BACS1024 INTRODUCTION TO COMPUTER SYSTEMS

Chapter 11: Virtual Memory

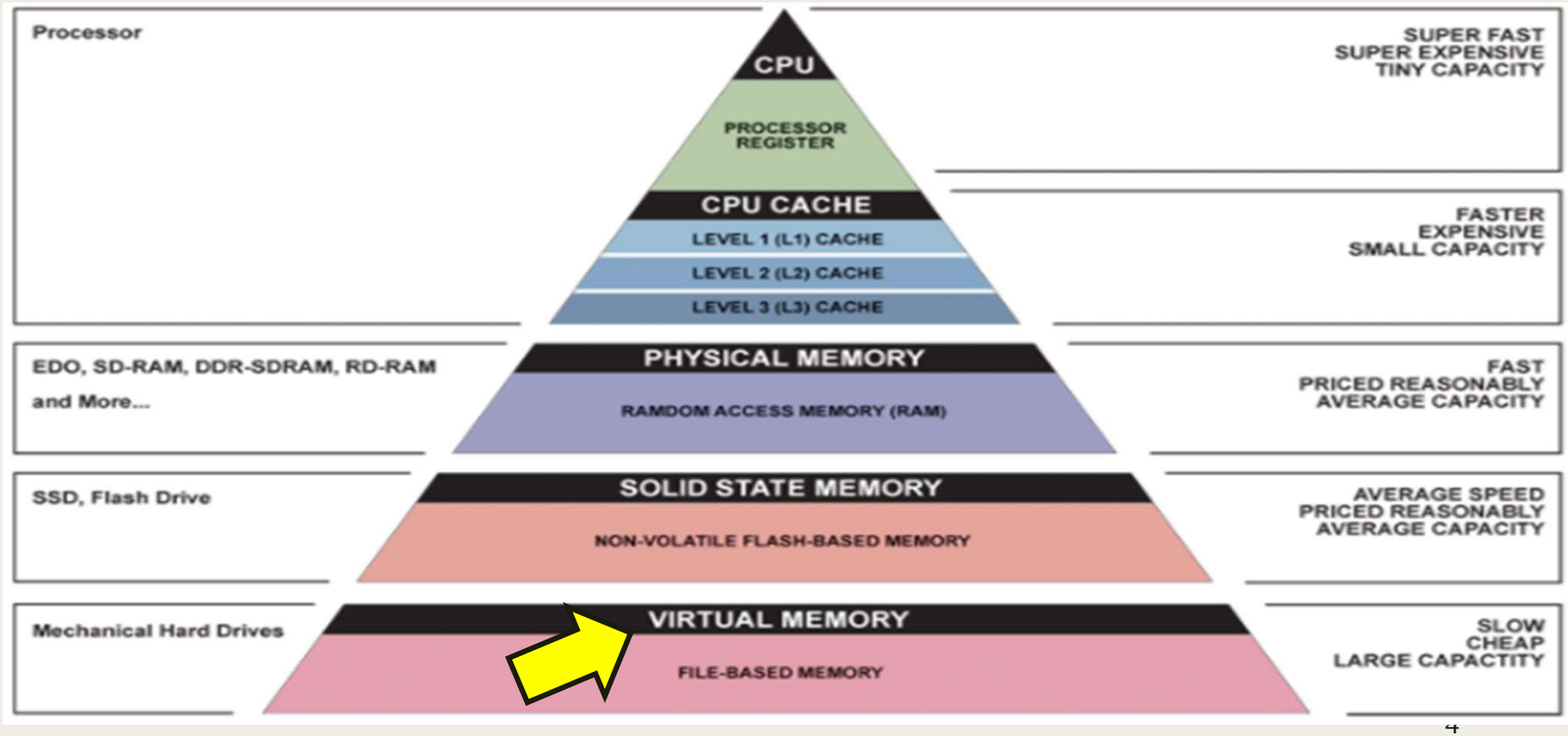
0. Overview

1. Virtual Memory Implementation
2. Demand Paging
3. Page Replacement Algorithms
4. Thrashing & Remedy

1. Virtual memory

1. Virtual Memory

- **Virtual memory**



1. Virtual Memory

- **Virtual memory (VM)**
  - Uses a combination of OS software & special purpose hardware to simulate a memory that meets the management needs of the modern computer system
  - Even though only a portion of each program is stored in memory, virtual memory gives the appearance that programs are being completely loaded in main memory during their entire processing time.
  - Virtual memory shared programs and subroutines are loaded “on demand,” reducing storage requirements of main memory.

1. Virtual Memory

- **Virtual memory (VM)**
  - **Advantages** of implementing virtual memory
    - Works well in a multiprogramming environment.
    - Job's size is no longer restricted to the size of main memory.
    - Memory is used more efficiently.
    - Eliminates external fragmentation when used with paging and eliminates internal fragmentation when used with segmentation.
    - Allows the sharing of code and data.
  - **Disadvantages** of implementing virtual memory
    - Increased processor hardware costs.
    - Increased overhead for handling paging interrupts.
    - Increased software complexity to prevent thrashing.

1. Virtual Memory

- **Virtual memory (VM)**
  - Implementation of virtual memory
    - VM is implemented through **demand paging** and **segmentation schemes**.

  (a) Paged memory allocation

  - Divides each incoming job into pages of equal size.
  - Works well if page size = size of memory block size (page frames) = size of disk section (sector, block).
  - Before executing a program, the Memory Manager:

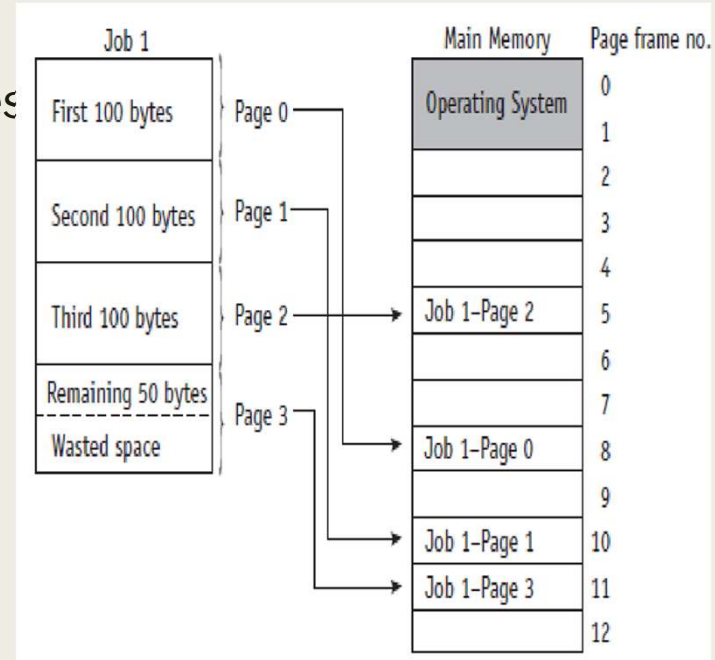
    1. Determines number of pages in program.
    2. Locates enough empty page frames in main memory.
    3. Loads all of the program's pages into them.

1. Virtual Memory

- **Virtual memory (VM)**

  (a) Paged memory allocation

  - At compilation time every job is divided into pages:
    - Page 0 contains the first hundred lines.
    - Page 1 contains the second hundred lines
    - Page 2 contains the third hundred lines.
    - Page 3 contains the last fifty lines.
  - Program has 350 lines.
    - Referred to by system as line 0 through line 349.



1. Virtual Memory

- **Virtual memory (VM)**

(a) Paged memory allocation

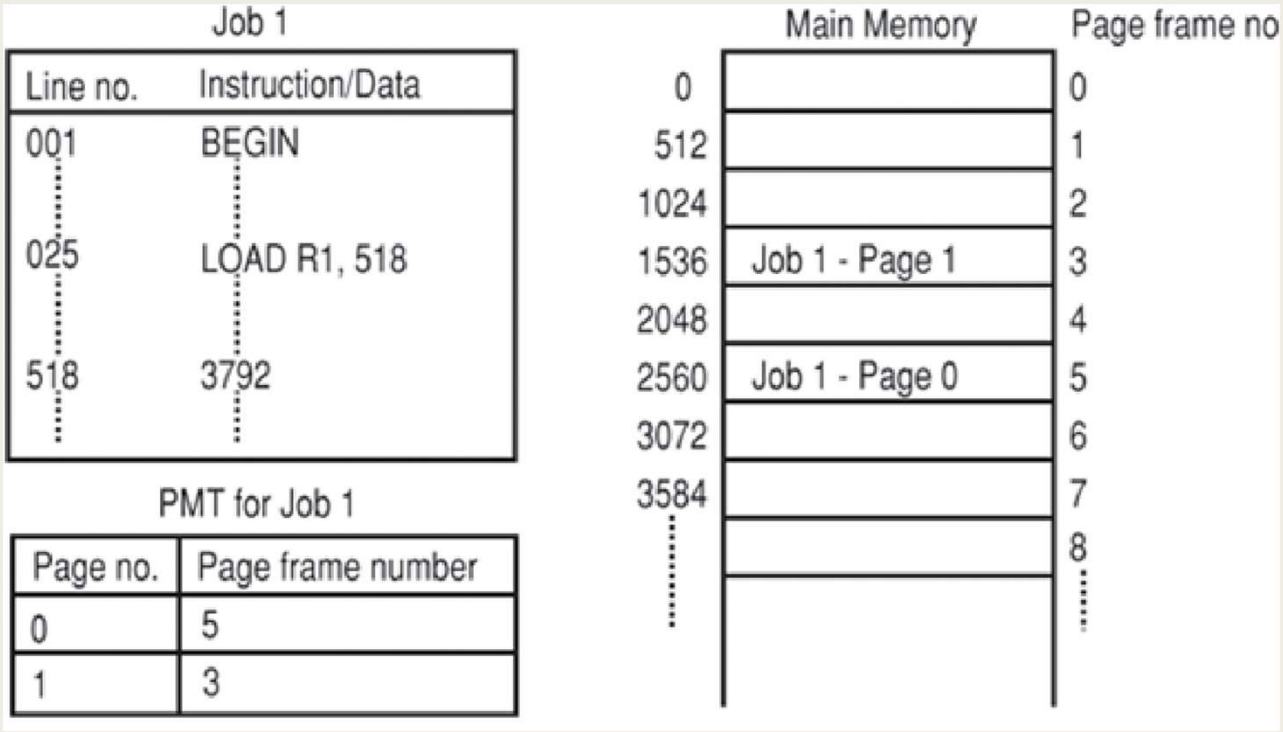
- Paging Requires 3 Tables to Track a Job's Pages

1. Job Table (JT) - 2 entries for each active job.
   - Size of job & memory location of its page map table.
   - Dynamic – grows/shrinks as jobs loaded/completed.
2. Page Map Table (PMT) - 1 entry per page.
   - Page number & corresponding page frame memory address.
   - Page numbers are sequential (Page 0, Page 1 …)
3. Memory Map Table (MMT) - 1 entry for each page frame.
   - Location & free/busy status.

1. Virtual Memory

- **Virtual memory (VM)**

  (a) Paged memory allocation

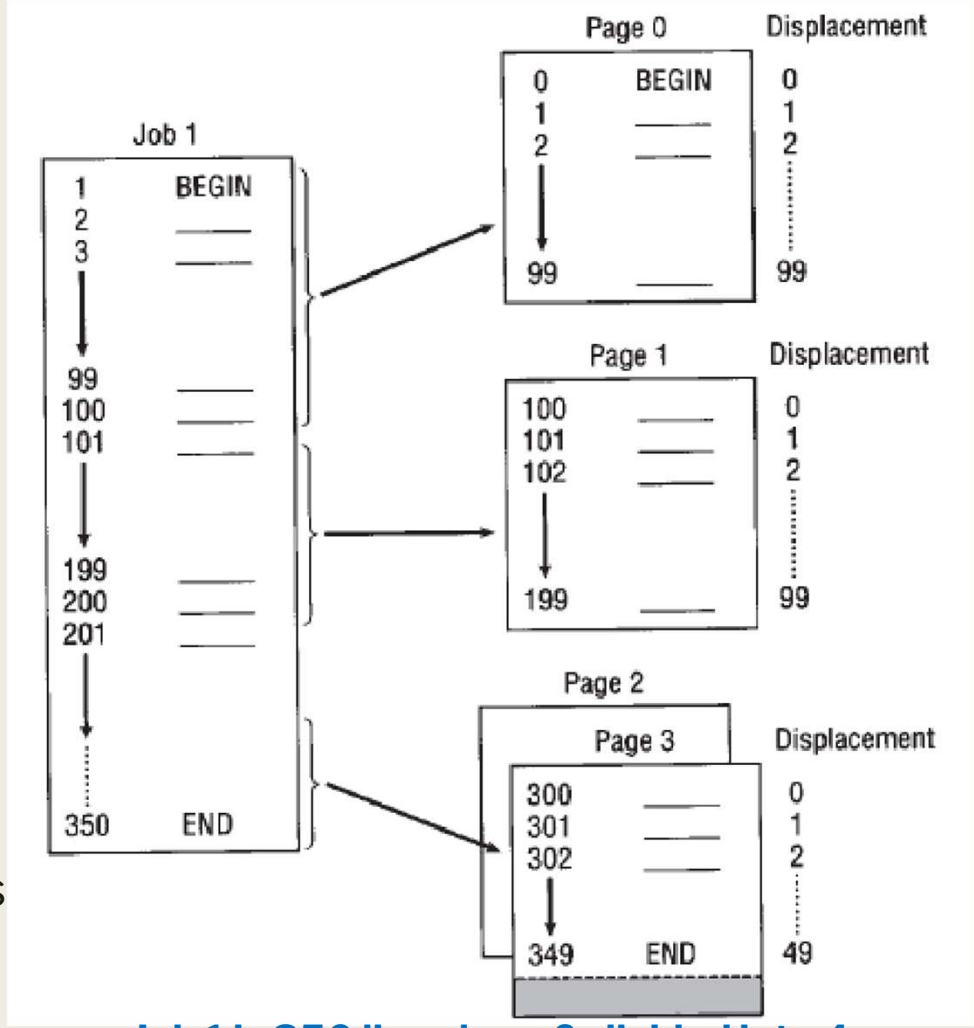


1. Virtual Memory

- **Virtual memory (VM)**

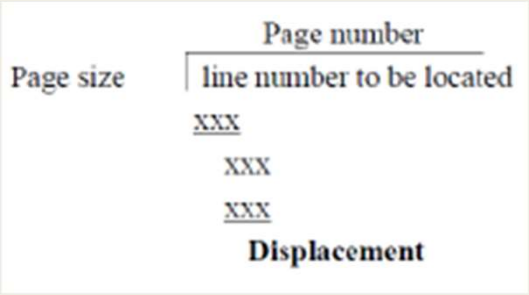
  (a) Paged memory allocation

  - The displacement (offset) of a line indicates how far away a line is from the beginning of its page.
    - Used to locate that line within its page frame.
    - Relative factor.
  - For example, lines 0, 100, 200, and 300 are first lines for pages 0, 1, 2, and 3 respectively so each has displacement of zero.



Job 1 is 350 lines long & divided into 4 pages

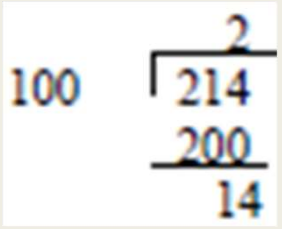
1. Virtual Memory



- **Virtual memory (VM)**

  (a) Paged memory allocation

  - To find the address of a given program line, divide the line number by the page size, keeping the remainder as an integer
  - E.g.: If we use 100 lines as the page size, the page number and the displacement (the location within that page) of Line 214 can be calculated:-



    - Quotient 2 ➔ Page Number
    - Remainder 2 ➔ Displacement
    - Line 214 is located on Page 2, 15 lines (Line 14) from the top of the page

1. Virtual Memory

- **Virtual memory (VM)**

  (a) Paged memory allocation

  - **Thrashing** Is a Problem With Paging
  - **Page fault** – a failure to find a page in memory.

1. Virtual Memory

- **Virtual memory (VM)**

  (a) Paged memory allocation

  - Advantages of paging
    - Allows jobs to be allocated in non-contiguous memory locations. Memory used more efficiently; more jobs can fit.
    - Reduces, but does not eliminate, internal fragmentation.
    - Size of page is crucial (not too small, not too large).
  - Disadvantages of paging
    - Increased overhead occurs.

1. Virtual Memory

- **Virtual memory (VM)**

  (b) Segmented memory allocation

  - Programmers commonly structure their programs in modules (logical groupings of code).
    - A segment is a logical unit such as: main program, subroutine, procedure, function, local variables, global variables, common block, stack, symbol table, or array.
    - Main memory is not divided into page frames because the size of each segment is different.
  - In a segmented memory allocation scheme, jobs are divided into a number of distinct logical units called segments, one for each module that contains pieces that performs related functions.
  - • Memory is allocated dynamically.

1. Virtual Memory

- **Virtual memory (VM)**

(b) Segmented memory allocation

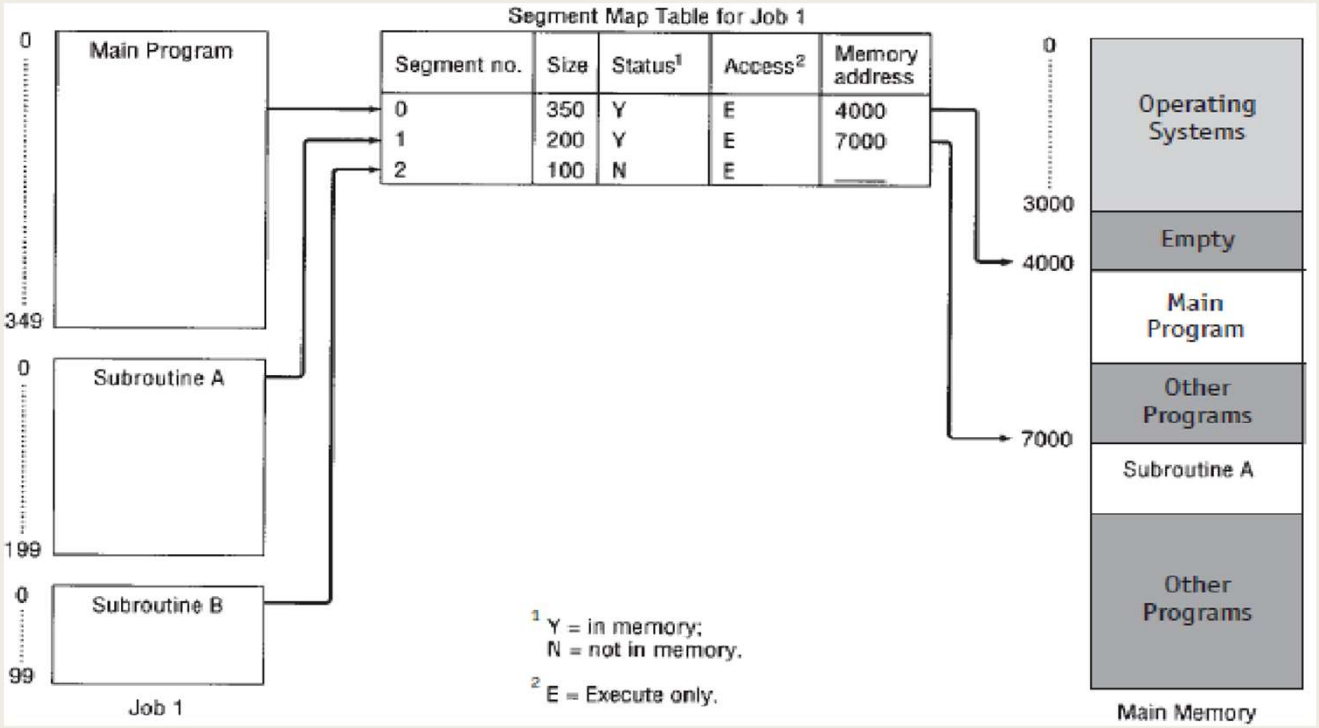
- **Segment Map Table (SMT)**
  - When a program is compiled, segments are set up according to program's structural modules.
  - Each segment is numbered and a Segment Map Table (SMT) is generated for each job. Contains segment numbers, their lengths, access rights, status, and (when each is loaded into memory) its location in memory.
- **Tables Used in Segmentation**
  - Memory Manager needs to track segments in memory:

    1. Job Table (JT) lists every job in process (one for whole system).
    2. Segment Map Table lists details about each segment (one for each job).
    3. Memory Map Table monitors allocation of main memory (one for whole system).

1. Virtual Memory

- **Virtual memory (VM)**

  (b) Segmented memory allocation



1. Virtual Memory

- **Virtual memory (VM)**

(b) Segmented memory allocation

- **Advantages of segmentation**
  - Compaction.
  - External fragmentation.
  - Secondary storage handling.
  - Memory is allocated dynamically.
- **Disadvantages of segmentation**
  - External fragmentation.
  - Costly memory management algorithms.

1. Virtual Memory

- **Virtual memory (VM)**
  - Implementation of virtual memory

Paging	Segmentation
Allows internal fragmentation within page frames	Doesn't allow internal fragmentation
Doesn't allow external fragmentation	Allows external fragmentation
Programs are divided into equal-sized pages	Programs are divided into unequal-sized segments
Absolute address calculated using page number and displacement	Absolute address calculated using segment number and displacement
Requires PMT	Requires SMT

2. Demand Paging

2. Demand Paging

- **Demand Paging**
  - Bring a page into memory only when it is needed, so less I/O and memory is needed.
  - Takes advantage of the fact that programs are written sequentially, so not all pages are needed at once. E.g.:
    - User-written error handling modules.
    - Mutually exclusive modules.
    - Certain program options are either mutually exclusive or not always accessible.
    - Many tables assigned fixed amount of address space even though only a fraction of table is actually used.
  - Demand paging has made **virtual memory** widely available. Can give appearance of an almost-infinite or nonfinite amount of physical memory.

2. Demand Paging

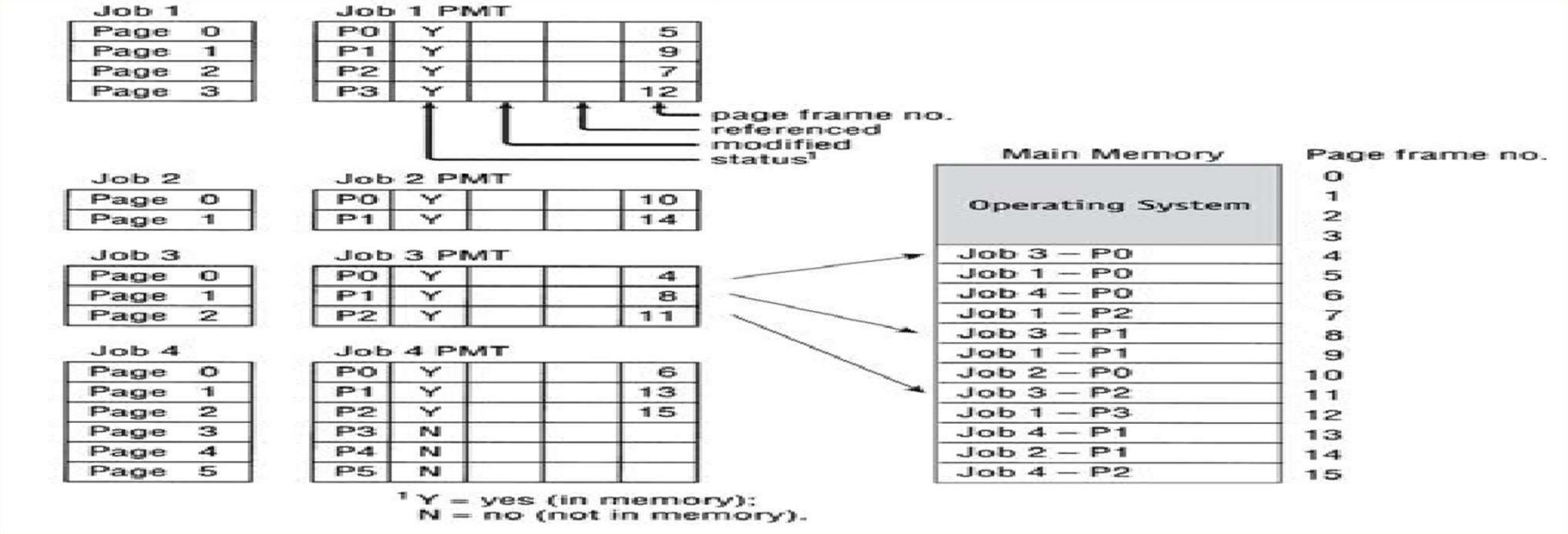
- **Demand Paging**
  - Requires use of a high-speed direct access storage device that can work directly with CPU.
  - How and when the pages are passed (or "**swapped**") depends on predefined policies that determine when to make room for needed pages and how to do so.
  - **Tables in Demand Paging**
    - **Job Table.**
    - **Page Map Table** (with 3 new fields).
      1. Determines if requested page is already in memory.
      2. Determines if page contents have been modified.
      3. Determines if the page has been referenced recently. Used to determine which pages should remain in main memory and which should be swapped out.
    - **Memory Map Table.**

2. Demand Paging

- **Demand Paging**
  - Requires use of a high-speed direct access storage device that can work directly with CPU.
  - How and when the pages are passed (or "**swapped**") depends on predefined policies that determine when to make room for needed pages and how to do so.
  - **Tables in Demand Paging**
    - **Job Table.**
    - **Page Map Table** (with 3 new fields).
      1. Determines if requested page is already in memory.
      2. Determines if page contents have been modified.
      3. Determines if the page has been referenced recently. Used to determine which pages should remain in main memory and which should be swapped out.
    - **Memory Map Table.**

2. Demand Paging

- **Demand Paging**



- 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.

2. Demand Paging

- **Demand Paging**
  - Advantages of demand paging
    - A job is no longer constrained by the size of physical memory (virtual memory).
    - Uses memory more efficiently than previous schemes because sections of a job used seldom or not at all aren't loaded into memory unless specifically requested.
  - Disadvantages of demand paging
    - Increased overhead caused by tables and page interrupts.

3. Page Replacement Policies

3. Paging Replacement Policies

- **Page Replacement Policies**
  - Policy that selects page to be removed is crucial to system efficiency. Policies used include:
    - **First-in first-out (FIFO) policy**
      - ✔ Best page to remove is the one that has been in memory the longest.
    - **Least-recently-used (LRU) policy**
      - ✔ Chooses pages least recently accessed to be swapped out.

3. Paging Replacement Policies

- **Page Replacement Policies**

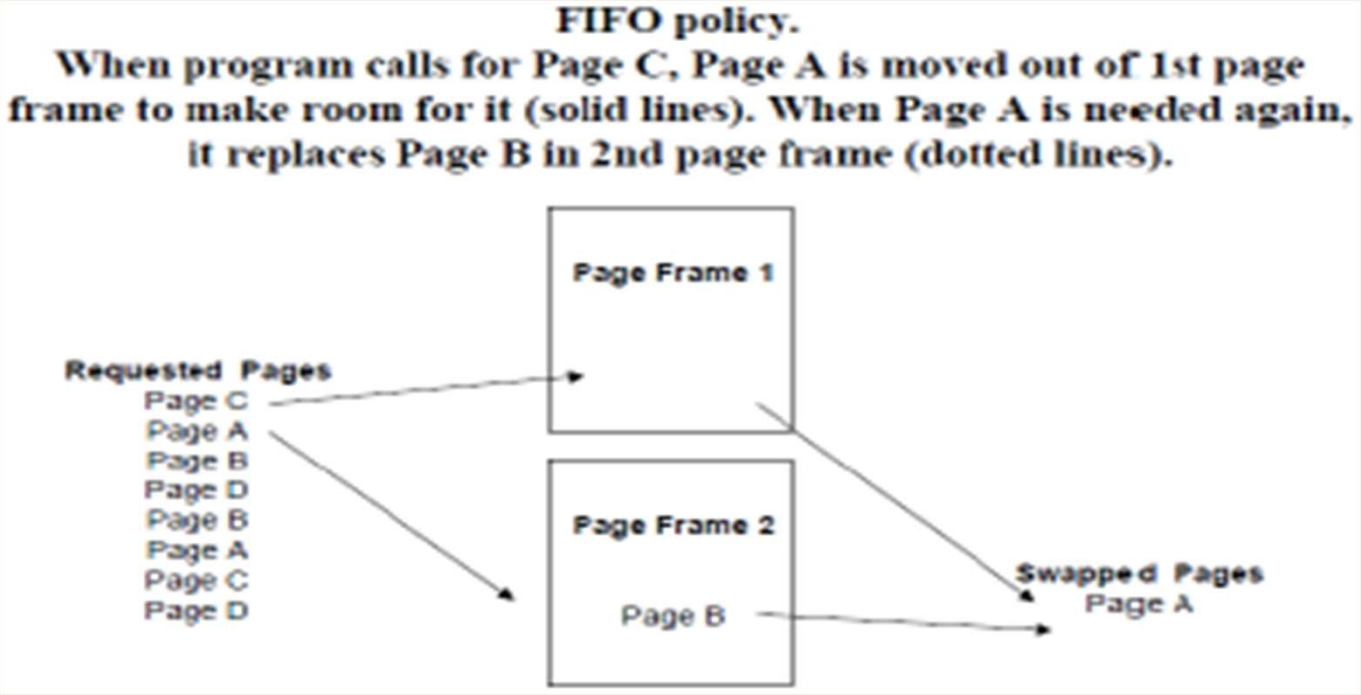
(a) First-in first-out (FIFO) policy

- High failure rate shown in previous example caused by:
  - Limited amount of memory available.
  - Order in which pages are requested by program (can't change).
- There is no guarantee that buying more memory will always result in better performance (FIFO anomaly or Belady's anomaly).

3. Paging Replacement Policies

- **Page Replacement Policies**

  (a) First-in first-out (FIFO) policy

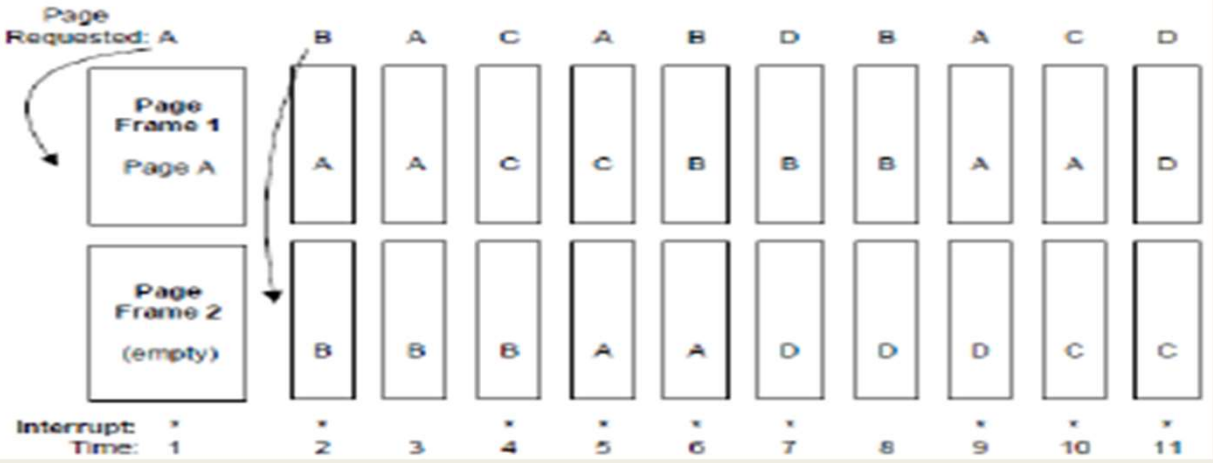


3. Paging Replacement Policies

- **Page Replacement Policies**

  (a) First-in first-out (FIFO) policy

How each page requested is swapped into 2 available page frames using FIFO. When program is ready to be processed all 4 pages are on secondary storage. Throughout program, 11 page requests are issued. When program calls a page that isn't already in memory, a page interrupt is issued (shown by *). 9 page interrupts result.



3. Paging Replacement Policies

- **Page Replacement Policies**

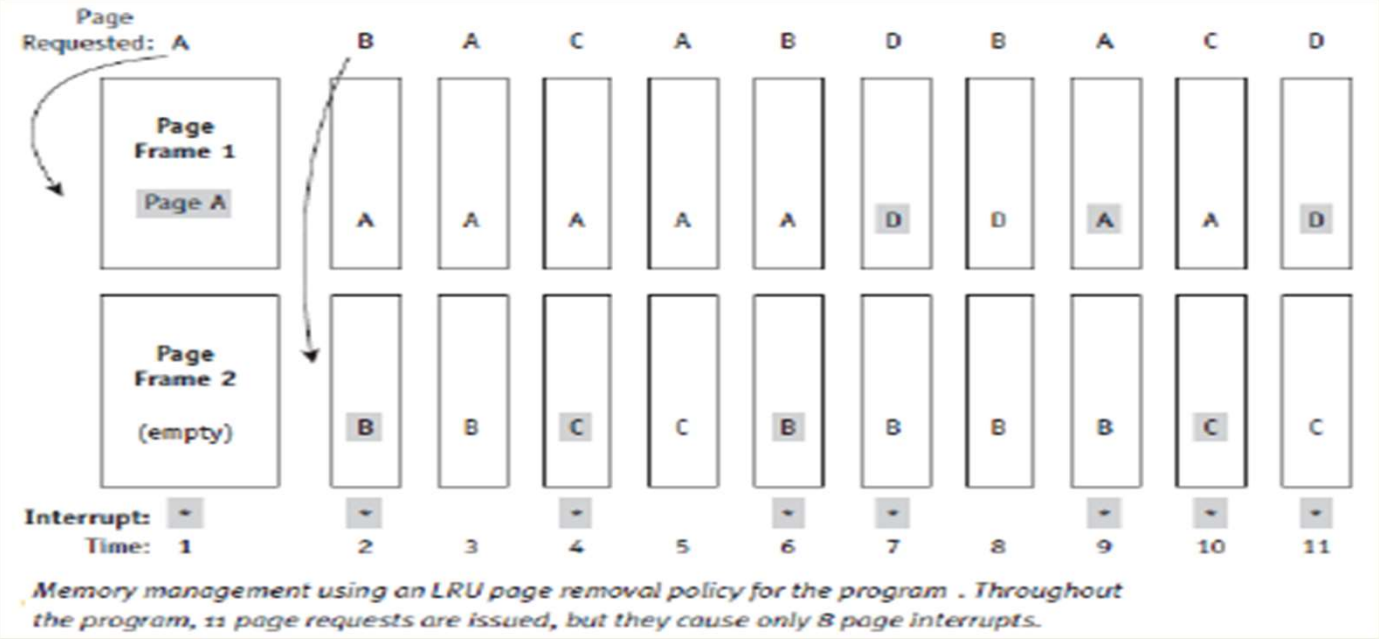
  (b) Least Recent Used (LRU) policy

  - ❑ The efficiency of LRU is only slightly better than with FIFO.
  - ❑ LRU is a stack algorithm removal policy – increasing main memory causes either a decrease in or same number of page interrupts.
  - ❑ LRU doesn't have same anomaly that FIFO does.

3. Paging Replacement Policies

- **Page Replacement Policies**

  (b) Least Recent Used (LRU) policy



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

4. Thrashing & Remedy

4. Thrashing & Remedy

- **Thrashing**
  - **Thrashing** Is a Problem With Demand Paging
  - Thrashing – an excessive amount of page swapping back and forth between main memory and secondary storage.
  - Effects:
    - Operation becomes inefficient.
    - Caused when a page is removed from memory but is called back shortly thereafter.
    - Can occur across jobs, when a large number of jobs are vying for a relatively few number of free pages.
    - Can happen within a job (e.g., in loops that cross page boundaries).

4. Thrashing & Remedy

- **Thrashing**
  - **Remedy**
    - Uses local page replacement where a process can only be allocated pages in its own region in memory.
    - If the swaps of a process increase also, the overall CPU utilization does not decrease much.
    - If other transactions have enough page frames in the partitions they occupy, they will continue to be processed efficiently

Chapter Review

Chapter Review

1. **Virtual Memory Implementation**
   - ❑ Paging
   - ❑ Segmentation
2. **Demand Paging**
3. **Page Replacement Algorithms**
   - ❑ FIFO
   - ❑ LRU
4. **Thrashing & Remedy**