# Memory Management

**Prof J P Misra**
**BITS, Pilani**

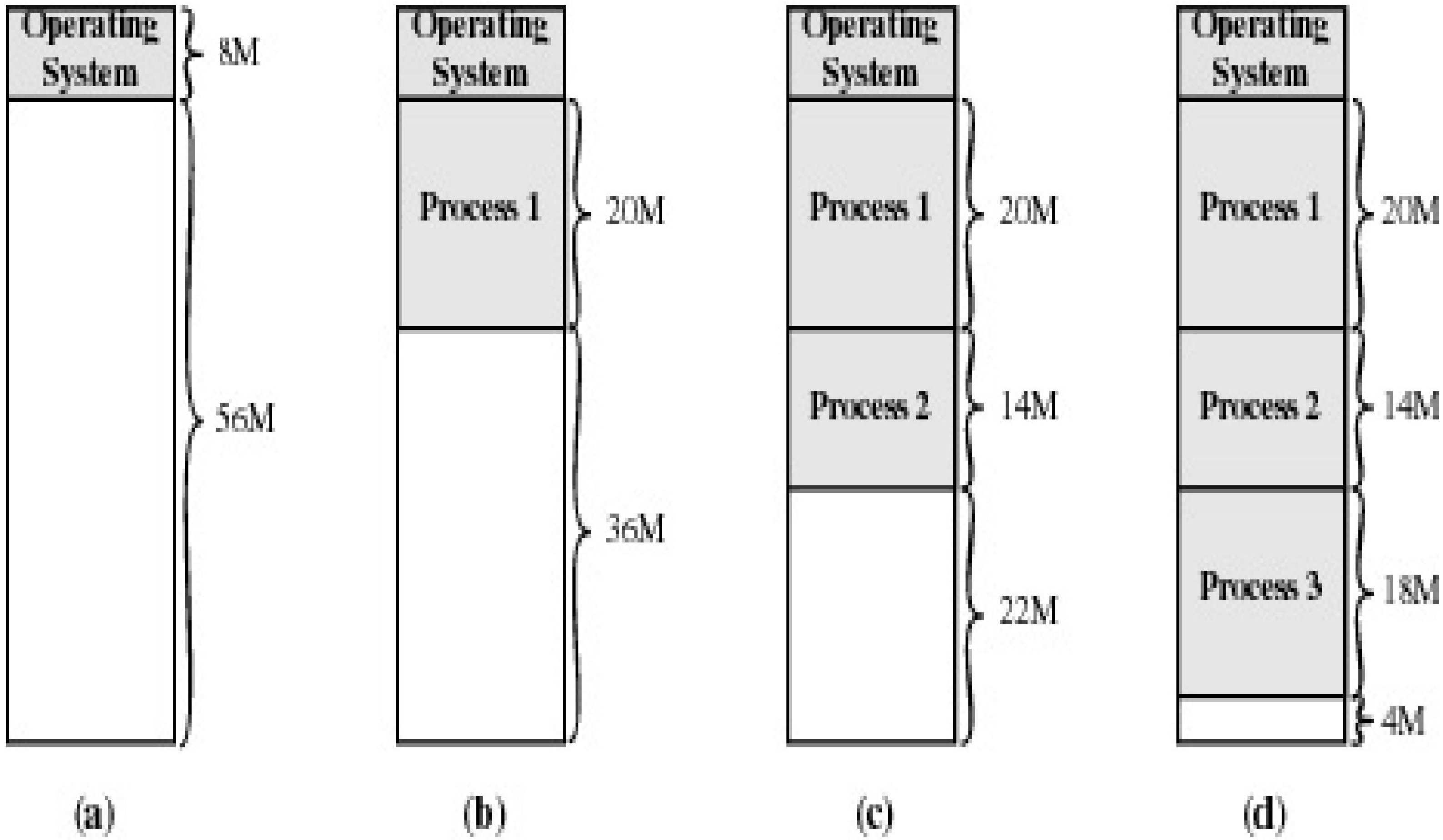
# We have learnt

- What is memory
- Types of memory
- Memory hierarchy
- Memory partioning

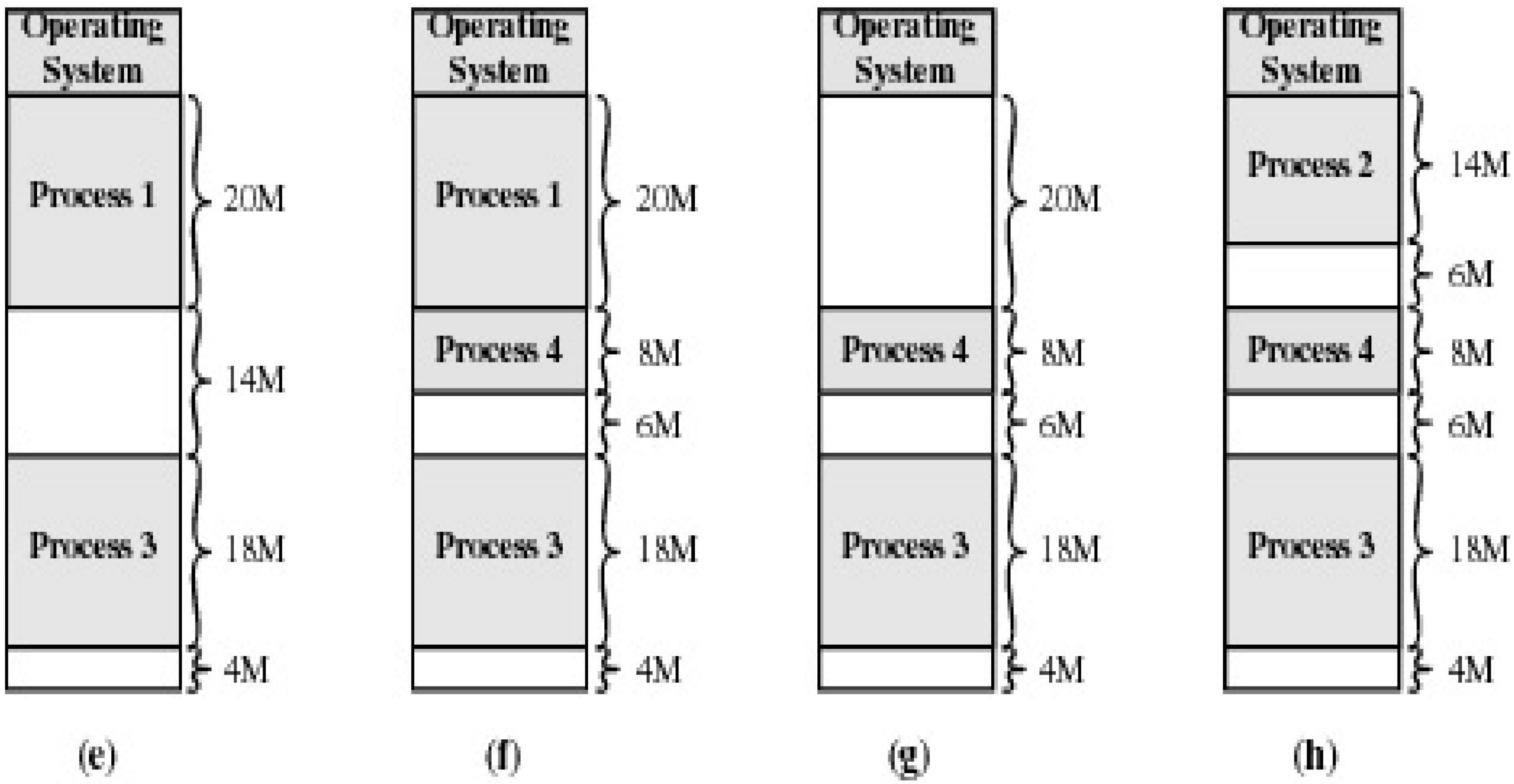
# Dynamic Partitioning

- Partitions are of variable length and number
- Process is allocated exactly as much memory as required

# Dynamic partitioning



# Dynamic partitioning : example



# Dynamic partitioning issues

- Dynamic partitioning leads to creation of unusable small size holes in memory known as external fragmentation
- To remove external fragmentation memory compaction can be used
  - Compaction is a technique which pushes all process towards one end of memory and creating a single large free memory block
- To support compaction , the system must support run time address binding

# Memory Allocation strategy

How to satisfy a request of size *n* from a list of free holes.

- **First-fit**: Allocate the *first* hole that is big enough
  - Fastest
  - May have many process loaded in the front end of memory that must be searched over when trying to find a free block
- **Next-fit**
  - More often allocate a block of memory at the end of memory where the largest block is found
  - The largest block of memory is broken up into smaller blocks

# Memory Allocation strategy

- **Best-fit** Allocate the *smallest* hole that is big enough; must search entire list, unless ordered by size.
  - Produces the smallest leftover hole
  - Compaction is required to obtain a large block at the end of memory
  - Since smallest block is found for process, the smallest amount of fragmentation is left memory compaction must be done more often

# Memory Allocation strategy

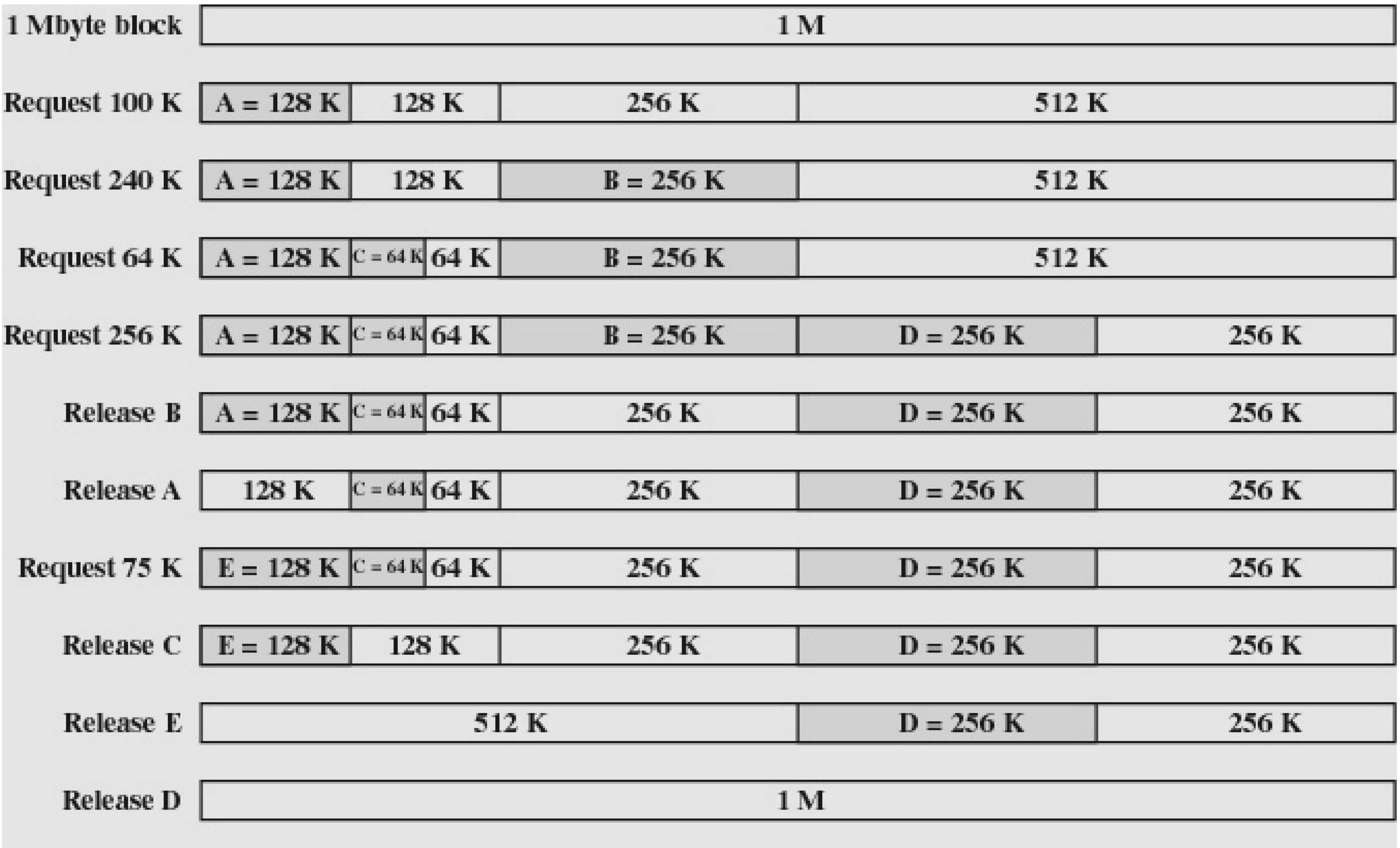
- **Worst-fit**: Allocate the *largest* hole; must also search entire list.
  - Produces the largest leftover hole.

- First-fit , next-fit and best-fit perform better than worst-fit in terms of speed and storage utilization.

# Buddy System

- In Buddy system memory blocks are available of size $2^K$ where $L <= K <= U$
- Entire space available is treated as a single block of $2^U$
- If a request of size s such that $2^{U-1} < S <= 2^U$, entire block is allocated
  - Otherwise block is split into two equal buddies
  - Process continues until smallest block greater than or equal to S is generated
- Buddy system is compromise to overcome the shortcoming of fixed and variable partitioning scheme

# Buddy System



# Fragmentation

innovate achieve lead

- External fragmentation –memory space exists to satisfy a request, but it is not contiguous.
- Internal fragmentation – allocated memory may be slightly larger than requested memory; The additional space is unused
  - 50 % rule. ( N allocated block & 0.5N waste)
- Reduce external fragmentation by compaction
  – Compaction is possible *only* if relocation is dynamic, and is done at execution time.
  – I/O problem
    – Latch job in memory while it is involved in I/O.
    – Do I/O only into OS buffers

# Underlying assumptions

- Allocate total required amount of memory to a process
- Allocate memory in contiguous manner

# Observations & solutions

- We realize that loading entire program in memory is wasteful as all the functionality of a program is not used simultaneously.
- Can We load on Demand ?
- When an attempt to bring code/data on demand
  - the memory may not be available at all
  - Memory may not be available in contiguous manner
- Relax the assumption of contiguous memory allocation

# On demand Memory management

- With Load on demand and Discontinuous allocation, we require

  – logical address to physical address mapping

  – Memory can be allocated in fixed size chunks or
  – Memory can be allocated in variable size chunks

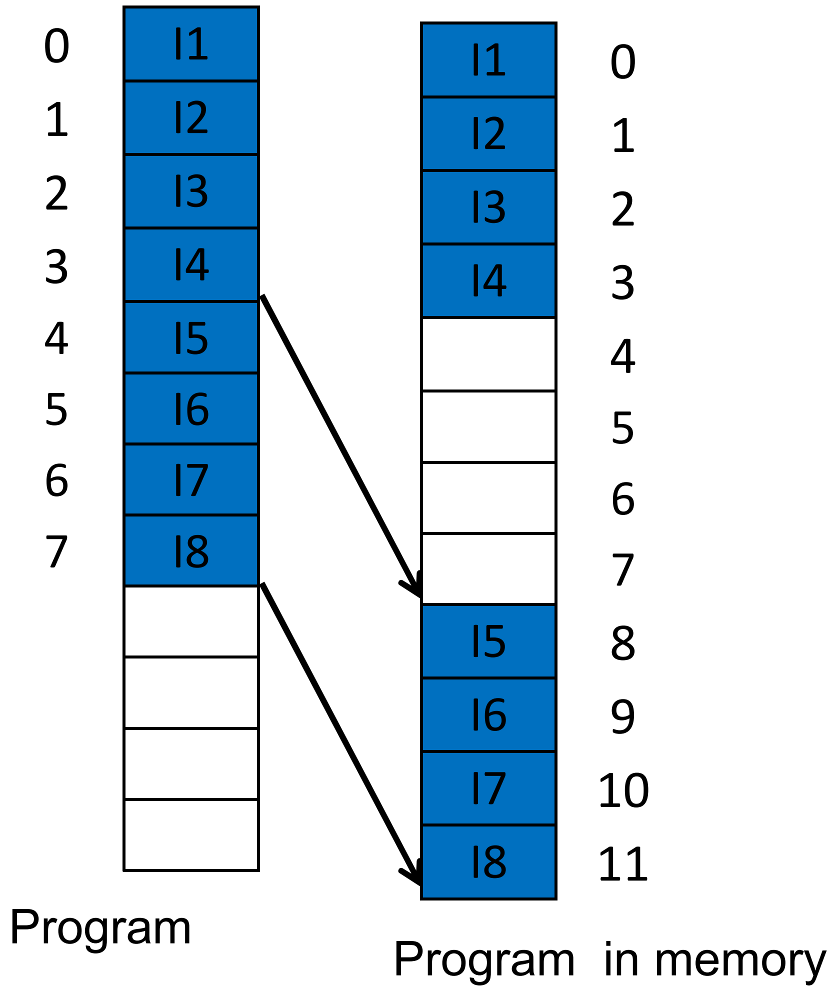
# Memory allocation issues

- Internal fragmentation (fixed Partition)
  - The system may not have enough memory to hold complete program( Overlaying technique  required)
- External fragmentation (variable Partition)
  - Compaction
- Contiguous allocation

# Dynamic Partitioning (fixed size)

- Memory is dynamically partitioned at run time and is allocated to processes.
- Logical address : is a reference to memory location independent of current assignment of data to memory
- The logical address space is divided into fixed size small chunks (usually 256 KB to 4 MB ) known as page
- The Physical memory is also divided into fixed size chunk and are known as frames.
- The size of frame and page is always equal for a given system
- The frame and page size are always power of 2. Mainly for easy address computation

# Program representation in Memory



# Memory allocation

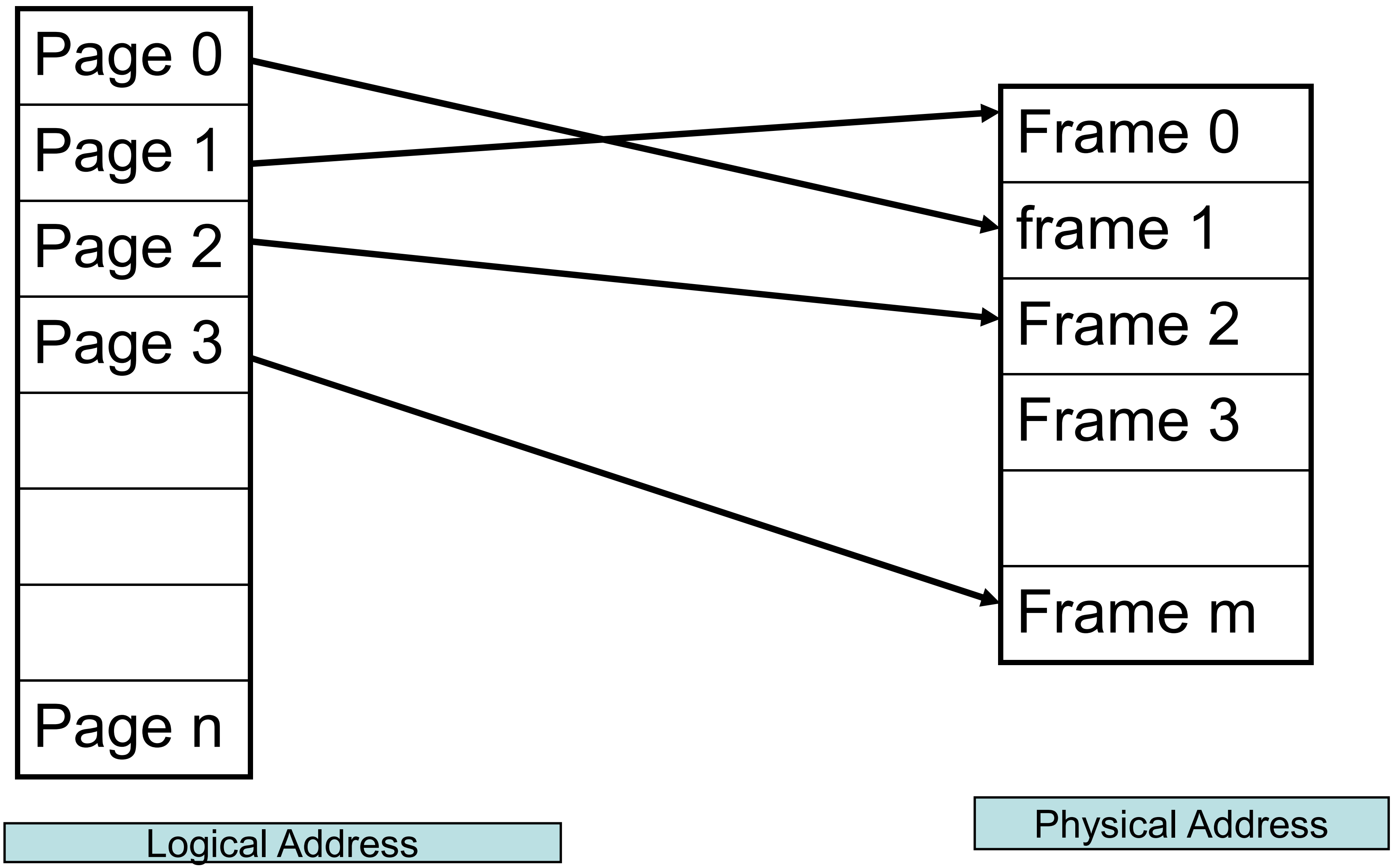
- When process pages are required to be loaded into system , OS allocates required number of frames where process pages are loaded.

- Frame need not be allocated in contiguous manner

- If we have a process of size 13KB and page size is 4KB then process is allocated 4 frames .

- Internal fragmentation occurs (18% unused space It is less sever as compared to fixed partitioning )

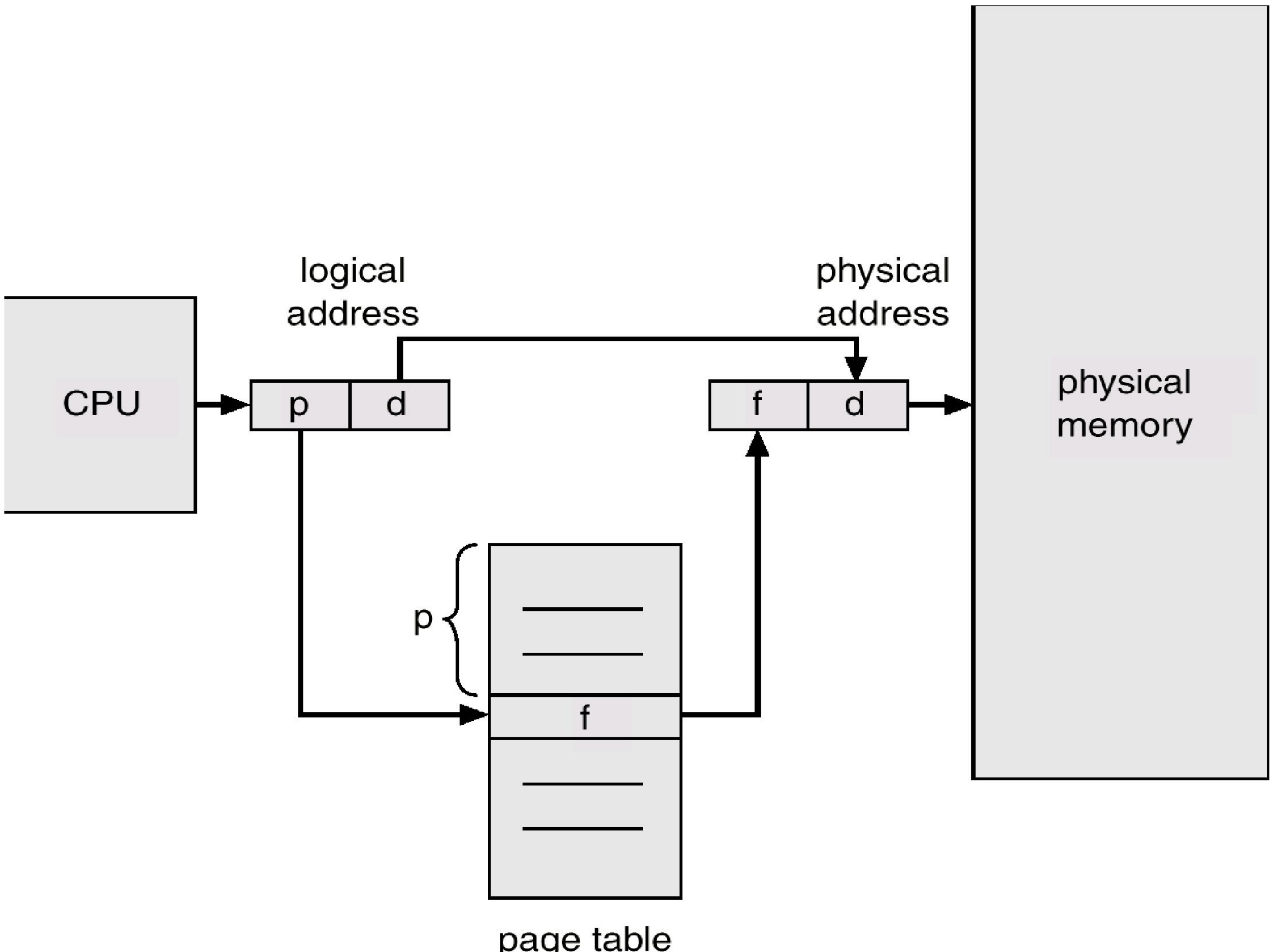
# Logical to Physical Address Mapping



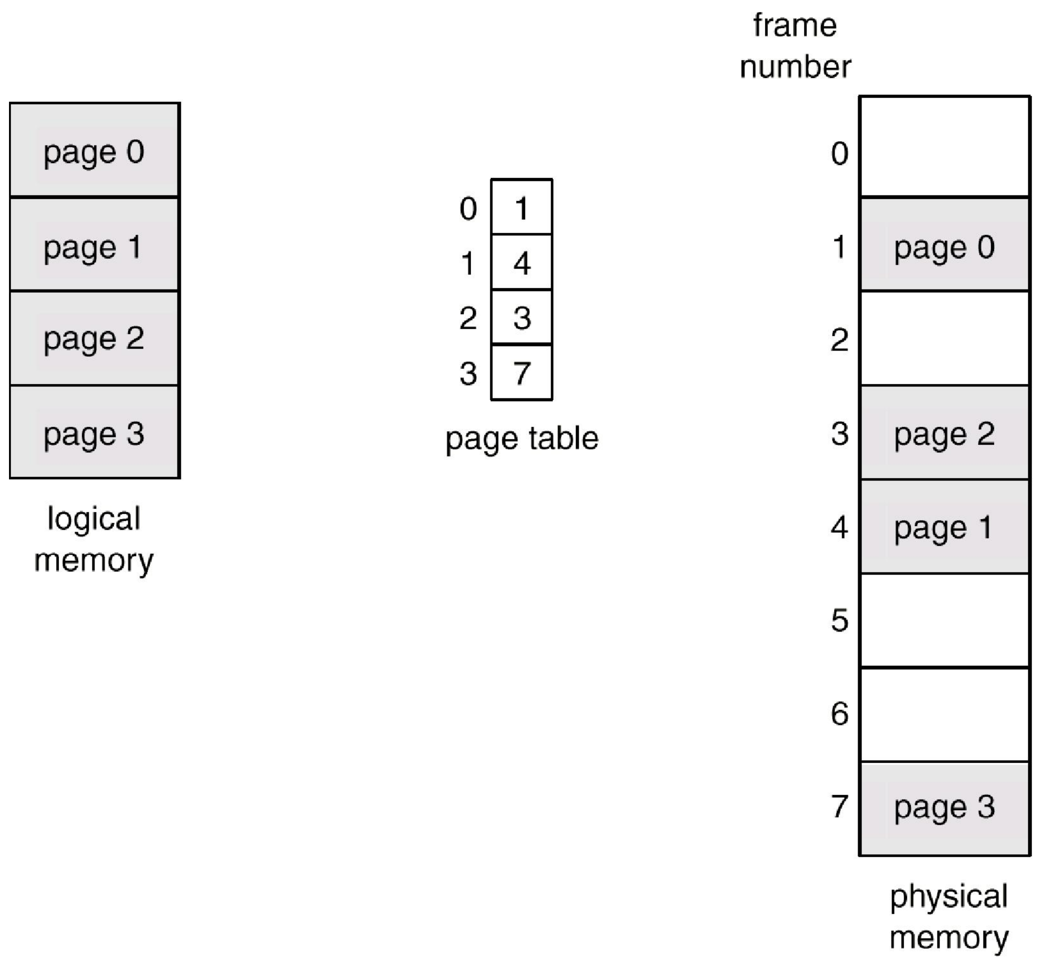
# Address Translation Example and Issues

- Consider 80X86 Processor
- Byte organized Memory
- 32 Bit IP (logical address space 4GB)
- Address lines 32 (maximum Physical Memory 4 GB)
- Page size 4KB

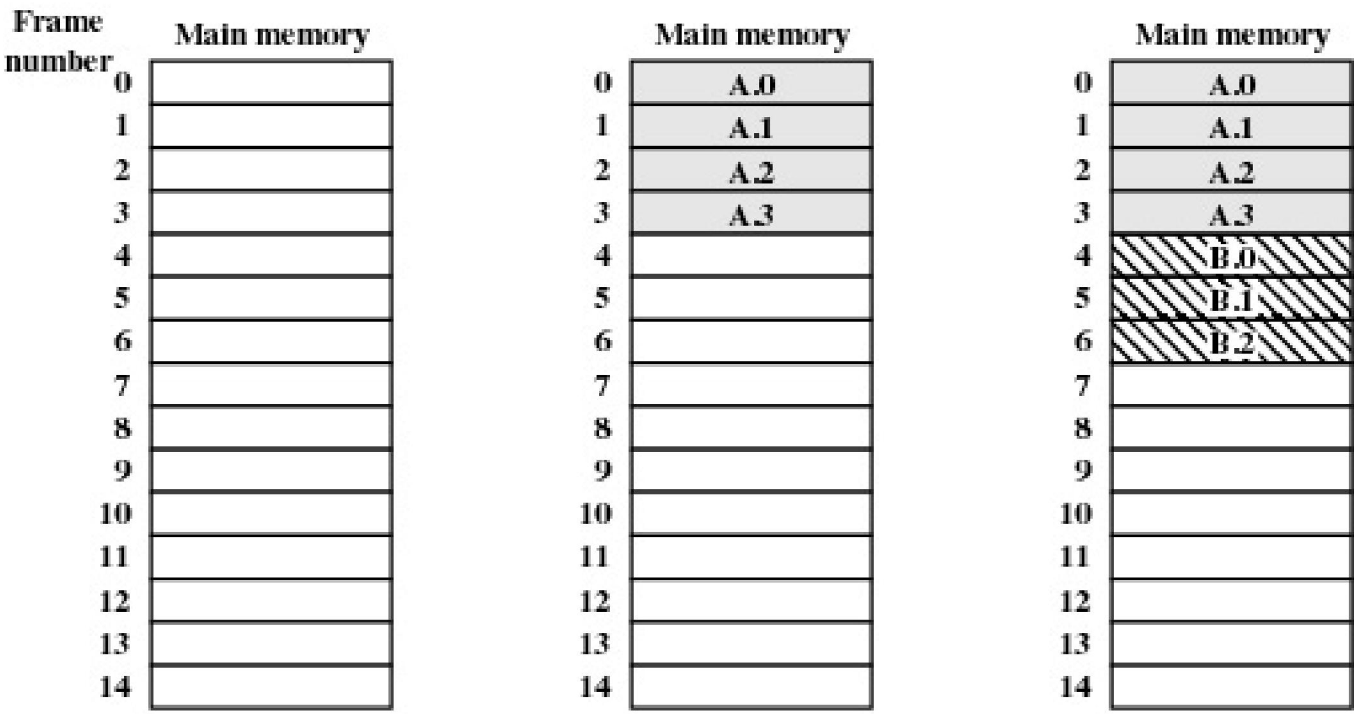
# Address Translation Architecture



# Paging Example

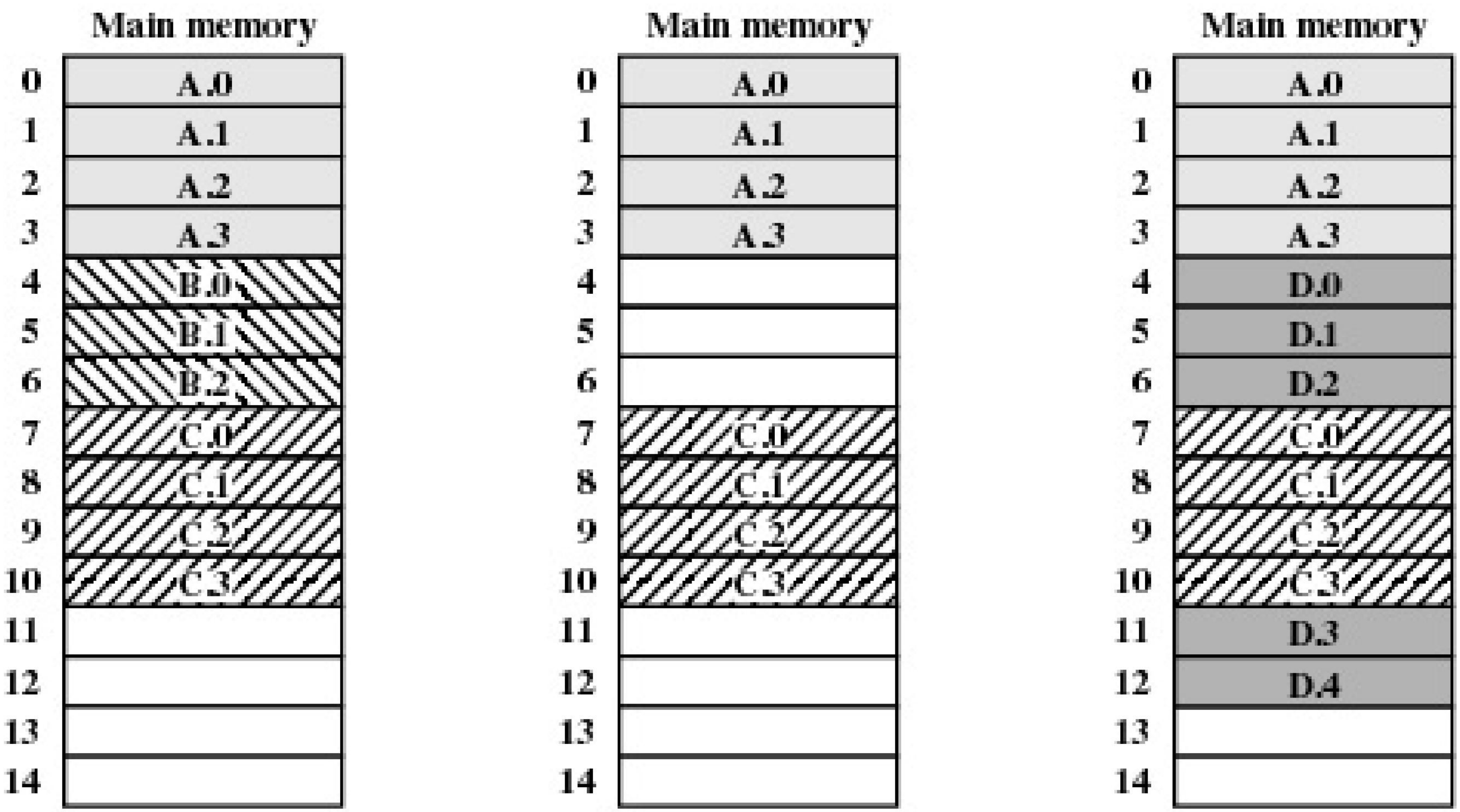


# Pages to free frame assignment




(a) Fifteen Available Frames (b) Load Process A (b) Load Process B

# Pages to free frame assignment




(d) Load Process C

(e) Swap out B

(f) Load Process D

| Page | Frame |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |

Process A page table

| Page | Frame |
|---|---|
| 0 | — |
| 1 | — |
| 2 | — |

Process B page table

| Page | Frame |
|---|---|
| 0 | 7 |
| 1 | 8 |
| 2 | 9 |
| 3 | 10 |

Process C page table

| Page | Frame |
|---|---|
| 0 | 4 |
| 1 | 5 |
| 2 | 6 |
| 3 | 11 |
| 4 | 12 |

Process D page table

| |
|---|
| 13 |
| 14 |

Free frame list

# Thank You