MONASH University

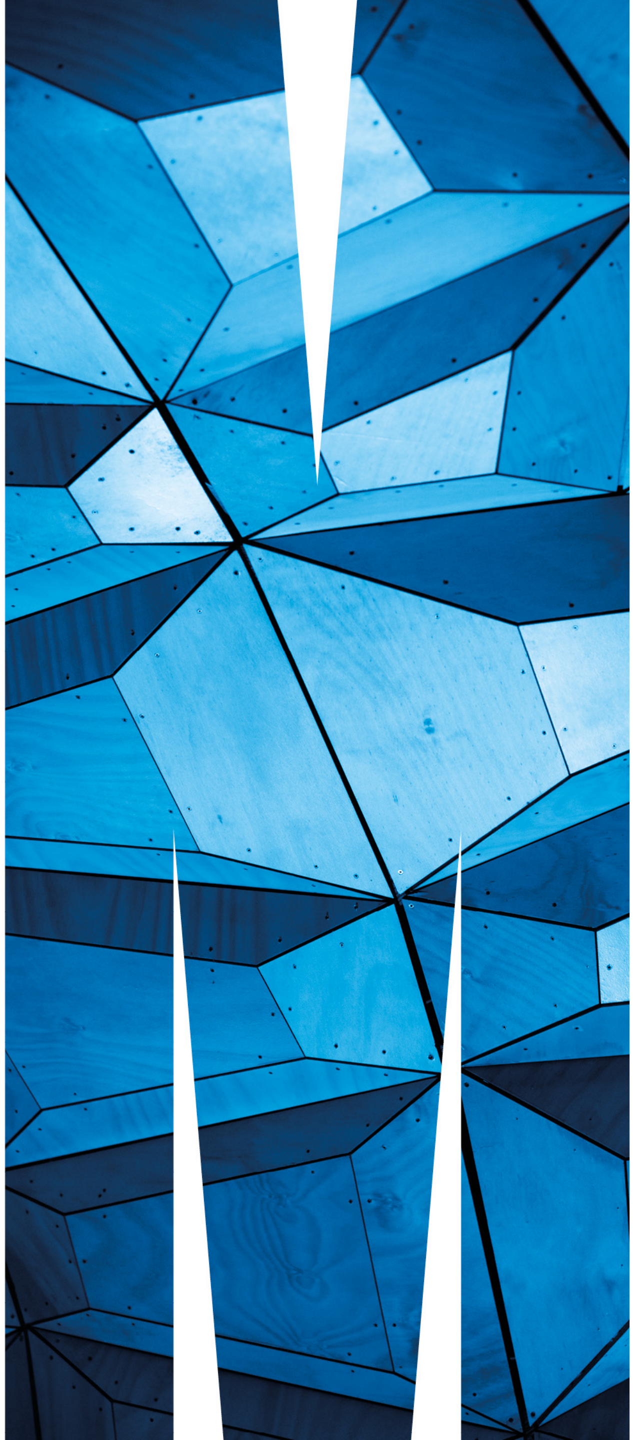
MONASH
INFORMATION
TECHNOLOGY

FIT2100 Semester 2 2017

Lecture 9: Virtual Memory (Reading: Stallings, Chapter 8)

Jojo Wong

GROUP OF EIGHT AUSTRALIA

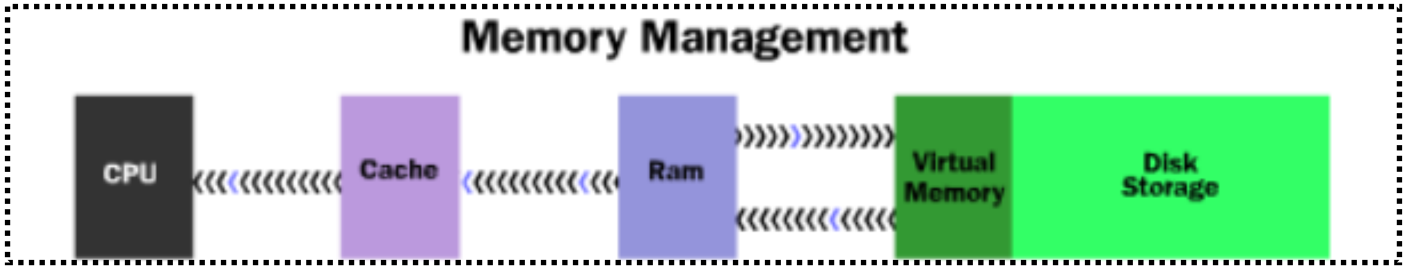


Lecture 9: Learning Outcomes

- Upon the completion of this lecture, you should be able to:
  - Define the concepts of virtual memory
  - Describe the hardware and control structures that support virtual memory
  - Describe various OS mechanisms used to implement virtual memory

Virtual Management: Terminology

Virtual memory	A storage allocation scheme in which secondary memory can be addressed as though it were part of main memory. The addresses a program may use to reference memory are distinguished from the addresses the memory system uses to identify physical storage sites, and program-generated addresses are translated automatically to the corresponding machine addresses. The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of secondary memory available and not by the actual number of main storage locations.
Virtual address	The address assigned to a location in virtual memory to allow that location to be accessed as though it were part of main memory.
Virtual address space	The virtual storage assigned to a process.
Address space	The range of memory addresses available to a process.
Real address	The address of a storage location in main memory.



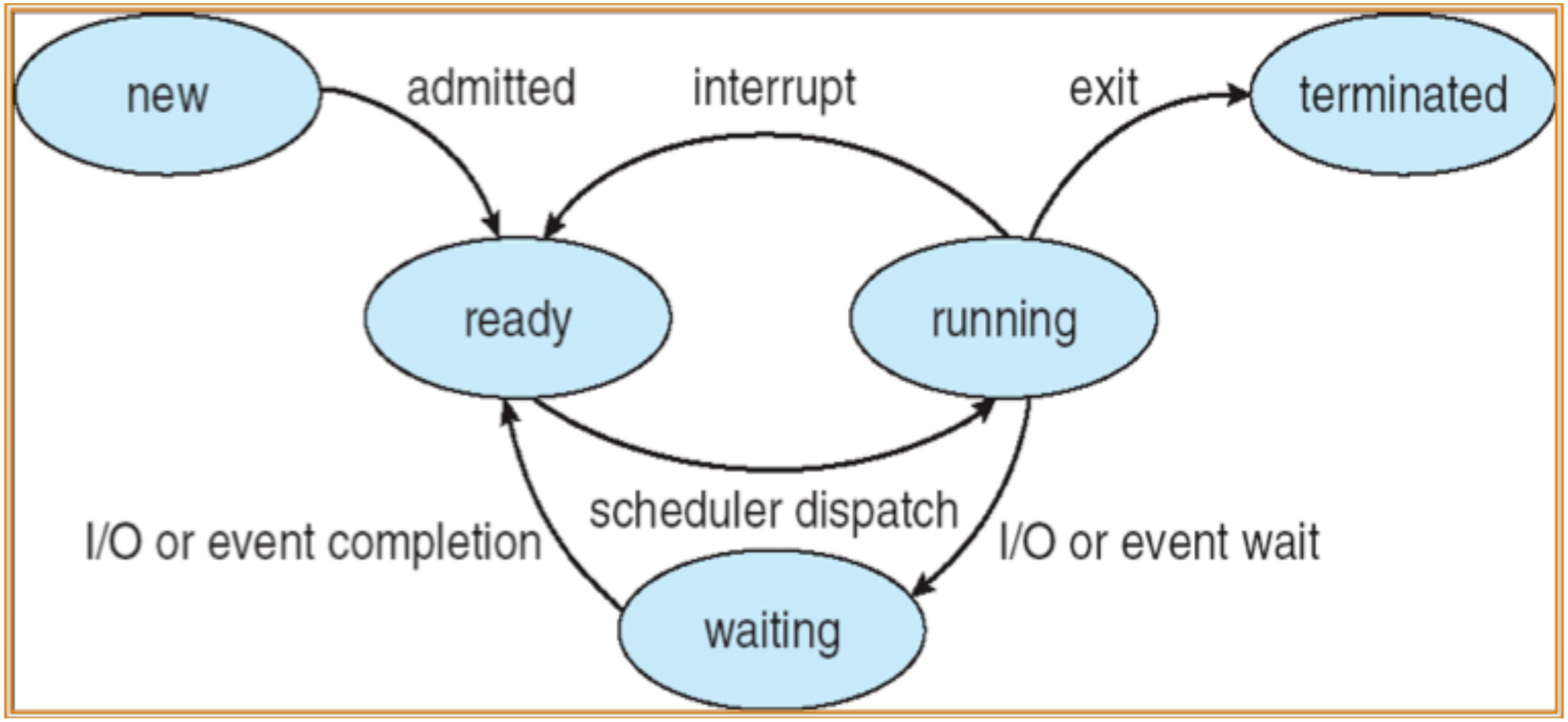
What are the hardware and control structures required?

Hardware and Control Structures

- Two characteristics fundamental to memory management:
  1) All memory references are logical addresses that are dynamically translated into physical addresses at run time
  2) A process may be broken up into a number of pieces that don't need to be contiguously located in main memory during execution

If these two characteristics are present, it is not necessary that all of the pages or segments of a process be in main memory during execution

Recap: States of a Process



Recap: Execution of a Process

- OS brings into main memory a few pieces (pages or segments) of the program
- **Resident set:** portion of process that is in main memory
- An interrupt is generated when a logical address is needed that is not in main memory

Memory access fault

- OS places the interrupted process in a *blocking* state

Recap: Execution of a Process (continue)

- Piece of the process that contains the logical address is brought into main memory
- OS issues a disk I/O read request
- Another process is dispatched to run while the disk I/ O read takes place
- An interrupt is issued when disk I/O is complete, which causes OS to place the affected process in the Ready state

MONASH University

What are the implications?

Implications

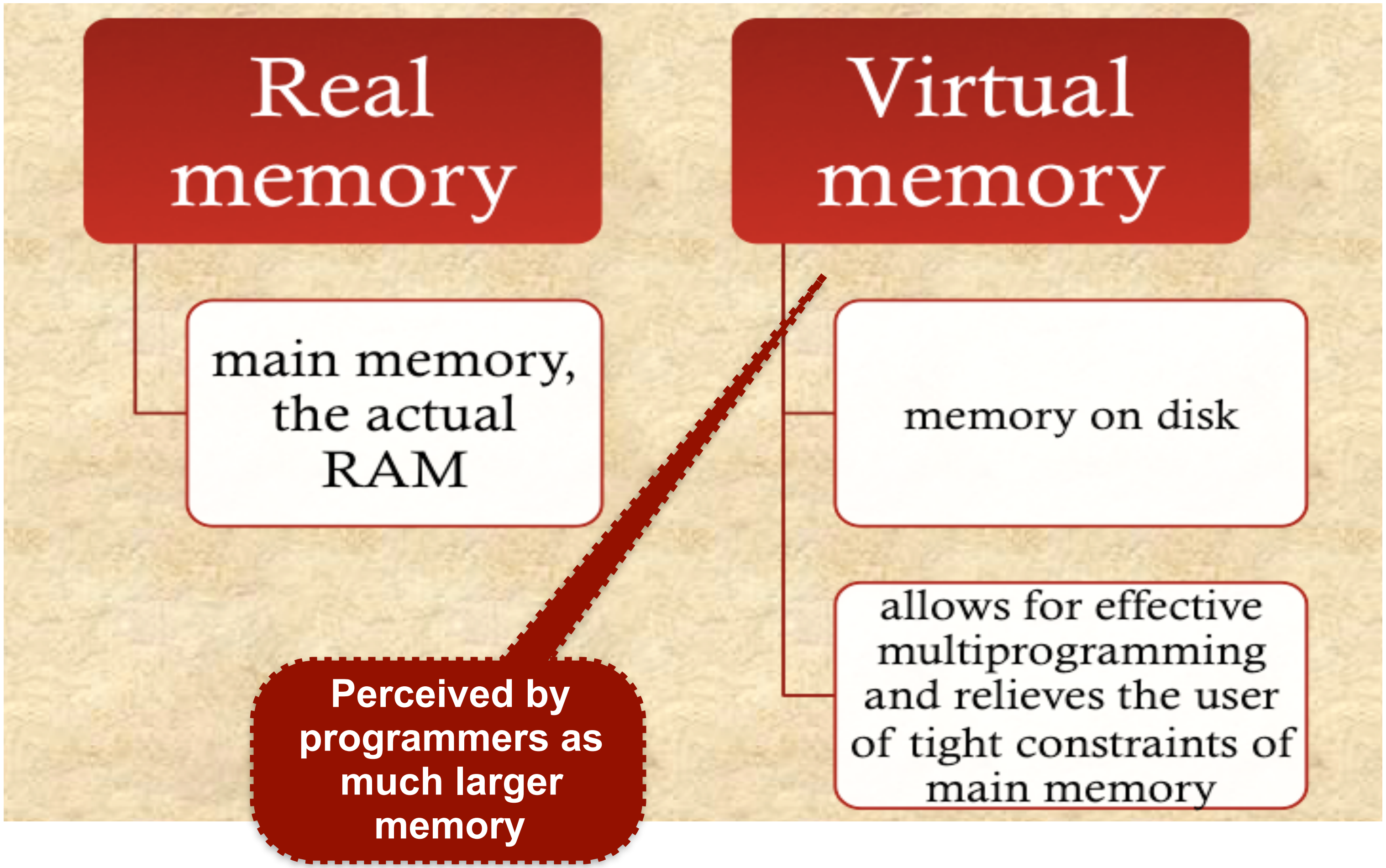
- More processes may be maintained in main memory
  - Only load in some of the pieces of each process
  - With so many processes in main memory — it is very likely a process will be in the Ready state at any particular time

- A process may be larger than all of main memory
  - OS automatically loads pieces of a large process into main memory as required

Virtual memory (with paging and segmentation)

What is virtual memory?

Real Memory vs Virtual Memory


Real memory
main memory, the actual RAM
Virtual memory
memory on disk
allows for effective multiprogramming and relieves the user of tight constraints of main memory
Perceived by programmers as much larger memory

Virtual Memory: Paging and Segmentation

Simple Paging	Virtual Memory Paging	Simple Segmentation	Virtual Memory Segmentation
Main memory partitioned into small fixed-size chunks called frames		Main memory not partitioned	
Program broken into pages by the compiler or memory management system		Program segments specified by the programmer to the compiler (i.e., the decision is made by the programmer)	
Internal fragmentation within frames		No internal fragmentation	
No external fragmentation		External fragmentation	
Operating system must maintain a page table for each process showing which frame each page occupies		Operating system must maintain a segment table for each process showing the load address and length of each segment	
Operating system must maintain a free frame list		Operating system must maintain a list of free holes in main memory	
Processor uses page number, offset to calculate absolute address		Processor uses segment number, offset to calculate absolute address	
All the pages of a process must be in main memory for process to run, unless overlays are used	Not all pages of a process need be in main memory frames for the process to run. Pages may be read in as needed	All the segments of a process must be in main memory for process to run, unless overlays are used	Not all segments of a process need be in main memory for the process to run. Segments may be read in as needed
	Reading a page into main memory may require writing a page out to disk		Reading a segment into main memory may require writing one or more segments out to disk

What is Thrashing?

A state in which the system spends most of its time swapping process pieces rather than executing instructions

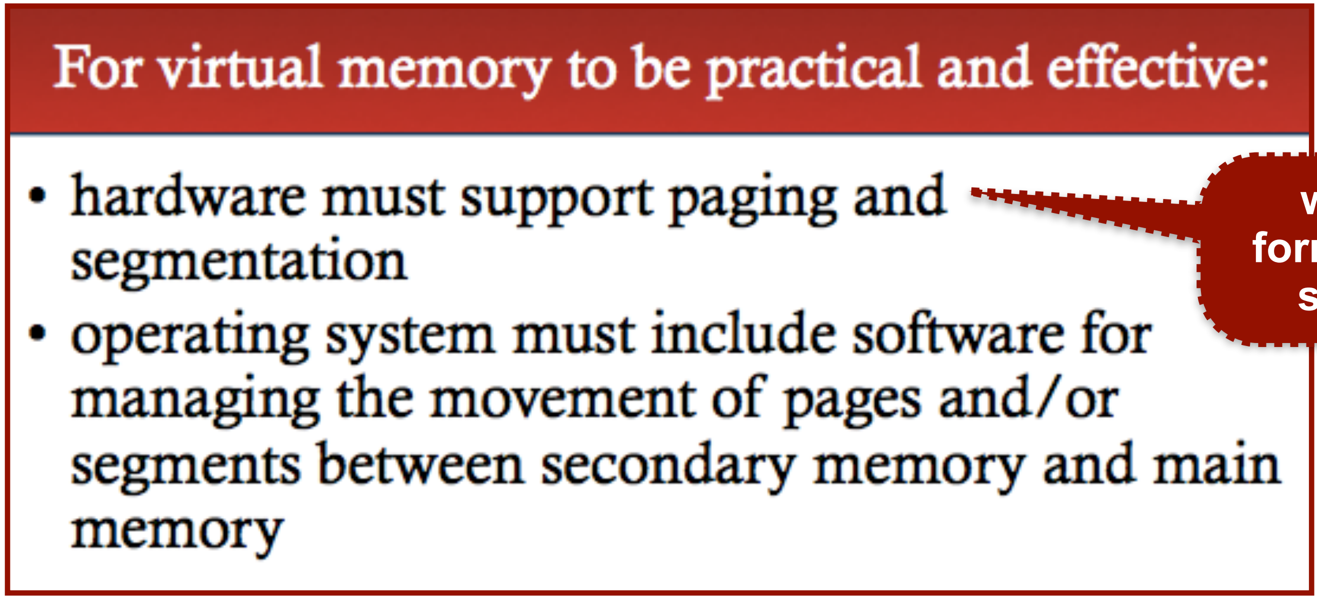
To avoid this, OS tries to guess, based on recent history, which pieces are least likely to be used in the near future

Principle of Locality

- Program and data references within a process tend to *cluster*
- Only a few pieces of a process will be needed over a short period of time
- **Temporal locality:** the instructions in a loop all reside in the cache (they are being used over and over within a short time).
- **Spatial locality:** accessing the elements of an array and also the instructions in a loop. Both are examples of words that are next to each other, and will appear in the cache next to each other.

It is possible to make intelligent guesses about which pieces will be needed in the future to avoid thrashing

Virtual Memory: What Support Needed?



For virtual memory to be practical and effective:

- hardware must support paging and segmentation
- operating system must include software for managing the movement of pages and/or segments between secondary memory and main memory

with some form of control structures

MONASH University

MONASH University

How *paging* is used to achieve virtual memory?

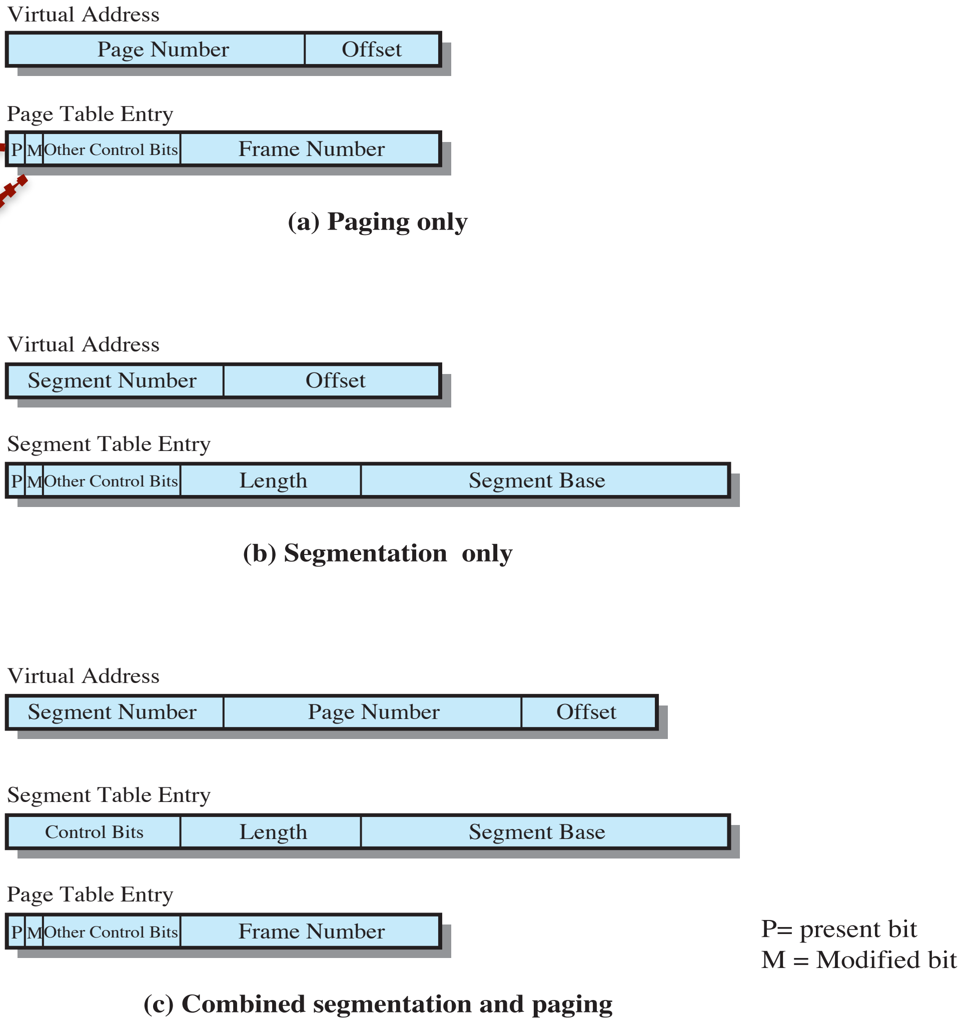
Virtual Memory: Paging

- The term virtual memory is usually associated with systems that employ paging
- Use of paging to achieve virtual memory was first reported for the Atlas computer (circa: 1962 — University of Manchester, UK)
- Each process has its own page table
  - Each page table entry contains the frame number of the corresponding page in main memory

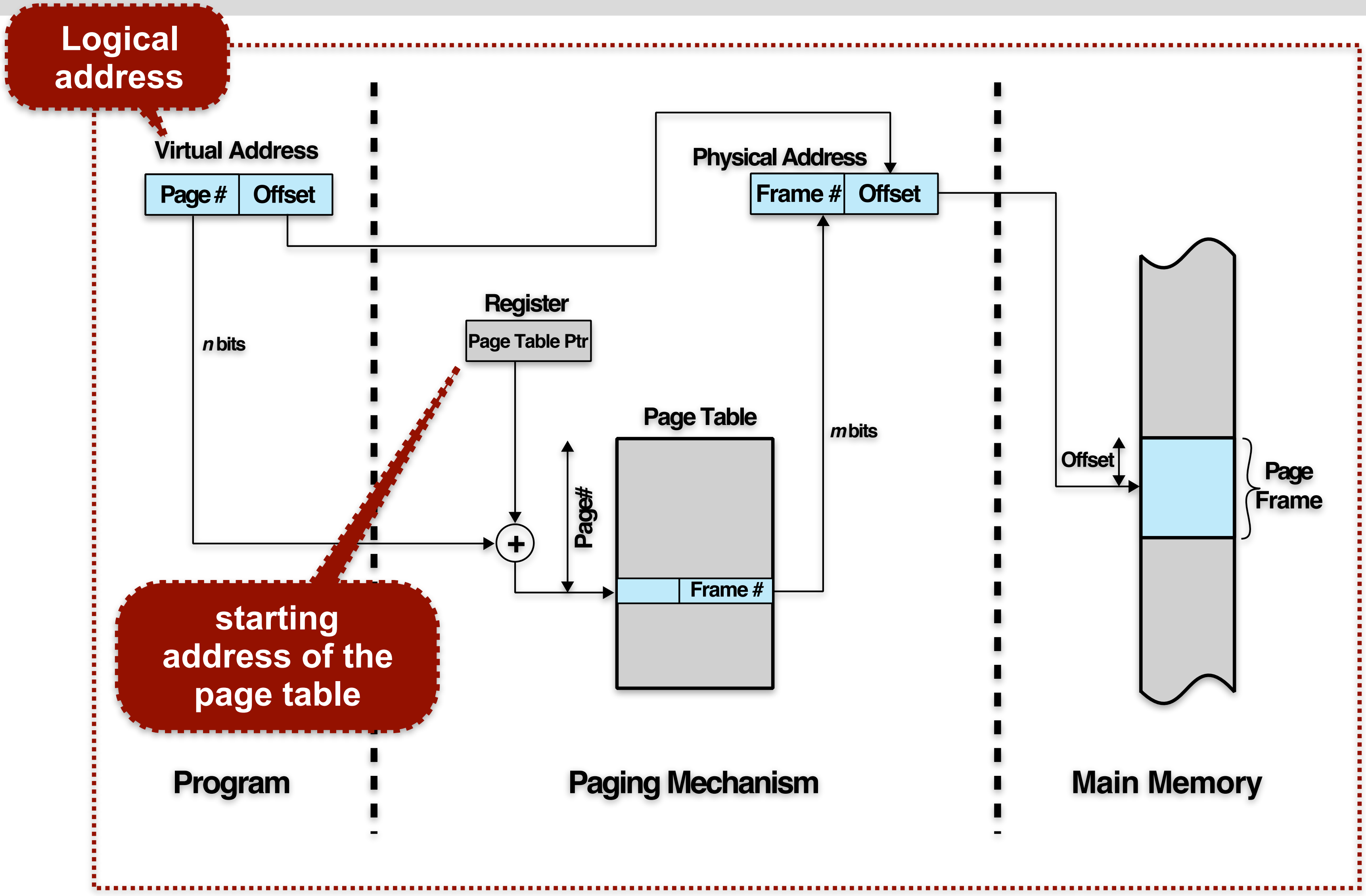
Memory Management: Formats

P (page) is present in main memory

M (modified bit) — if page content is modified need to write back



Paging System: Address Translation



Paging System: Page Size

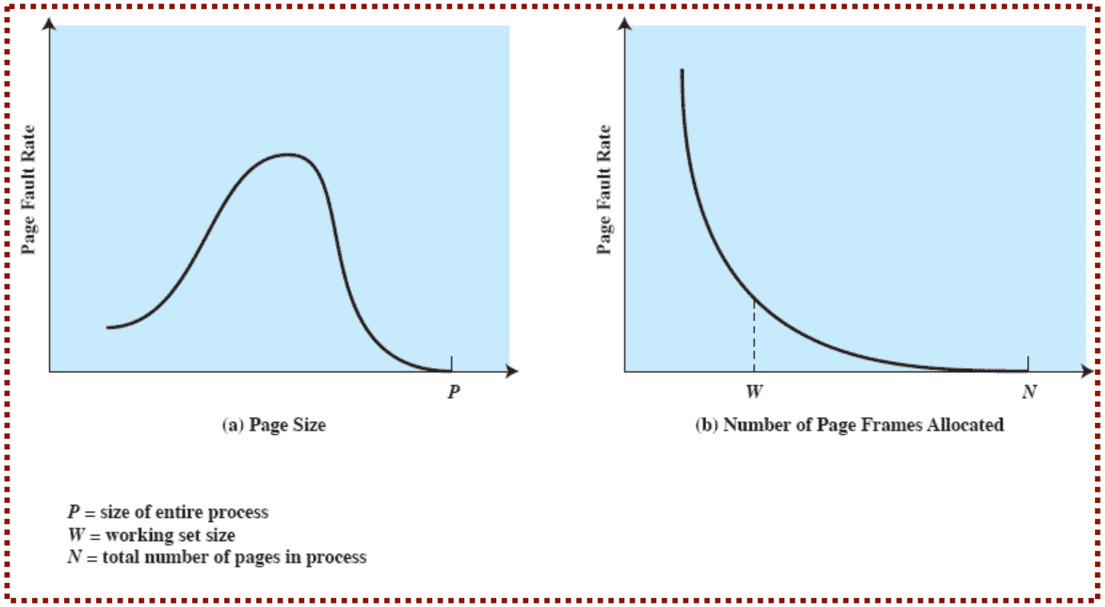
- The smaller the page size, the lesser the amount of internal fragmentation
  - However, more pages are required per process
  - More pages per process means larger page tables
- For large programs in a heavily multi-programmed environment some portion of the page tables of active processes must be in *virtual memory* instead of main memory

double page fault

Physical characteristics of most secondary-memory devices favour a larger page size for more efficient block transfer of data

MONASH University

Paging Behaviour



- This is based on the principle of locality. If the page size is very small, then ordinarily a relatively large number of pages will be available in main memory for a process. After a time, the pages in memory will all contain portions of the process near recent references. Thus, **the page fault rate should be low**.
- As the size of the page is increased, each individual page will contain locations further and further from any particular recent reference. Thus the effect of the principle of locality is weakened and **the page fault rate begins to rise**. Eventually, however, the page fault rate will begin to fall as the size of a page approaches the size of the entire process.

How *segmentation* is used to achieve virtual memory?

Virtual Memory: Segmentation

Segmentation

- Allows programmers to view memory as consisting of multiple address spaces or segments
- Segments are of unequal and dynamic size

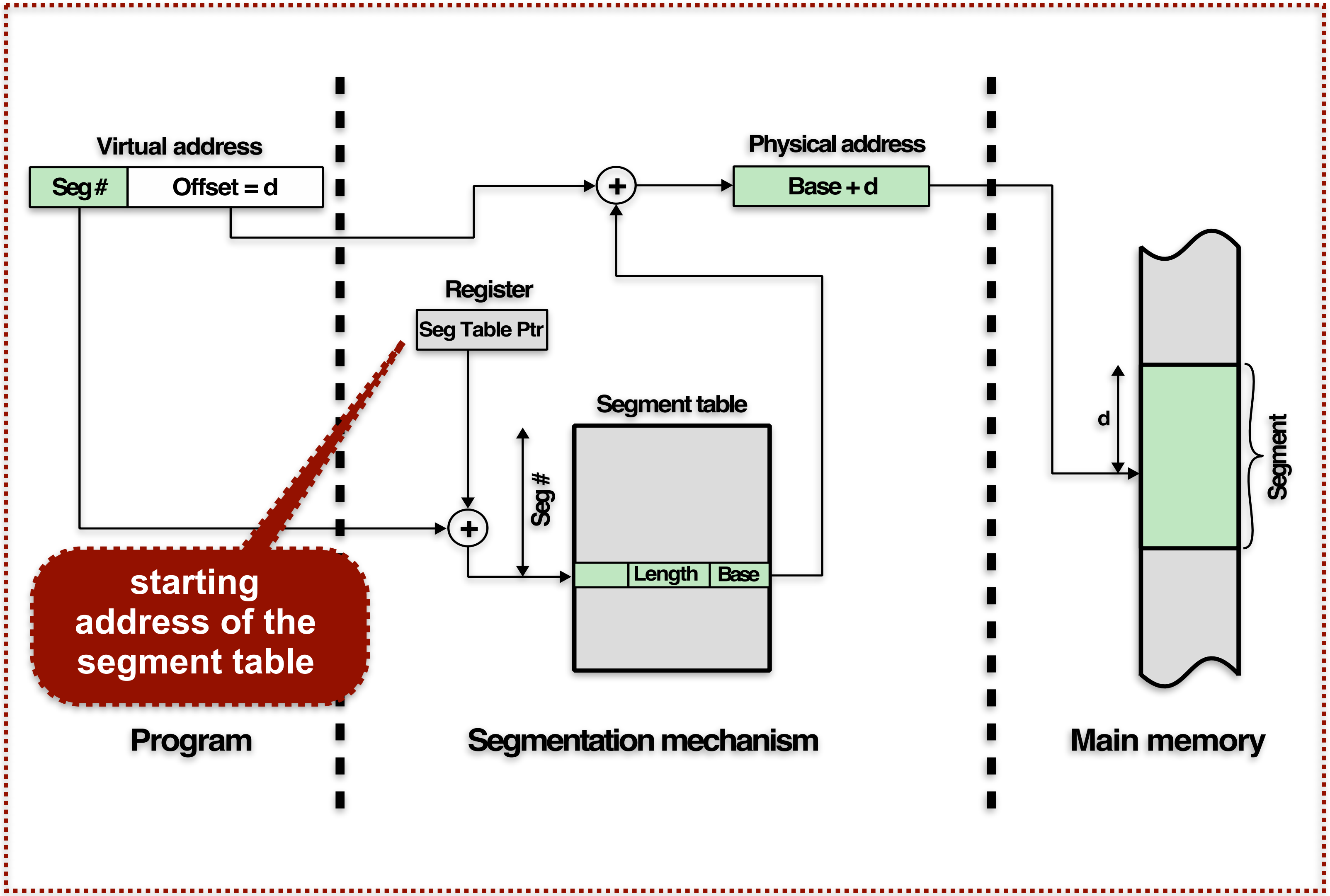
Advantages

- Simplifies handling of growing data structures
- Allows programs to be altered and recompiled independently
- Lends itself to protection and sharing data among processes

Segmentation System: Organisation

- Each segment table entry contains the starting address of the corresponding segment in main memory and the length of the segment
- A bit is needed to determine if the segment is already in main memory
- Another bit is needed to determine if the segment has been modified since it was loaded in main memory

Segmentation System: Address Translation



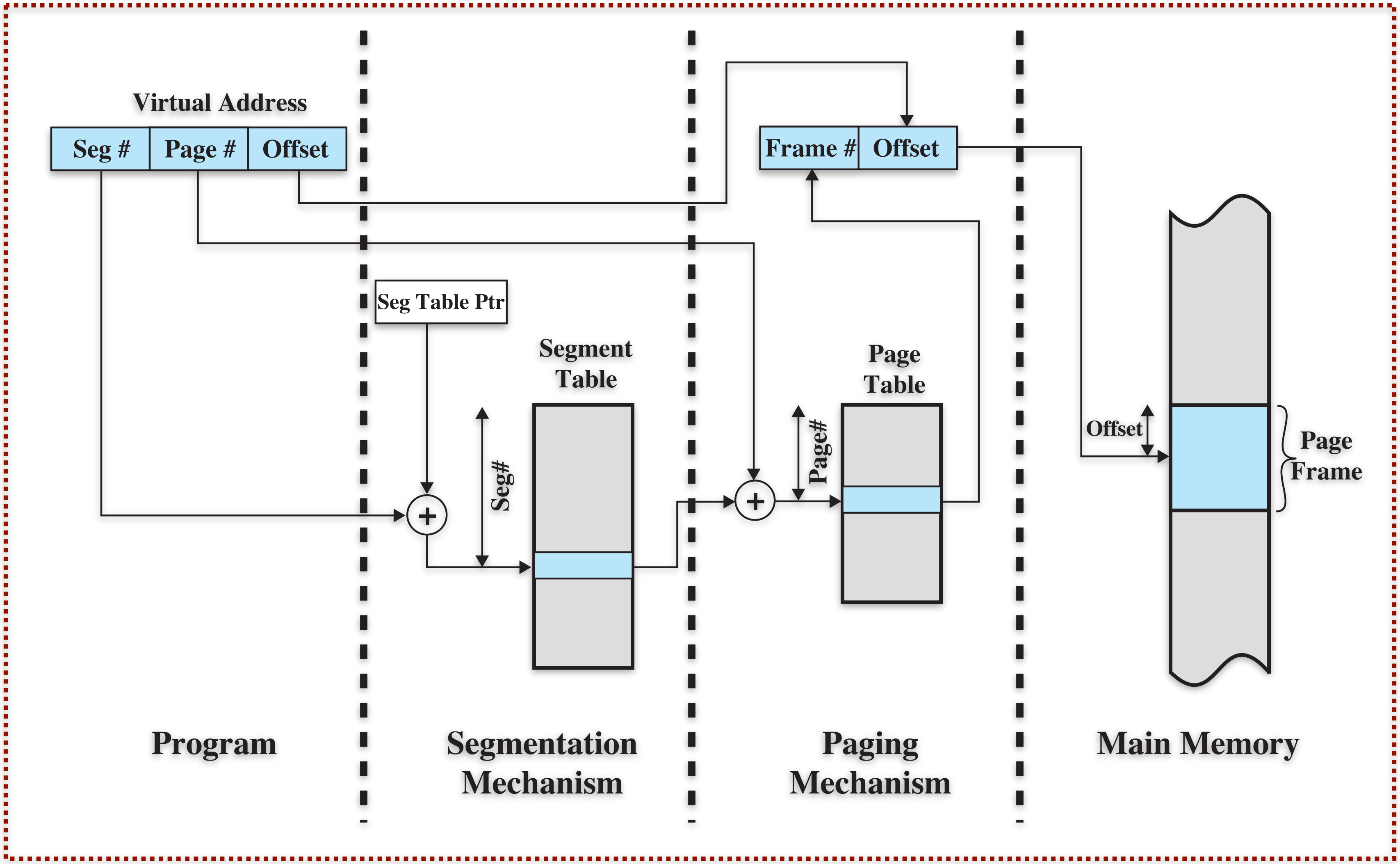
Combined System: Paging and Segmentation

A user's address space is broken up into a number of segments. Each segment is broken up into a number of fixed-sized pages which are equal in length to a main memory frame.

Segmentation is visible to programmers

Paging is transparent to programmers

Segmentation/Paging System: Address Translation



How does OS support the implementation of virtual memory?

Operating System: Memory Management

The design of the memory management portion of an OS depends on three fundamental areas of choice:

- **Whether or not to use virtual memory techniques**
- **The use of paging or segmentation or both**
- **The algorithms employed for various aspects of memory management**

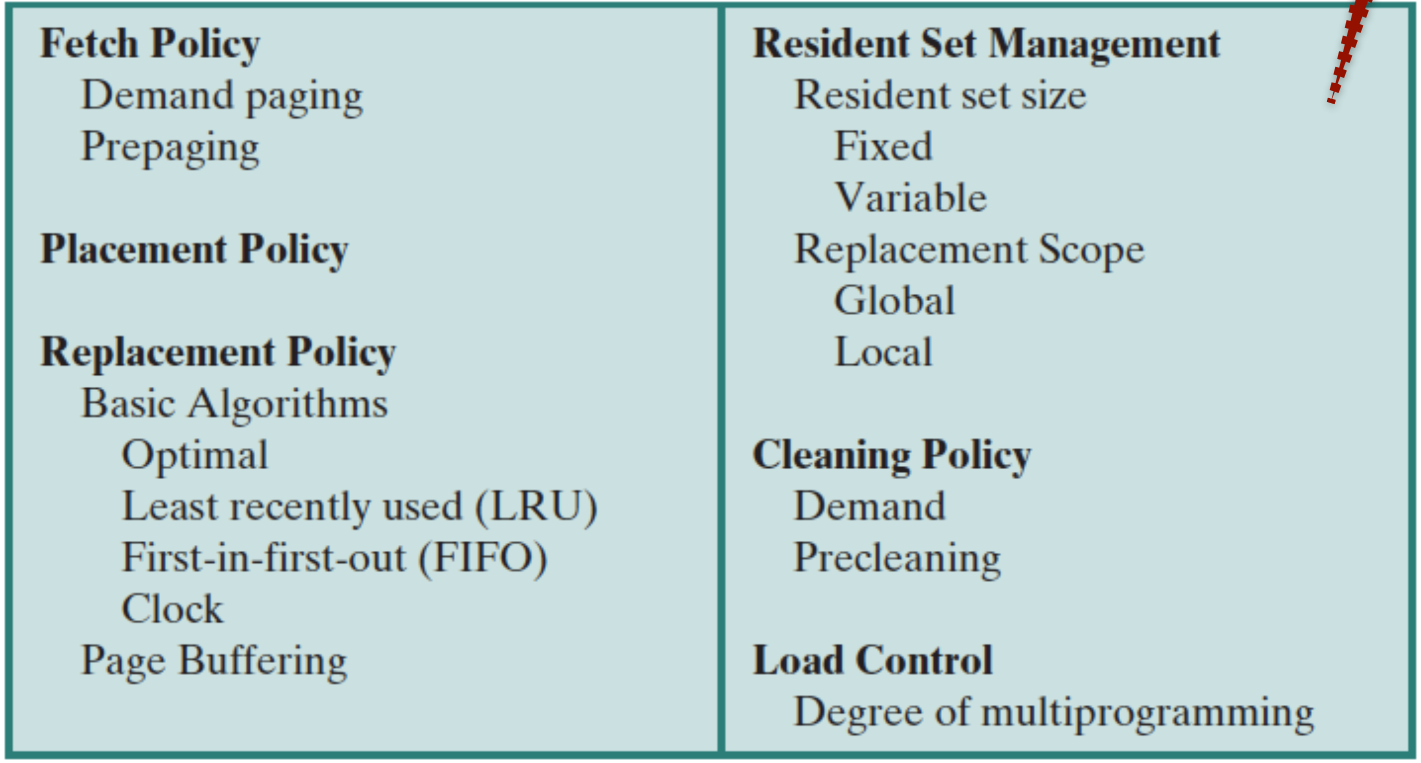
Supported by OS

MONASH University

Virtual Memory: OS Policies

No definitive policy that works best

- Key performance issue: minimise page fault rate



Fetch Policy
- Demand paging
- Prepaging

Placement Policy

Replacement Policy
- Basic Algorithms
  - Optimal
  - Least recently used (LRU)
  - First-in-first-out (FIFO)
  - Clock
- Page Buffering

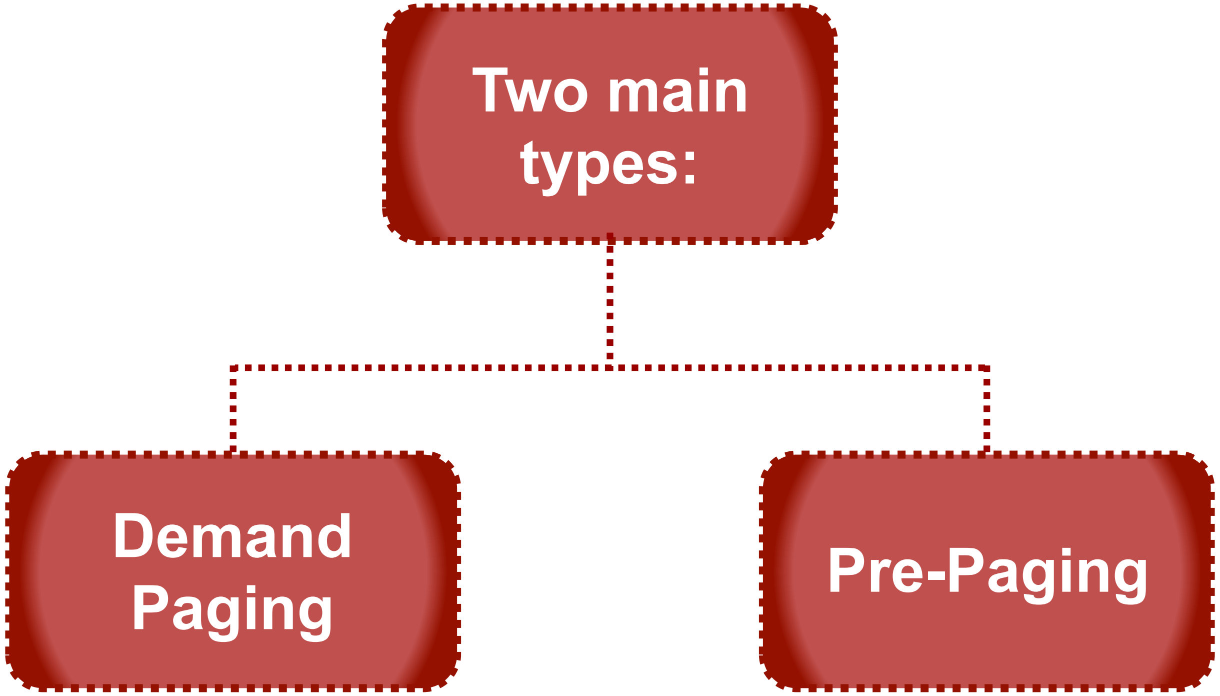
Resident Set Management
- Resident set size
  - Fixed
  - Variable
- Replacement Scope
  - Global
  - Local

Cleaning Policy
- Demand
- Precleaning

Load Control
- Degree of multiprogramming

Virtual Memory: Fetch Policy

- Determine when a page should be brought into main memory



Fetch Policy: Demand Paging

- Only bring pages into main memory when a reference is made to a location on the page
- Many page faults when a process is first started
- Principle of locality suggests that as more and more pages are brought in — most future references will be to pages that have recently been brought in
  - Page faults should drop to a very low level

Fetch Policy: Pre-Paging

- Pages other than the one demanded by a page fault are brought in
- Exploits the characteristics of most secondary memory devices
  - If pages of a process are stored contiguously in secondary memory it is more efficient to bring in a number of pages at one time
- Ineffective if extra pages brought in are not referenced
- Should not be confused with “swapping”

For suspended processes

MONASH University

Virtual Memory: Placement Policy

- Determine where in real memory a process piece is to reside
- Important design issue in a segmentation system

best-fit, first-fit, and next-fit

- Paging or combined paging with segmentation placing is *irrelevant* — because hardware can perform functions with equal efficiency

Hardware for address translation and main memory access

Virtual Memory: Replacement Policy

- Deals with the selection of a page in main memory to be replaced when a new page must be brought in
- Objective: the page that is removed should be the page least likely to be referenced in the near future
- The more elaborate the replacement policy — the greater the hardware and software overhead to implement it

Replacement Policy: Frame Locking

- When a frame is locked — the page currently stored in that frame may not be replaced
- Kernel of the OS as well as key control structures are held in locked frames
- I/O buffers and time-critical areas may be locked into main memory frames
    - Can be achieved by associating a lock bit with each frame

What are the different page replacement algorithms?

Replacement Algorithms

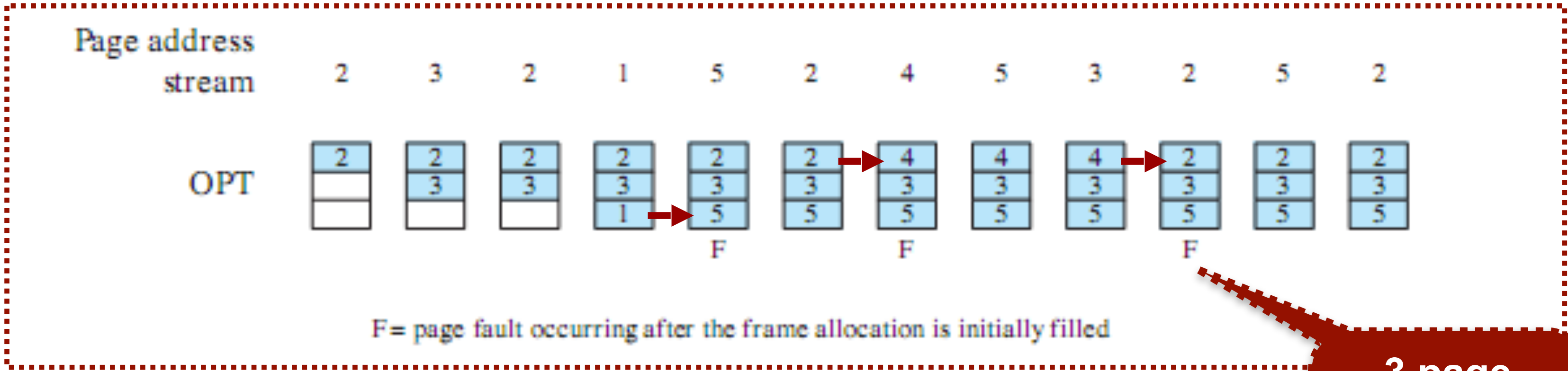
- Various algorithms used for the selection of a page to replace:

  - Optimal policy
  - Least recently used (LRU) policy
  - First-in-first-out (FIFO) policy
  - Clock policy

Replacement Algorithm: Optimal Policy

future knowledge required

- Select the replacement page for which the time to the next reference is the longest
- Produces three page faults after the frame allocation has been filled



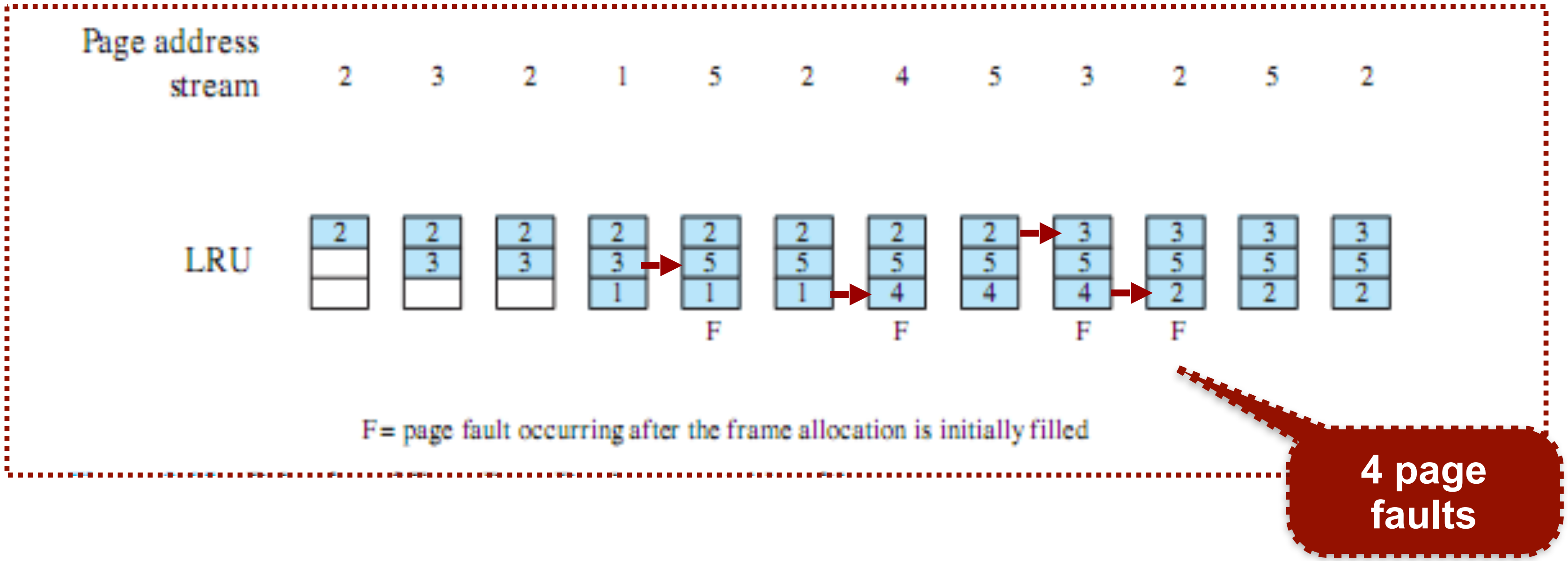
F= page fault occurring after the frame allocation is initially filled

3 page faults

Replacement Algorithm: Least Recently Used (LRU)

- Replace the page that has not been referenced for the longest time
- By the *principle of locality*, this should be the page least likely to be referenced in the near future
- Difficult to implement:
  - Tag each page with the time of last reference — requires a great deal of overhead

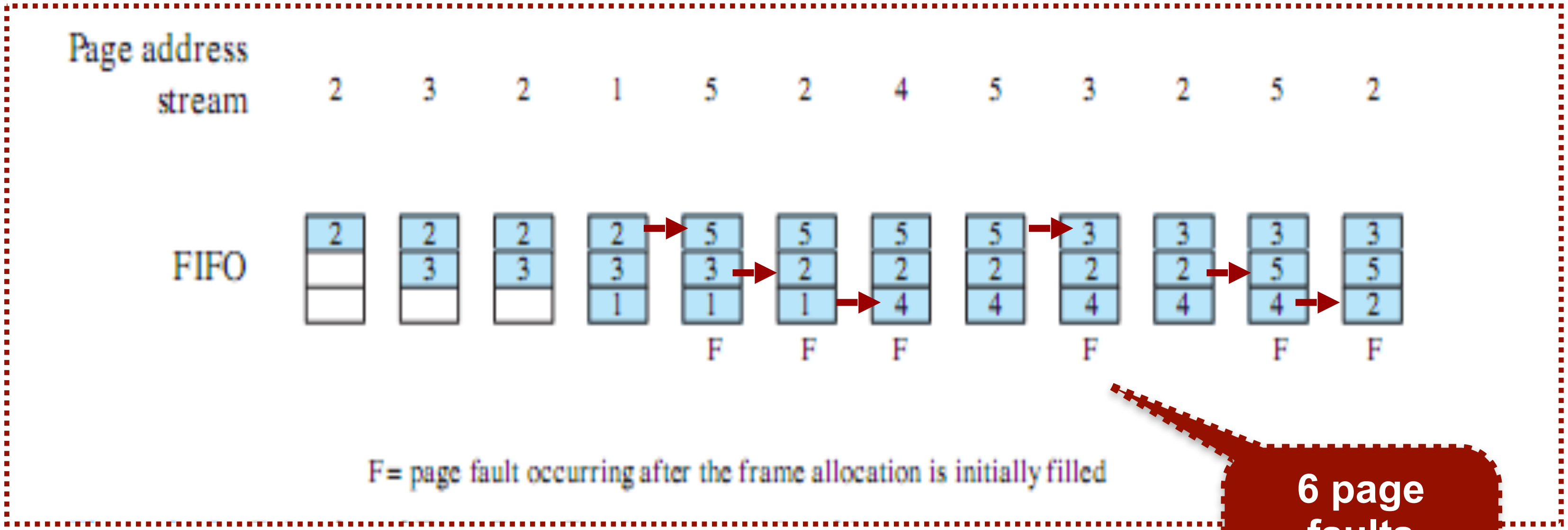
Replacement Algorithm: Least Recently Used (LRU)


Page address stream
2 3 2 1 5 2 4 5 3 2 5 2
LRU
F
F
F
F
F= page fault occurring after the frame allocation is initially filled
4 page faults

Replacement Algorithm: First-In-First-Out (FIFO)

- *Simplest* replacement policy to implement
- Page that has been in memory the longest is replaced
- Treat page frames allocated to a process as a circular buffer
- Pages are removed in round-robin style

Replacement Algorithm: First-In-First-Out (FIFO)



F= page fault occurring after the frame allocation is initially filled

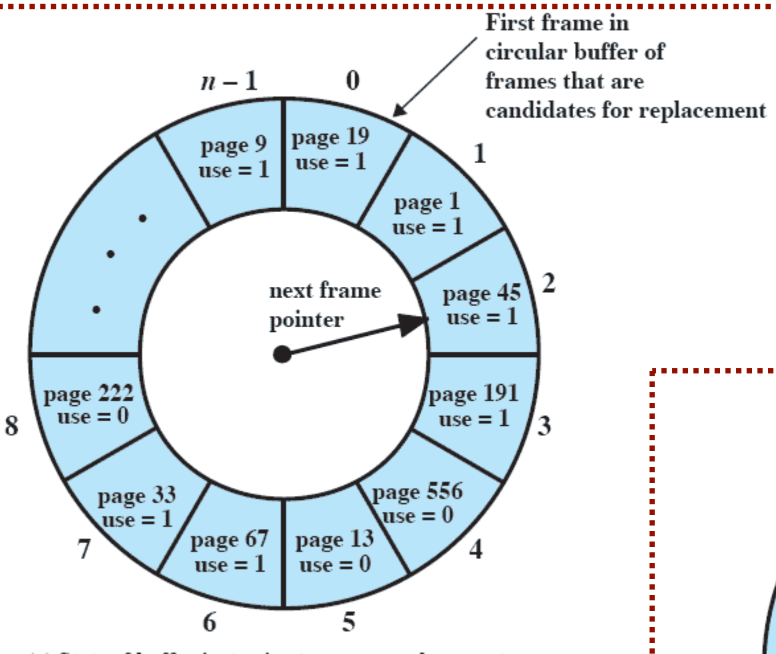
6 page faults

Replacement Algorithm: Clock Policy

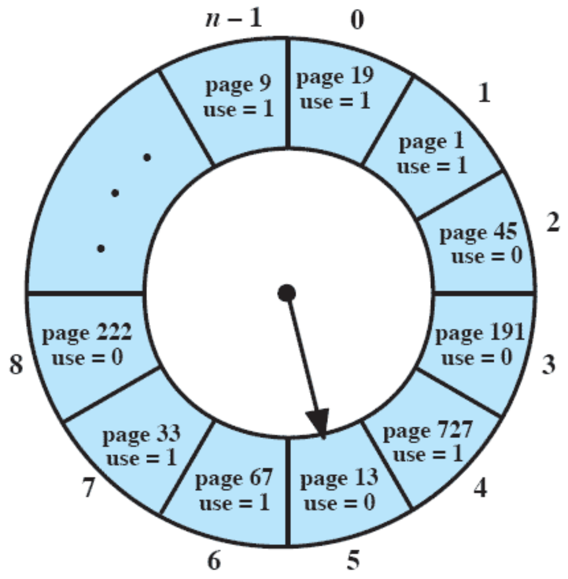
- Requires the association of an additional bit with each frame — referred to as the use bit
- When a page is first loaded in memory or referenced, the use bit is set to 1
- The set of frames is considered to be a circular buffer, with a pointer associated with it
- Any frame with a use bit of 1 is passed over by the algorithm
- Page frames visualised as laid out in a *circle*

First frame encountered with use bit of 0 is replaced

Replacement Algorithm: Clock Policy



(a) State of buffer just prior to a page replacement

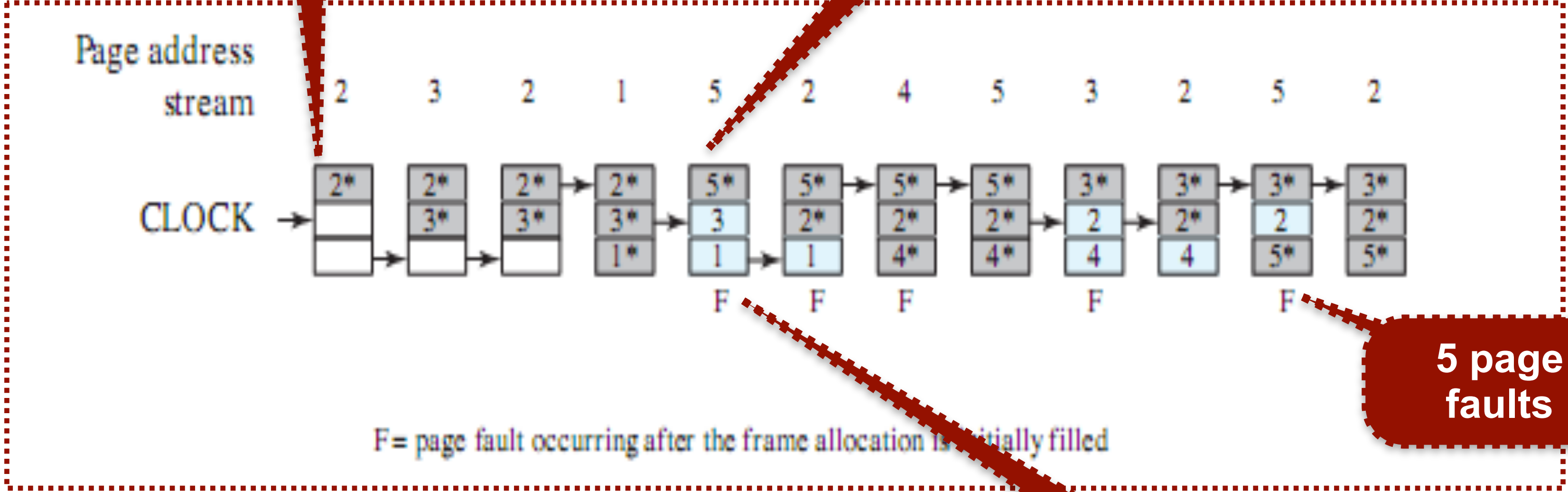


(b) State of buffer just after the next page replacement

Replacement Algorithm: Clock Policy

* implies that page is referenced (bit is 1) and not yet passed

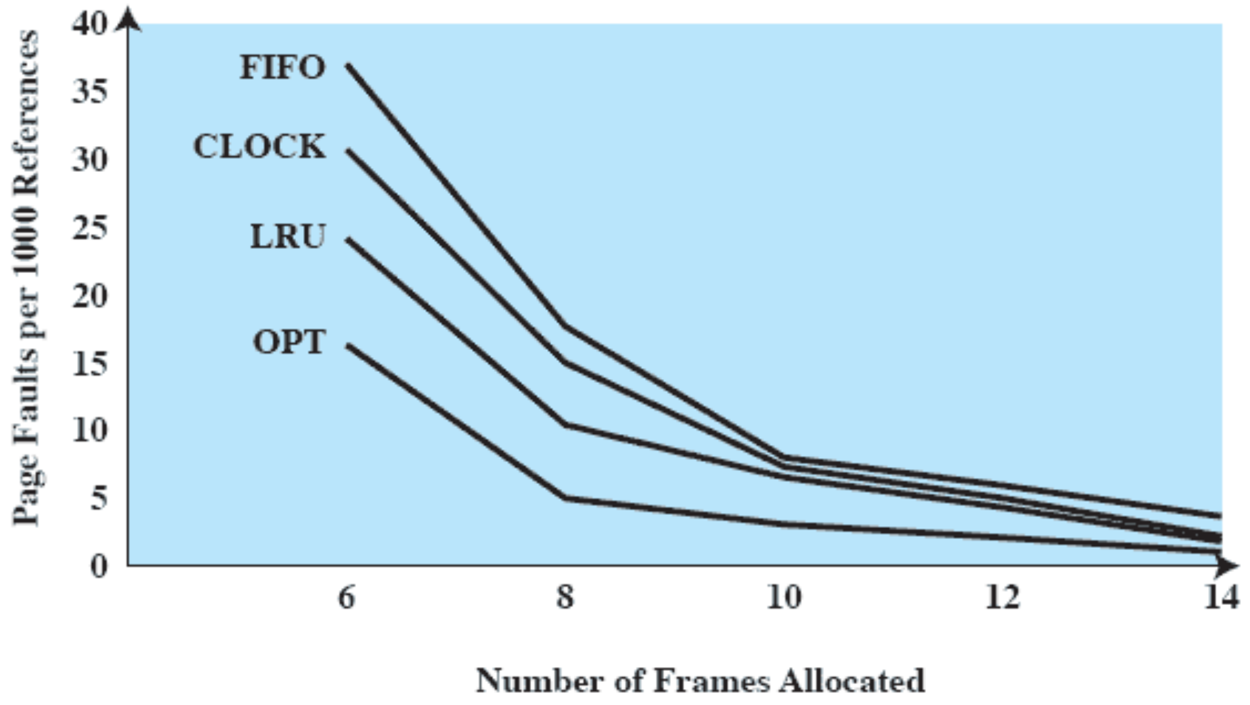
Pass over 2, 3, and 1; and come back to replace 2 with 5



5 page faults

When you pass over, reset the bit to 0 (no * in the example)

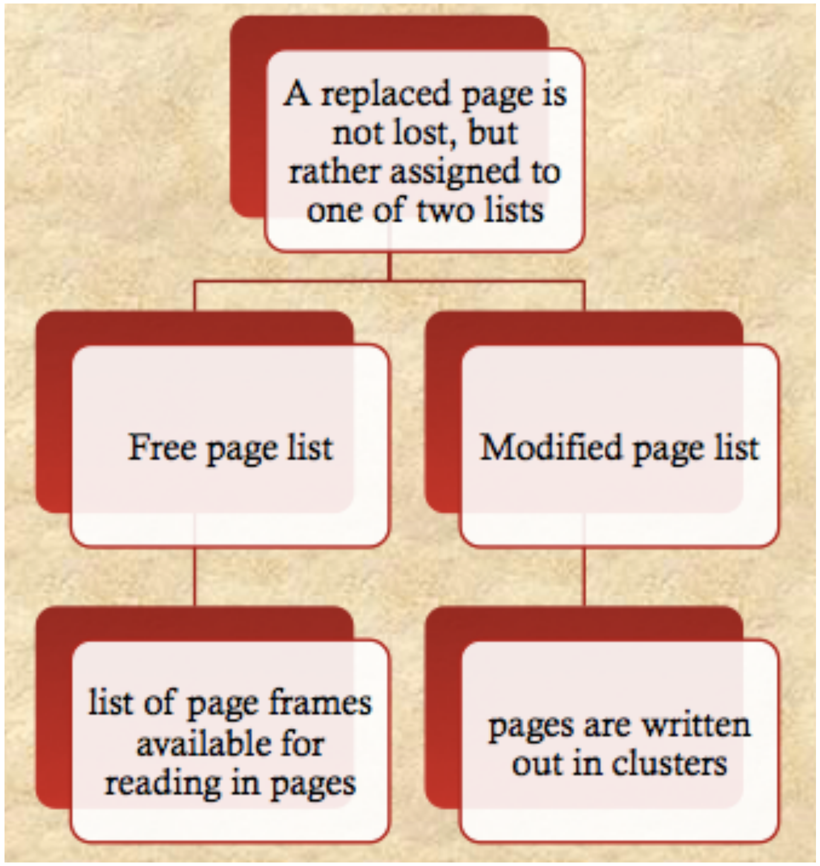
Comparison: Replacement Algorithms



Comparison of Fixed-Allocation, Local Page Replacement Algorithms

What is Page Buffering?

- Improve paging performance
- Allow the use of simpler page replacement policy (FIFO)
- The replaced page is removed from the page table — and placed in one of the two lists


A replaced page is not lost, but rather assigned to one of two lists
Free page list
Modified page list
list of page frames available for reading in pages
pages are written out in clusters

MONASH University

How resident set is managed by OS?

Resident Set Management

- OS must decide how many pages to bring into main memory

at least one ready process

- The smaller the amount of memory allocated to each process, the more processes can reside in memory
- Small number of pages loaded increases page faults
- Beyond a certain size, further allocations of memory will not affect the page fault rate

Resident Set Management

Fixed Allocation

- Give a process a fixed number of frames in main memory within which to execute
- When a page fault occurs, one of the pages of that process must be replaced

Variable Allocation

- Allow the number of page frames allocated to a process to be varied — over the lifetime of the process
- Related to the concept of replacement scope

Replacement Scope: Local vs Global

Local Replacement

- Chooses only among the resident pages of the process that generated the page fault

Global Replacement

- Considers all unlocked pages in main memory — regardless of which process owns a particular page

Both strategies are activated by a page fault when there are no free page frames

Resident Set Management: Summary

	Local Replacement	Global Replacement
Fixed Allocation	•Number of frames allocated to a process is fixed. •Page to be replaced is chosen from among the frames allocated to that process.	•Not possible.
Variable Allocation	•The number of frames allocated to a process may be changed from time to time to maintain the working set of the process. •Page to be replaced is chosen from among the frames allocated to that process.	•Page to be replaced is chosen from all available frames in main memory; this causes the size of the resident set of processes to vary.

Summary of Lecture 9

- With the use of virtual memory:
  - All address references are logical references that are translated at run time to real addresses
  - A process can be broken up into pieces
  - Two approaches are paging and segmentation
  - Management scheme requires both hardware and software support

Next week: I/O Management and Disk Scheduling