Operating Systems I

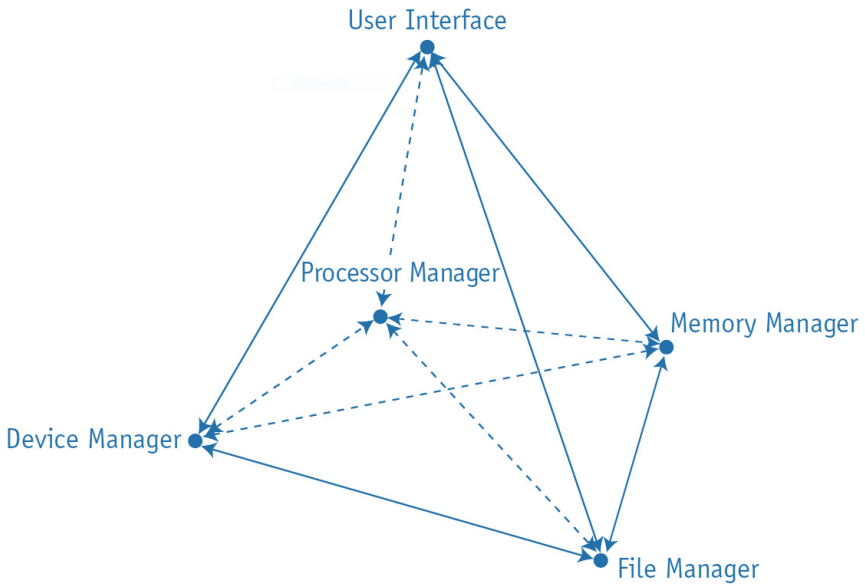
LECTURE III: VIRTUAL MEM. (1/3)

Siv Ratha | siv.ratha89@gmail.com

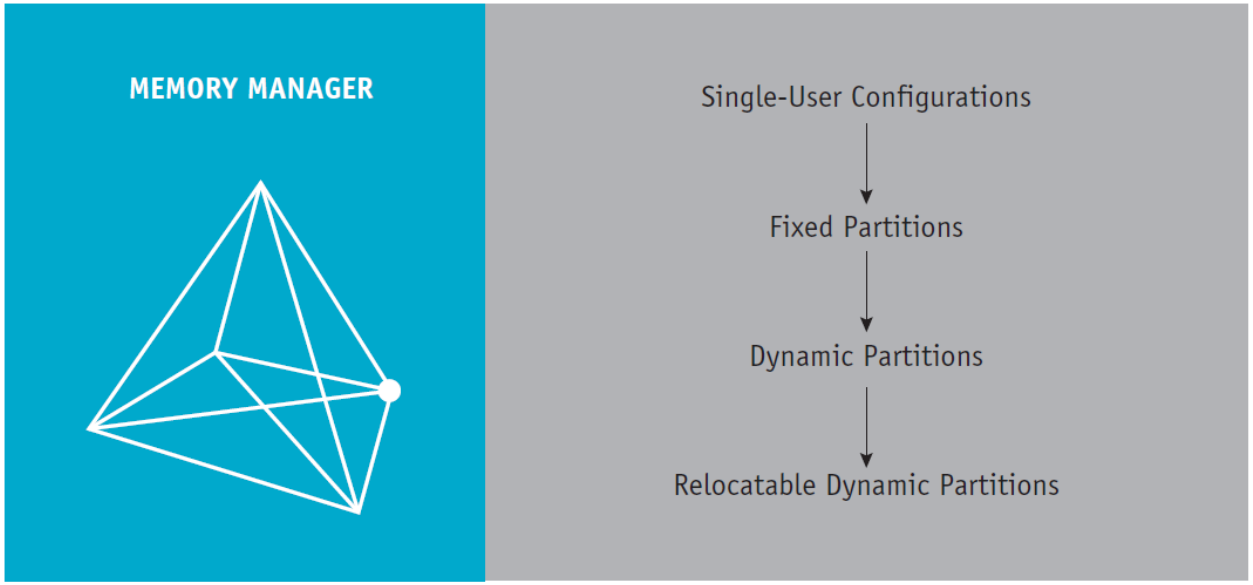
Department of Information and Communication Engineering
Institute of Technology of Cambodia

Review

- There are 4 essential managers (software) of every major operating system:
  - Memory manager (in charge of the use of main memory - RAM).
  - Processor manager
  - Device manager
  - File manager
  - Network manager



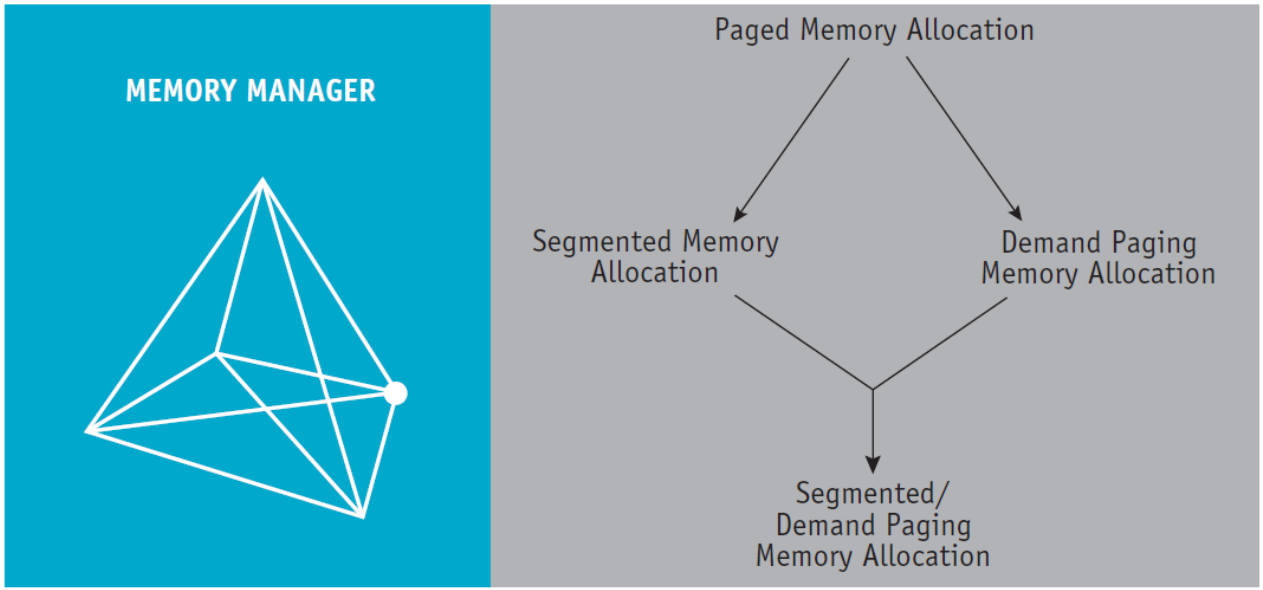
Memory Manager



"Memory is the primary and fundamental power, without which there could be no other intellectual operation."

—Samuel Johnson (1709–1784)

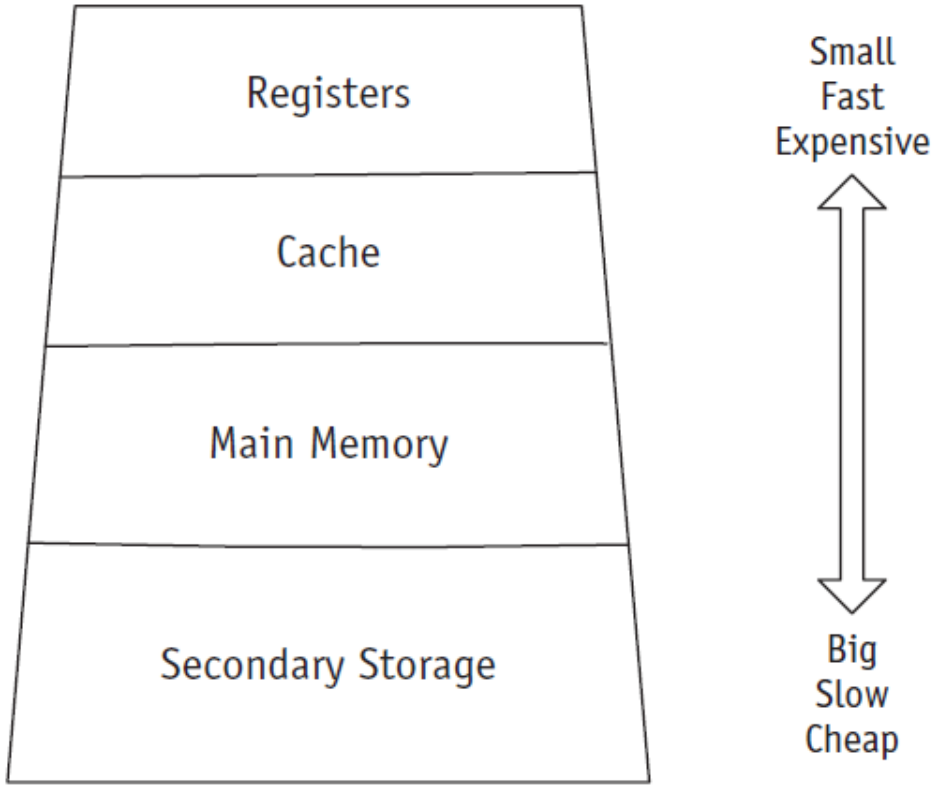
Virtual Memory



"Nothing is so much strengthened by practice, or weakened by neglect, as memory."

—Quintillian *(A.D. 35–100)*

Hierarchy of Data Storage



New Management Schemes

- Why? To remove the restriction of storing the programs contiguously, and most of them eliminate the requirement that the entire program reside in memory during its execution.
- The 4 new memory management schemes:
  - Paged memory allocation
  - Demand paging memory allocation
  - Segmented memory allocation
  - Segmented / demand paging memory allocation

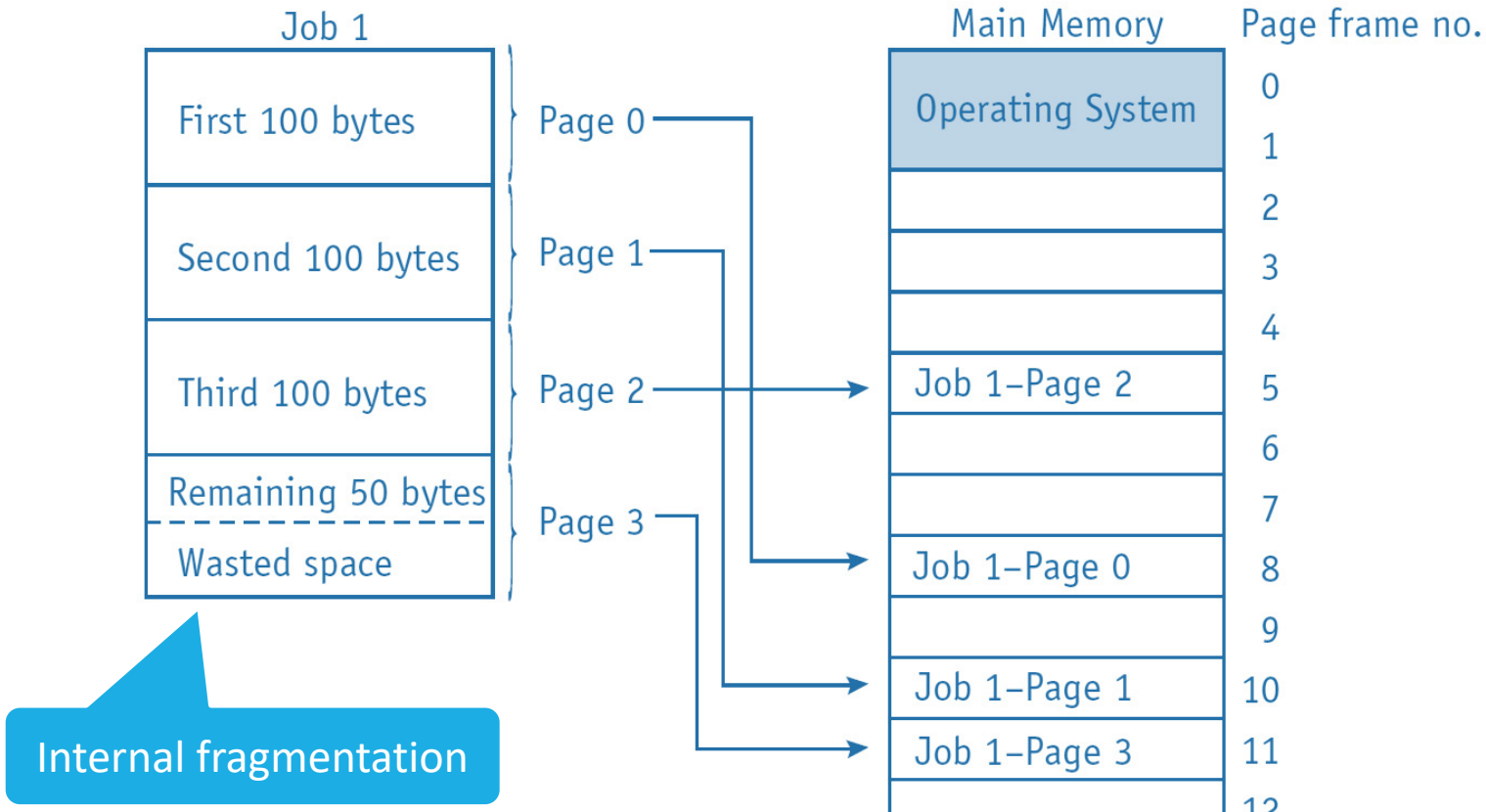
The Paged Memory (1/6)

- Paged memory allocation is based on the concept of dividing jobs into units of equal size and each unit is called a page.
- Works quite efficiently when the pages, sectors, and page frames are all the same size (the number of bytes that can be stored in each of them) which is usually determined by the disk's sector size.
- Therefore, one sector will hold one page of job instructions (or data) and fit into one page frame of memory.
- For many decades, the standard size of a page, page frame, and sector was identical at 512-bytes. Now the standard is moving to sectors of 4096-bytes (4K).

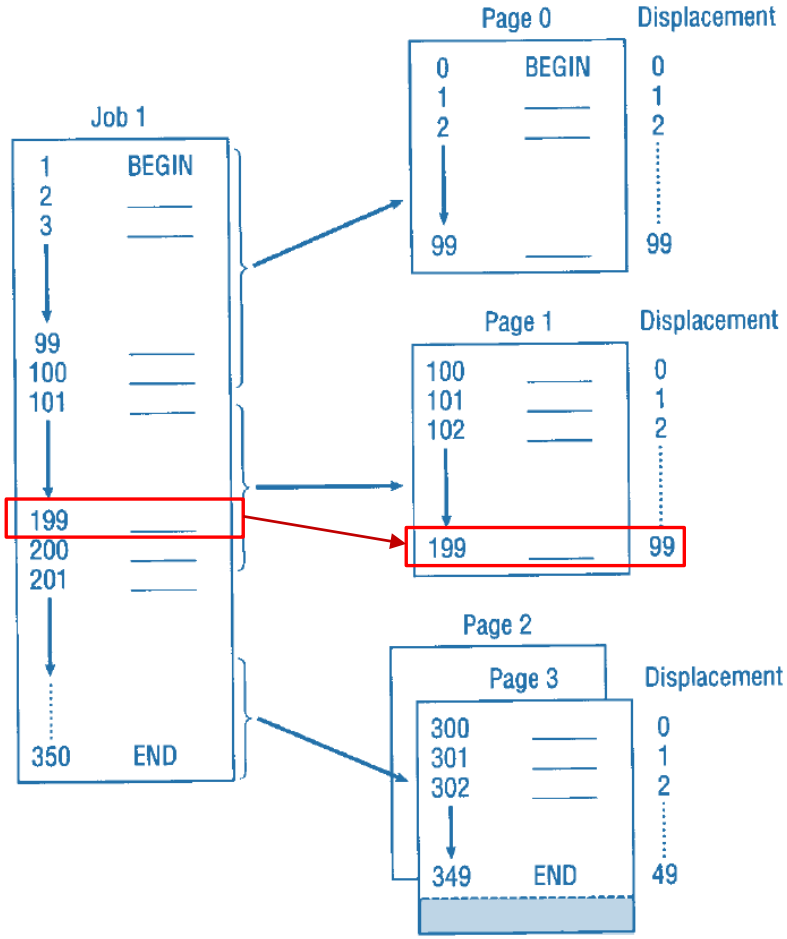
The Paged Memory (2/6)

- Before executing a program, a basic Memory Manager:
  - 1. Determines the number of pages in the program
  - 2. Locates enough empty page frames in main memory
  - 3. Loads all of the program's pages into those frames
- With the paged memory allocation, pages can be loaded in noncontiguous page frames. In fact, each page can be stored in any available page frame anywhere in main memory.
- There is no external fragmentation between page frames.
- Memory Manager now needs a mechanism to keep track of them—and that means enlarging the size, complexity, and overhead of the operating system software.

The Paged Memory (3/6)



The Paged Memory (4/6)



- Job 1 has 350 bytes.
- Size of page and frame is 100.

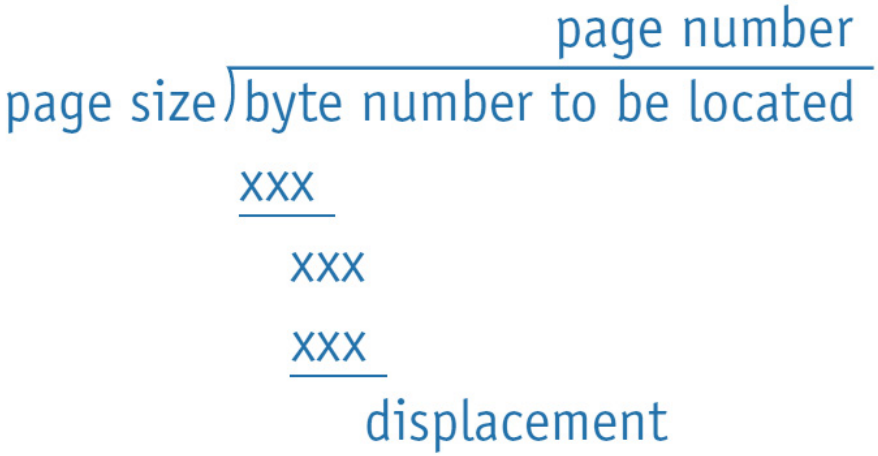
Find page number and displacement of byte 199?

* Page => 1
* Displacement => 99

The Paged Memory (5/6)

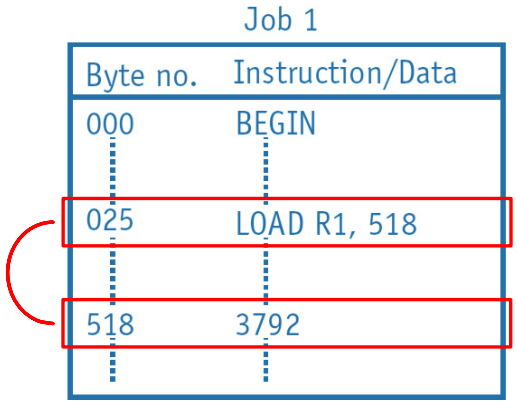
- Displacement (offset): in a paged or segmented memory allocation environment, the difference between a page's relative address and the actual machine language address.

BYTE_NUMBER_TO_BE_LOCATED / PAGE_SIZE = PAGE_NUMBER



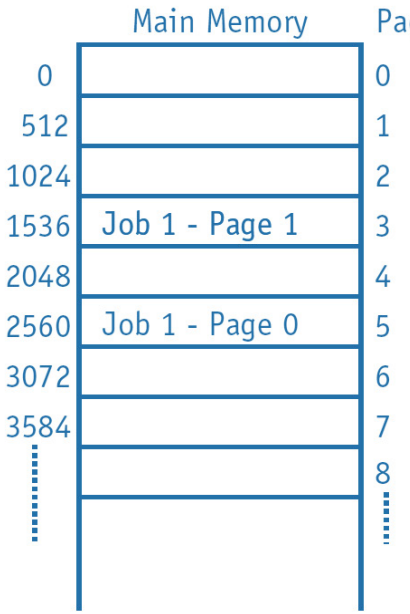
The Paged Memory (6/6)

Size of page and frame is 512 bytes. Find actual address of (LOAD R1, 518)?



PMT for Job 1

Page no.	Page frame number
0	5
1	3



* 518/512
=> Page = 1, Displacement = 6

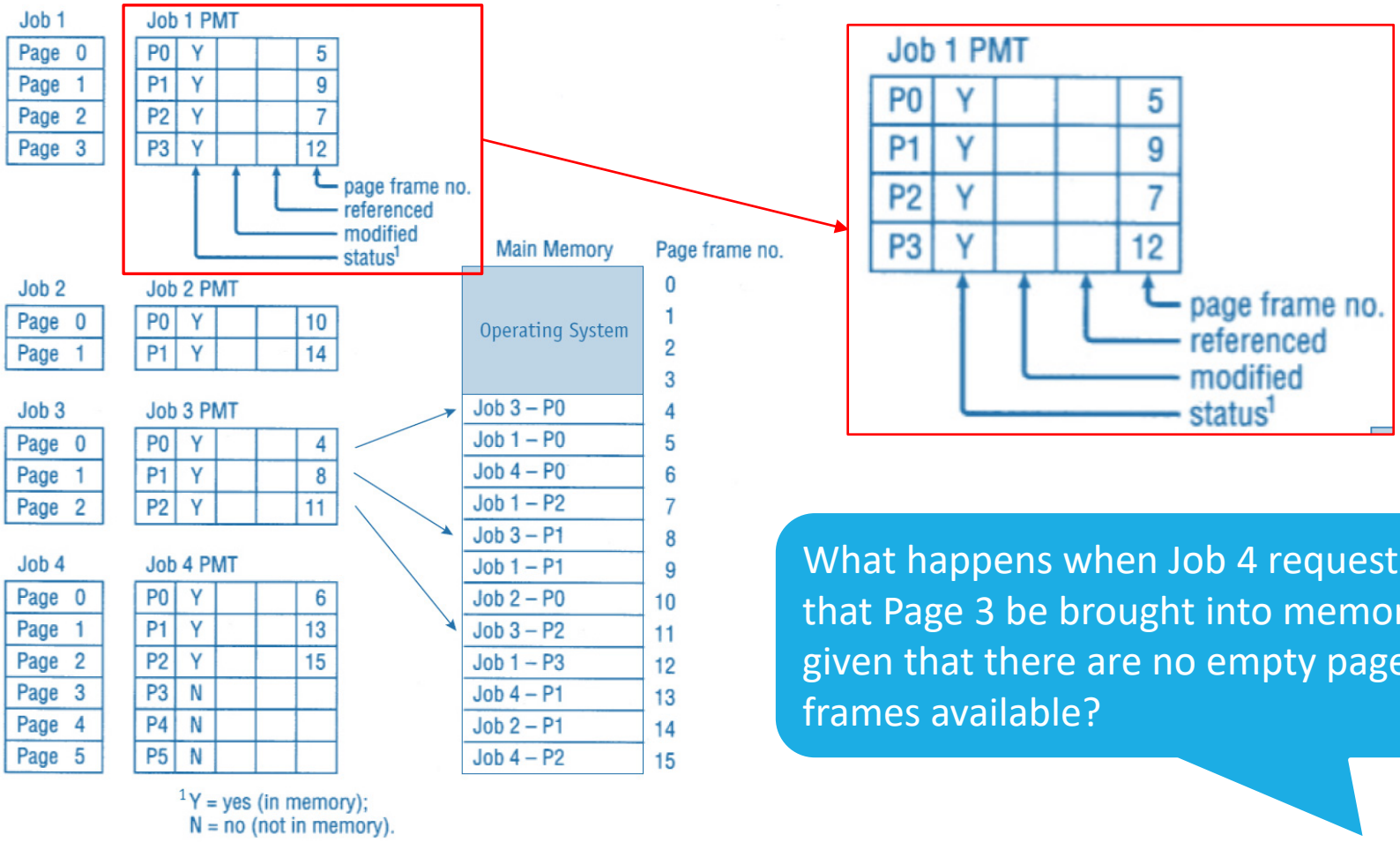
* PTM (Page 1 - Frame 3)
=> Address of Page 1 = 512*3 = 1,536

* Therefore, address of (LOAD R1, 518)
=> 1,536 + 6 = 1,542

Demand Paging Memory (1/5)

- Introduced the concept of loading only a part of the program into memory for processing.
- Was the first widely used scheme that removed the restriction of having the entire job in memory from the beginning to the end of its 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, and if they're never needed, they're never loaded.
- Examples: See page 67 of ref. book.

Demand Paging Memory (2/5)



What happens when Job 4 requests that Page 3 be brought into memory, given that there are no empty page frames available?

Demand Paging Memory (3/5)

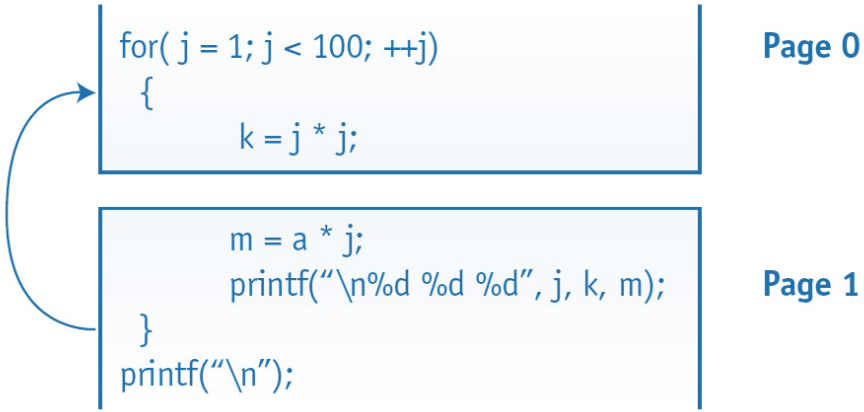
- How and when the pages are passed (also called swapped) between main memory and secondary storage depends on predefined policies that determine when to make room for needed pages and how to do so.
- The page fault handler determines whether there are empty page frames in memory so that the requested page can be immediately copied from secondary storage.
- If all page frames are busy, then the page fault handler must decide which page will be swapped out. (This decision is determined by the predefined policy for page removal.) Then the swap is made.

Demand Paging Memory (4/5)

- When there is an excessive amount of page swapping between main memory and secondary storage, the operation becomes inefficient. This phenomenon is called **thrashing**.
- Thrashing is caused when pages are frequently removed from memory but are called back shortly thereafter, thrashing uses a great deal of the computer's time and accomplishes very little.
- Thrashing can occur across jobs, when a large number of jobs are vying for a relatively low number of free pages (that is, the ratio of job pages to free memory page frames is high), or it can happen within a job.

Demand Paging Memory (5/5)

- Example of thrashing:
  - If there is only one free frame, swap will happened 199 times.
  - If a page can't be found in memory, it will cause a "**page fault**"; this example would generate 199 page faults (and swaps).



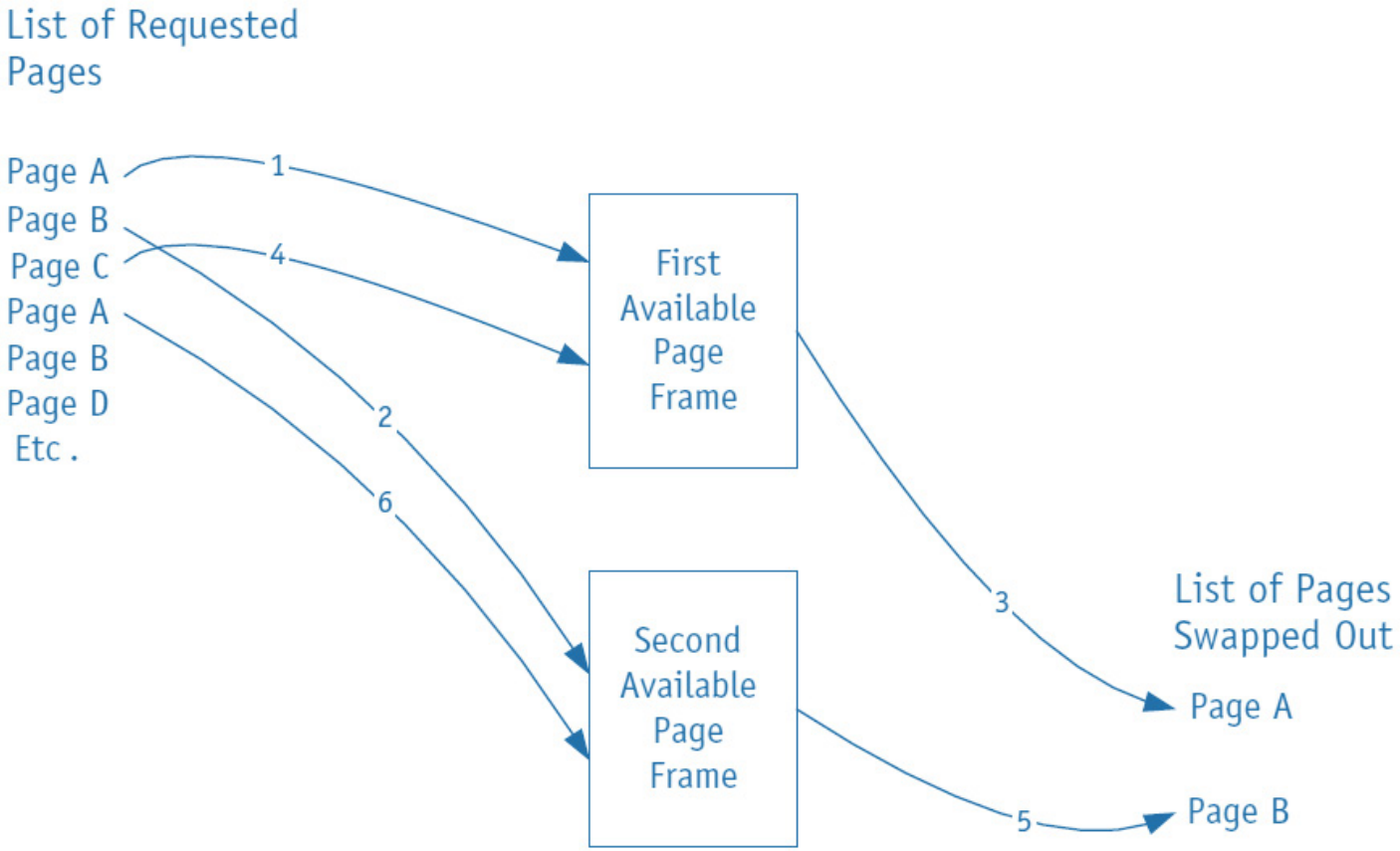
Page Replacement Policies and Concepts

- Two of the most well-known algorithms are first-in first-out and least recently used.
  - The first-in first-out (FIFO) policy is based on the assumption that the best page to remove is the one that has been in memory the longest.
  - The least recently used (LRU) policy chooses the page least recently accessed to be swapped out.

FIFO

- The first-in first-out (FIFO) page replacement policy will remove the pages that have been in memory the longest, that is, those that were “first in.”

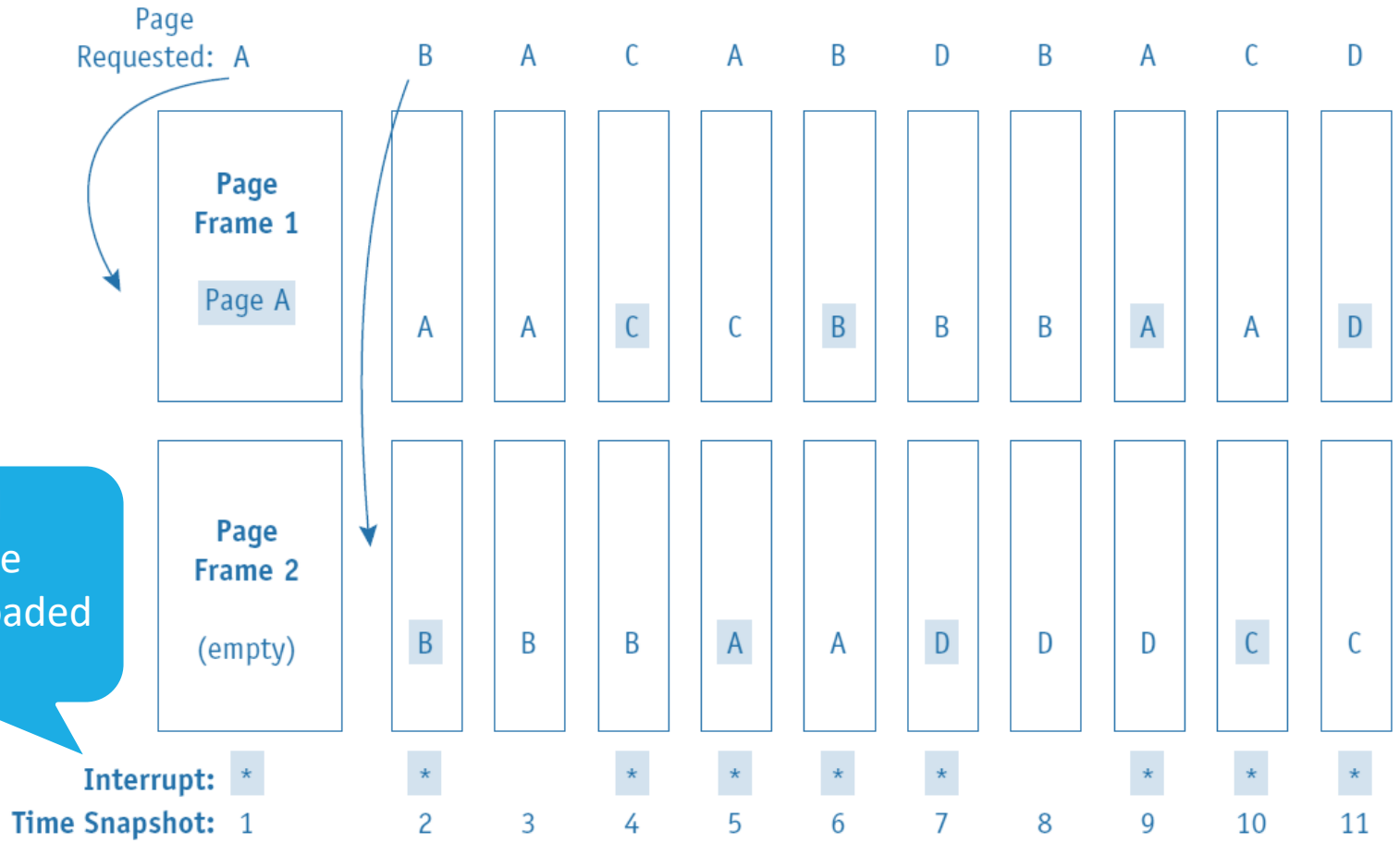
FIFO – Example 1 (1/2)



FIFO – Example 1 (2/2)

- For figure in the last slide:
  - Step 1: Page A is moved into the first available page frame.
  - Step 2: Page B is moved into the second available page frame.
  - Step 3: Page A is swapped into secondary storage.
  - Step 4: Page C is moved into the first available page frame.
  - Step 5: Page B is swapped into secondary storage.
  - Step 6: Page A is moved into the second available page frame.

FIFO – Example 2 (1/2)



It is generated anytime a page needs to be loaded into memory.

FIFO – Example 2 (2/2)

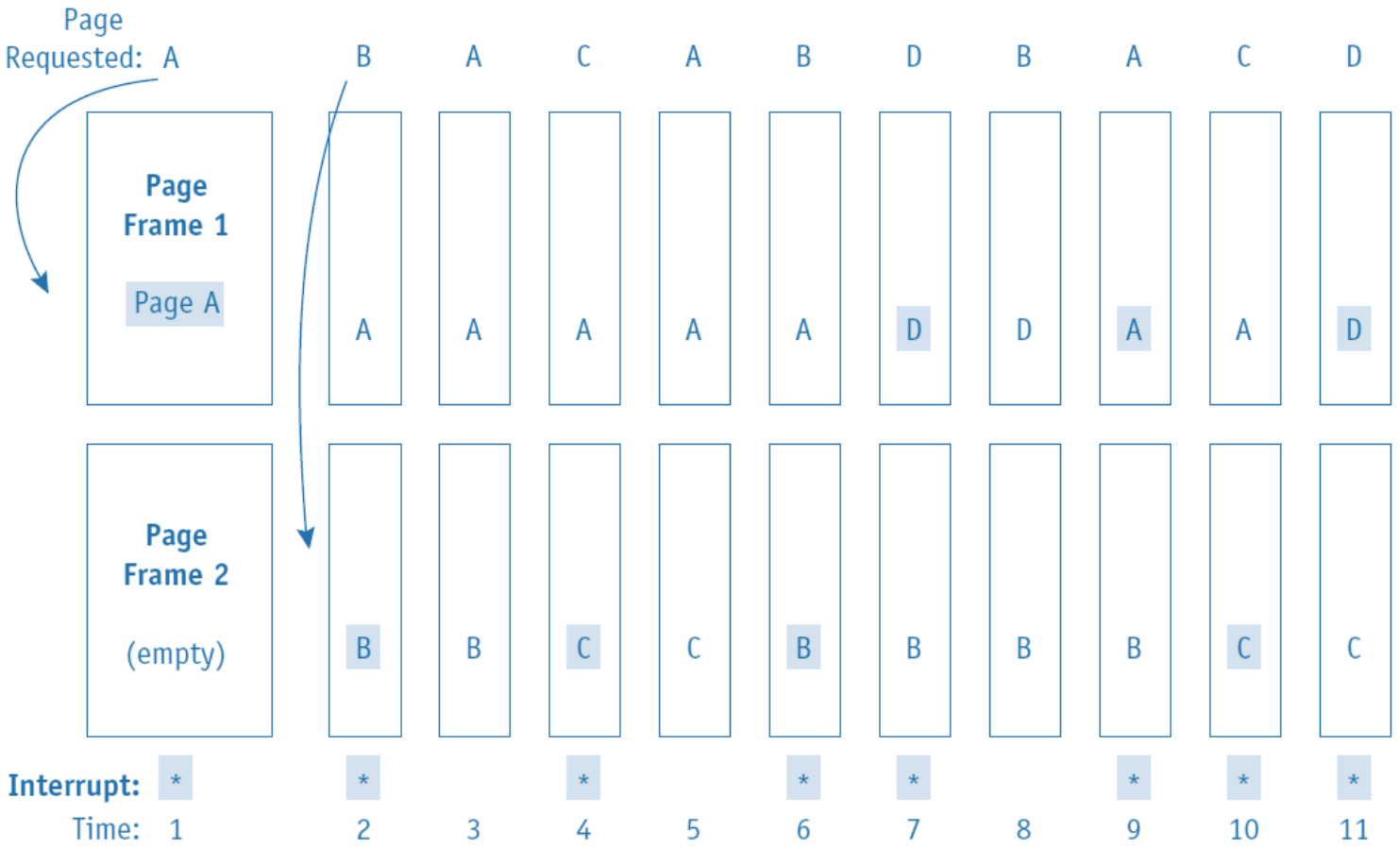
- The efficiency of this configuration is dismal due to the limited number of page frames available.
- There are 9 page interrupts out of 11 page requests, so failure rate is 9/11 (about 82%).
- The failure rate of this system is 9/11, which is 82 percent. Stated another way, the success rate is 2/11, or 18 percent. A failure rate this high is usually unacceptable.

Failure_Rate = Number_of_Interrupts / Page_Requests_Made

LRU

- The least recently used (LRU) page replacement policy swaps out the pages that show the least recent activity, figuring that these pages are the least likely to be used again in the immediate future.

LRU – Example (1/2)



LRU – Example (2/2)

- The failure rate of this system is 8/11, which is 73 percent, so the success rate is 3/11, or 27 percent.
- Compared to FIFO, the failure rate is decreased.

Homework

1. Enjoy your time!
2. Nothing is required to submit!