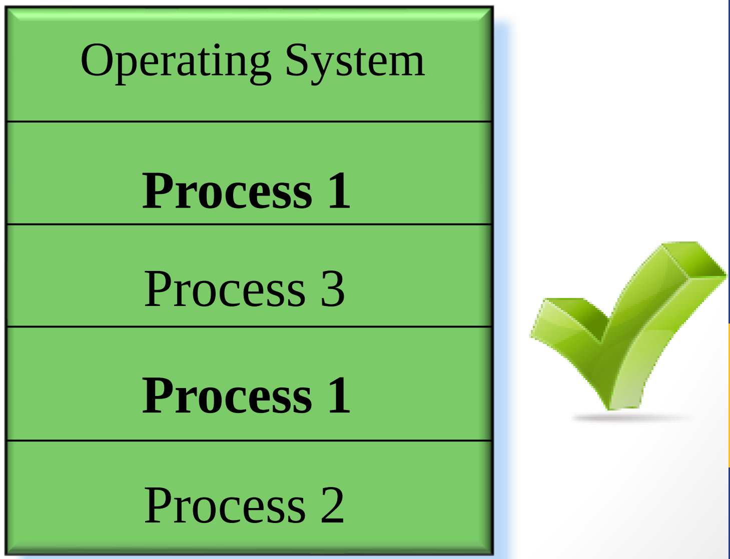
# Variable-Partition Memory Allocation

- Reducing External Fragmentation
  - Coalescing – merge **adjacent** holes to form a single, larger hole.
  - Memory compaction – relocate all occupied areas of memory to one end of main memory. This leaves a single large free memory hole.
  - Non-contiguous memory allocation (Next Lecture)
    - Paging
    - Segmentation
    - Segmentation with paging

# Non-contiguous Memory Allocation

- A program is divided into blocks or segments that the system may place in **nonadjacent** slots in main memory.

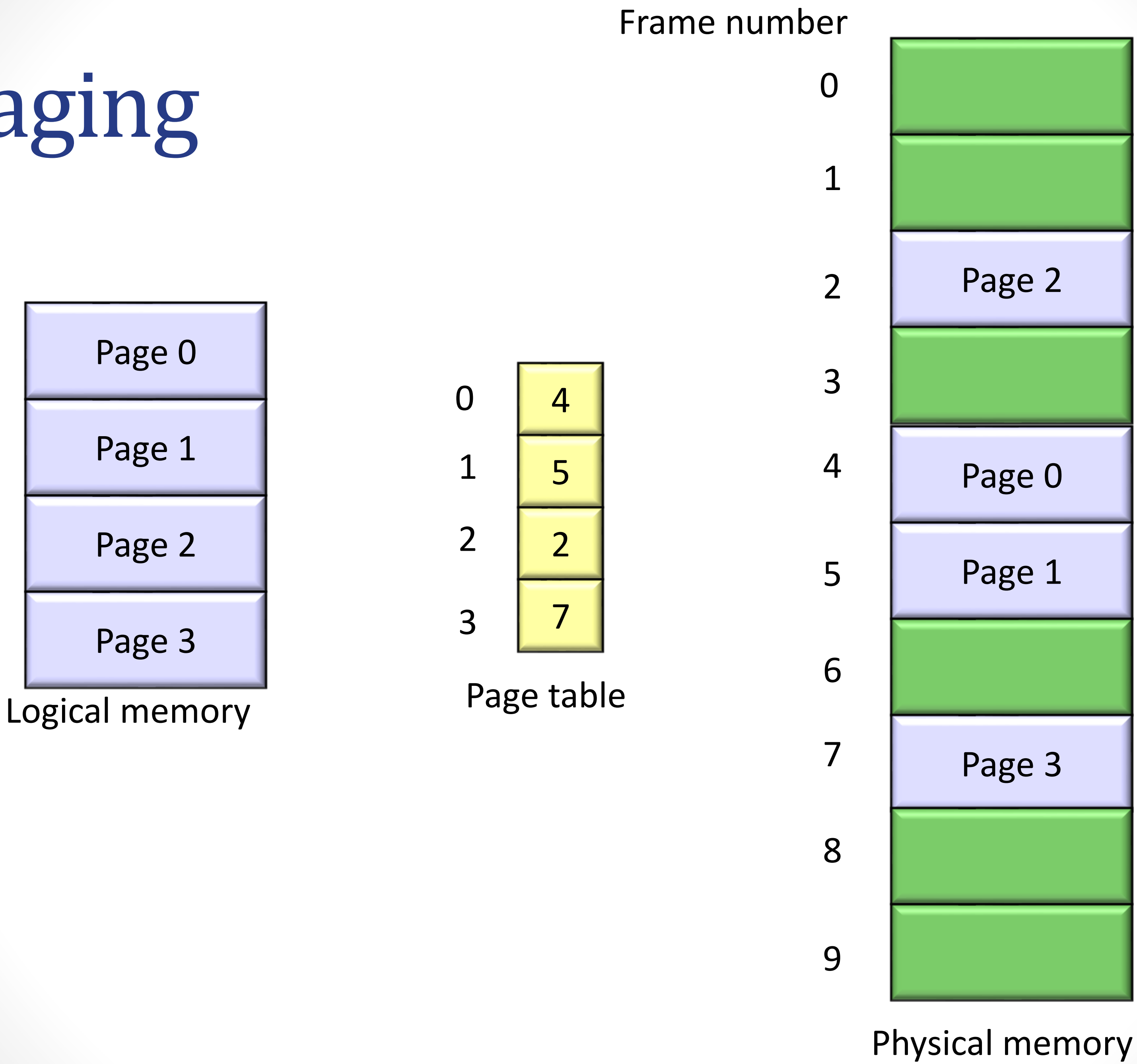


# Paging

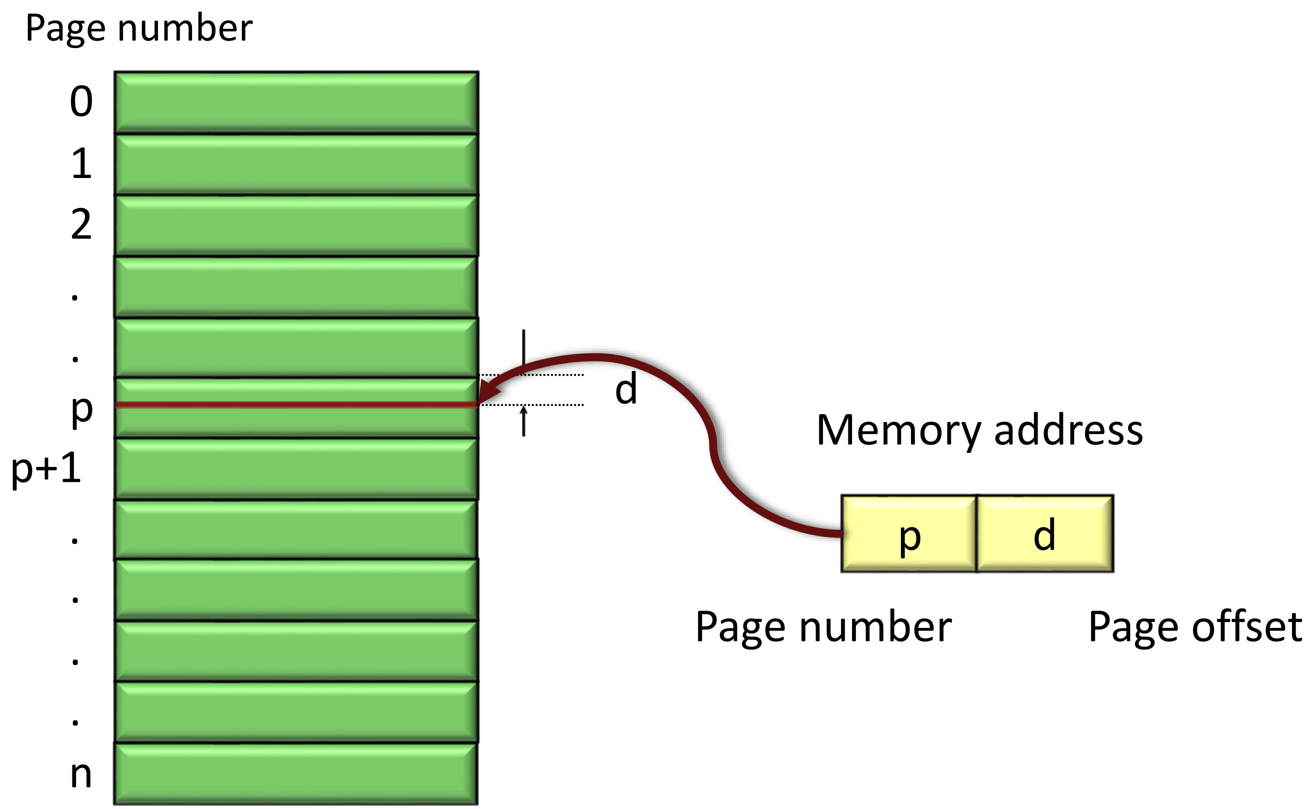
- Basic method
  - Break physical memory into fixed-sized blocks called **frames**.
  - Break logical memory into fixed-sized blocks called **pages.**
  - page size = frame size

# Paging




CM1205

# Representation of Memory Addresses

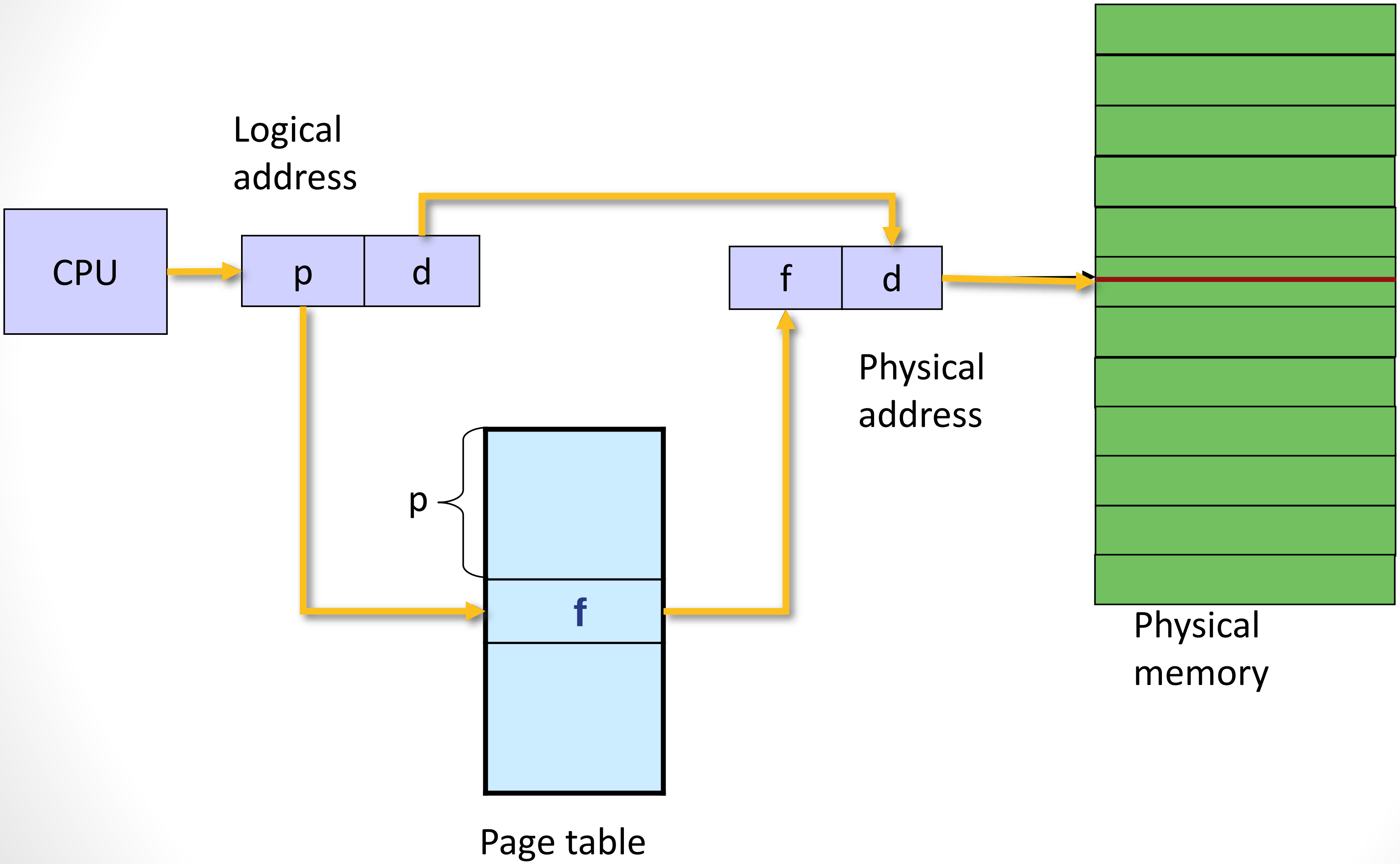


# Representation of Memory Addresses

- The page (frame) size is defined by the hardware. It is typically a power of 2.
  - If the size of logical address space is $2^n$ bytes, and a page size is $2^m$ bytes:
    - The high-level *n-m* bits of the logical address are used for the page number
    - The remaining *m* bits are used for the page offset.

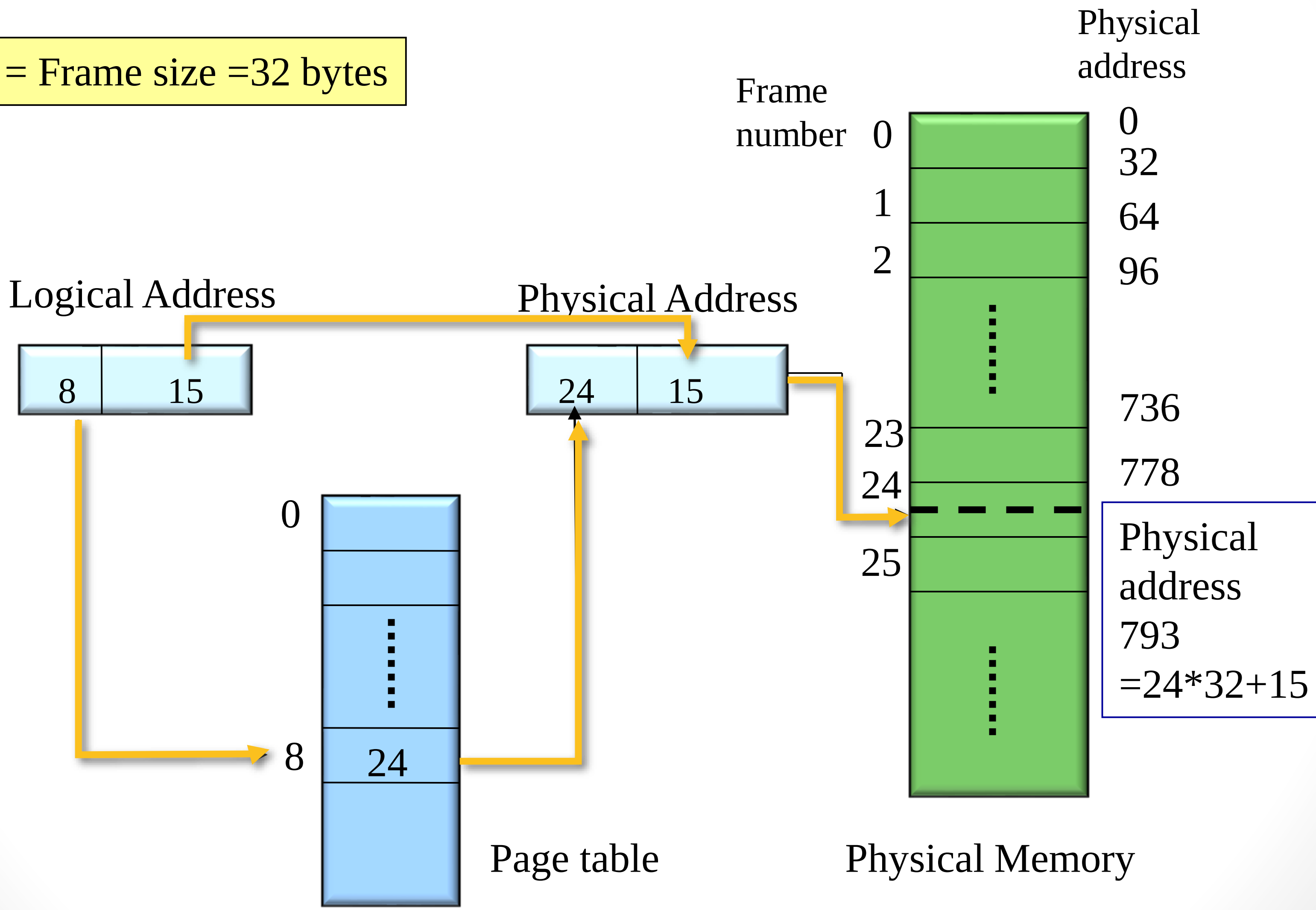
| Page number | Page offset |
|---|---|
| p | d |
| n-m bits | m bits |

# Paging Hardware

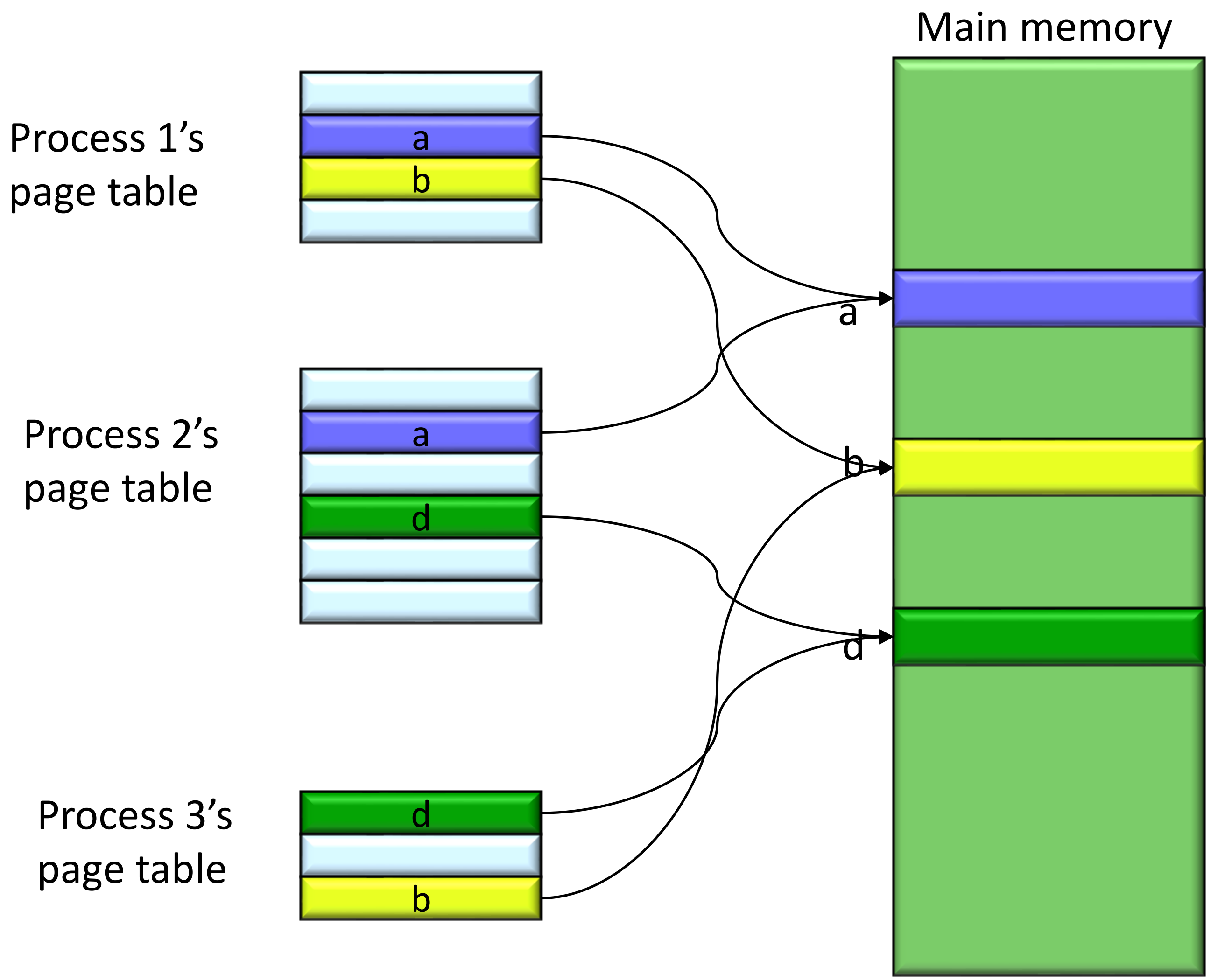


# An Example

Page size = Frame size =32 bytes



# Sharing in a Paging System



# Fragmentation

- It may cause internal fragmentation
  - Consider page size = 4KB, and a process requesting 5KB of memory.
  - Larger page size → worse problem
  - Very small page size?

# Summary

- Contiguous-Memory Allocation
  - Fixed-Partition
  - Variable-Partition
- Non-contiguous Memory Allocation
  - Paging