BITS Pilani
Pilani Campus

Computer Organization and Software Systems

Session 14

Prof. C R Sarma

Last Session

Contact Hour	List of Topic Title	Text/Ref Book/external resource

23-24	• Dead Lock detection
	• Memory Management
	  • Fixed Partition
	  • Dynamic Partition	

Today's Session

Contact Hour	List of Topic Title	Text/Ref Book/external resource

25-26	• Memory Management
	  • Paging
	  • Virtual Memory
		T2

Paging

- Fixed size and dynamic partitions are inefficient
  - Fixed size ➔ internal fragmentation
  - Dynamic➔ external fragmentation
- Paging helps to reduce internal fragmentation and completely eliminates external fragmentation

Paging

- Paging permits the physical address space of a process to be non- contiguous.
- Divide physical memory into fixed-sized blocks called **frames** (size is power of 2).
- Divide logical memory into blocks of same size called **pages**.
- Keep track of all free frames.
- To run a program of size *n* pages, need to find *n* free frames and load program.
- Set up a page table to translate logical to physical addresses.

Example

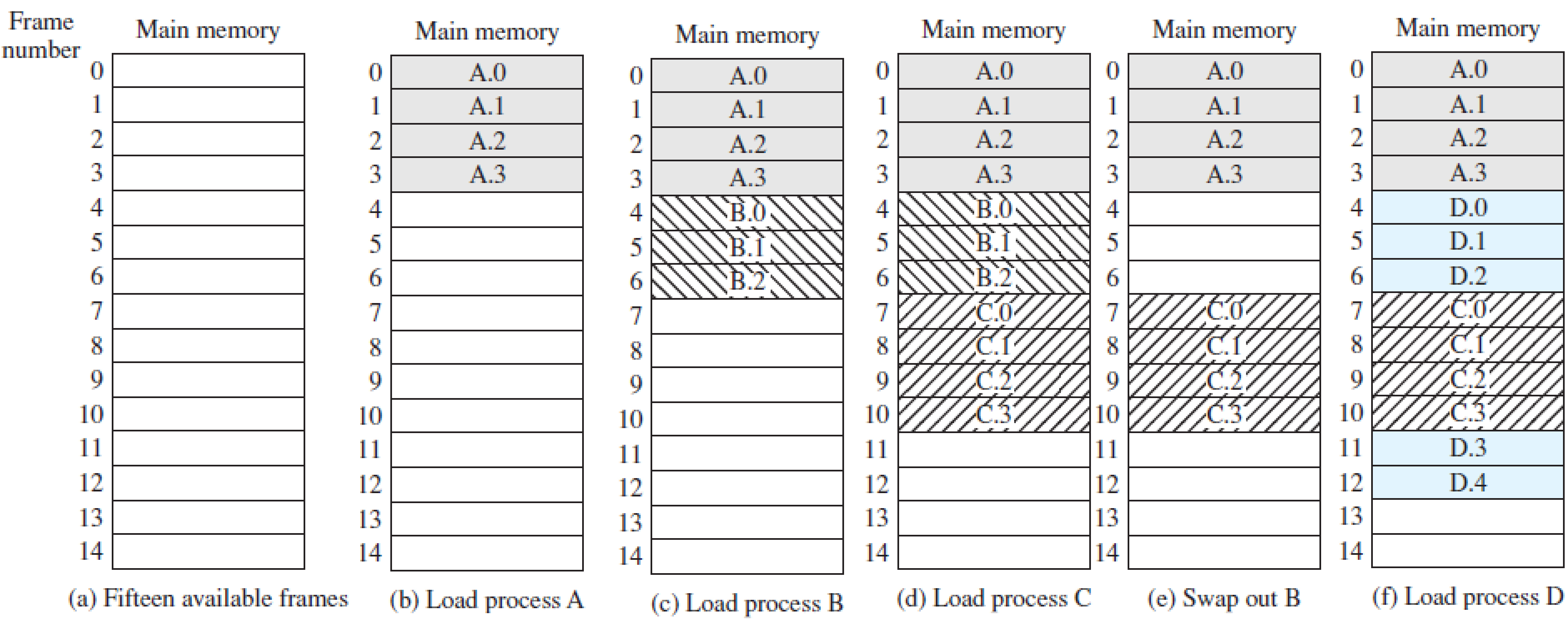
Process A : 4 pages

Process B : 3 pages

Process C : 4 Pages

Process D : 5 Pages

Main Memory : 15 frames

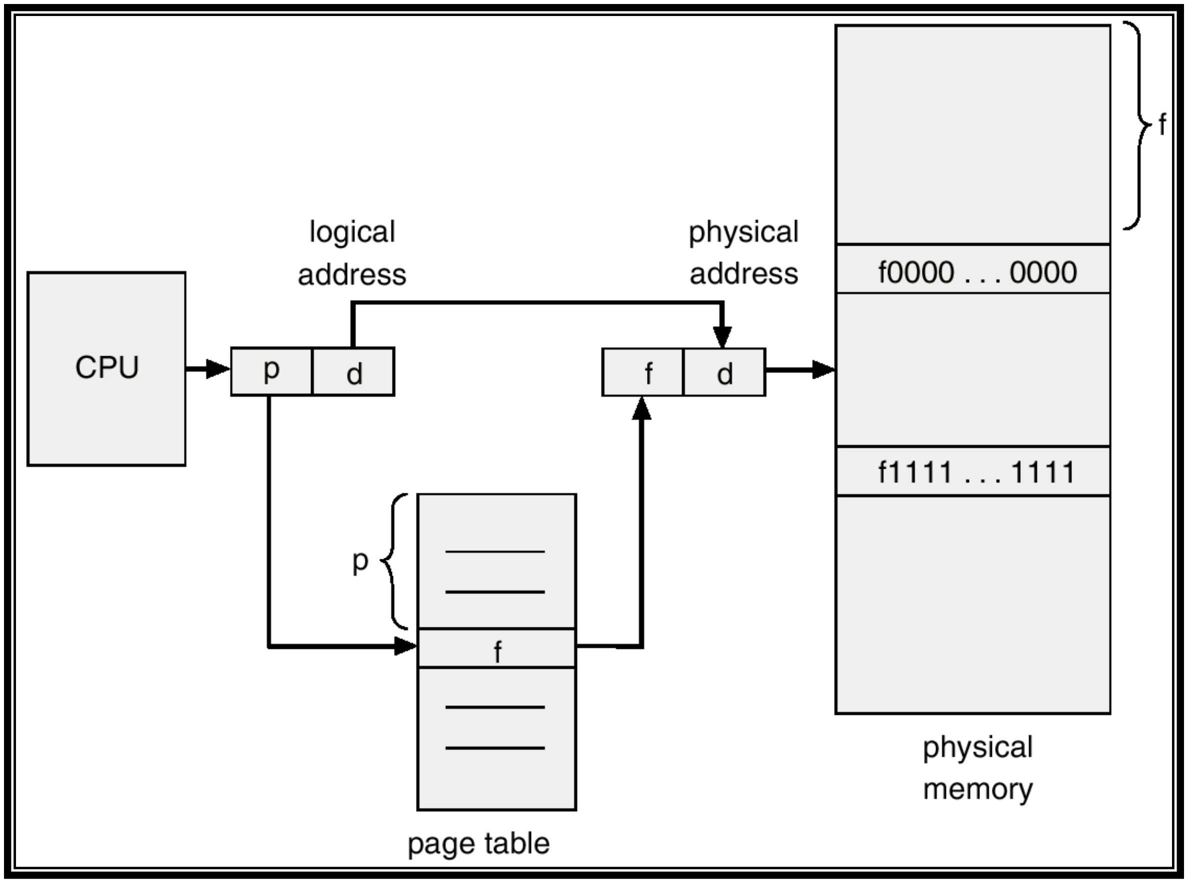


Address Translation Scheme

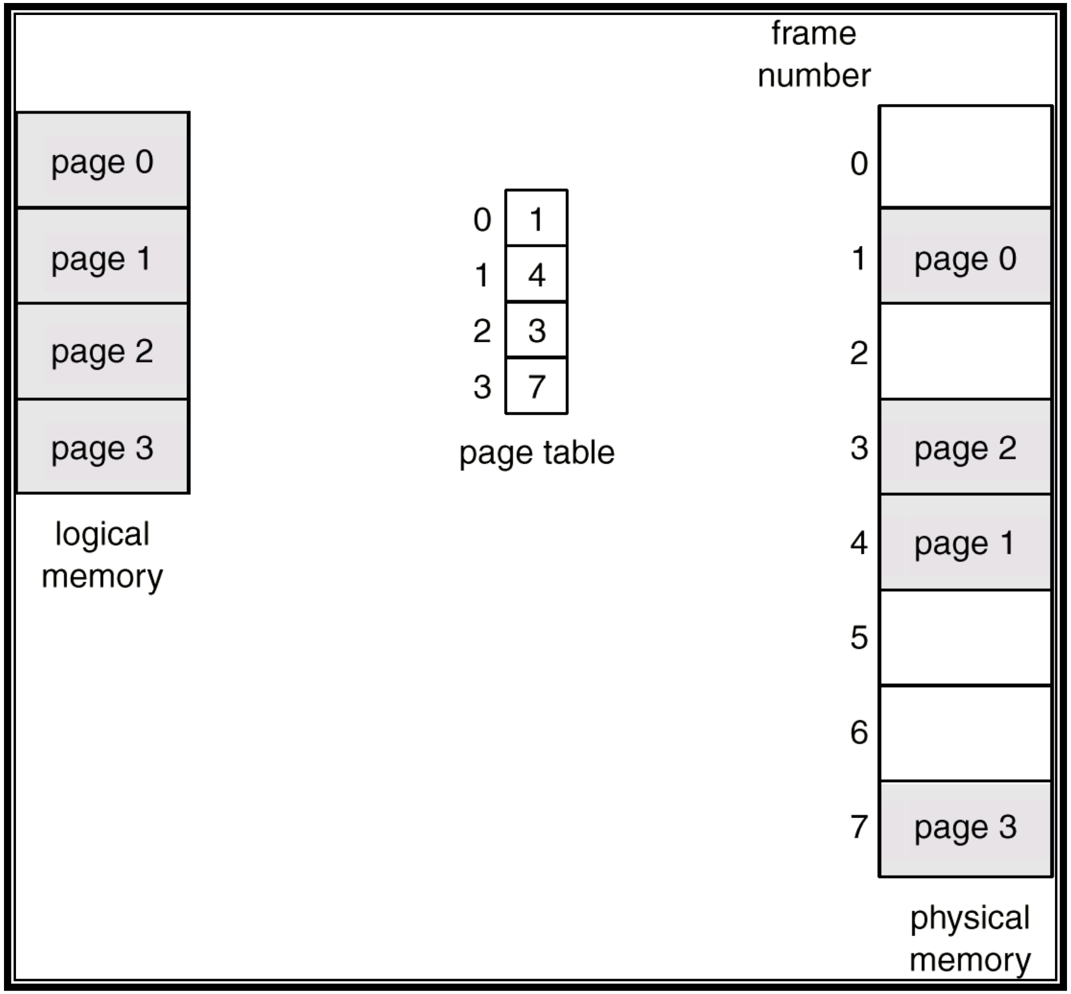
- Address generated by CPU is divided into:
  - *Page number (p)* – used as an index into a *page table* Page table contains base address of each page in physical memory.
  - *Page offset (d)* – combined with base address to define the physical memory address that is sent to the memory unit.
  - For given logical address space 2^m *and page size* 2^n

page number	page offset
p	*d*
m - n	*n*

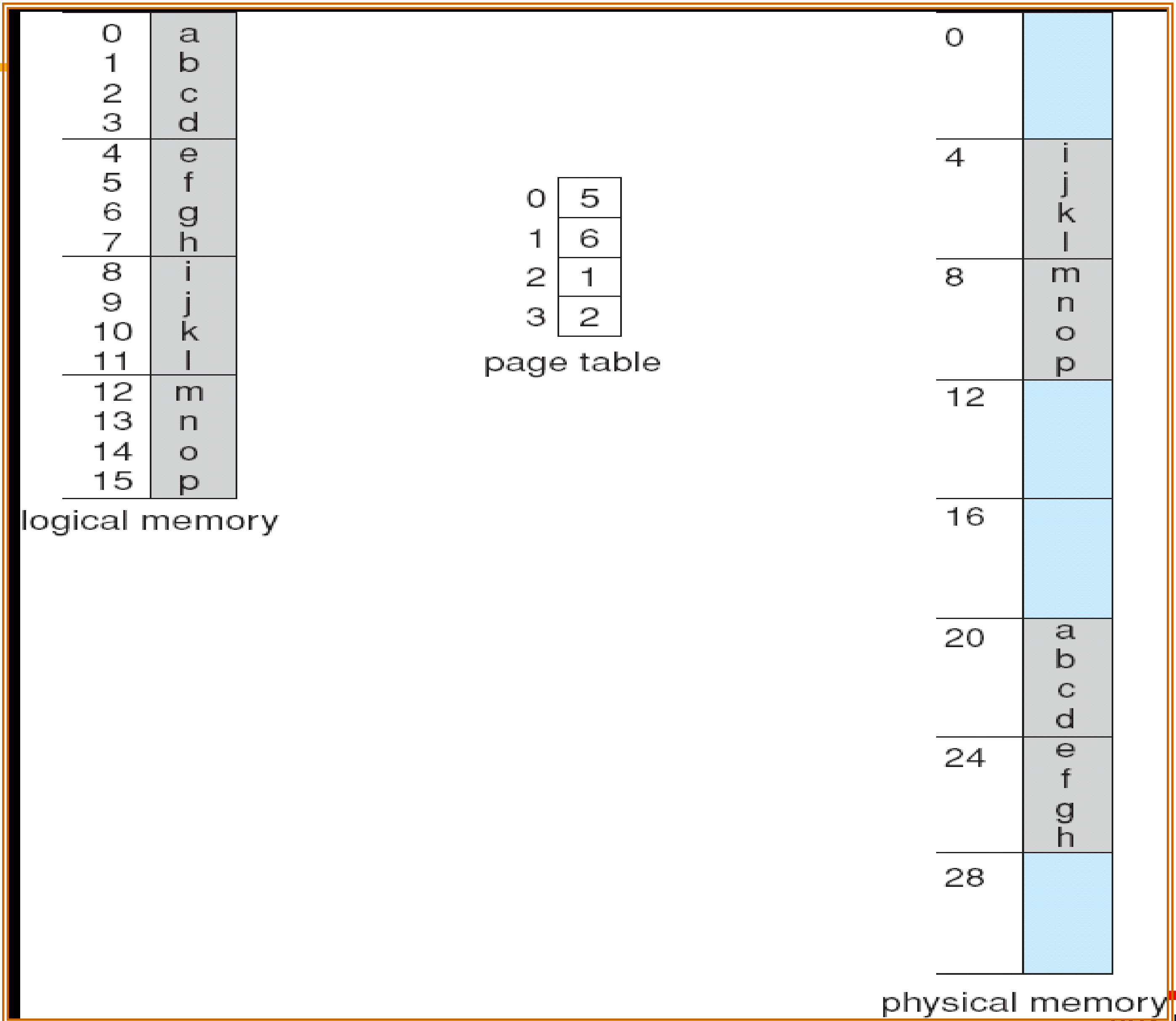
Address Translation Architecture



Paging Example



Paging Example



Important points….

- Paging is a form of dynamic relocation
- No external fragmentation but may have some internal fragmentation
- Small or large page size ?
- Page size determines
  - Memory wastage due to internal fragmentation
  - Size of the page table for a process
  - disk I/O data transfer
- small page : increases paging table overhead !!!!, internal fragmentation decreases
- large page : Decreases paging table overhead, internal fragmentation increases, disk I/O is more efficient when the number of data being transferred is larger

Contd…

- page size is usually 4 KB to 8 KB
- Needs to maintain the allocation details of main memory ➔ frame table

Problem

Consider a logical address space of 64 pages of 1,024 words each, mapped onto a physical memory of 32 frames.

a. How many bits are there in the logical address?
b. How many bits are there in the physical address?

Problem

Assuming a 1-KB page size, what are the page numbers and offsets for the following address references (provided as decimal numbers):

a. 3085

b. 42095

c. 215201

d. 650000

e. 2000001

Problem

Consider a computer system with a 32-bit logical address and 4-KB page size. The system supports up to 512 MB of physical memory. Find out #pages and #frames.

Example

Consider a simple paging system with the following parameters: 2^{32} bytes of physical memory; page size of 2^{10} bytes; 2^{16} pages of logical address space.

a. How many bits are in a logical address?

b. How many bytes in a frame?

c. How many bits in the physical address specify the frame?

d. How many entries in the page table?

e. How many bits in each page table entry? Assume each page table entry contains a valid/invalid bit

innovate achieve lead

BITS Pilani
Pilani Campus

Virtual Memory

Introduction

- How to increase the degree of multiprogramming ?
- Virtual memory is a technique that allows the execution of processes that are not completely in memory.
- Motivation:
  - Programs often have code to handle unusual error conditions which is almost never executed.
  - Arrays, lists, and tables are often allocated more memory than they actually need.
  - Certain options and features of a program may be used rarely
  - Principle of locality
  - trashing is a condition where system spends more time in swapping than executing instructions

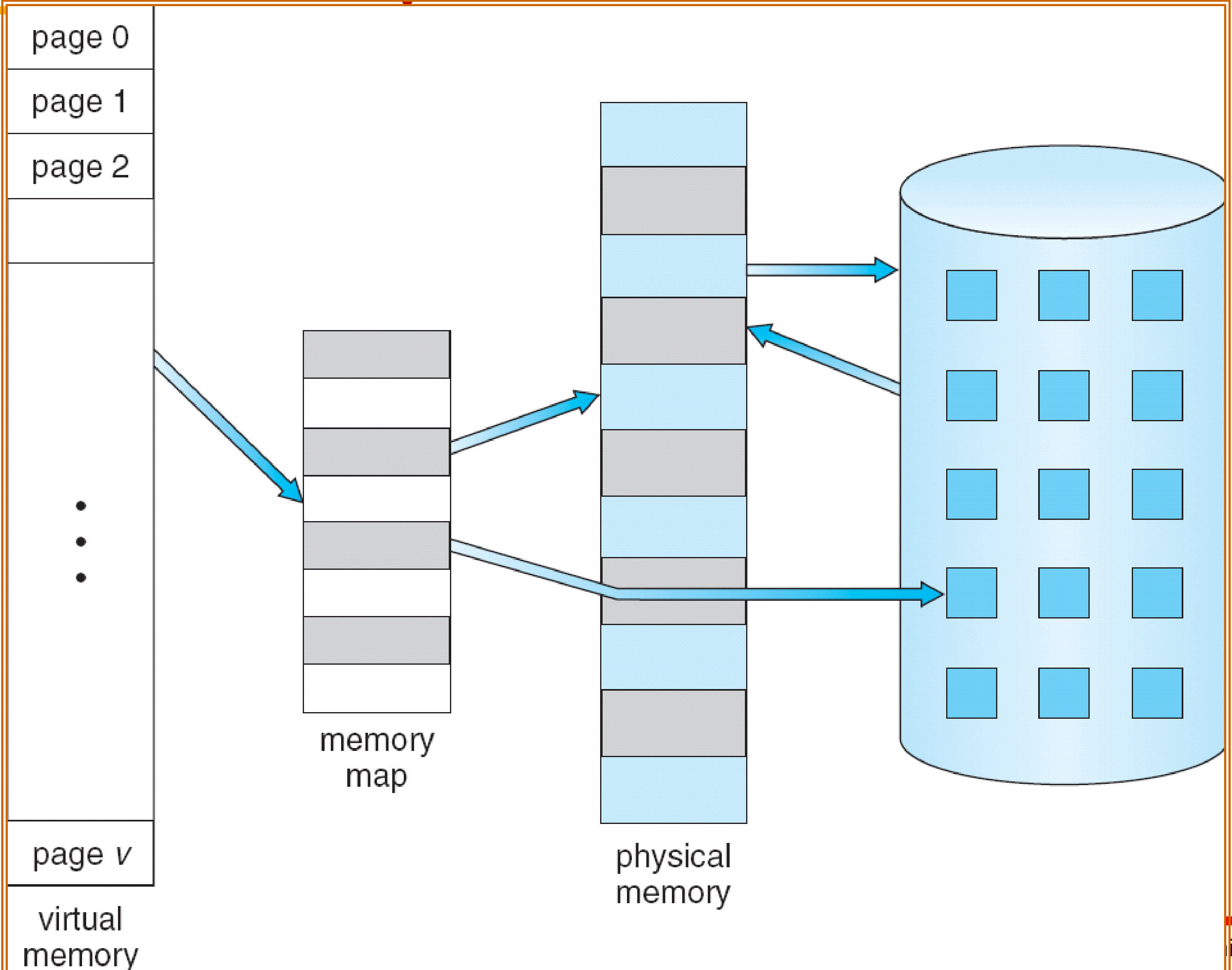
Contd…

- Advantages:
  - A program would no longer be constrained by the amount of physical memory that is available
  - Because each user program could take less physical memory, more programs could be run at the same time
  - increases the CPU utilization and throughput
  - Less I/O would be needed to load or swap each user program into memory, so each user program would run faster

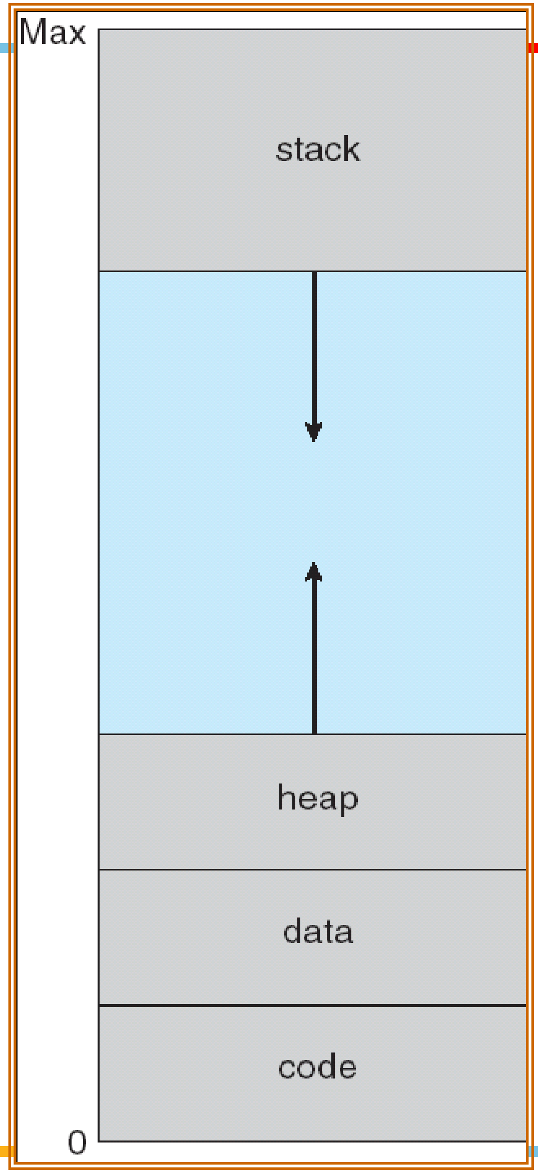
Background

- **Virtual memory** – separation of user logical memory from physical memory.
  - Only part of the program needs to be in memory for execution
  - Logical address space can therefore be much larger than physical address space
  - Allows address spaces to be shared by several processes
  - Allows for more efficient process creation

Virtual Memory That is Larger Than Physical Memory

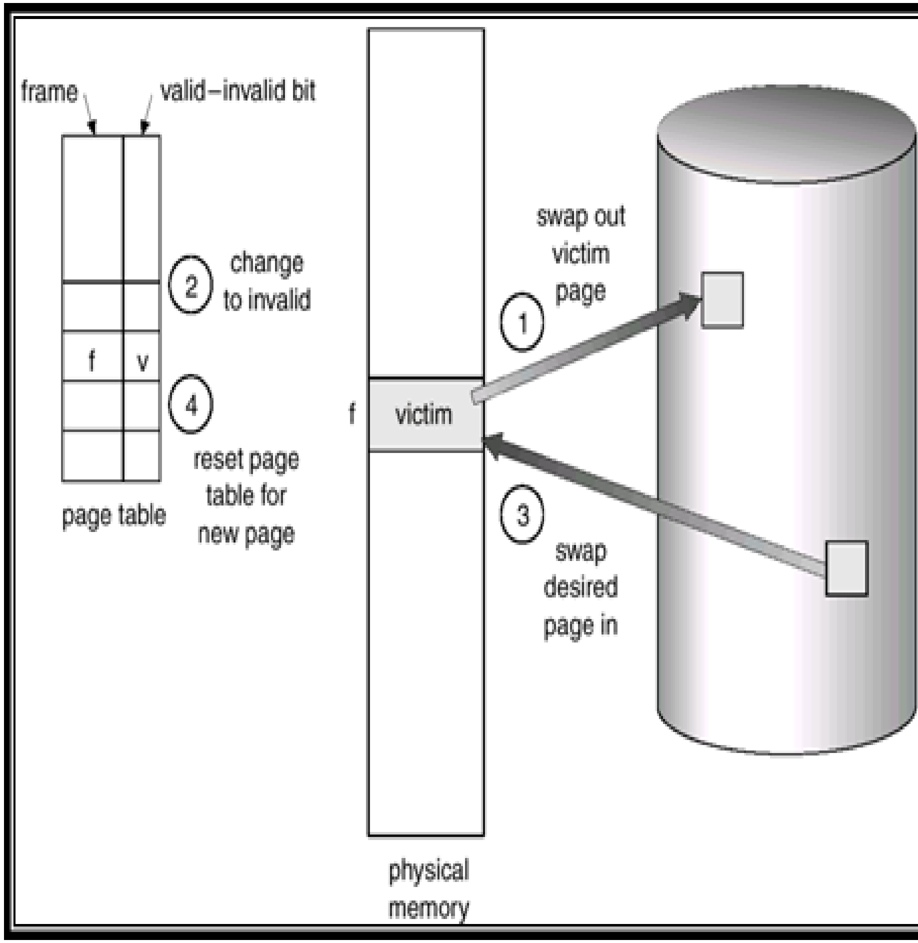


Virtual-address Space



What happens if there is no free frame?

- Page replacement – find some page in memory, but not really in use, swap it out
  – use a page replacement algorithm to select a *victim* frame
  – performance – want an algorithm which will result in minimum number of page faults
- Same page may be brought into memory several times



Contd…

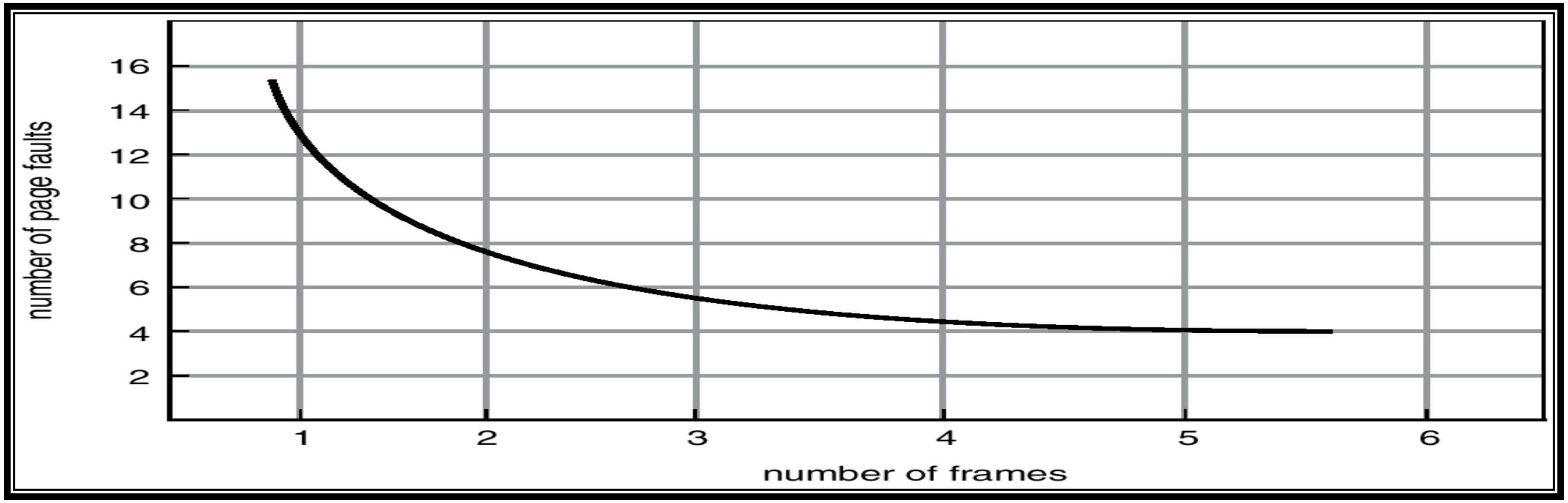
- if no frames are free, *two page transfers (one out and one in) are required.*
- Main drawback : doubles the page-fault service time and increases the effective access time accordingly.
- Use **modify (dirty) bit** to reduce overhead of page transfers – only modified pages are written to disk
- Demand paging requires two algorithms:
  - frame – allocation algorithm
  - page – replacement algorithm
- Frame allocation algorithm : handles frame allocation to each process.
- page – replacement algorithm : deals with the frames that are to be replaced

Procedure

1. Find the location of the desired page on the disk.
2. Find a free frame:
   a) If there is a free frame, use it.
   b) If there is no free frame, use a page-replacement algorithm to select a victim frame.
   c) Write the victim frame to the disk; change the page and frame tables accordingly.
3. Read the desired page into the newly freed frame; change the page and frame tables.
4. Restart the user process.

Page Replacement Algorithms

- Want lowest page-fault rate.



- Evaluate algorithm by running it on a particular string of memory references (reference string) and computing the number of page faults on that string.
- In all our examples, the reference string is
  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.

First-In-First-Out (FIFO) Algorithm

Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

3 frames (3 pages can be in memory at a time per process)

1	1	4	5	
2	2	1	3	9 page faults
3	3	2	4	

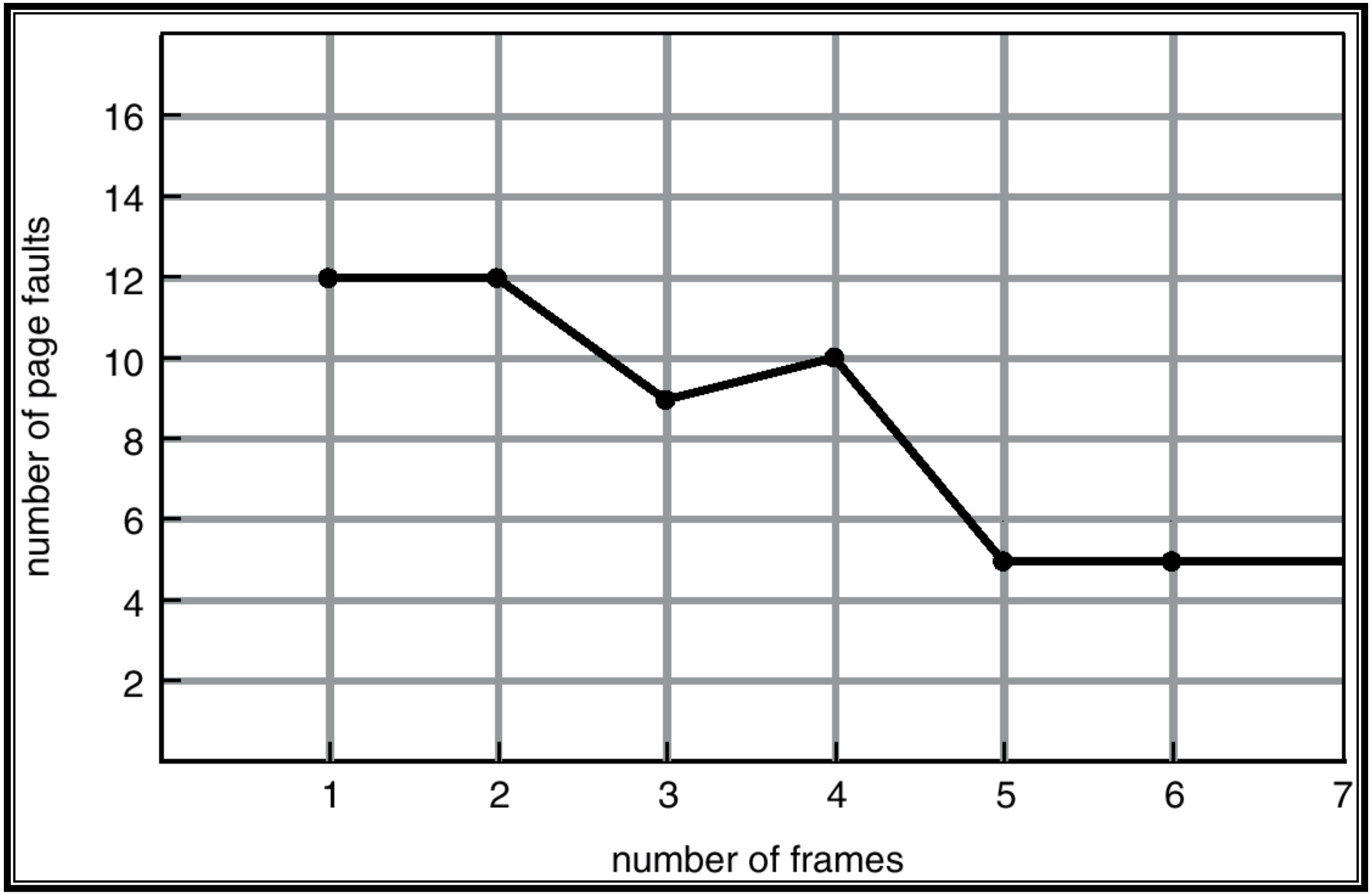
4 frames

1	1	5	4	
2	2	1	5	10 page faults
3	3	2		
4	4	3		

FIFO Replacement – Belady's Anomaly

- more frames $\Rightarrow$ more page faults

FIFO Illustrating Belady's Anamoly



Example 2

Consider the following sequence of address

0100, 0432, 0101,0612, 0102, 0103, 0104, 0101, 0611, 0102, 0103, 0104,0101,0610, 0102, 0103, 0104, 0101, 0609, 0102, 0105

Assume 100 byte page.

Find out the reference string.

Replacement Algorithm

Given page reference string: 1,2,3,4,2,1,5,6,2,1,2,3,7,6,3

Compare the number of page faults for FIFO, Optimal and LRU page replacement algorithm

Solution : FIFO

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3

Problem3: Replacement Algorithm

Given page reference string: 1,2,3,4,2,1,5,6,2,1,2,3,7,6,3

Compare the number of page faults for FIFO, Optimal and LRU page replacement algorithm

Solution : Optimal ➔ **Replace page that will not be used for longest period of time**

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3

Problem3: Replacement Algorithm

Given page reference string: 1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2

Compare the number of page faults for FIFO, Optimal and LRU page replacement algorithm

Solution : LRU

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3

Problem

Consider the following reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

For <u>Optimal and LRU</u> replacement algorithm with frames = 2, 3, 4, check whether it exhibits Belady's Anomaly or not.

Belady's anomaly : more frames $\Rightarrow$ more page faults

Optimal

1	2	3	4	1	2	5	1	2	3	4	5

1	2	3	4	1	2	5	1	2	3	4	5

1	2	3	4	1	2	5	1	2	3	4	5

LRU

1	2	3	4	1	2	5	1	2	3	4	5

1	2	3	4	1	2	5	1	2	3	4	5

1	2	3	4	1	2	5	1	2	3	4	5

Buddy System

The **buddy system** is a memory allocation and management algorithm

- It manages memory in **power of two increments**.
- Splitting memory into halves and to try to give a best fit

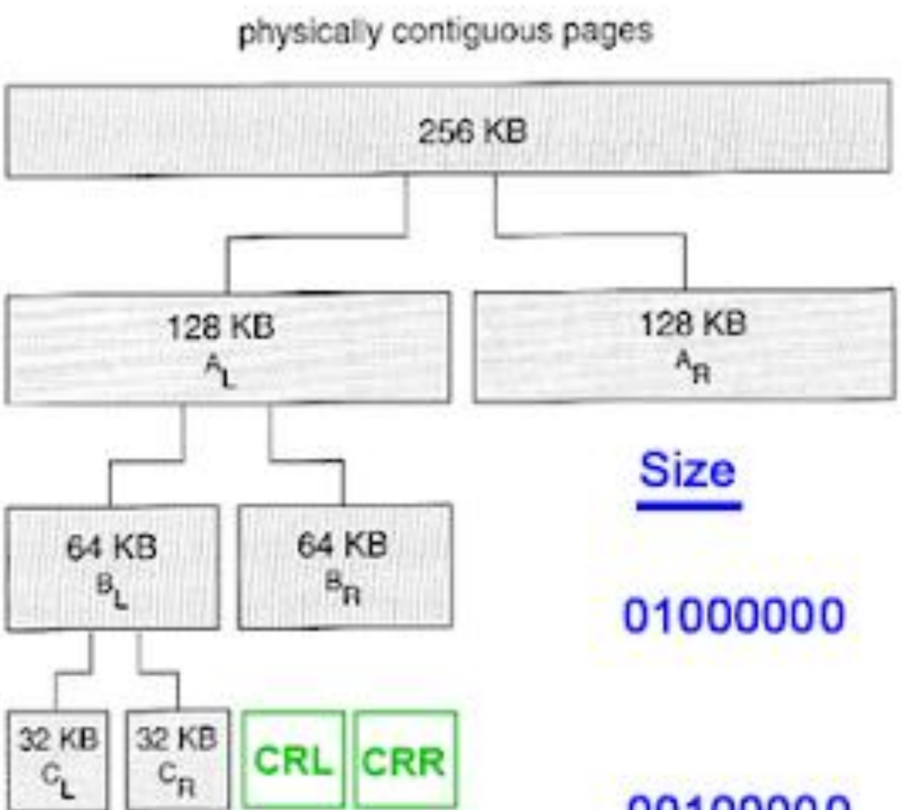
Provides two operations:

- Allocate(2k) : Allocates a block of 2k and marks it as allocated
- Free(A): Marks the previously allocated block A as free and merge it with other blocks to form a larger block

- Algorithm: Assume that a process A of size "X" needs to be allocated

- **If 2K-1< X <=2K:** Allocate the entire block
- **Else:** Recursively divide the block equally and test the condition at each time, when it satisfies, allocate the block.

Cont..



Buddy system allocation.

Size	Buddy Addresses	
	00000000	
	00000000	10000000
01000000	00000000	01000000
00100000	00000000	00100000
	0100000	0110000

Problem 1: Buddy System

Consider a memory block of 32 K. Perform the following:

Allocate (A: 7.5K)

Allocate (B: 1.2K)

Allocate (C: 3.3K)

Allocate (D: 12.9K)

Allocate (E: 3.2K)

Free (C)

Free (B)

Allocate (F: 1.6K)

Allocate (G: 1.8K)

Consider a memory block of 32 K. Perform the following:

Allocate (A: 7.5K)	Free (C)
Allocate (B: 1.2K)	Free (B)
Allocate (C: 3.3K)	Allocate (F: 1.6K)
Allocate (D: 12.9K)	Allocate (G: 1.8K)
Allocate (E: 3.2K)	

- 32K Memory Block

32K

- Allocate (A: 7.5K)

8k (A)	8K	16K

- Allocate (B: 1.2K)

8k (A)	2K(B)	2K	4K	16K

- Allocate (C: 3.3K)

8k (A)	2K(B)	2K	4K(C)	16K

- Allocate (D: 12.9K)

8k (A)	2K(B)	2K	4K(C)	16K(D)

Consider a memory block of 32 K. Perform the following:

Allocate (A: 7.5K)
Allocate (B: 1.2K)
Allocate (C: 3.3K)
Allocate (D: 12.9K)
Allocate (E: 3.2K)
Free (C)
Free (B)
Allocate (F: 1.6K)
Allocate (G: 1.8K)

As we don’t have partition to allocate the process E., there exist wait condition and we need to wait until a process which is previously allocated becomes Free before the next allocate request is raised.

Problem 2 Home work

Consider a memory block of 16K. Perform the following:

Allocate (A: 3.5K)

Allocate (B: 1.2K)

Allocate (C: 1.3K)

Allocate (D: 1.9K)

Allocate (E: 3.2K)

Free (C)

Free (B)

Allocate (F: 1.6K)

Allocate (G: 1.8K)