O/S Design Decisions

- Fetch Policy = what page & when
- Placement Policy = in what frame
- Replacement Policy = out of what frame
- Resident Set Management = how many frames per process
- Cleaning Policy = when to write out to disk
- Load control = how many processes (CPU scheduling)

O/S Design Decisions

Fetch Policy
- Demand
- 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

Fetch Policy

- Fetch Policy
  - Determines when a page should be brought into memory
  - Demand paging only brings pages into main memory when a reference is made to a location on the page
    - Many page faults when process first started
  - Prepaging brings in more pages than needed
    - More efficient to bring in pages that reside contiguously on the disk

Placement Policy

- Determines where in real memory a process piece is to reside
- Important in a segmentation system
- Paging or combined paging with segmentation hardware performs address translation

Replacement Policy

- Placement Policy
  - Which page is replaced?
  - Page removed should be the page least likely to be referenced in the near future
  - Most policies predict the future behavior on the basis of past behavior

Replacement Policy

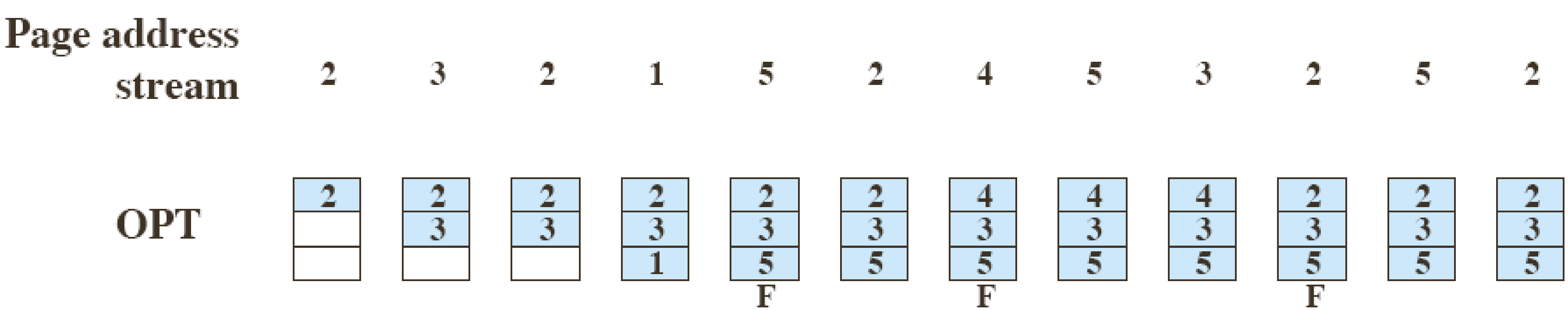
- Frame Locking
  - If frame is locked, it may not be replaced
  - Kernel of the operating system: the page handler itself!
  - Control structures
  - I/O buffers
  - Associate a lock bit with each frame

Basic Replacement Algorithms

- Optimal policy (OPT)
- Least Recently Used (LRU)
- First-In-First-Out (FIFO)
- Clock

Basic Replacement Algorithms

- Optimal policy
  - Selects for replacement that page for which the time to the next reference is the longest
  - Impossible to have perfect knowledge of future events

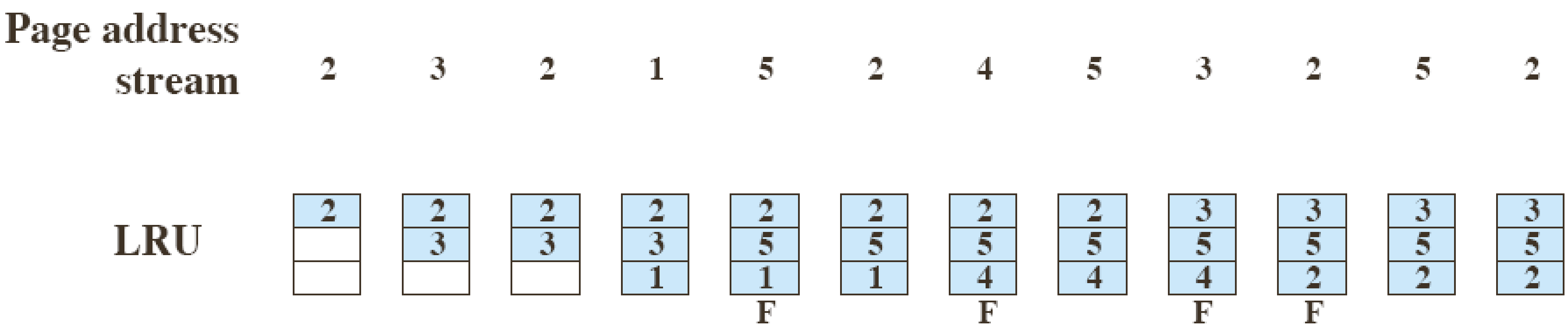


Basic Replacement Algorithms

- Least Recently Used (LRU)
  - Replaces 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
  - Each page could be tagged with the time of last reference. This would require a great deal of overhead.

Basic Replacement Algorithms

- Least Recently Used (LRU)



Basic Replacement Algorithms

- First-in, first-out (FIFO)
  - Treats page frames allocated to a process as a circular buffer
  - Pages are removed in round-robin style
  - Simplest replacement policy to implement
  - Page that has been in memory the longest is replaced
  - These pages may be needed again very soon

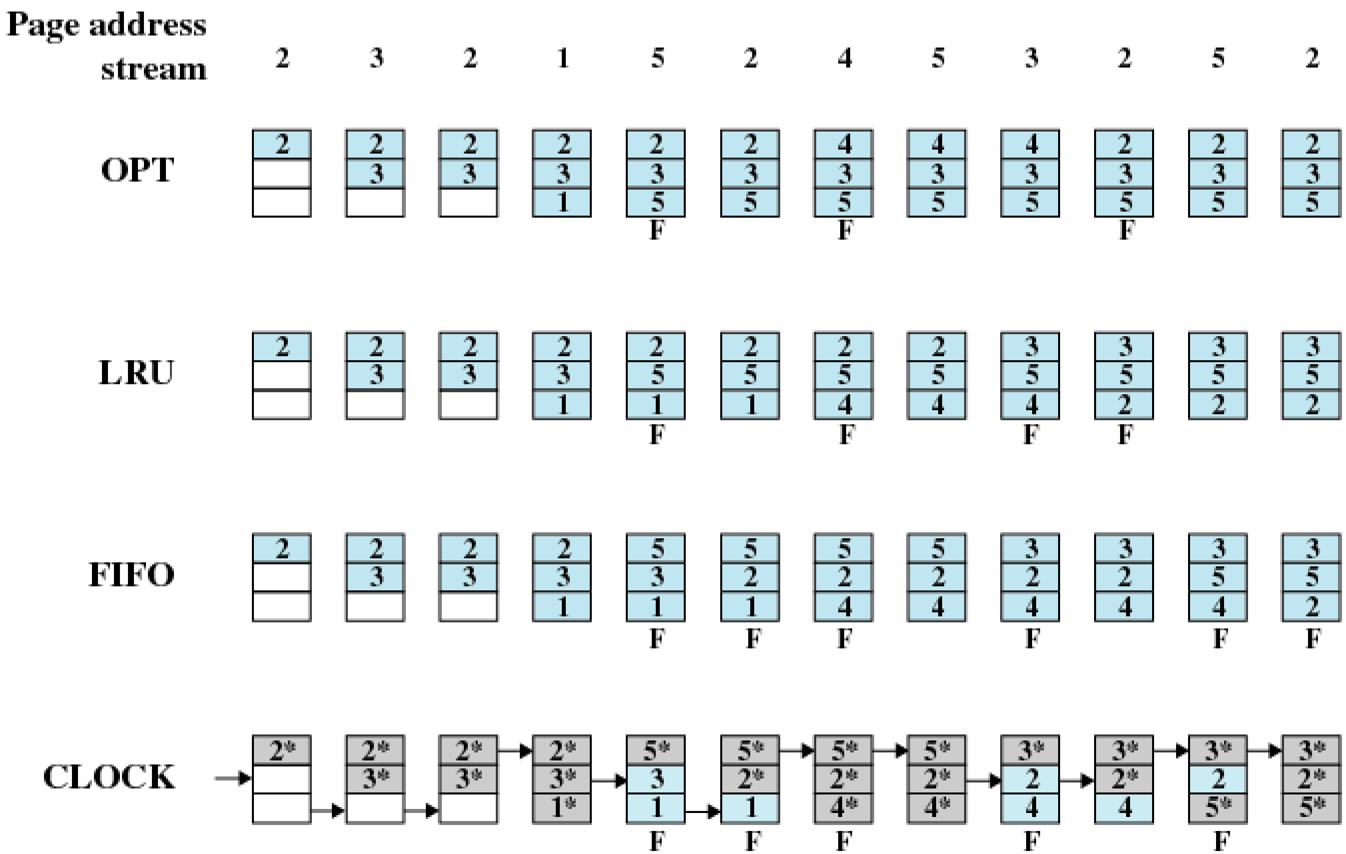
Basic Replacement Algorithms

- First-in, first-out (FIFO)

Page address stream	2	3	2	1	5	2	4	5	3	2	5	2
FIFO	2	2	2	2	5	5	5	5	3	3	3	3
		3	3	3	3	2	2	2	2	2	5	5
				1	1	1	4	4	4	4	4	2
					F	F	F		F		F	F

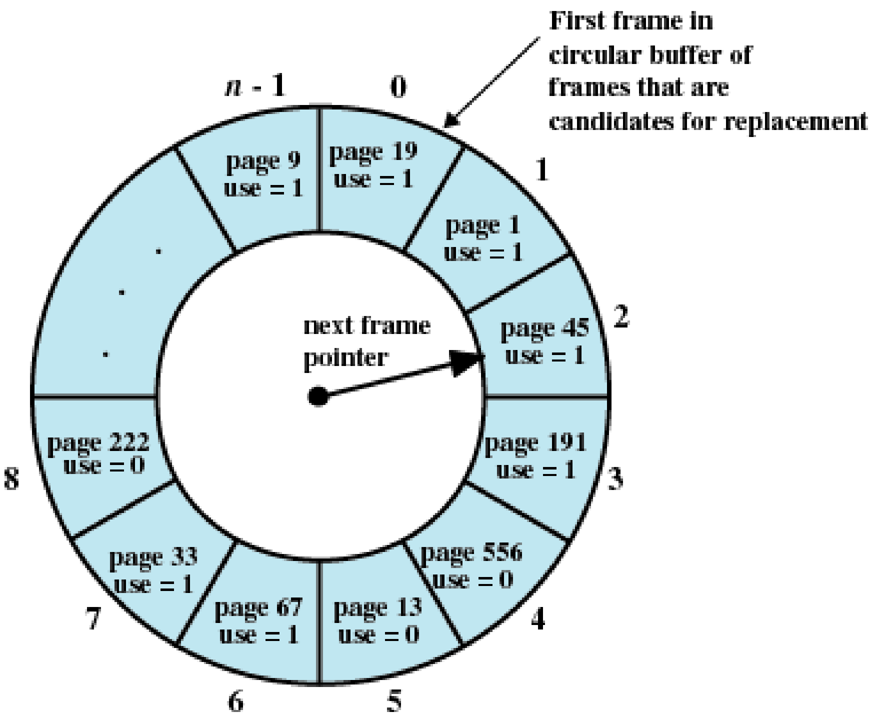
Basic Replacement Algorithms

- Clock Policy
  - Additional bit called a use bit
  - When a page is first loaded in memory, the use bit is set to 1
  - When the page is referenced, the use bit is set to 1
  - When it is time to replace a page, the first frame encountered with the use bit set to 0 is replaced.
  - During the search for replacement, each use bit set to 1 is changed to 0

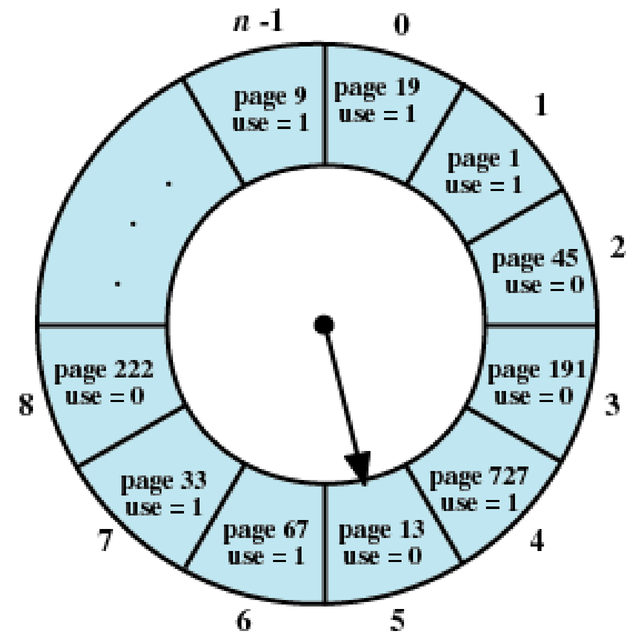


F = page fault occurring after the frame allocation is initially filled

Figure 8.15 Behavior of Four Page-Replacement Algorithms



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



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

Figure 8.16 Example of Clock Policy Operation

Comparison of Placement Algorithms

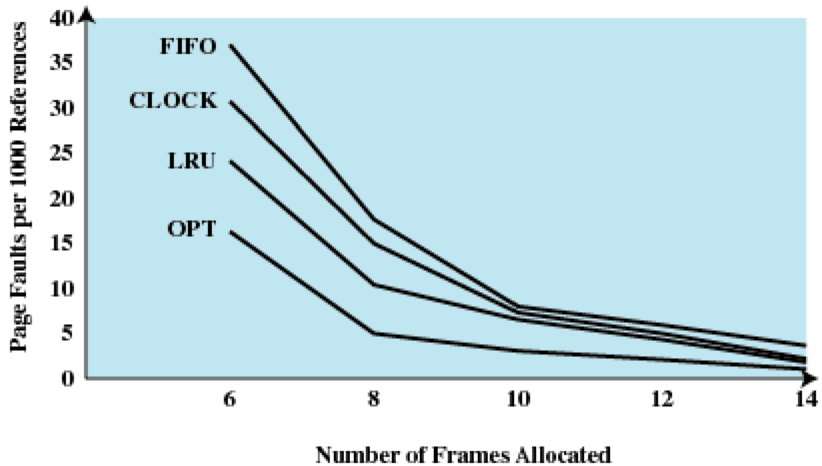


Figure 8.17 Comparison of Fixed-Allocation, Local Page Replacement Algorithms

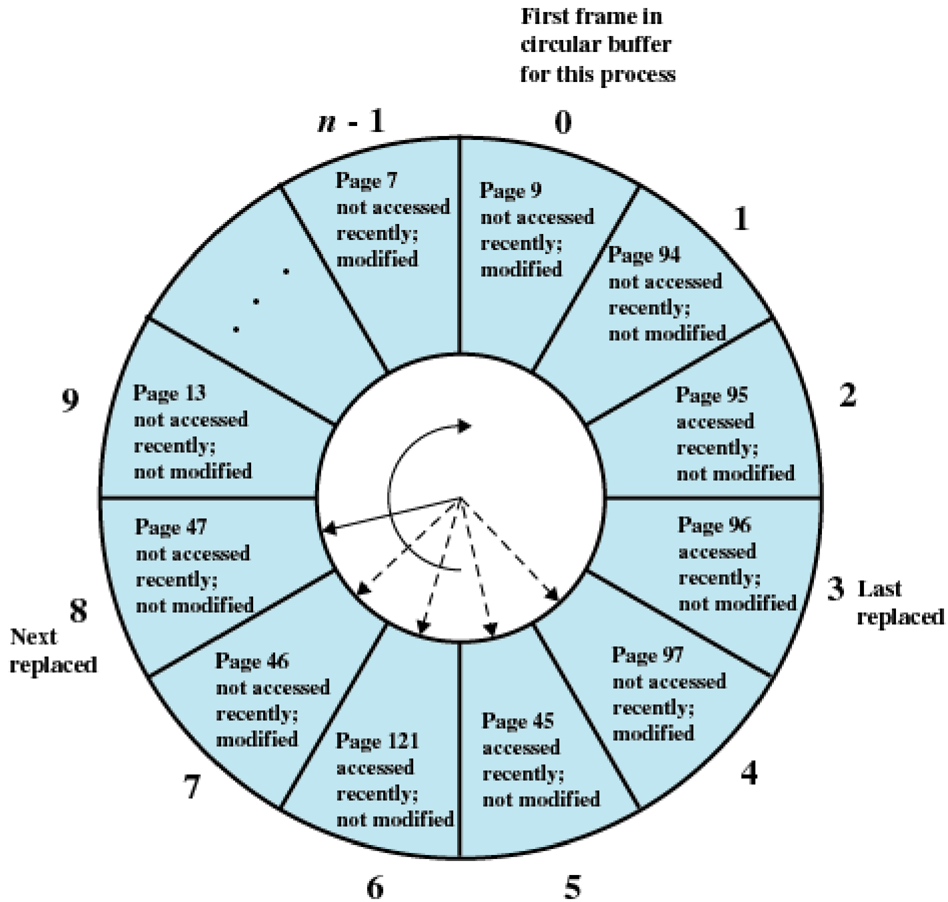


Figure 8.18 The Clock Page-Replacement Algorithm [GOLD89]

Resident Set Size

- Fixed-allocation
  - Gives a process a fixed number of pages within which to execute
  - When a page fault occurs, one of the pages of that process must be replaced
- Variable-allocation
  - Number of pages allocated to a process varies over the lifetime of the process

Resident Set Size

- Local scope
  - Replace page only within the process that faulted
- Global scope
  - Replace page in any frame across all processes

Resident Set Size

	Local Replacement	Global Replacement
Fixed Allocation	•Number of frames allocated to 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.

Fixed Allocation, Local Scope

- Decide ahead of time the amount of allocation to give a process
- One replacement clock or queue per process
- If allocation is too small, there will be a high page fault rate
- If allocation is too large there will be too few programs in main memory

Variable Allocation, Global Scope

- Easiest to implement
- Adopted by many operating systems
- Operating system keeps list of free frames
- Free frame is added to resident set of process when a page fault occurs
- If no free frame, replaces one from another process

Variable Allocation, Local Scope

- One global clock or queue for all processes
- When new process added, allocate number of page frames based on application type, program request, or other criteria
- When page fault occurs, select page from among the resident set of the process that suffers the fault
- Reevaluate allocation from time to time

Variable Allocation, Local Scope

Sequence of Page References	Window Size, Δ 2	3	4	5
24	24	24	24	24
15	24 15	24 15	24 15	24 15
18	15 18	24 15 18	24 15 18	24 15 18
23	18 23	15 18 23	24 15 18 23	24 15 18 23
24	23 24	18 23 24	•	•
17	24 17	23 24 17	18 23 24 17	15 18 23 24 17
18	17 18	24 17 18	•	18 23 24 17
24	18 24	•	24 17 18	•
18	•	18 24	•	24 17 18
17	18 17	24 18 17	•	•
17	17	18 17	•	•
15	17 15	17 15	18 17 15	24 18 17 15
24	15 24	17 15 24	17 15 24	•
17	24 17	•	•	17 15 24
24	•	24 17	•	•
18	24 18	17 24 18	17 24 18	15 17 24 18

Cleaning Policy

- The opposite of Fetch Policy
- Demand cleaning
  - A page is written out only when it has been selected for replacement
- Precleaning
  - Pages are written out in batches

Cleaning Policy

- Best approach uses page buffering
  - Replaced pages are placed in two lists
    - Modified and unmodified
  - Pages in the modified list are periodically written out in batches
  - Pages in the unmodified list are either reclaimed if referenced again or lost when its frame is assigned to another page

Load Control

- Determines the number of processes that will be resident in main memory
- Too few processes, many occasions when all processes will be blocked and much time will be spent in swapping
- Too many processes will lead to thrashing

Multiprogramming

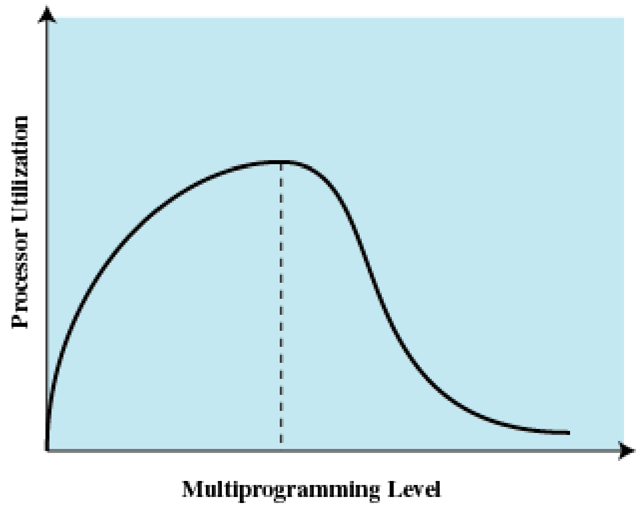


Figure 8.21 Multiprogramming Effects

Process Suspension

- Lowest priority process
- Faulting process
  - This process does not have its working set in main memory so it will be blocked anyway
- Last process activated
  - This process is least likely to have its working set resident

Process Suspension

- Process with smallest resident set
  - This process requires the least future effort to reload
- Largest process
  - Obtains the most free frames
- Process with the largest remaining execution window