Main memory management: virtual memory

Memory Manager

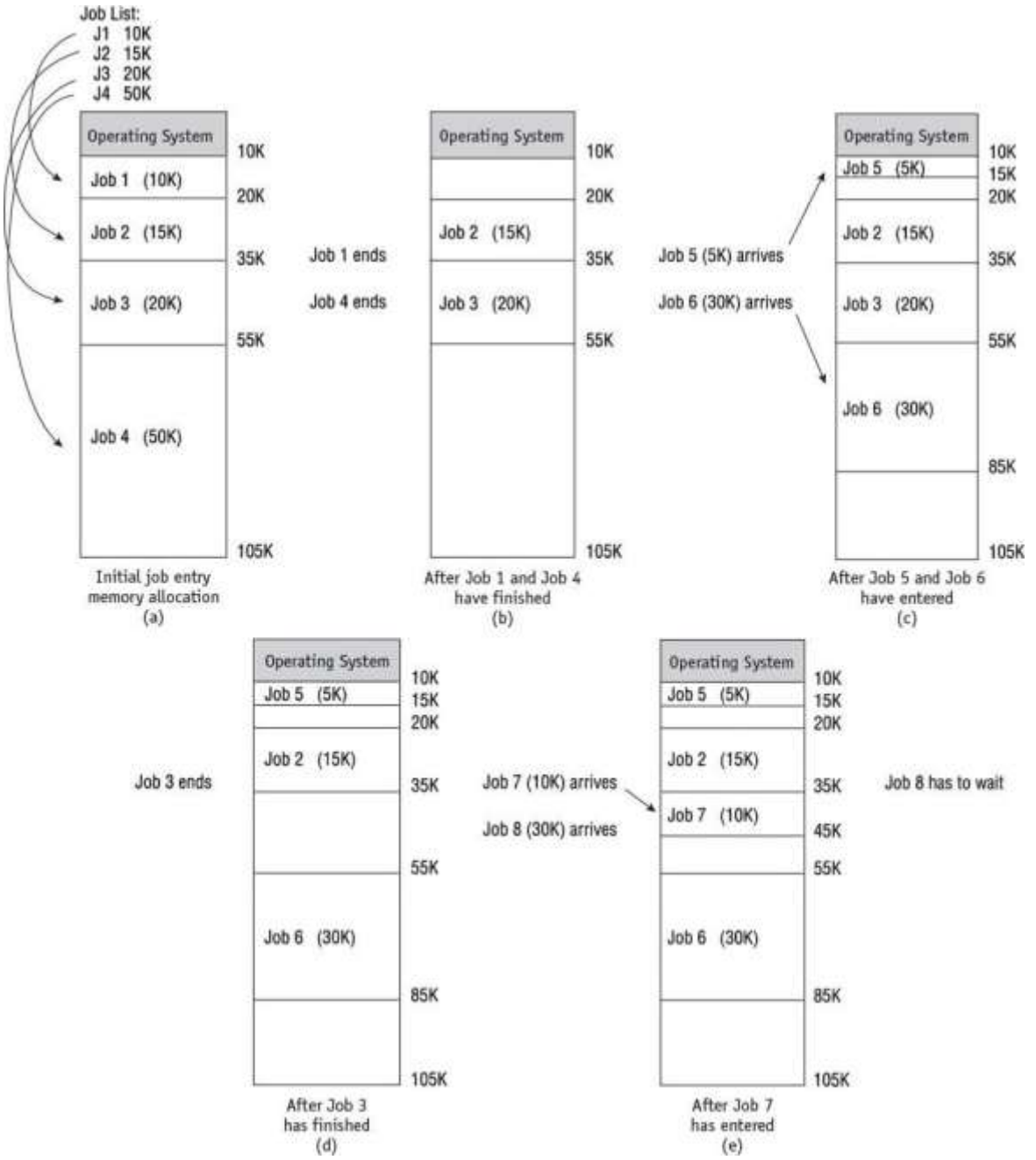
- In charge of main memory
  - Random Access Memory (RAM)
- Responsibilities include:
  - Preserving space in main memory occupied by operating system
  - Checking validity and legality of memory space request (can not use memory used by others programs)
  - Setting up memory tracking table
    - To keep track of who is using which section of memory
  - De-allocating memory to reclaim it

Old types of memory allocation

- Two main types:
  - Fixed partition system
  - Dynamic partitions system

- The main limitations of these basic memory allocation mechanisms are:
  - The program must be contiguous.
  - internal and external fragmentation (fragmentation refer to main memory space that can be utilised due to the nature of partitions)

- **Dynamic memory**
- Main memory use during dynamic partition allocation. Five snapshots (a-e) of main memory as eight jobs are submitted for processing and allocated space
- *Job 8 has to wait* (e) even though there's enough free memory between partitions to accommodate it.
- External fragmentation (unused memory between blocks)



Paged Memory Allocation

- Memory manager tasks: prior to program execution
  - Determine number of pages in program
  - Locate “enough” empty page frames (frames) in main memory
  - Load all program pages into page frames

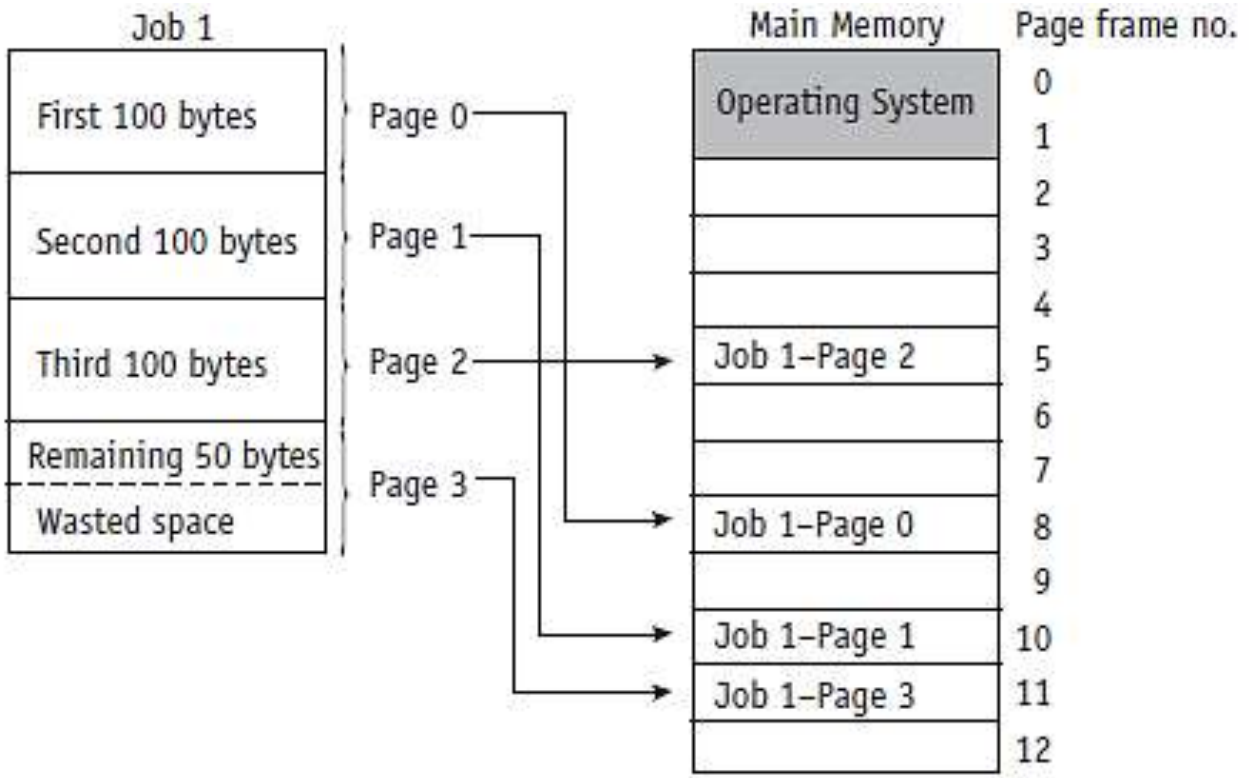
Paged Memory Allocation (cont'd.)

- Unlike the previous allocation mechanisms programs can be stored in *noncontiguous* page frames [in main memory]
  - **Advantages**:
    - more efficient memory use; no external fragmentation however, it does not completely eliminate internal fragmentation
  - **New problem**:
    - keeping track of a job's pages (what page corresponds to what "page" frame.
    - This increases operating system overhead

Paged Memory Allocation (cont'd.)

(figure 3.1)

- In this *simplified example*, each page frame can hold 100 bytes.
- This job, at 350 bytes long, is divided among four page frames



Internal fragmentation only in job's last frame: 50 bytes

Paged Memory Allocation (cont'd.)

- **Entire program**: required in memory during its execution
- *Three* tables required for tracking pages:
  1. Job Table (JT),
  2. Page Map Table (PMT), and
  3. Memory Map Table (MMT)

- These are stored in main memory operating system's area

Paged Memory Allocation (cont'd.)

- **Job Table**: information for each active job
  1. Job size
  2. Memory location: job's PMT

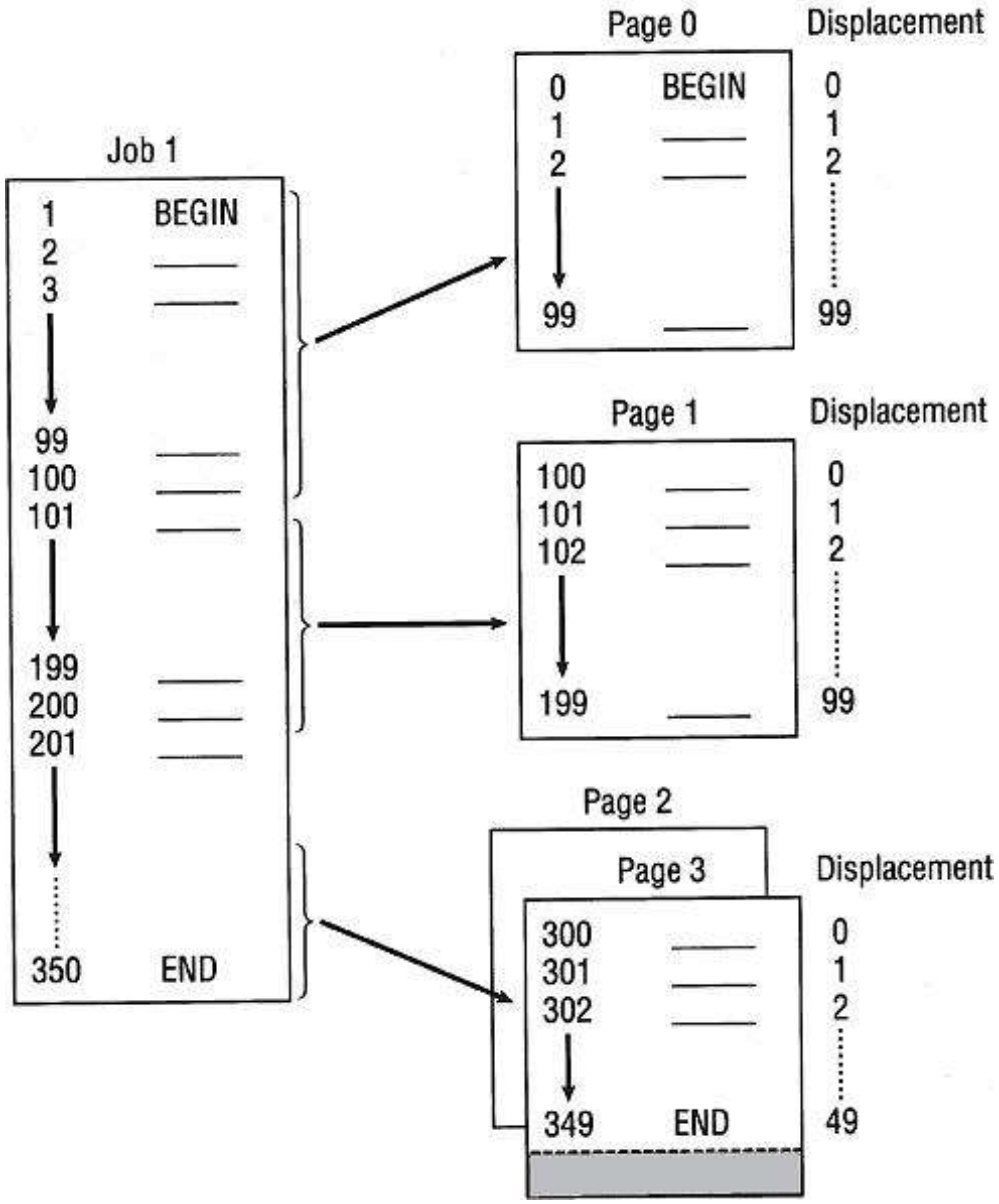
- **Page Map Table**: information for each page
  - Page number: beginning with Page 0
  - Memory address (Frame number)

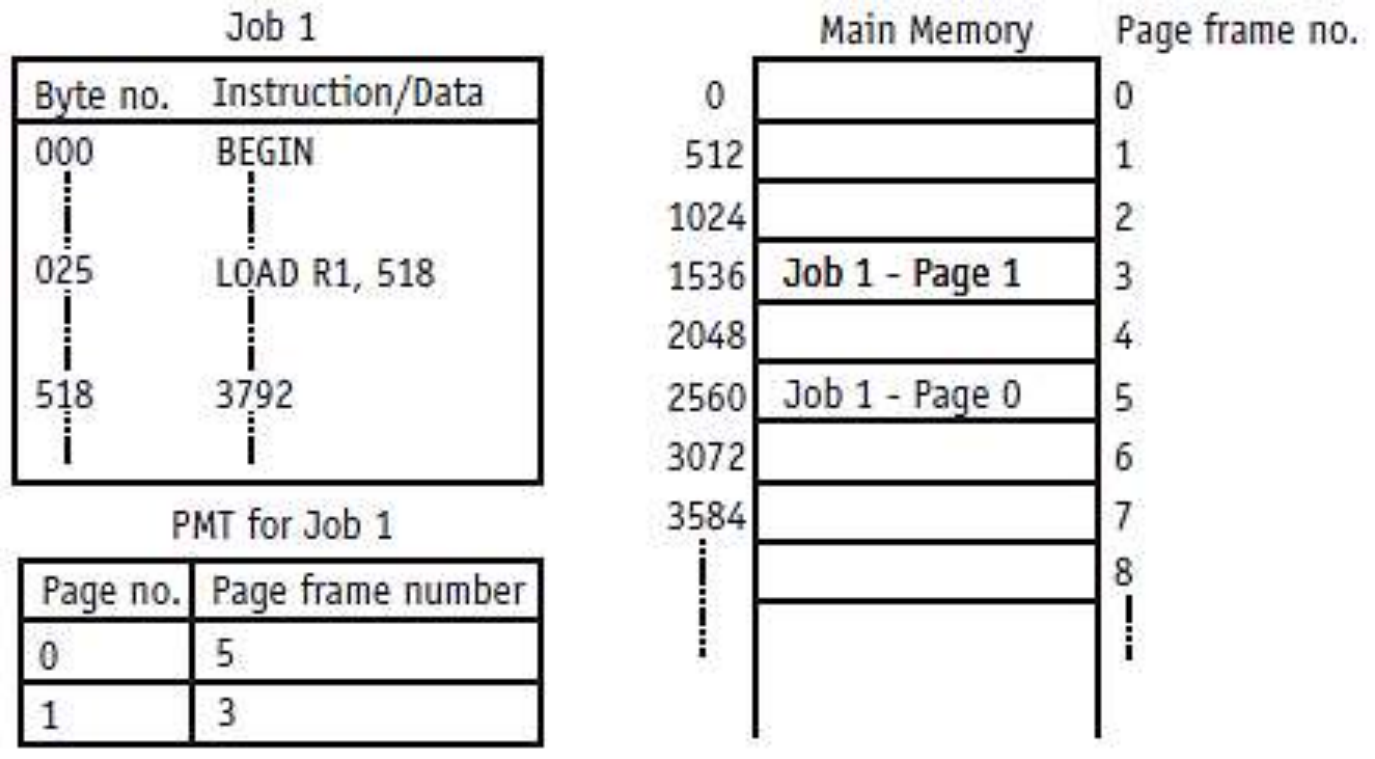
- **Memory Map Table**: basically entry for *each* page frame
  - *Location* of frame
  - Free/busy status

(figure 3.2)
This job is 350 bytes long and is divided into four pages of 100 bytes.

Each that are loaded into four page frames in memory.

The values 1..350 are referred to as the **logical address** (the way the CPU interoperates a program)





(figure 3.3): This system has page frame and page sizes of 512 bytes each. The PMT shows where the job's two pages are loaded into available page frames in main memory: the logical address is that in the job. E.g. **518;** the physical address is the actual location of this instruction.. In the above example frame 3 position 1536+6 = 1542

Paged Memory Allocation (cont'd.)

- Advantages
  - Efficient memory use: job allocation in noncontiguous memory
- Disadvantages
  - Increased overhead: address resolution
  - Internal fragmentation: last page
- Page size: crucial
  - Too small: very long PMTs
  - Too large: excessive internal fragmentation

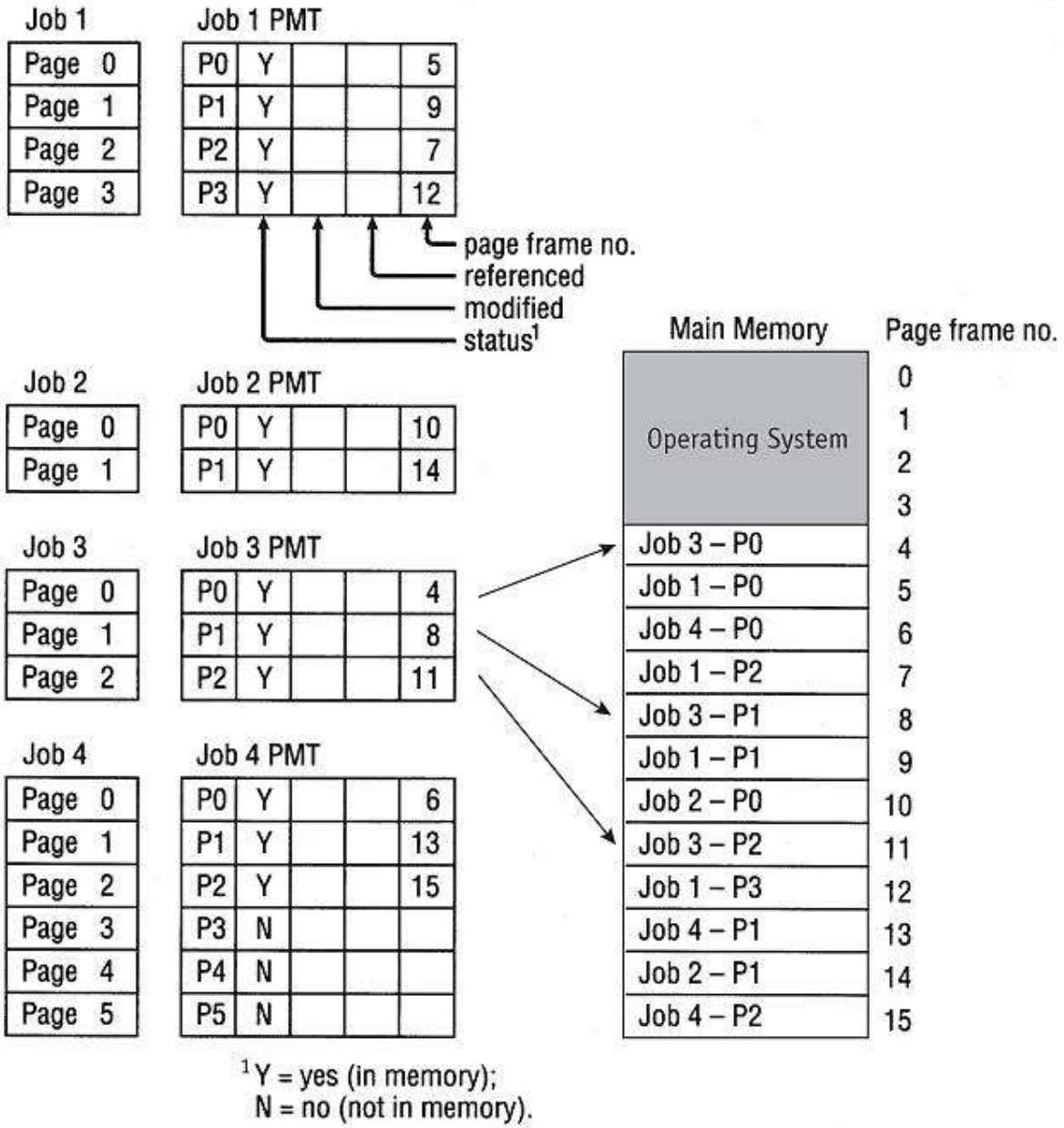
Virtual Memory Allocation

- Virtual memory or **demand** paged memory allocation: Loads only a part of the program into memory
  - Removes restriction: entire program in memory
  - However, Requires high-speed page access

- Exploits programming techniques
  - Modules:
    - All pages: not needed simultaneously
    - Examples
      - Error-handling modules instructions

Demand Paging swapping

- Algorithm implementation: uses Job Table, Page Map Table, and Memory Map Table
- However the “demand” Page Map Table (has 3 **extra** fields): field 1 to 3:
  - **Field 1** says if it is in memory and relates to saving time going from secondary storage to main memory.
  - **Field 2** (modified) is related to outputting frame to secondary storage
  - **Filed 3** referenced: shows recently active versus not active pages. Indicates which pages stay in memory and which can be swapped out.



(figure 3.5)
Demand paging *swapping* 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,
- the page frame number for each page currently in main memory.

Demand Paging Memory Allocation (cont'd.)

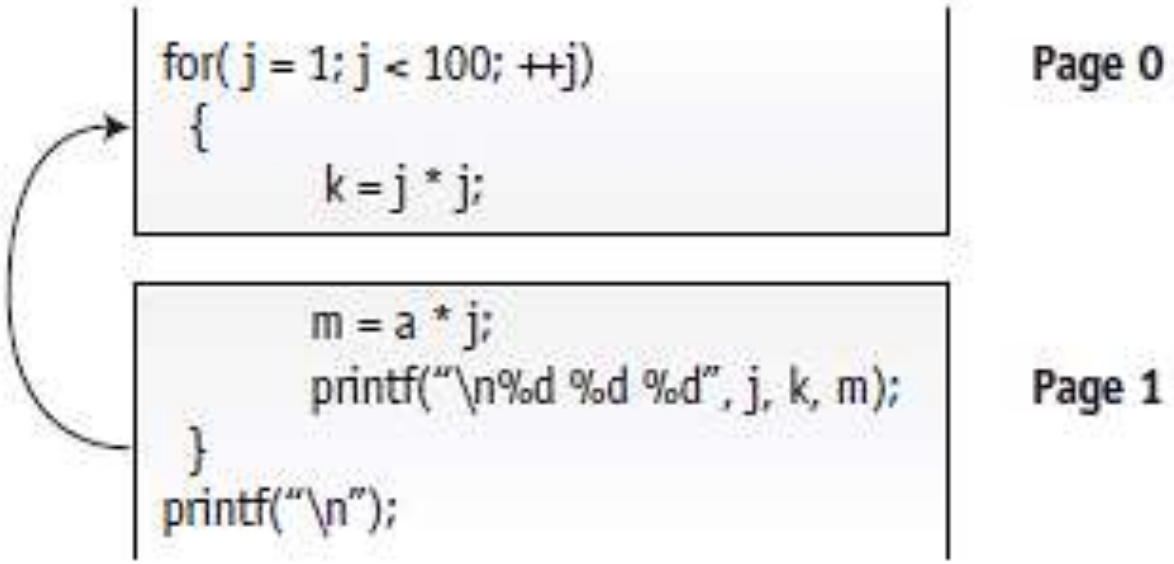
- Swapping process:
  - Find the page number
  - Determine page status: already in memory
  - Resident memory page: exchanged with secondary storage page if modified
    - Resident page: copied to disk (if modified bit set to 1)
  - A New page: written into available page frame

Demand Paging Memory Allocation

- **Page fault**: failure to find page in memory
  - Generate a page interrupt (stops execution of the processing)
- **Page fault handler**: part of operating system
  - Determines if empty page frames in memory
    - Yes: requested page copied from secondary storage
    - No: swapping (dependent on the *predefined* page removal algorithm)
- **Tables updated** when page swap occurs
  - PMT for both jobs (page swapped out; page swapped in)
    - If frame modified write to disk

Demand Paging Memory Allocation

- **Thrashing (**Excessive page swapping)
  - Results in inefficient operation of memory management
  - Main memory pages: *removed frequently; called back soon thereafter*
  - Occurs across jobs
    - Large number of jobs: limited free pages (a lot of swapping would be required to satisfy all jobs)
  - Occurs within a job
    - If Loops crossing page boundaries (figure 3.6)
    - In this example Swapping (generation of page faults) will occur 100 times so useful processing is degraded by a factor of 100

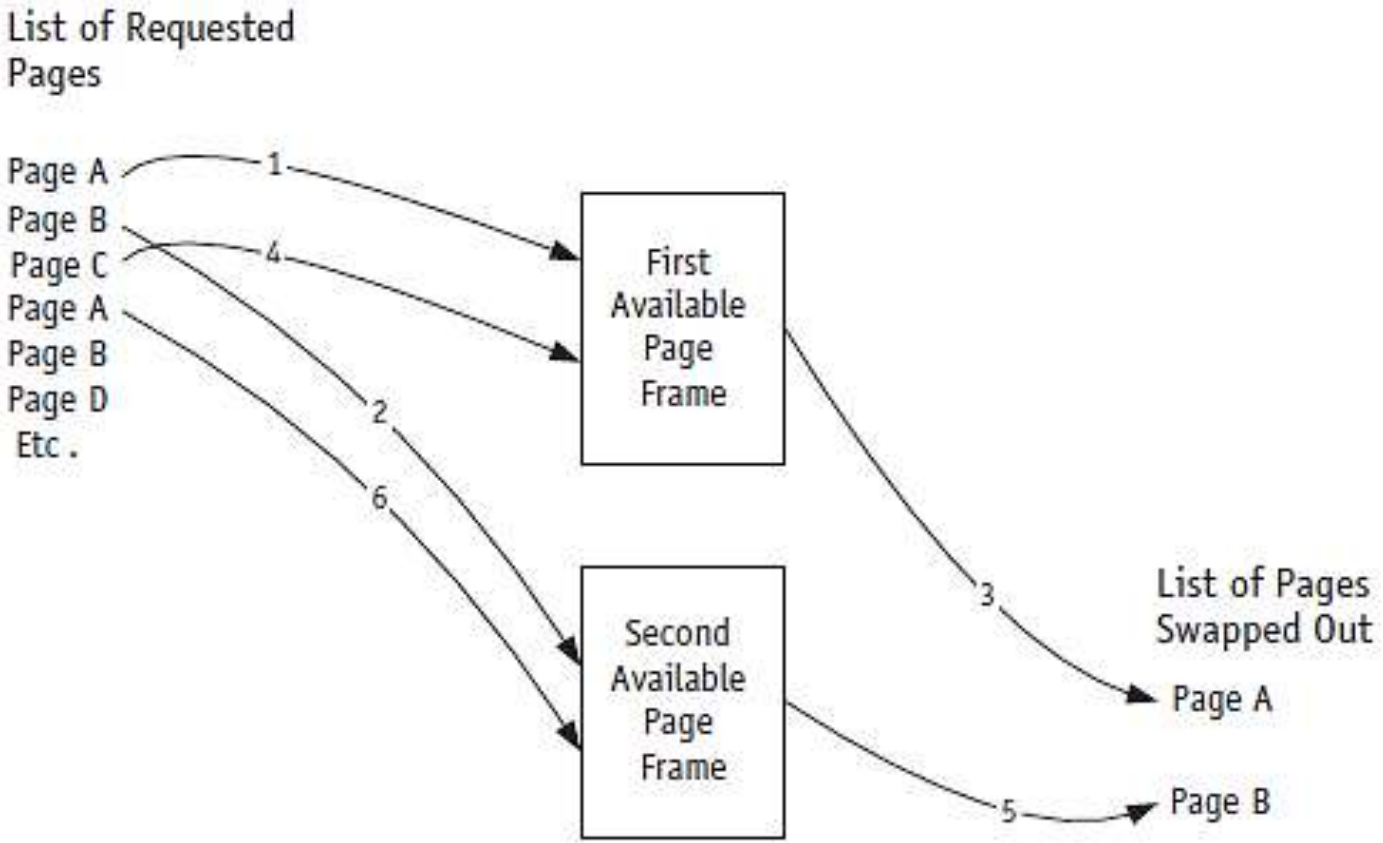


(figure 3.6)
An example of demand paging that causes a page swap each time the loop is executed and results in thrashing. If only a single page frame is available, this program will have one page fault each time the loop is executed.

Page Replacement (swapping) policies and concepts

- Page replacement policy
  - Crucial to system efficiency; number of generate page faults/page interrupts is minimal

- Two well-known algorithms
  - **First-in first-out** (FIFO) policy – *uses queues*
    - Best page to remove: page in memory longest
  - **Least Recently Used** (LRU) policy
    - Best page to remove: page least recently accessed (based on principle of locality: page used will likely be used again quite soon)
    - Least being the *longest time* since it was *last used*.



(figure 3.7)

First, Pages A and B are loaded into the two available page frames. When Page C is needed, the first page frame is emptied so C can be placed there. Then Page B is swapped out so Page A can be loaded there.

The Mechanics of Paging (cont'd.)

- Page Map Table: bit meaning
  - *Status bit*: page currently in memory
  - *Referenced bit*: page referenced recently
    - Determines page to swap: LRU algorithm
  - *Modified bit*: page contents altered
    - Determines if page must be rewritten to secondary storage when swapped out
- Bits checked when swapping
  - FIFO: *modified* and *status* bits
  - LRU: *all* bits (status, modified, and reference bits)

The Mechanics of Paging

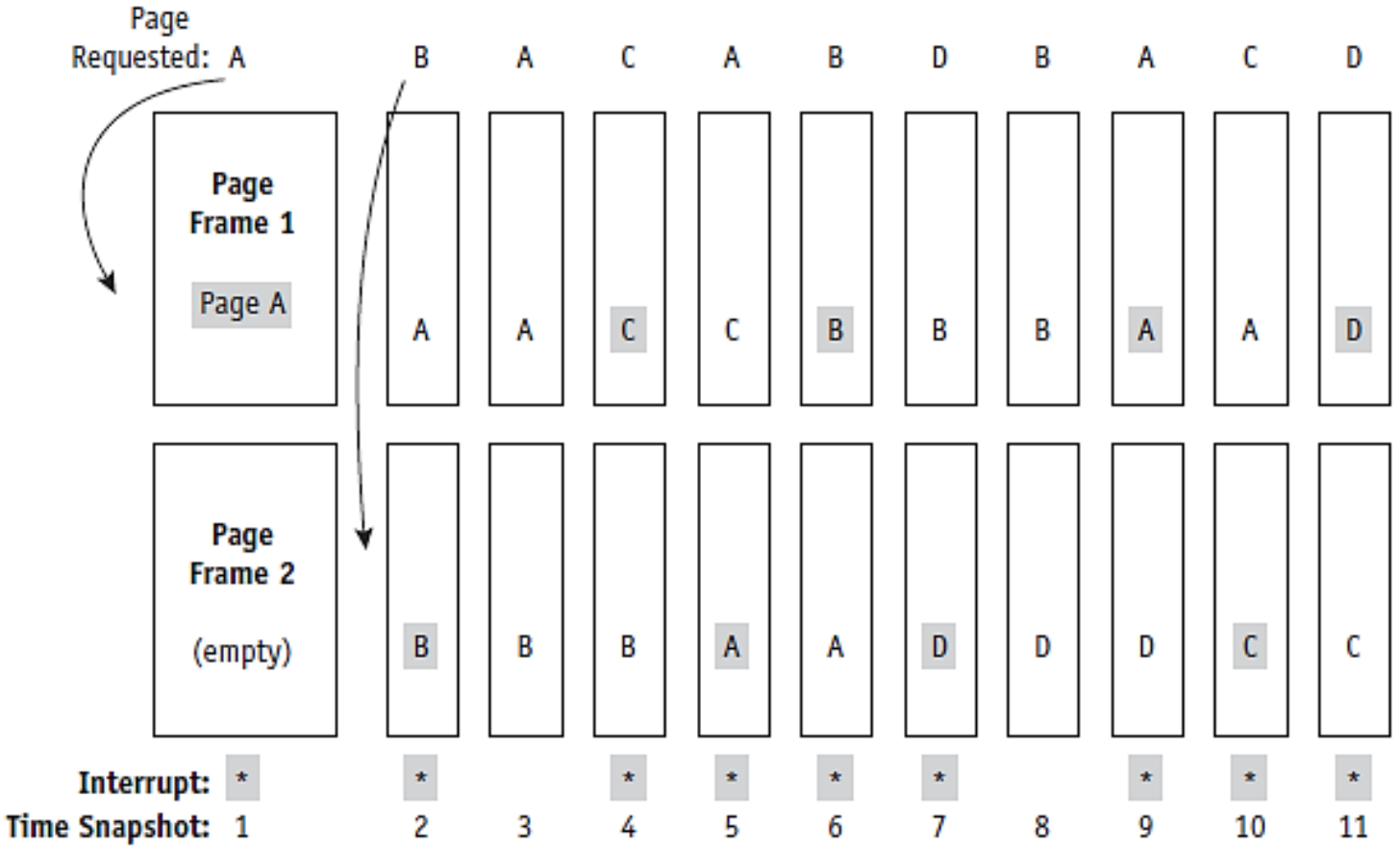
- Page swapping
  - Memory manage requires specific information: Page Map Table

Page No.	Status Bit	Modified Bit	Referenced Bit	Page Frame No.
0	1	1	1	5
1	1	0	0	9
2	1	0	0	7
3	1	0	1	12

(table 3.3)

Page Map Table for Job 1 shown in *Figure 3.5*.

A 1 = Yes and 0 = No.

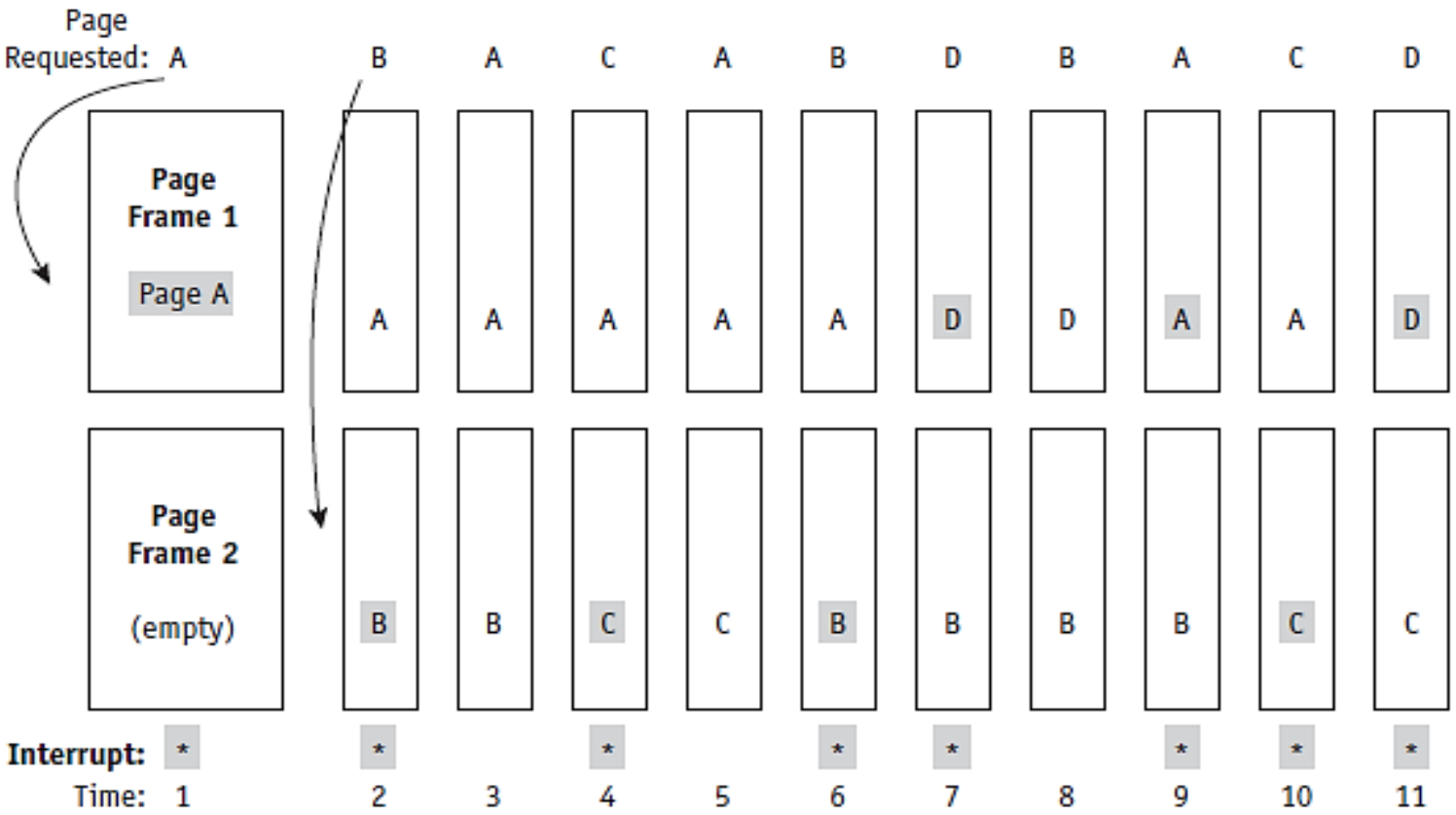


(figure 3.8)

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.

Consider swapping in C of the 2 pages A/B A was first in so it is removed.

What is the failure rate

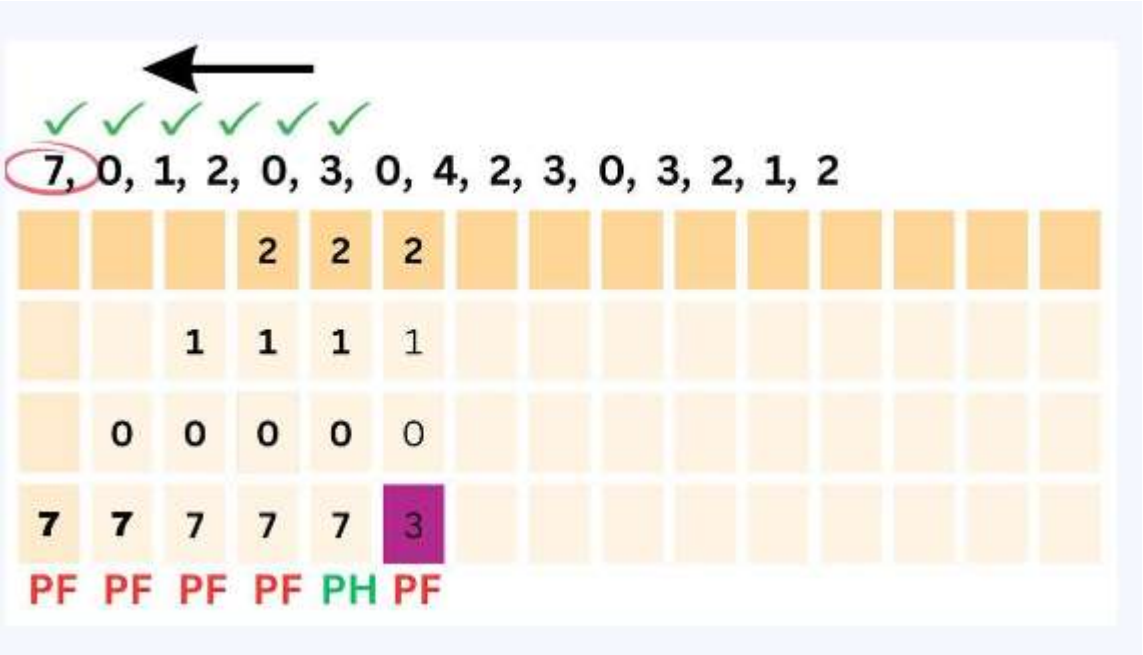


(figure 3.9)

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.

Take for example C being swapped in: of the two pages A/B B has the longest time since it was last used (least recently used) so B is chosen for swap out

LRU swap example



- If the **page is in memory**, **remove** it from its **current position** in the list and append it to the **end of the list** (since it was the most recently used page).
- If the **page is not in memory**, **add it to the end of the list if there is space** (i.e., the list is not yet full), or
  else remove the least recently used page (i.e., the first page in the list) and **replace it with the new page at the end of the list**.

How to Determine which Page Frame is Least Recently Used

- This algorithm is based on the principle of locality:
  - Pages recently referenced are likely to be referenced again in the near future.
  - Two types of *locality of reference*:
    - *Spatial* locality (code on the same page)
    - *Temporal* locality (code that are used in loops: these can be on the same or neighbouring pages)
  - This limits the amount of swapping required.

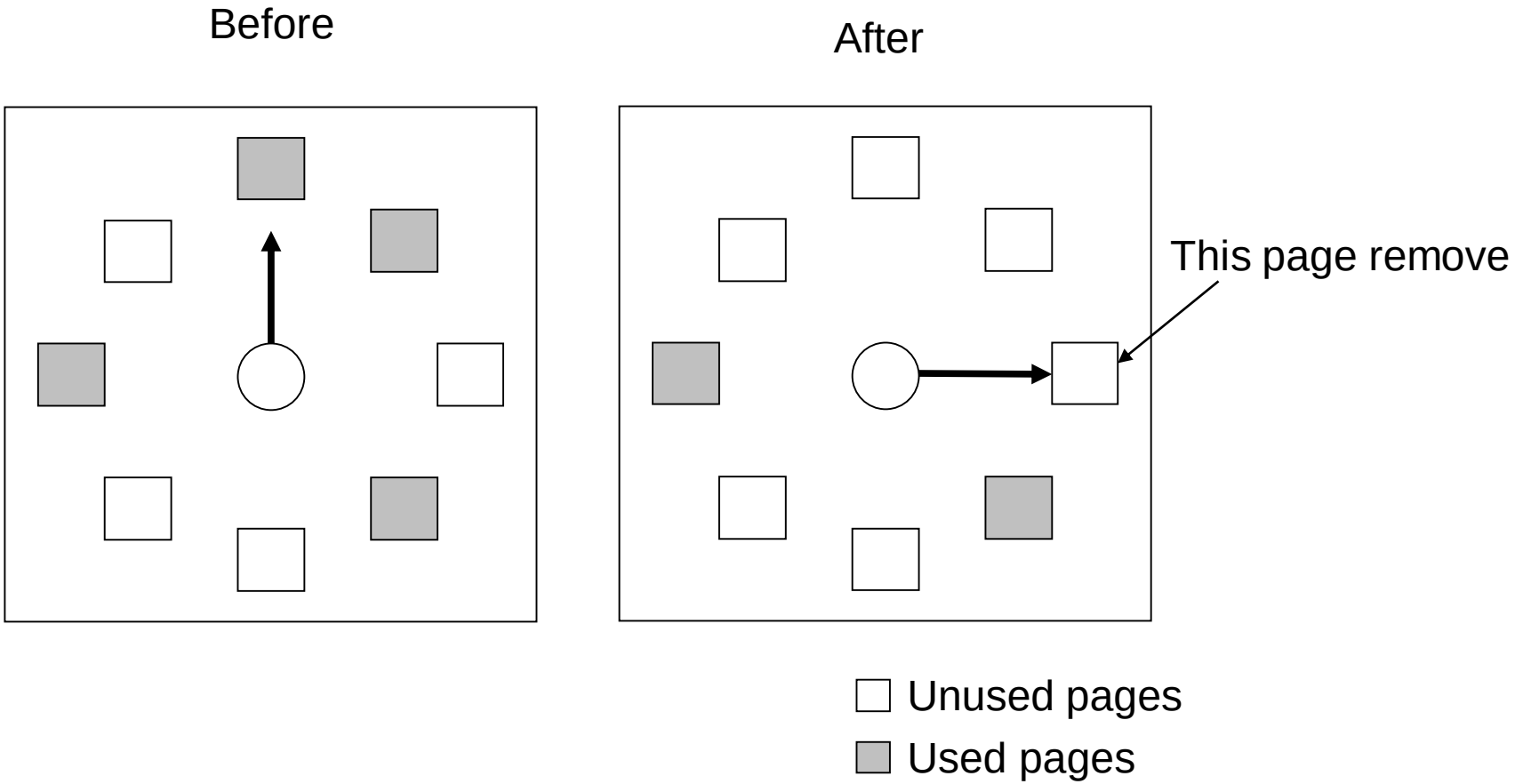
The LRU Clock policy technique

- Clock policy is a method used to establish easily and quickly if a page is recently referenced (used)

- Only requires one bit to record when a page has been referenced
  - memory management unit needs to set this whenever a page is referenced
- Scan referenced bits in a circular manner looking for a page which has not been referenced
  - i.e (referenced bit of the PMT = 0)

- If a bit is =1 reset reference bit to 0 as each entry is examined if bit is 0 it is targeted for removal.

Clock policy-scanning for a page to remove



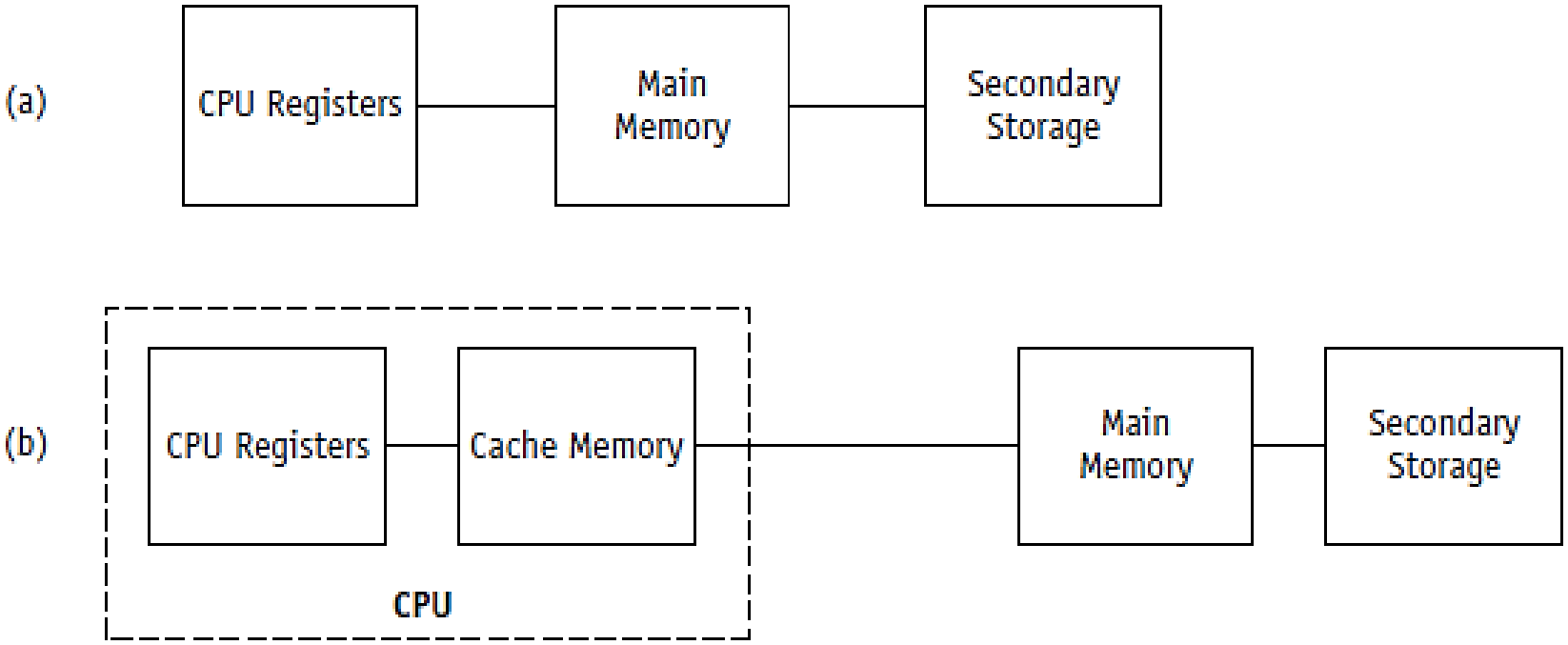
Cache Memory

- Small, high-speed intermediate memory unit
- Computer system's performance increased
  - Faster processor access compared to main memory
  - Stores frequently used data and instructions
- Data/instructions: move between main memory and cache.
  - Methods similar to paging algorithms

Cache Memory

- Small, high-speed intermediate memory unit
- Computer system's performance increased
  - Faster processor access compared to main memory
  - Stores frequently used data and instructions
- Data/instructions: move between main memory and cache.
  - Methods similar to paging algorithms

CPU registers: speed up lookup process by storing PMT/SMT of recently entries for active jobs



(figure 3.19)
Comparison of (a) the traditional path used by early computers between main memory and the CPU and (b) the path used by modern computers to connect the main memory and the CPU via cache memory.

Cache Memory (cont'd.)

- Referenced instructions/data are loaded from main memory into cache memory.
- First cache memory is searched for instructions/data and if not found the search continues in main memory.
- Then one of the page in cache is chosen to be swapped out…
- Optimal cache and replacement algorithm
  - Results in 80-90% of all requests in cache possible

Potential Exam Questions

- Describe, using a suitable example, the relationship between logical and physical memory. **(4 marks)**
- What is the purpose of the: modified field, status field and the referenced field in the PMT of the demand page memory management system **(6 marks)**

Sample exam questions

- Distinguish, *using suitable examples* the difference between the FIFO and the LRU page swapping algorithms (you could also be given example similar to the ones covered in class)

  (12 marks).

- What is the purpose of cache memory

  (4 marks)

Sample exam Question

- Explain how the least recently used swapping algorithm is implemented using for example the clock policy referencing technique

  (8 marks)

- **or**
- Explain how a cache is used in virtual memory management and improves efficiency.

  (8 marks)