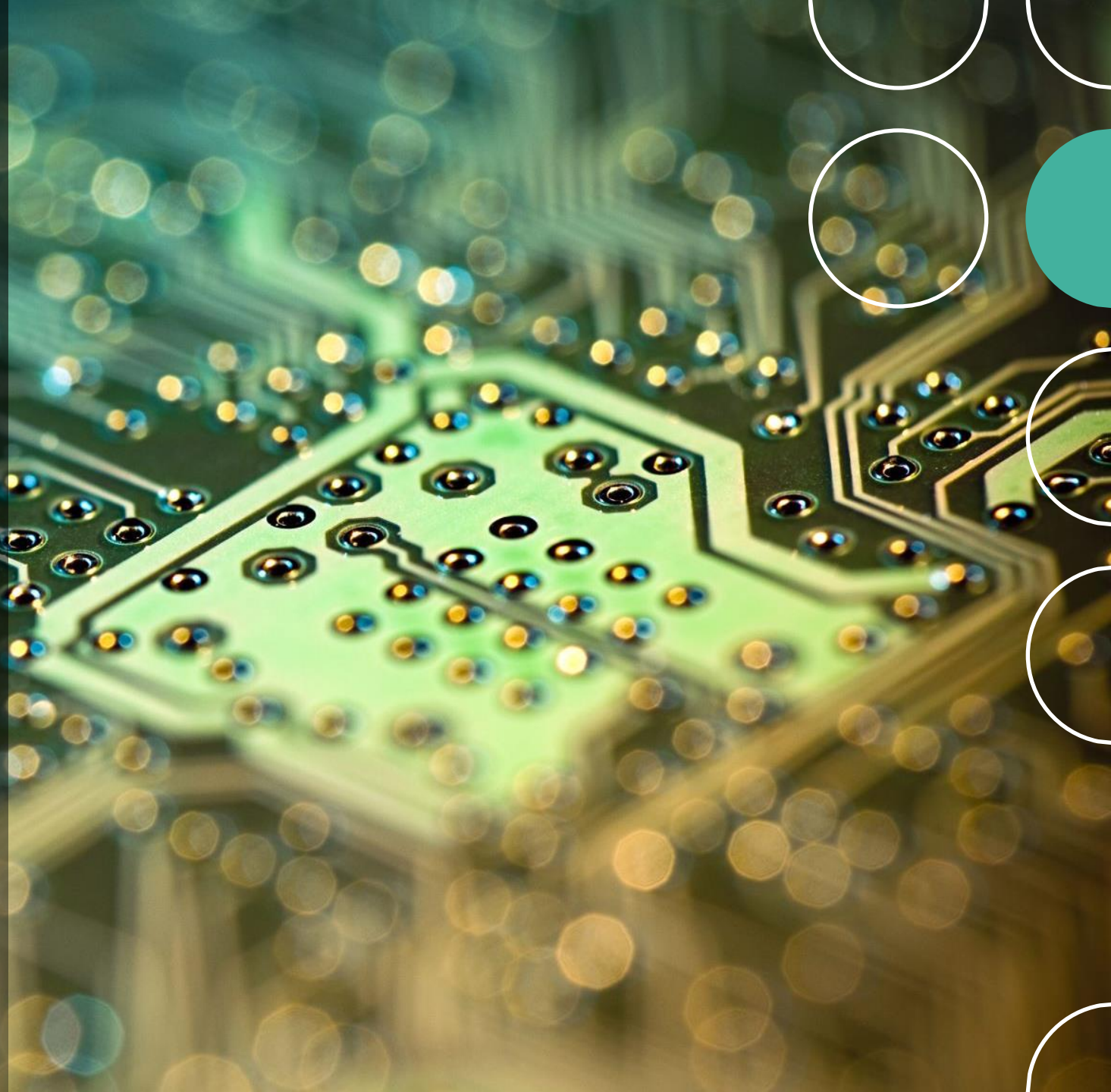
COS3023 Operating Systems and Concurrency

Topic 3- Virtual Memory (Part 2)

Lecturer : Ms Sha



Previous lecture

- What is memory *compaction*?
- What is fragmentation?
- Name two types of fragmentation.

Learning Objectives

After completing this chapter, you should be able to describe:

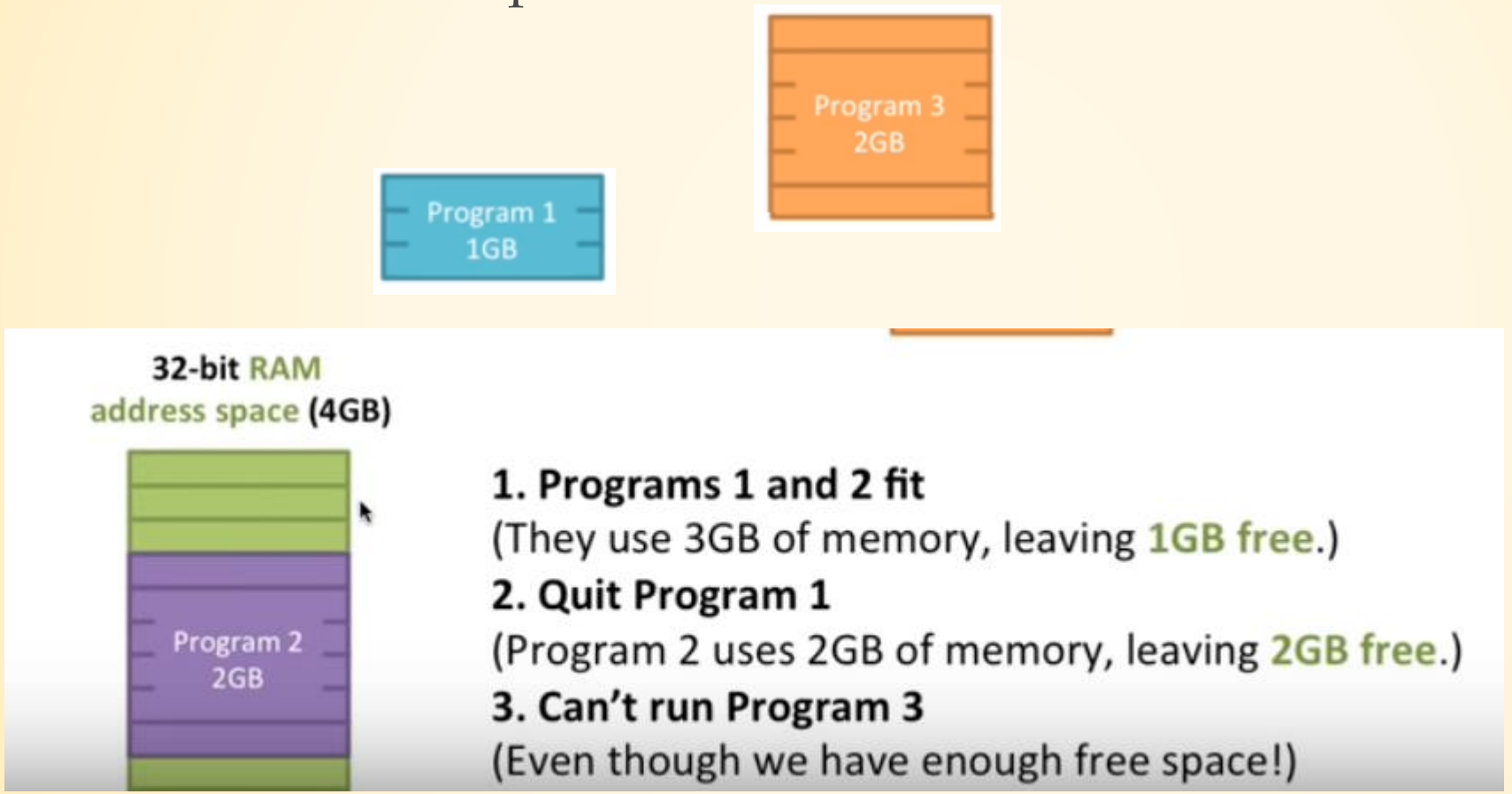
- The basic functionality of the memory allocation methods covered in this chapter: paged, demand paging, segmented, and segmented/demand paged memory allocation
- The influence that these page allocation methods have had on virtual memory
- The difference between a first-in first-out page replacement policy, a least-recently-used page replacement policy, and a clock page replacement policy
- The mechanics of paging and how a memory allocation scheme determines which pages should be swapped out of memory
- The concept of the working set and how it is used in memory allocation schemes
- The impact that virtual memory had on multiprogramming
- Cache memory and its role in improving system response time

Problem with memory

1. Not enough RAM
2. Holes in address spaces
3. Programs writing over each other – crash

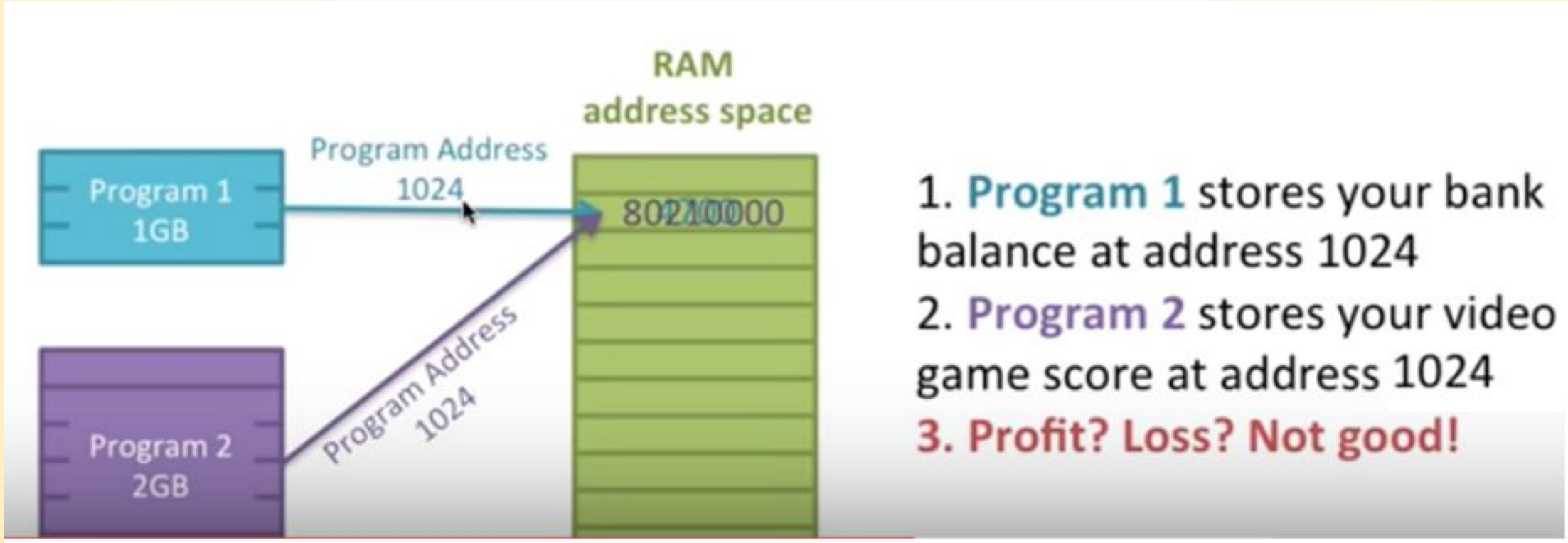
Problem with memory

1. Not enough RAM
2. Holes in address spaces



Problem with memory

3. Programs writing over each other – crash



Problem with memory

Key to the problem – " same memory space ". All program used the same memory space.

Solution – Using virtual memory

- We give each program it's own memory space.
  - Separately map each program's memory space to the RAM memory space.

Virtual Memory

- Separate memory space
- What the program sees/uses

Physical Memory

- Physical RAM in computer. Eg : 2GB of RAM

Virtual address – what the program uses

Physical address – what the hardware uses to talk to the RAM

Virtual Memory

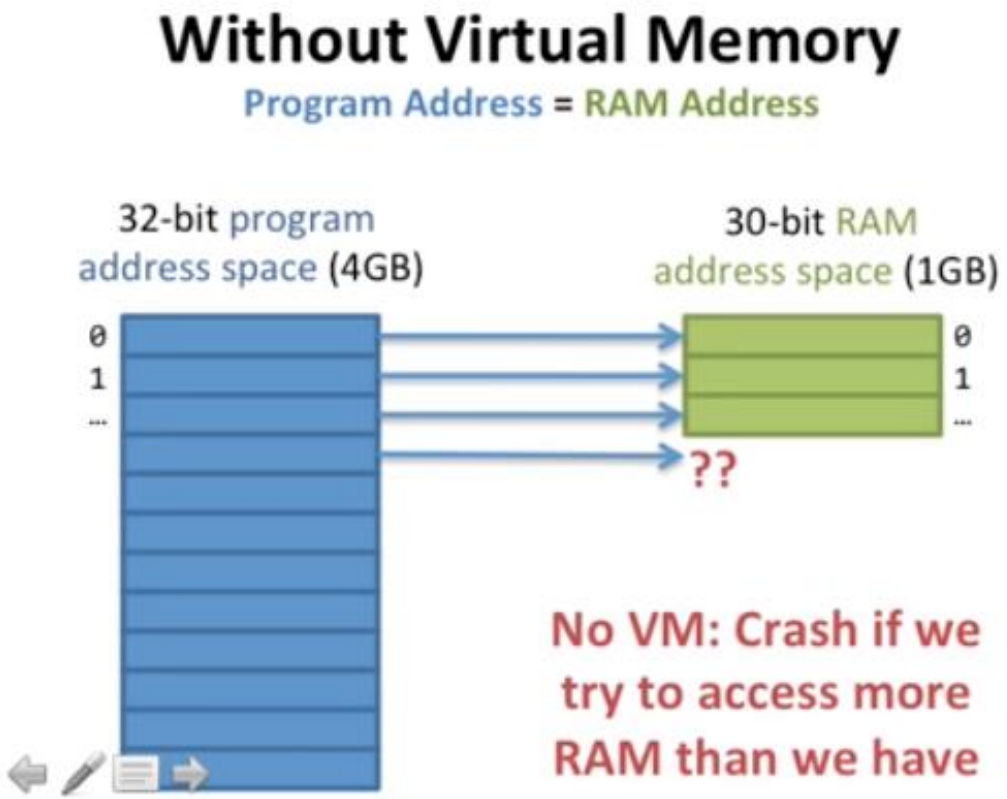
- Virtual memory is a feature of an operating system that enables a computer to be able to compensate shortages of physical memory by transferring pages of data from random access memory to disk storage.
- This process is done temporarily and is designed to work as a combination of RAM and space on the hard disk

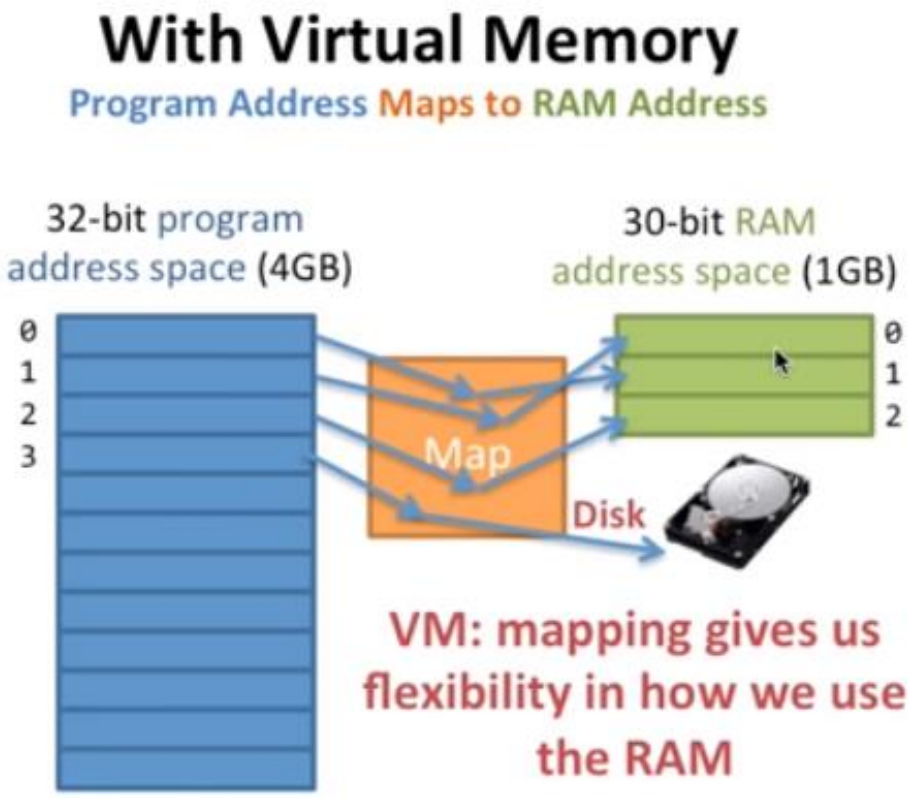
How the Virtual Memory can solve the problems?

Virtual Memory is a layer of indirection

"Any problem in computer science can be solved by adding indirection."

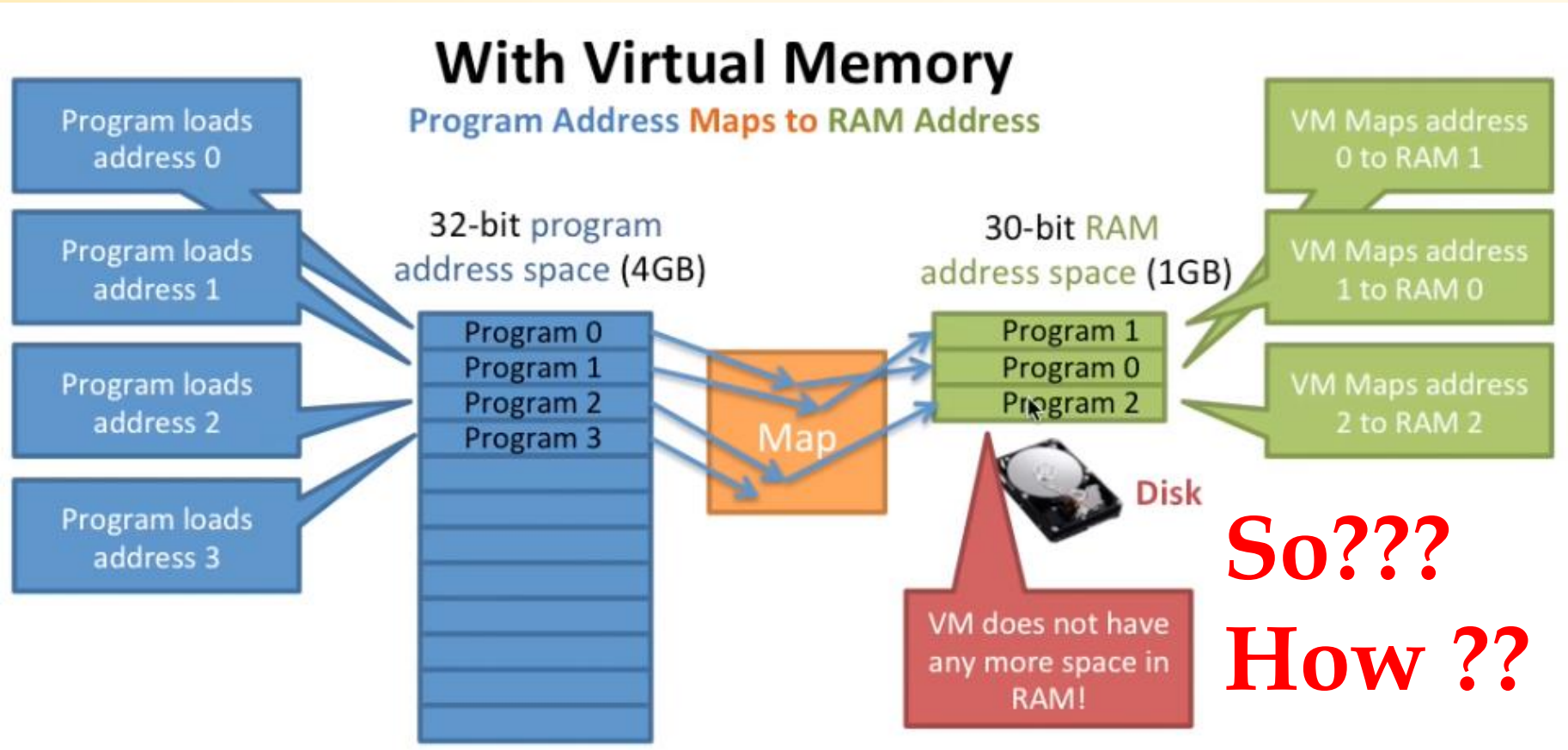
Virtual memory takes **program addresses** and **maps** them to **RAM addresses**





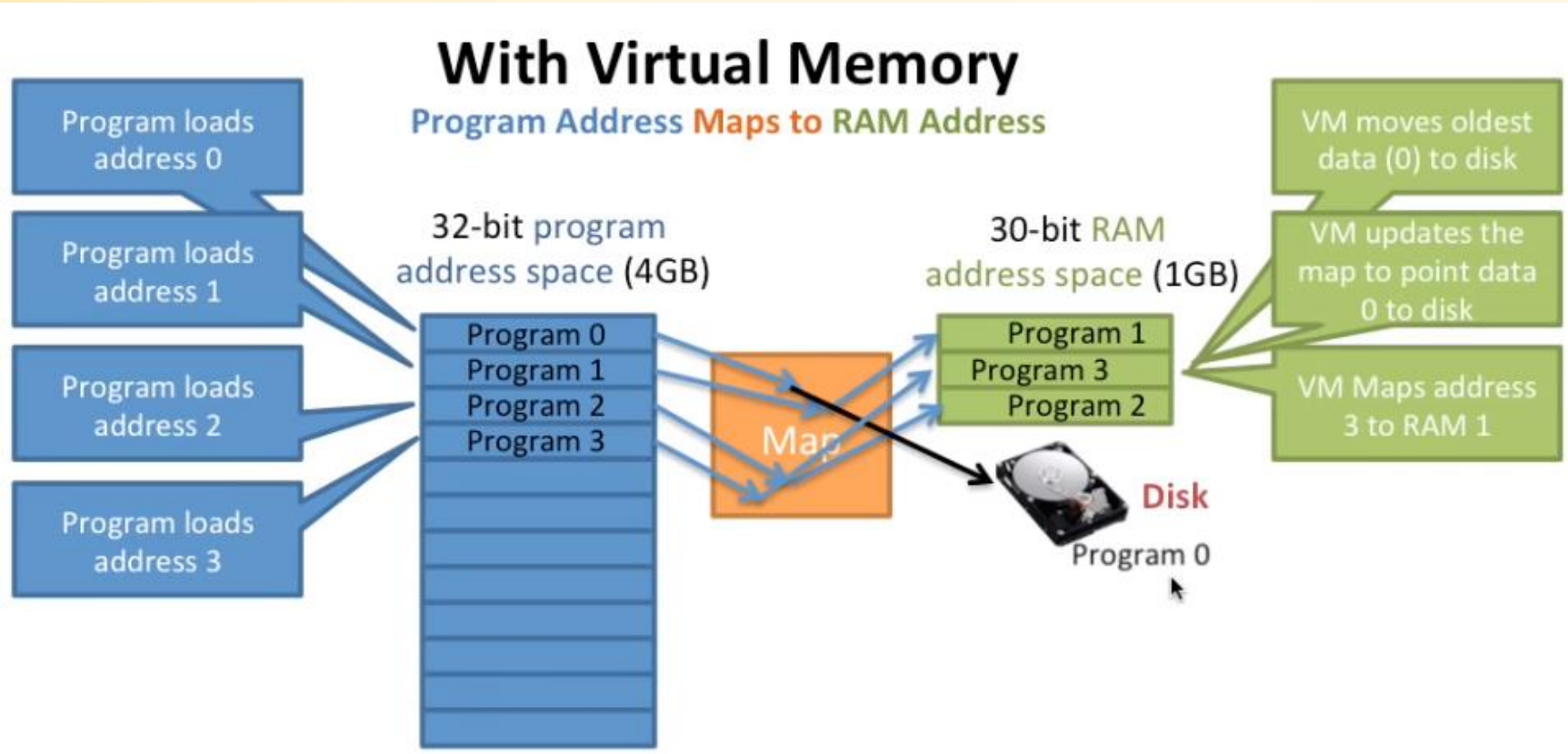
Solving the problems: #1 not enough memory

- **Map** some of the **program's address space** to the **disk**
- When we need it, we bring it into memory



Solving the problems: #1 not enough memory

- **Map** some of the **program's address space** to the **disk**
- When we need it, we bring it into memory

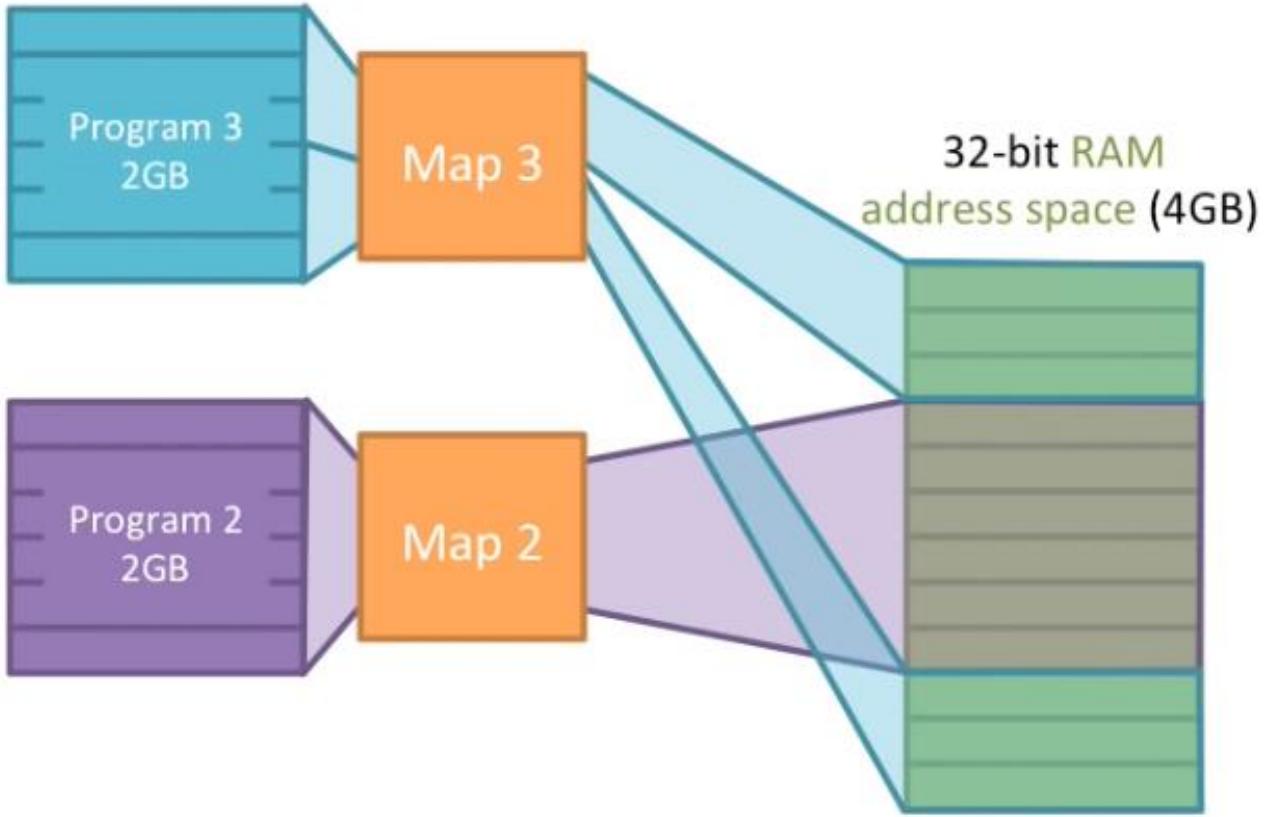


This is the solution !!

Mapping lets us use our disk to give the illusion of unlimited

Solving the problems: #2 holes in the address space

- How do we use the holes left when programs quit?
- We can **map** a program's addresses to RAM addresses however we like



With Virtual Memory

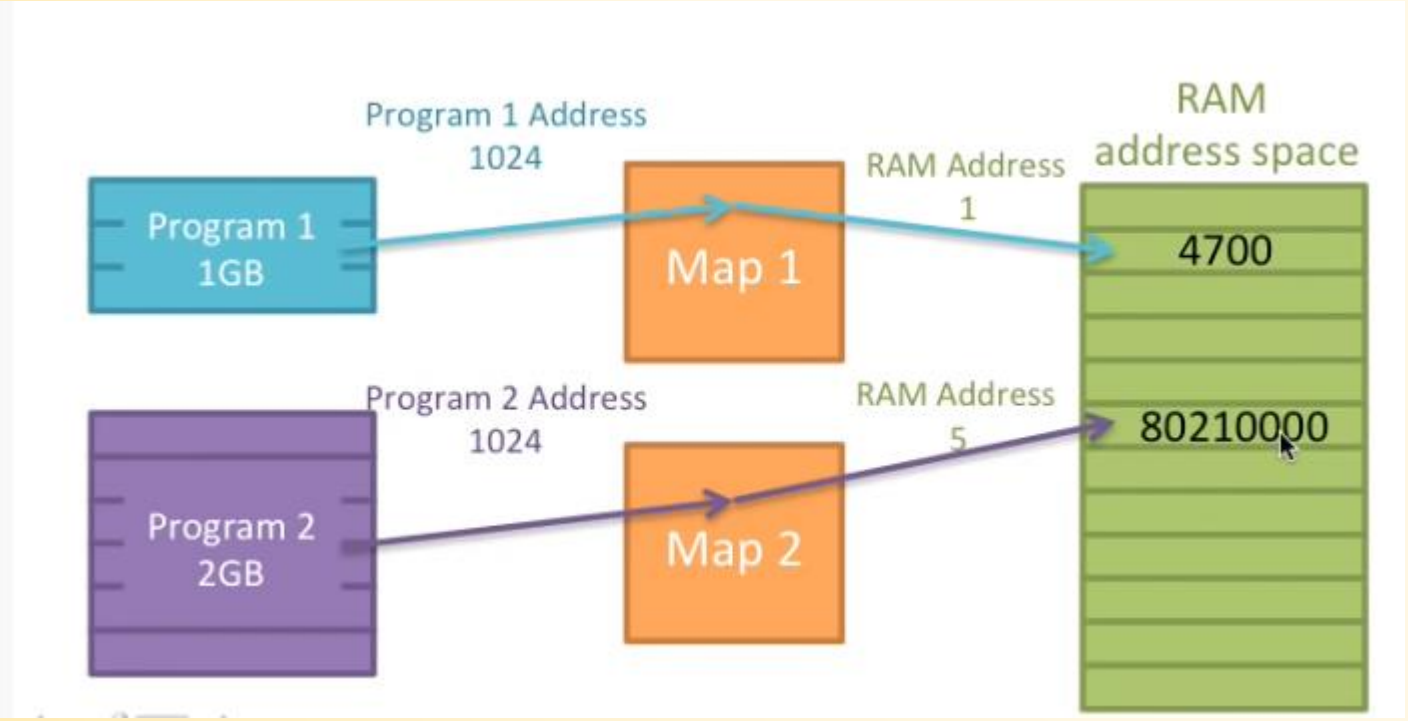
Program Address Maps to RAM Address

Each program has its own mapping.

Mappings lets us put our program data wherever we want in the RAM.

Solving the problems: #3 keeping programs secure

- **Program 1's** and **Program 2's** addresses **map to** different **RAM addresses**
- Because each program has its own address space, they cannot access each other's data: security and reliability!



With Virtual Memory

Program Address Maps to RAM Address

1. **Program 1** stores your bank balance at address 1024
1B. **VM maps** it to **RAM address 1**

2. **Program 2** stores your video game score at address 1024
2B. **VM maps** it to **RAM address 5**

3. Neither can touch the other's data!

Virtual Memory

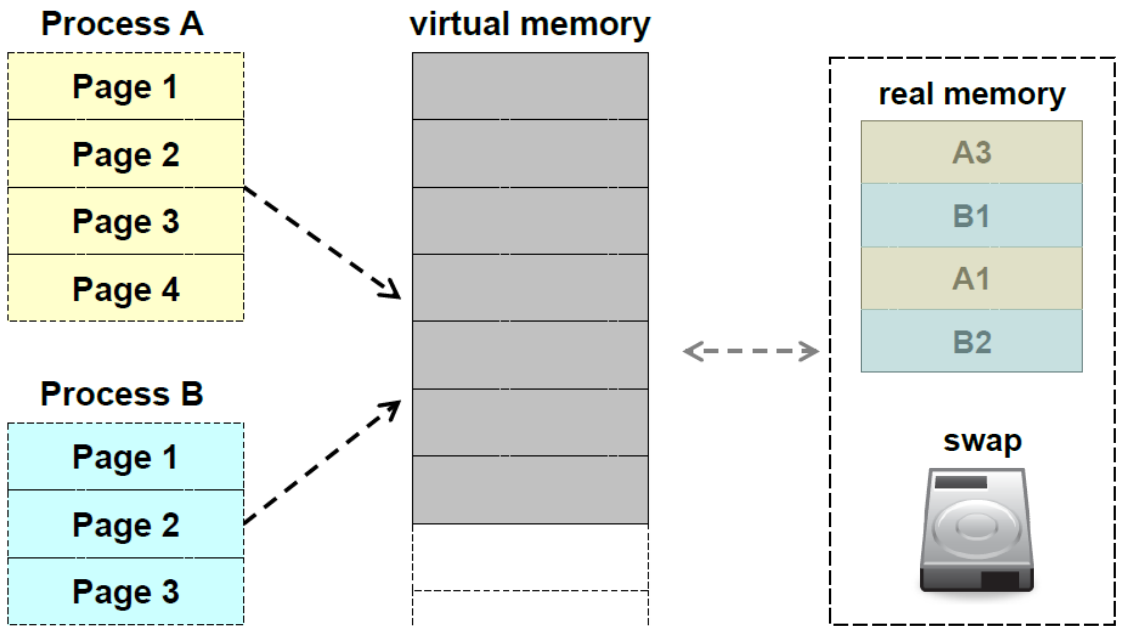
- A process could be much larger than the actual physical memory (i.e. ***real memory***)
- Memory management can load only few process pages/segments in the *real memory*, and keep the remaining ones in secondary storage (i.e. swap space)
- In this way, a large amount of memory (i.e. ***virtual memory***) becomes available to the user in a transparent way

4. Four memory allocation schemes

- paged,
- demand paging,
- segmented,
- segmented/demand paged allocation

1.Paged Memory Allocation

- Before a job is loaded into memory, it is divided into parts called pages that will be loaded into memory locations called page frames.
- based on the concept of dividing each incoming job into pages of **equal size**.



NOTE:
Size of page must equal to size of frame
=> Each page is 10kb, each frame also must be 10kb

1.Paged Memory Allocation

- Before executing a program, the Memory Manager prepares it by:

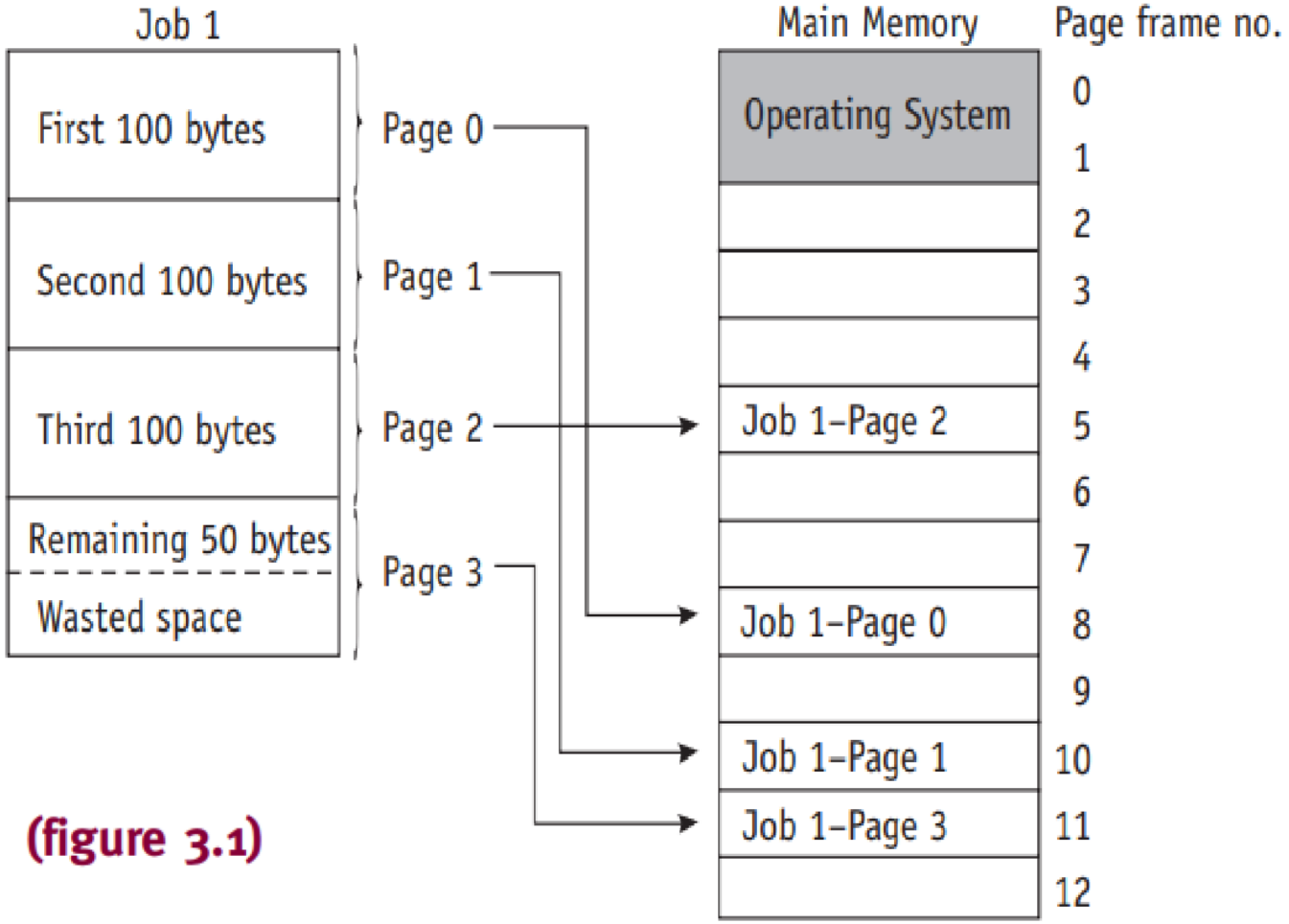
1. Determining the number of pages in the program
2. Locating enough empty page frames in main memory
3. Loading all of the program's pages into them

1.Paged Memory Allocation

- each page can be stored in any available page frame anywhere in main memory.
- Main memory is used more efficiently because an empty page frame can be used by any page of any job.
- However, Because a job's pages can be located anywhere in main memory, the Memory Manager now needs a mechanism to keep track of them.

1. Paged Memory Allocation

Programs that are too long to fit on a single page are split into equal-sized pages that can be stored in free page frames. In this example, each page frame can hold 100 bytes. Job 1 is 350 bytes long and is divided among four page frames, leaving internal fragmentation in the last page frame. (The Page Map Table for this job is shown later in Table 3.2.)



(figure 3.1)

1. Paged Memory Allocation

- In Figure 3.1 (with seven free page frames), the operating system can accommodate jobs that vary in size from 1 to 700 bytes because they can be stored in the seven empty page frames.
- But a job that is larger than 700 bytes can’t be accommodated until Job 1 ends its execution and releases the four page frames it occupies.
- And a job that is larger than1100 bytes will never fit into the memory

1.Paged Memory Allocation

- So, advantage - noncontiguous storage, memory is used more efficiently and more jobs can fit in the main memory .
- disadvantage - it still requires that the entire job be stored in memory during its execution. Overhead is increased and internal fragmentation is still a problem.

1.Paged Memory Allocation

- Memory Manager uses tables to keep track job's pages
- The tables are:

1. Job Table,
2. Page Map Table,
3. Memory Map Table

(Do some research on the content of the tables)

- The Job Table lists every job being processed (one for the whole system).
- The Segment Map Table lists details about each segment (one for each job).
- The Memory Map Table monitors the allocation of main memory (one for the whole system).

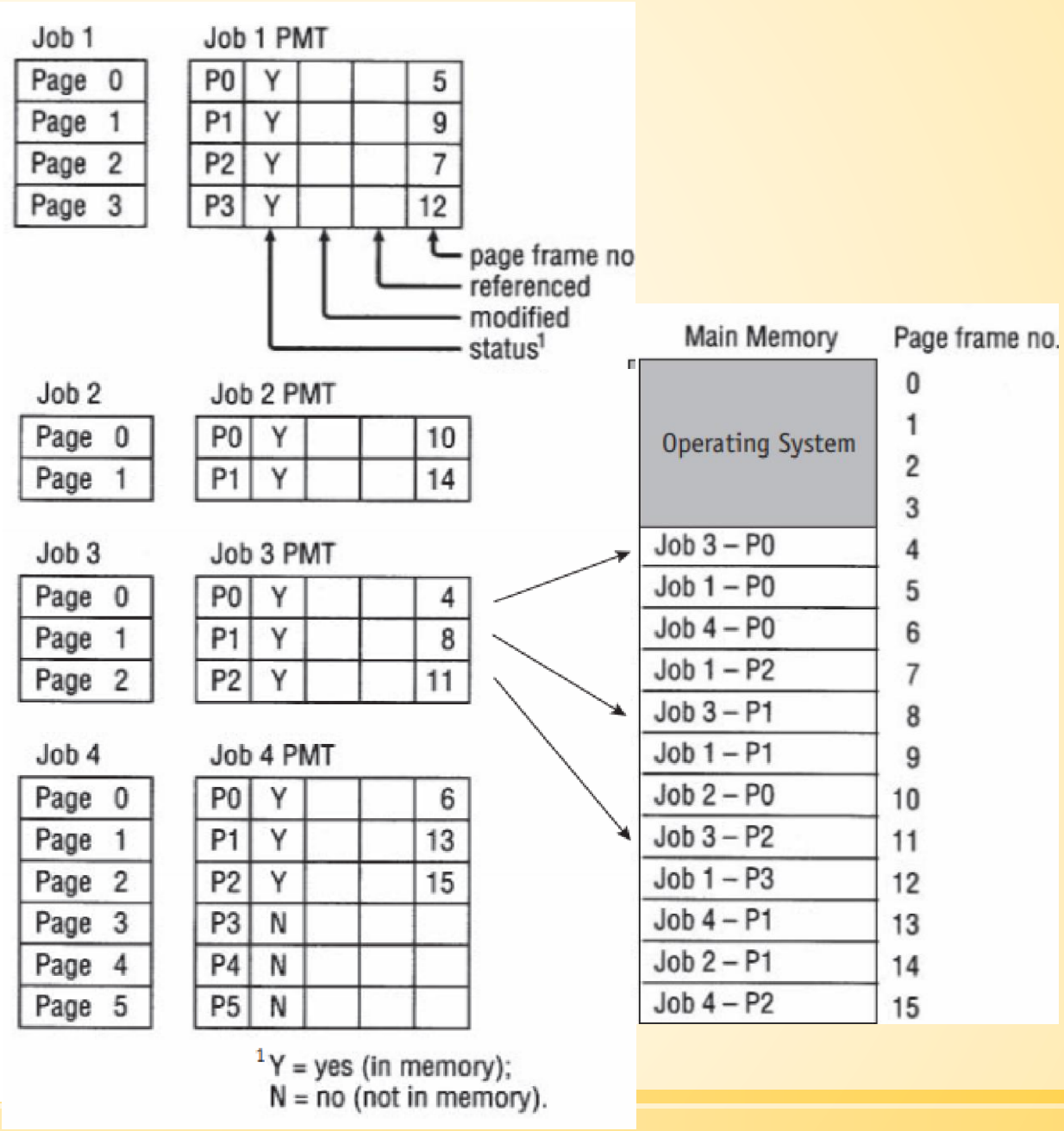
2. Demand Paging

- Introduced the concept of loading only a part of the program into memory for processing.
- Jobs are still divided into equally sized pages that initially reside in secondary storage.
- When the job begins to run, its pages are brought into memory only as they are needed.

2. Demand Paging

(figure 3.5)

Demand paging requires that the Page Map Table for each job keep track of each page as it is loaded or removed from main memory. Each PMT tracks the status of the page, whether it has been modified, whether it has been recently referenced, and the page frame number for each page currently in main memory. (Note: For this illustration, the Page Map Tables have been simplified. See Table 3.3 for more detail.)



2. Demand Paging- Page Replacement Policies and Concepts

Algorithms:

1. First-In First-Out (FIFO)
2. Least Recently Used. (LRU)
3. Optimal Page Replacement

To illustrate the difference between FIFO and LRU, let imagine a dresser drawer filled with your favourite sweaters. ?? Your decision will be based on a sweater removal policy

2. Demand Paging- Page Replacement Policies and Concepts

1.first-in first-out (FIFO)

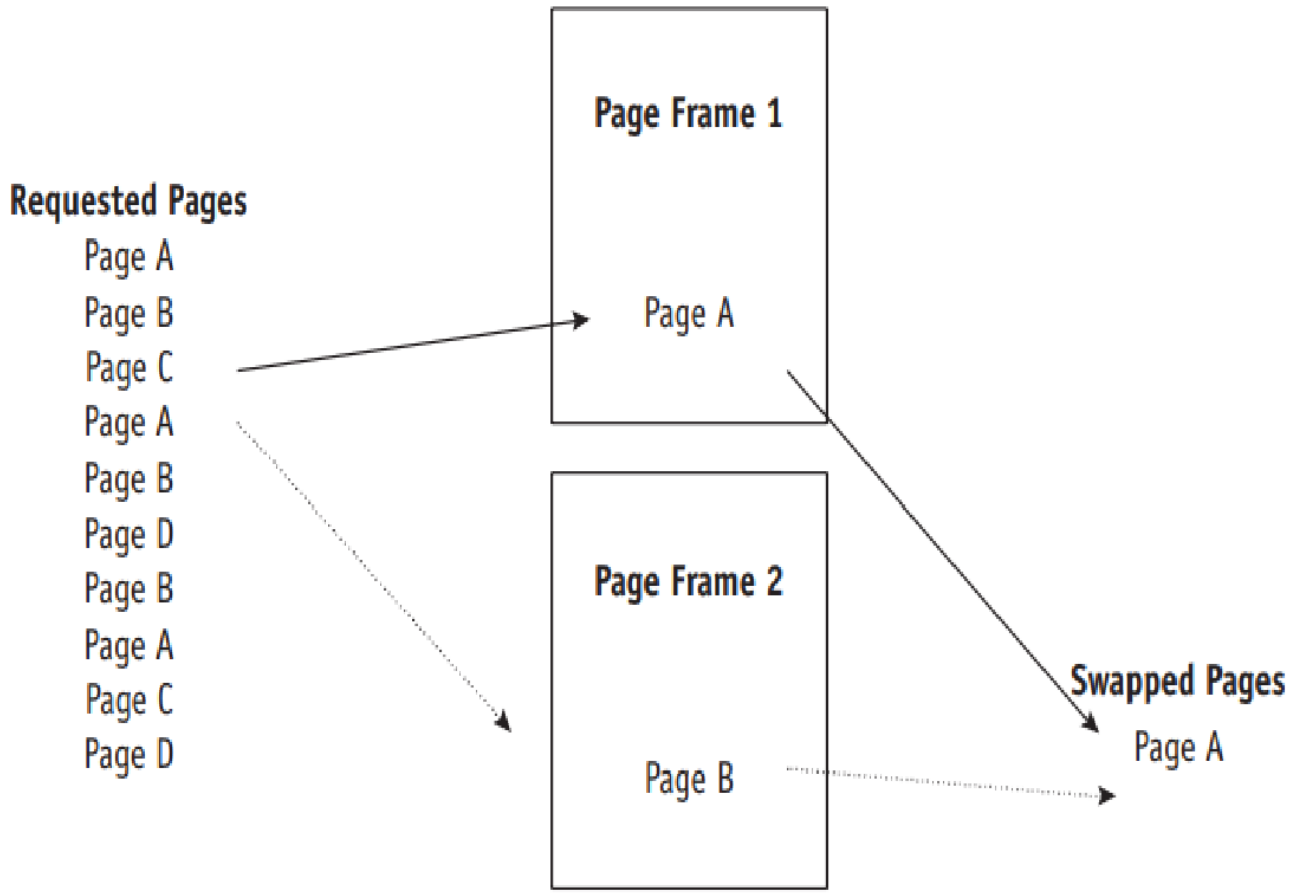
Best page to remove is the one that has been in memory the longest.

2. Demand Paging- Page Replacement Policies and Concepts

1.first-in first-out (FIFO)

(figure 3.7)

The FIFO policy in action with only two page frames available. When the program calls for Page C, Page A must be moved out of the first page frame to make room for it, as shown by the solid lines. When Page A is needed again, it will replace Page B in the second page frame, as shown by the dotted lines. The entire sequence is shown in Figure 3.8.

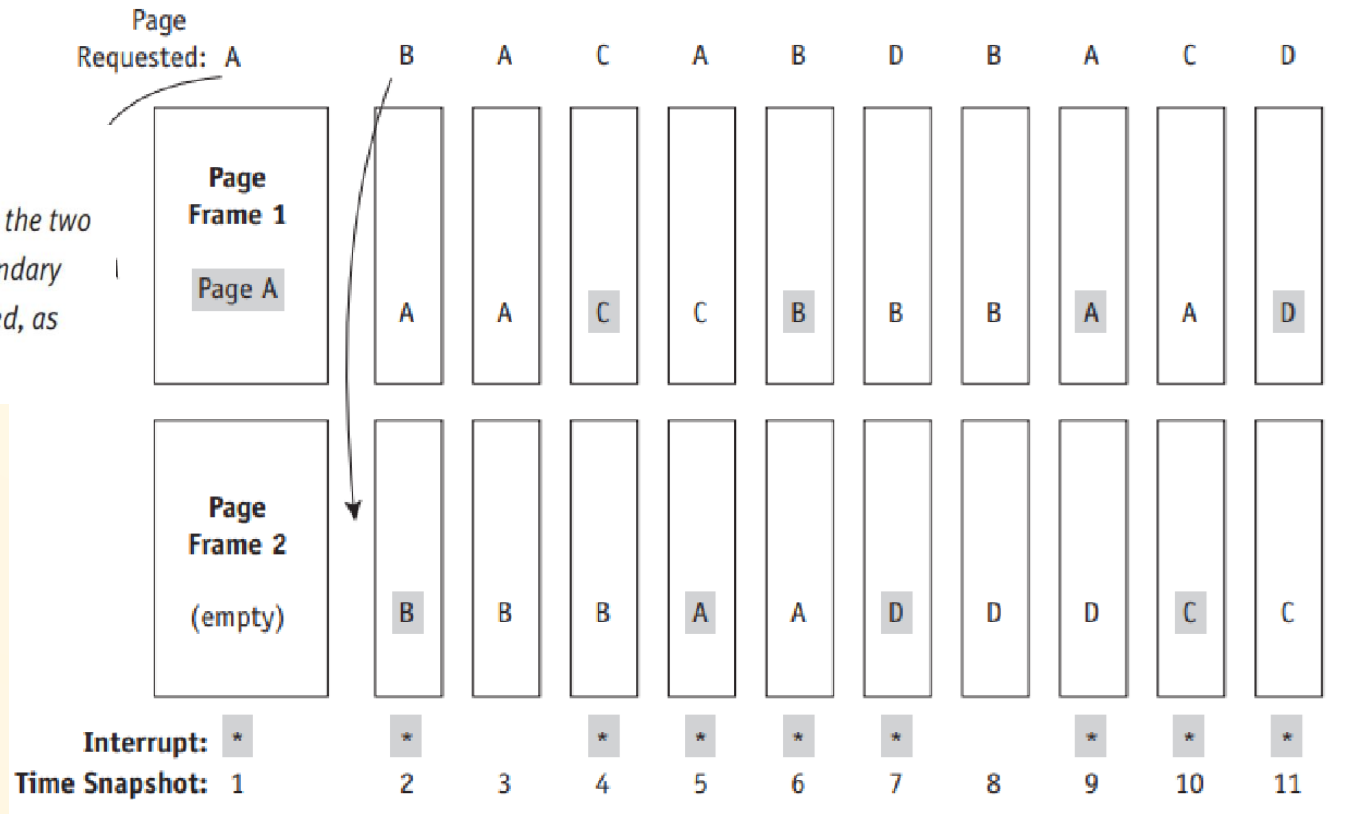


2. Demand Paging- Page Replacement Policies and Concepts

1.first-in first-out (FIFO)

FIFO algorithm works by following a job with four pages (A,B, C, D) as it is processed by a system with only two available page frames. The job to be processed needs its pages in the following order:

A, B, A, C, A, B, D, B, A, C, D.



(figure 3.8)

Using a FIFO policy, this page trace analysis shows how each page requested is swapped into the two available page frames. When the program is ready to be processed, all four pages are in secondary storage. When the program calls a page that isn't already in memory, a page interrupt is issued, as shown by the gray boxes and asterisks. This program resulted in nine page interrupts.

Failure rate = 9/11
= 82%

Success rate = 2/11
= 18%

Section checkout -01

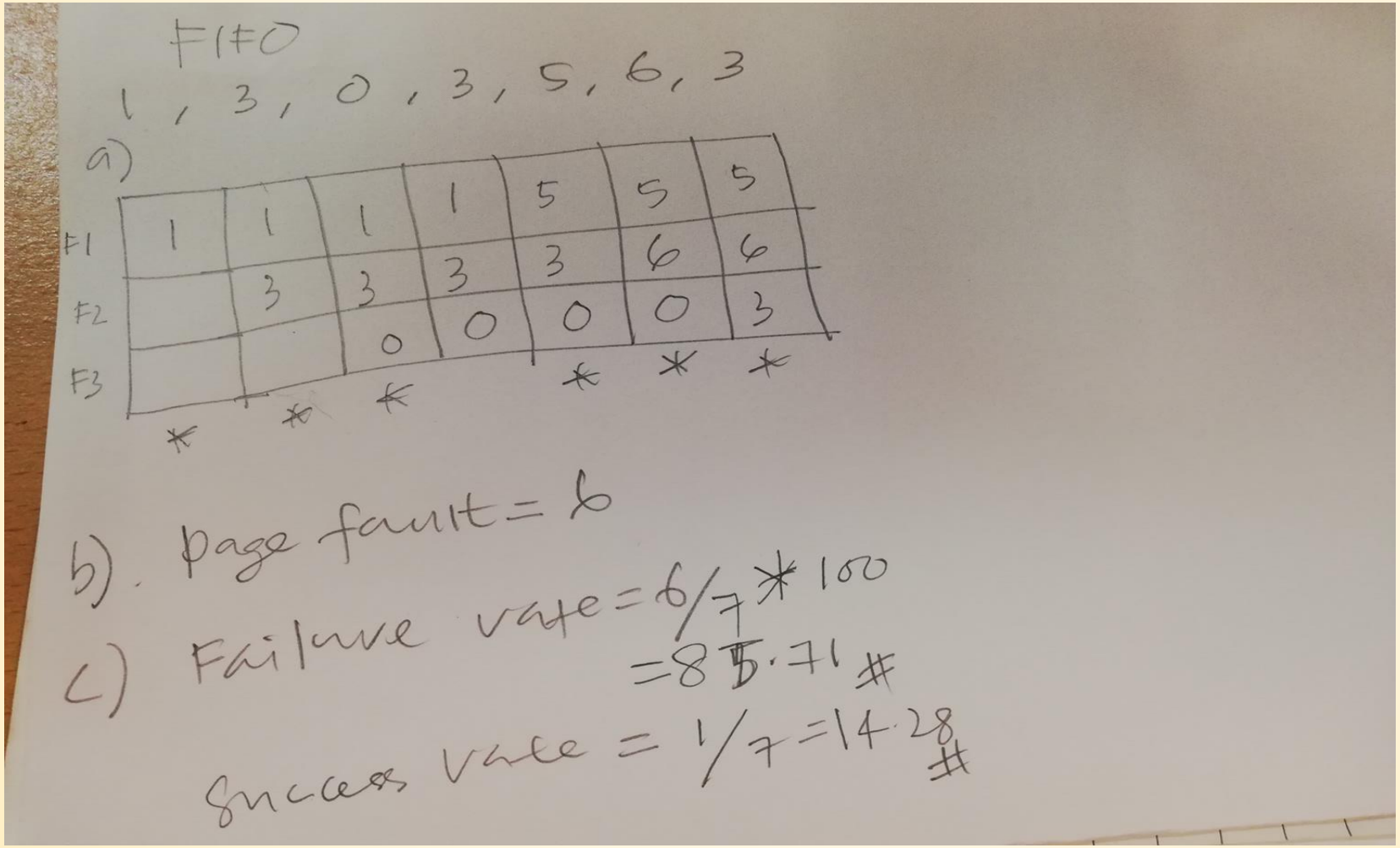
Consider a reference string:

1, 3, 0, 3, 5, 6, 3 of frame size 3.

Using FIFO algorithm:

a. Draw the page trace analysis diagram.
b. determine number of page faults.
c. Calculate the failure and success ratios.

Solution


FIFO
1, 3, 0, 3, 5, 6, 3
a)
F1
F2
F3
1 1 1 1 5 5 5
3 3 3 3 6 6
0 0 0 0 3
* * * * * * *
b). Page fault = 6
c) Failure rate = 6/7 * 100
= 85.71 #
Success rate = 1/7 = 14.28 #

Section checkout -01

Consider a reference string:

3, 2, 1, 3, 4, 1, 6, 2, 4 of frame size 3.

Using FIFO algorithm:

a. Draw the page trace analysis diagram.
b. determine number of page faults.
c. Calculate the failure and success ratios.

Section checkout- 02

Consider a reference string:

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1 of frame size 3.

Using FIFO algorithm:

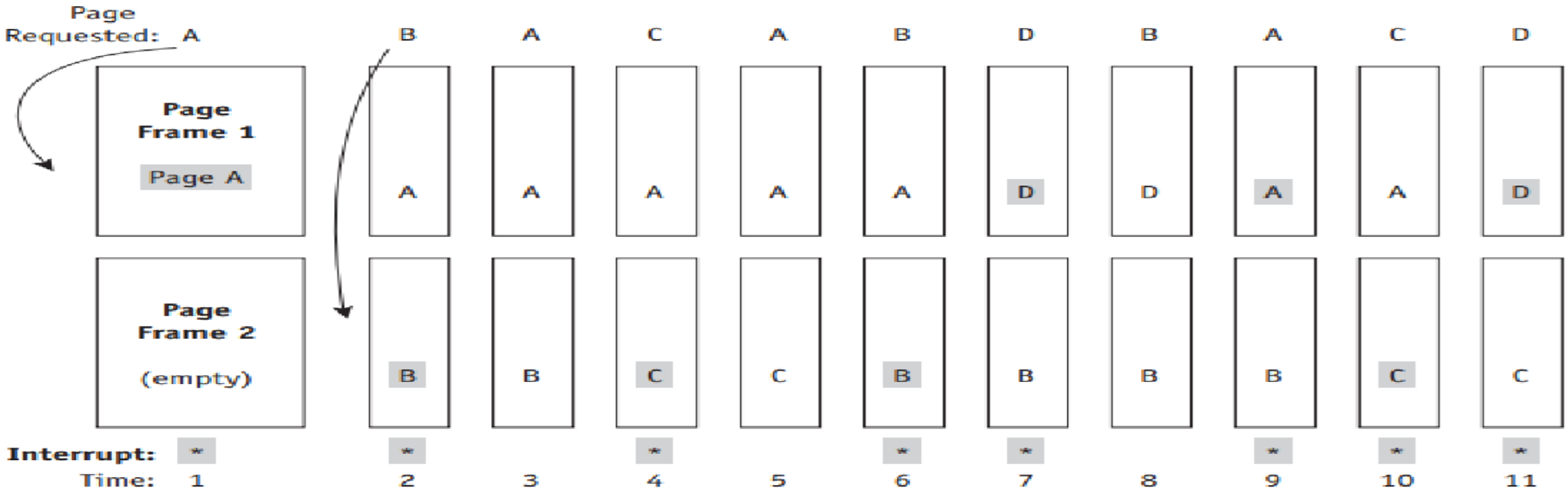
a. Draw the page trace analysis diagram.
b. determine number of page faults.
c. Calculate the failure and success ratios.

2. Demand Paging- Page Replacement Policies and Concepts

2. Least Recently Used. (LRU)

- chooses the page least recently accessed to be swapped out. Swaps out the pages that show the least amount of recent activity

2. Demand Paging- Page Replacement Policies and Concepts



(figure 3.9)

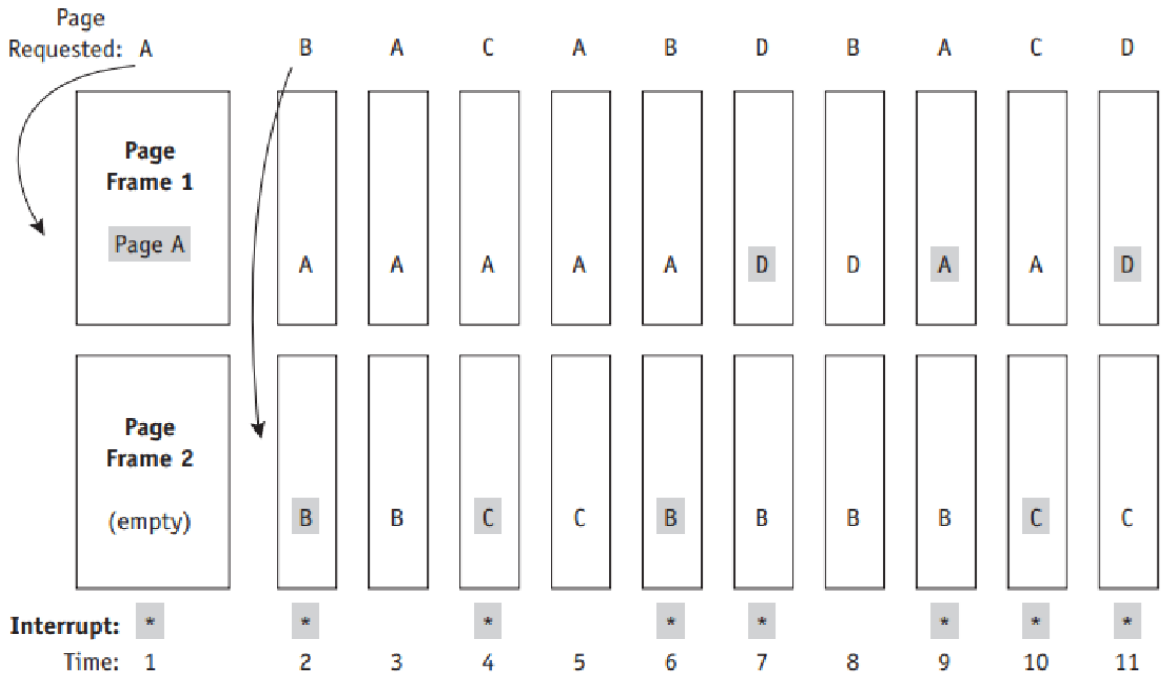
Memory management using an LRU page removal policy for the program shown in Figure 3.8. Throughout the program, 11 page requests are issued, but they cause only 8 page interrupts.

Least Recently Used

Failure rate = 8/11
= 73%

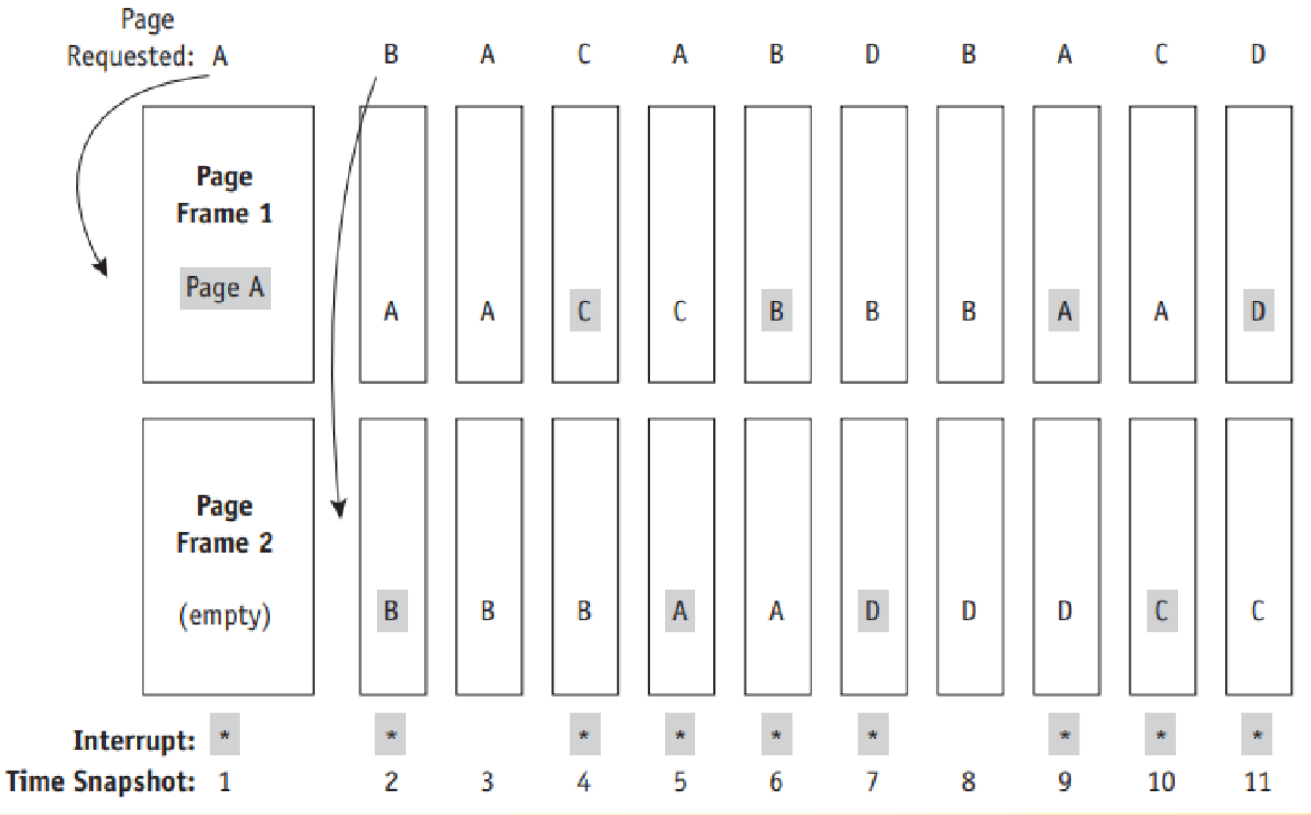
Success rate = 3/11
= 27%

2. Demand Paging- Page Replacement Policies and Concepts



(figure 3.9)

Memory management using an LRU page removal policy for the program shown in Figure 3.8. Throughout the program, 11 page requests are issued, but they cause only 8 page interrupts.



Least Recently Used

Failure rate = 8/11
= 73%

Success rate = 3/11
= 27%

Least Recently Used

5 0 1 2 0 3 2 0 3 4 1 0 5 0 4 3 2 1 2 0 1

String	5	0	1	2	0	3	2	0	3	4	1	0	5	0	4	3	2	1	2	0	1
Frame 3			1	1	1	3	3	3	3	3	3	0	0	0	0	3	3	3	3	0	0
Frame 2		0	0	0	0	0	0	0	0	0	1	1	1	1	4	4	4	1	1	1	1
Frame 1	5	5	5	2	2	2	2	2	2	4	4	4	5	5	5	5	2	2	2	2	2
Miss/Hit	M	M	M	M	H	M	H	H	H	M	M	M	M	H	M	M	M	M	H	M	H

Section checkout

Consider a reference string: 4, 7, 6, 1, 7, 6, 1, 2, 7, 2. the number of frames in the memory is 3.

Using **LRU** algorithm:

a. Draw the page trace analysis diagram.

b. determine number of page faults.

c. Calculate the failure and success ratios.

Section checkout

1. What is Virtual Memory
2. List the four memory allocation schemes
3. Describe the first two schemes

2. Demand Paging- Page Replacement Policies and Concepts

3. Optimal Page Replacement

In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.

2. Demand Paging- Page Replacement Policies and Concepts

3. Optimal Page Replacement

Example: Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3 with 4 page frame. Find number of page fault.

7	**0**	**1**	**2**	**0**	**3**	**0**	**4**	**2**	**3**	**0**	**3**	**2**	**3**
7	7	7	7	7	**3**	**3**	3	3	3	3	3	3	3
	0	0	0	0	0	0	0	0	0	0	0	0	0
		1	1	1	1	1	**4**	**4**	**4**	**4**	**4**	**4**	**4**
			2	2	2	2	2	2	2	2	2	2	2
*	*	*	*		*		*						

frames

No. of page faults = 6

2. Demand Paging- Page Replacement Policies and Concepts

3. Optimal Page Replacement

- Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**
- 0 is already there so —> **0 Page fault.**
- when 3 came it will take the place of 7 because it is not used for the longest duration of time in the future.—>**1 Page fault.**
- 0 is already there so —> **0 Page fault.**
- 4 will takes place of 1 —> **1 Page Fault.**

Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

2. Demand Paging- Page Replacement Policies and Concepts

3. Optimal Page Replacement

Not possible in practice as the operating system cannot know future requests.

Section checkout-FIFO,LRU, Optimal Page

Consider a reference string: 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1 the number of frames in the memory is 4.

Using **FIFO, LRU, and Optimal Page** algorithm:

a. Draw the page trace analysis diagram.

b. determine number of page faults.

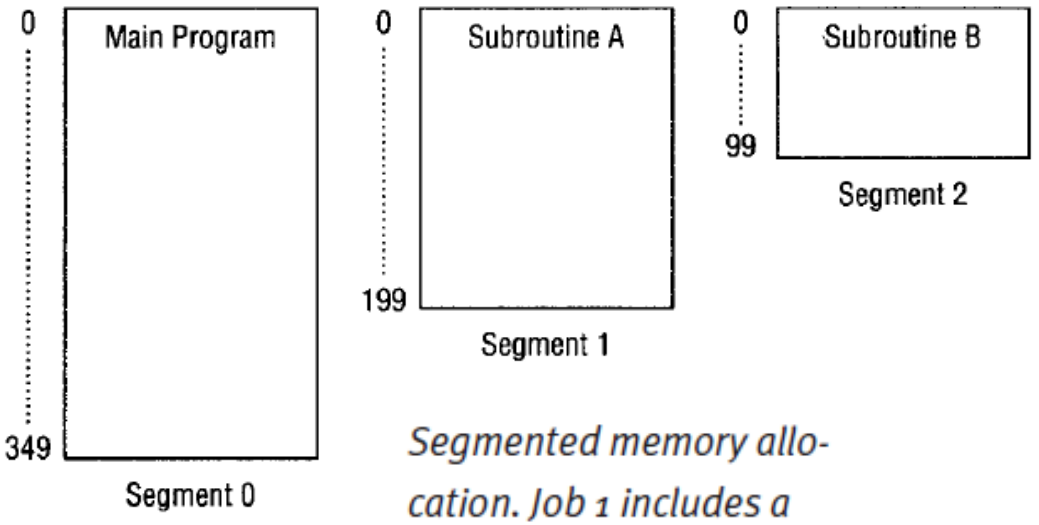
c. Calculate the failure and success ratios.

3. Segmented Memory Allocation

- The concept of segmentation is based on the common practice by programmers of structuring their programs in modules—logical groupings of code (subroutines)
- each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions.

3. Segmented Memory Allocation

Logical view of segmentation:



Segmented memory allocation. Job 1 includes a main program, Subroutine A, and Subroutine B. It is one job divided into three segments.

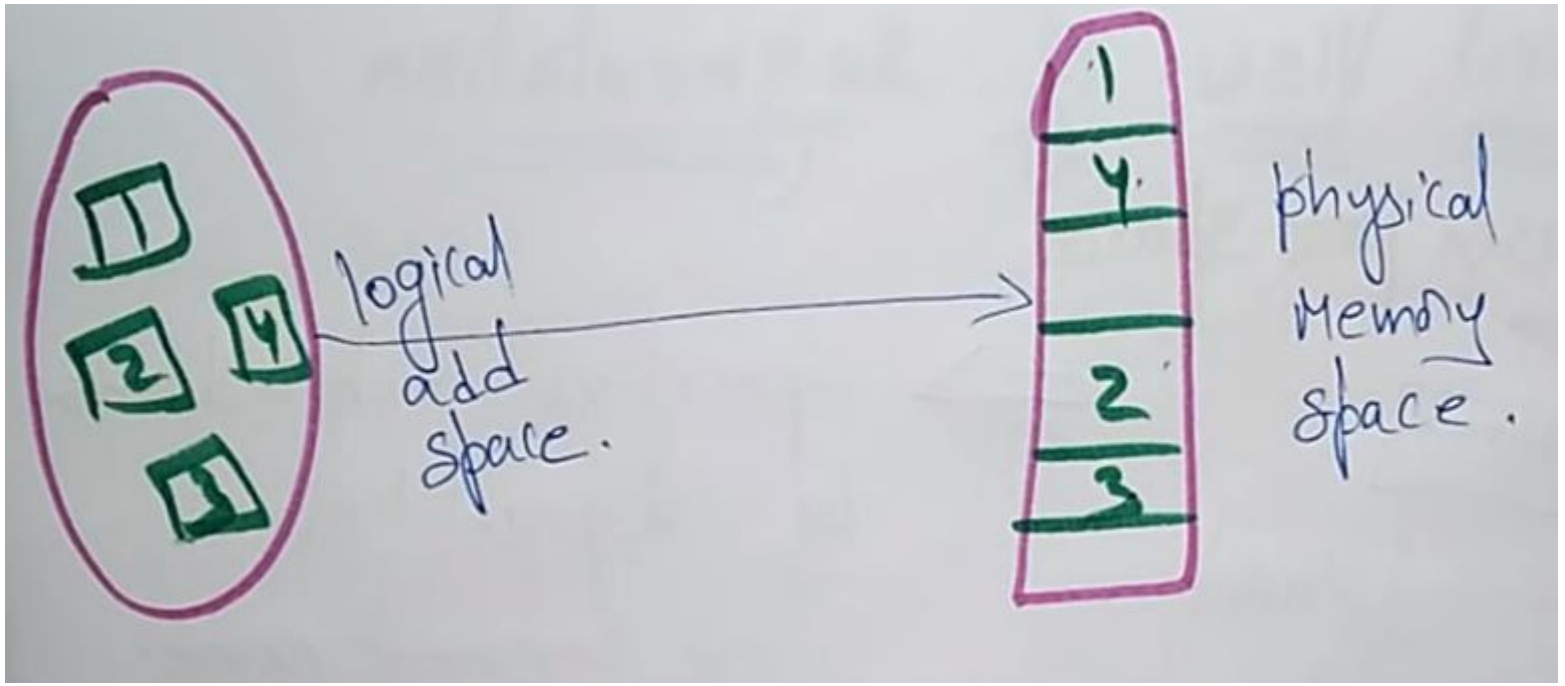
- specifies each segment address by 2 quantities:

  a) segment name

  b) segment offset

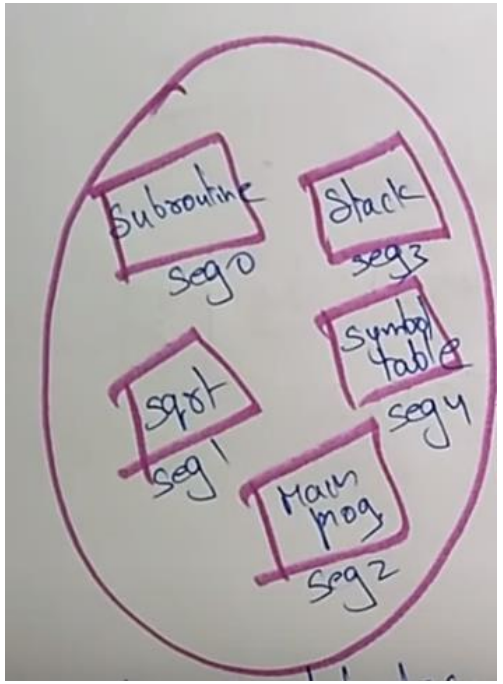
3. Segmented Memory Allocation

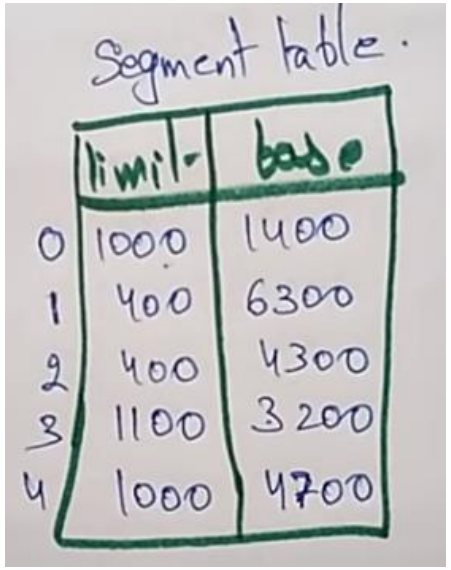
Logical view of segmentation:

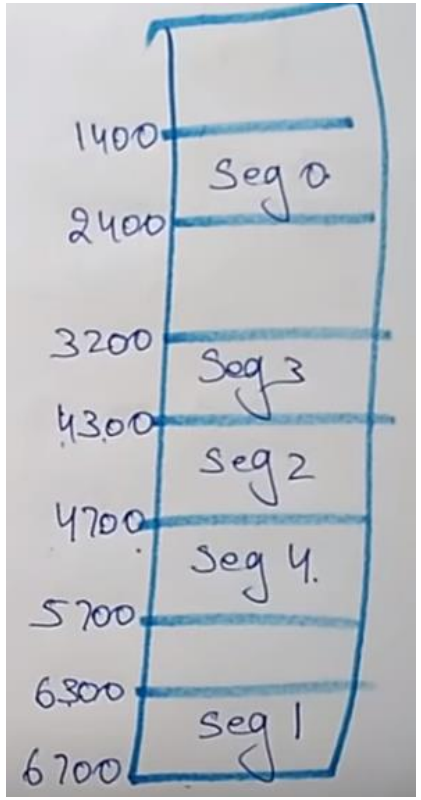


3. Segmented Memory Allocation

Logical view of segmentation:





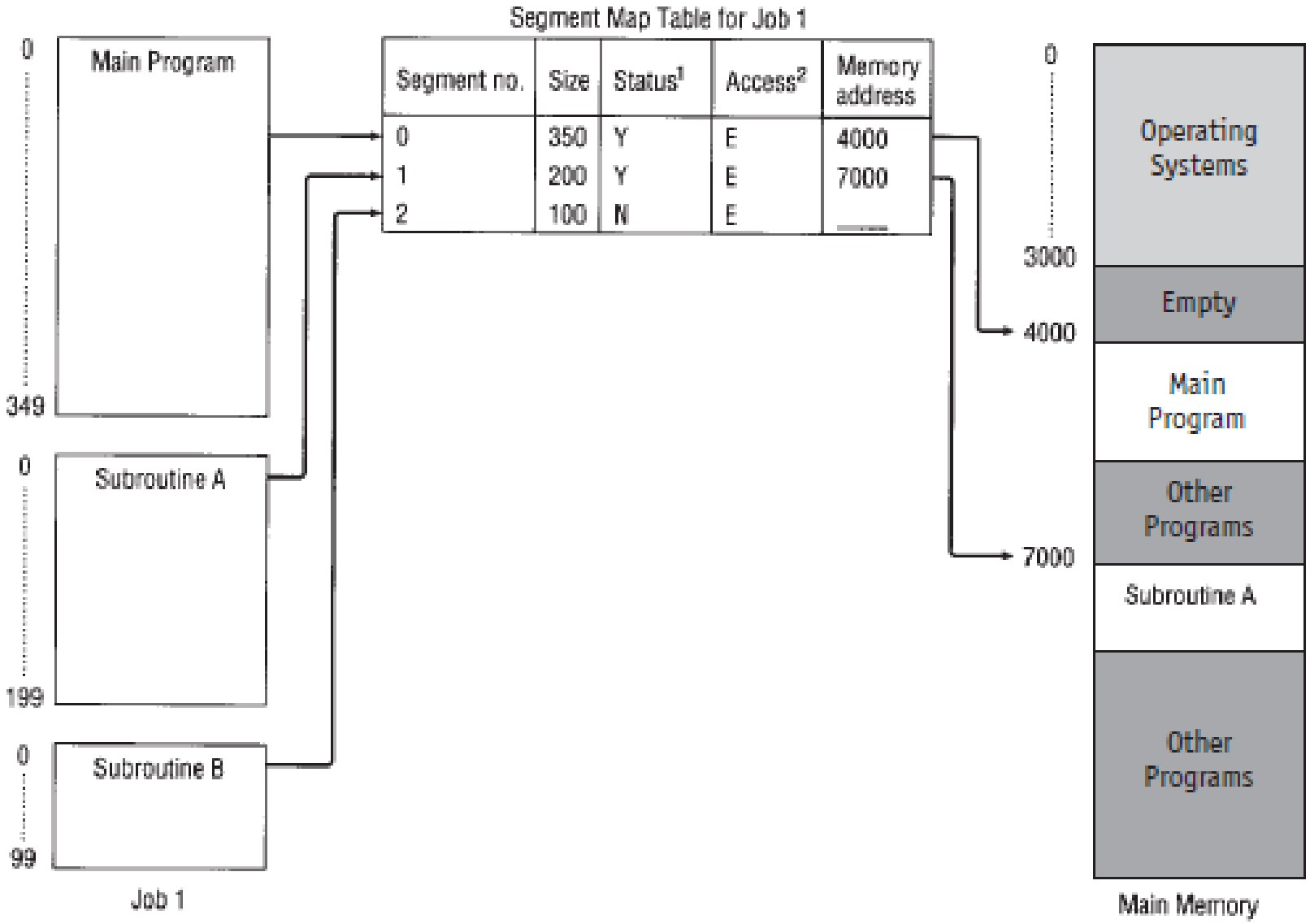


3. Segmented Memory Allocation

- When a program is compiled or assembled, the segments are set up according to the program's structural modules.
- Each segment is numbered and a Segment Map Table (SMT) is generated for each job.
- SMT it contains the segment numbers, their lengths, access rights, status, and (when each is loaded into memory) its location in memory.

Segment Map Table for Job 1

Segment no.	Size	Status[1]	Access[2]	Memory address
0	350	Y	E	4000
1	200	Y	E	7000
2	100	N	E	____



[1] Y = in memory;
N = not in memory.

[2] E = Execute only.

3. Segmented Memory Allocation

The Memory Manager needs to keep track of the segments in memory. This is done with three tables:

- The Job Table lists every job being processed (one for the whole system).
- The Segment Map Table lists details about each segment (one for each job).
- The Memory Map Table monitors the allocation of main memory (one for the whole system).

4. Segmented/Demand Paged Memory Allocation

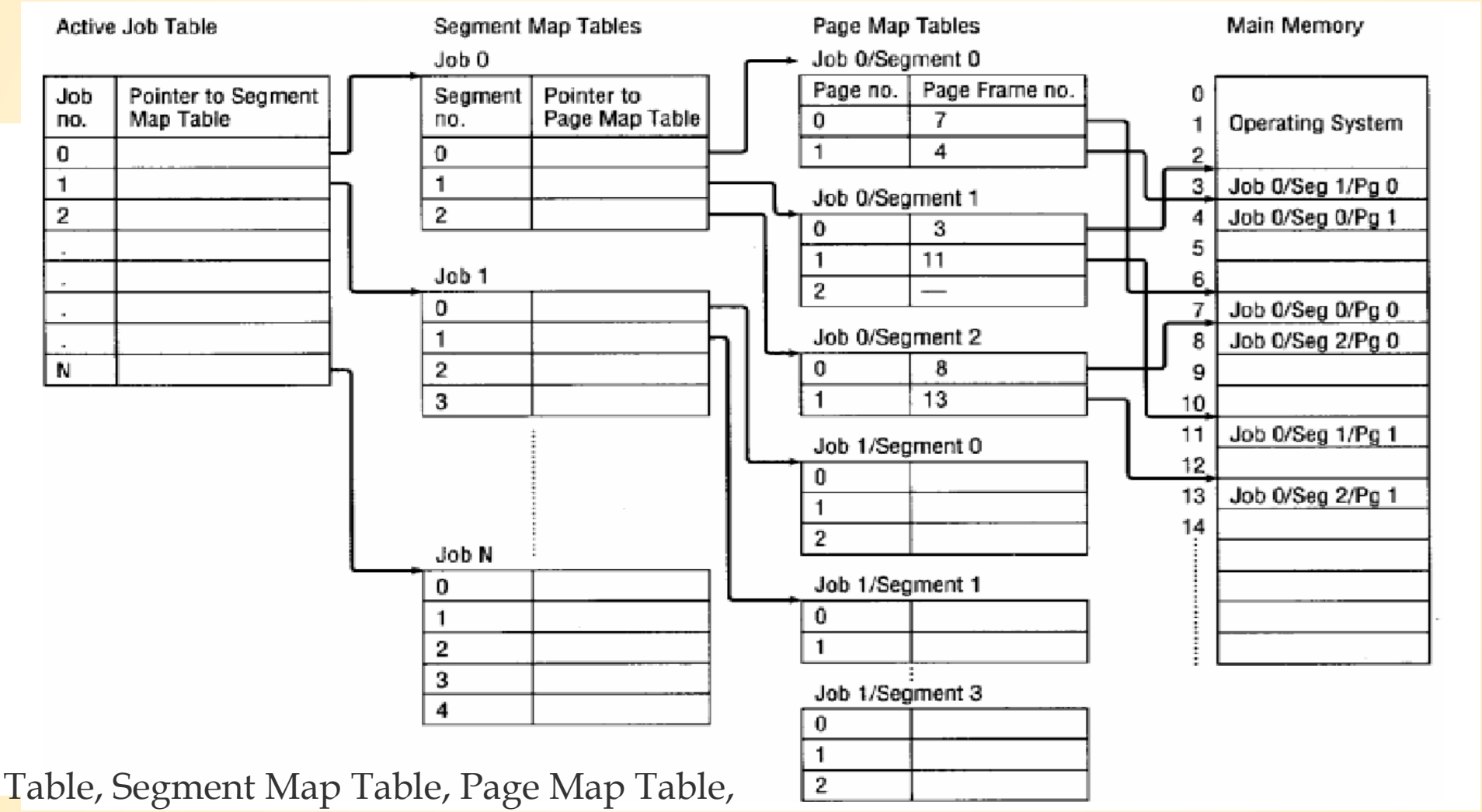
- It is a combination of segmentation and demand paging.
- The algorithms used by the demand paging and segmented memory management schemes are applied here with only minor modifications.
- This allocation scheme doesn't keep each segment as a single contiguous unit but subdivides it into pages of equal size, smaller than most segments, and more easily manipulated than whole segments.

4. Segmented/Demand Paged Memory Allocation

This scheme, requires four tables:

- The Job Table lists every job in process (one for the whole system).
- The Segment Map Table lists details about each segment (one for each job).
- The Page Map Table lists details about every page (one for each segment).
- The Memory Map Table monitors the allocation of the page frames in main memory (one for the whole system).

4. Segmented/Demand Paged Memory Allocation



How the Job Table, Segment Map Table, Page Map Table, and main memory interact in a segment/paging scheme.

Exercise

What are the differences between?

a) Page table and Segment table

b) First –fit placement and Best-fit placement

c) Contiguous and non contiguous

d) Segmentation and paging storage

Exercise

Given that main memory is composed of three page frames for public use and that a seven-page program (with pages a, b, c, d, e, f, g) requests pages in the following order:

a. Using the FIFO page removal algorithm, do a page trace analysis indicating page faults with asterisks (*). Then compute the failure and success ratios.

Exercise

b. Increase the size of memory so it contains four page frames for public use. Using the same page requests as above and FIFO, do another page trace analysis and compute the failure and success ratios.

THANK YOU