Virtual Memory

Chapter 8

Hardware and Control Structures

- Memory references are dynamically translated into physical addresses at run time
  - A process may be swapped in and out of main memory such that it occupies different regions
- A process may be broken up into pieces that do not need to located contiguously in main memory
  - All pieces of a process do not need to be loaded in main memory during execution

Execution of a Program

- Operating system brings into main memory a few pieces of the program
- Resident set - portion of process that is in main memory
- An interrupt is generated when an address is needed that is not in main memory
- Operating system places the process in a blocking state

Execution of a Program

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

Advantages of Breaking up a Process

- 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

Types of Memory

- Real memory
  - Main memory
- Virtual memory
  - Memory on disk
  - Allows for effective multiprogramming and relieves the user of tight constraints of main memory

Thrashing

- Swapping out a piece of a process just before that piece is needed
- The processor spends most of its time swapping pieces rather than executing user instructions

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
- Possible to make intelligent guesses about which pieces will be needed in the future
- This suggests that virtual memory may work efficiently

Support Needed for Virtual Memory

- Hardware must support paging and segmentation
- Operating system must be able to management the movement of pages and/or segments between secondary memory and main memory

Paging

- Each process has its own page table
- Each page table entry contains the frame number of the corresponding page in main memory
- A bit is needed to indicate whether the page is in main memory or not

Modify Bit in Page Table

- Another modify bit is needed to indicate if the page has been altered since it was last loaded into main memory
- If no change has been made, the page does not have to be written to the disk when it needs to be swapped out

Page Table Entries

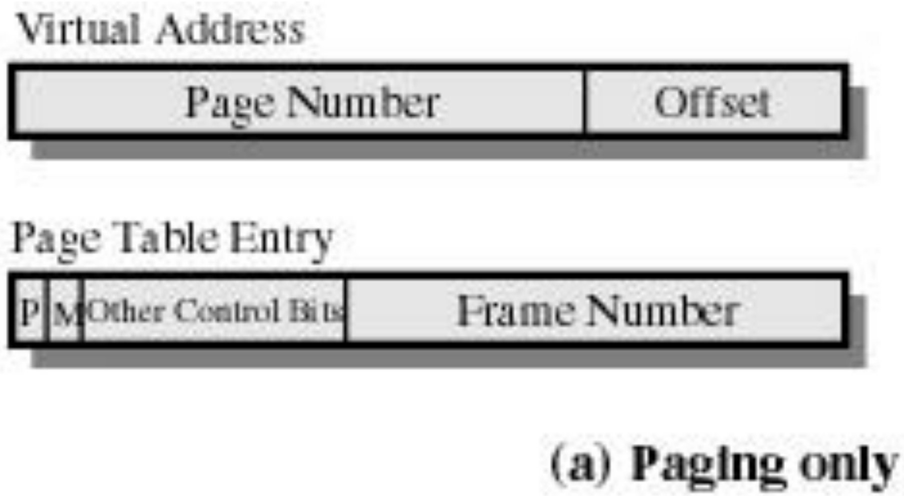


Figure 8.2 Typical Memory Management Formats

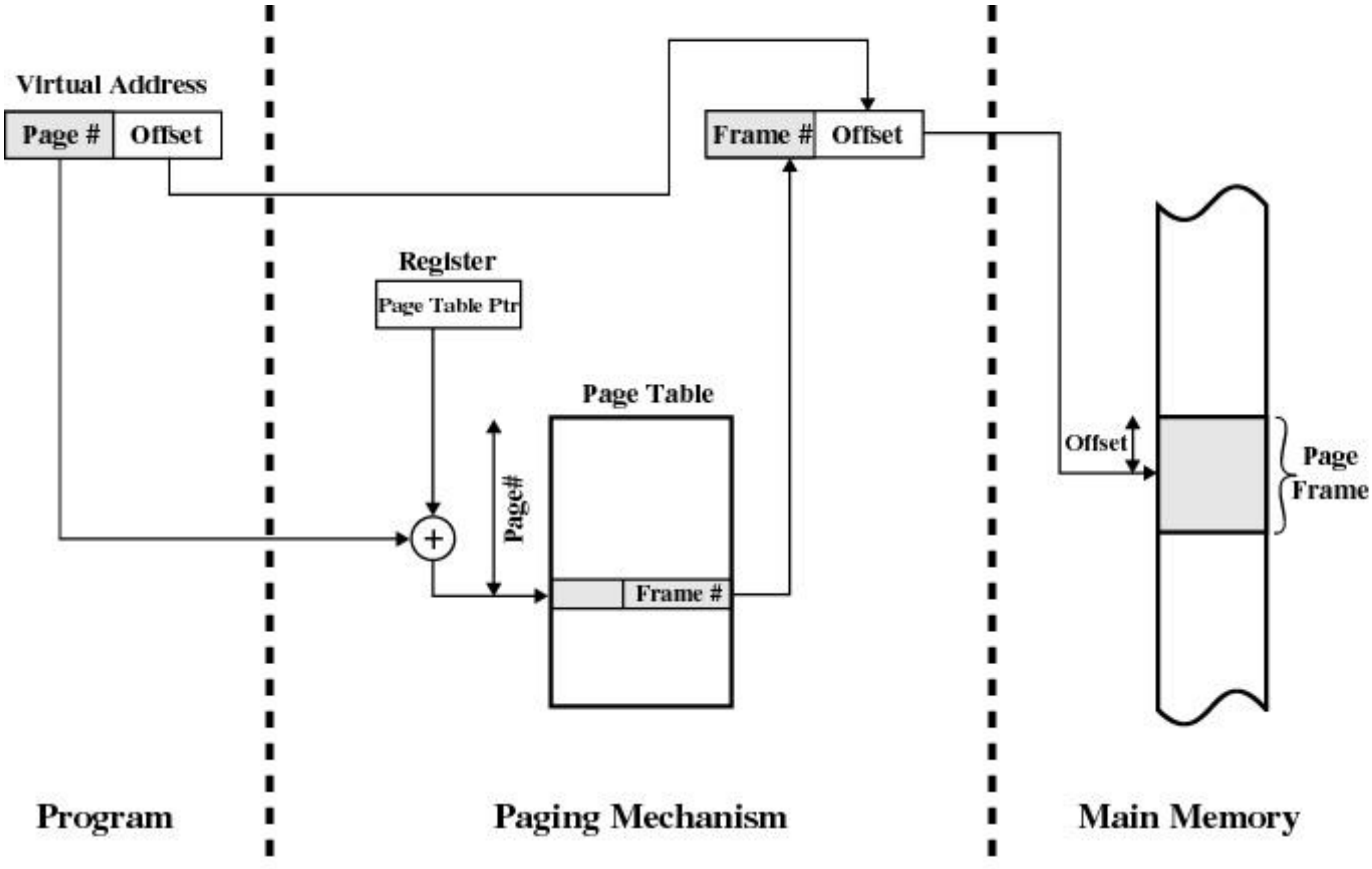


Figure 8.3 Address Translation in a Paging System

Two-Level Scheme for 32-bit Address

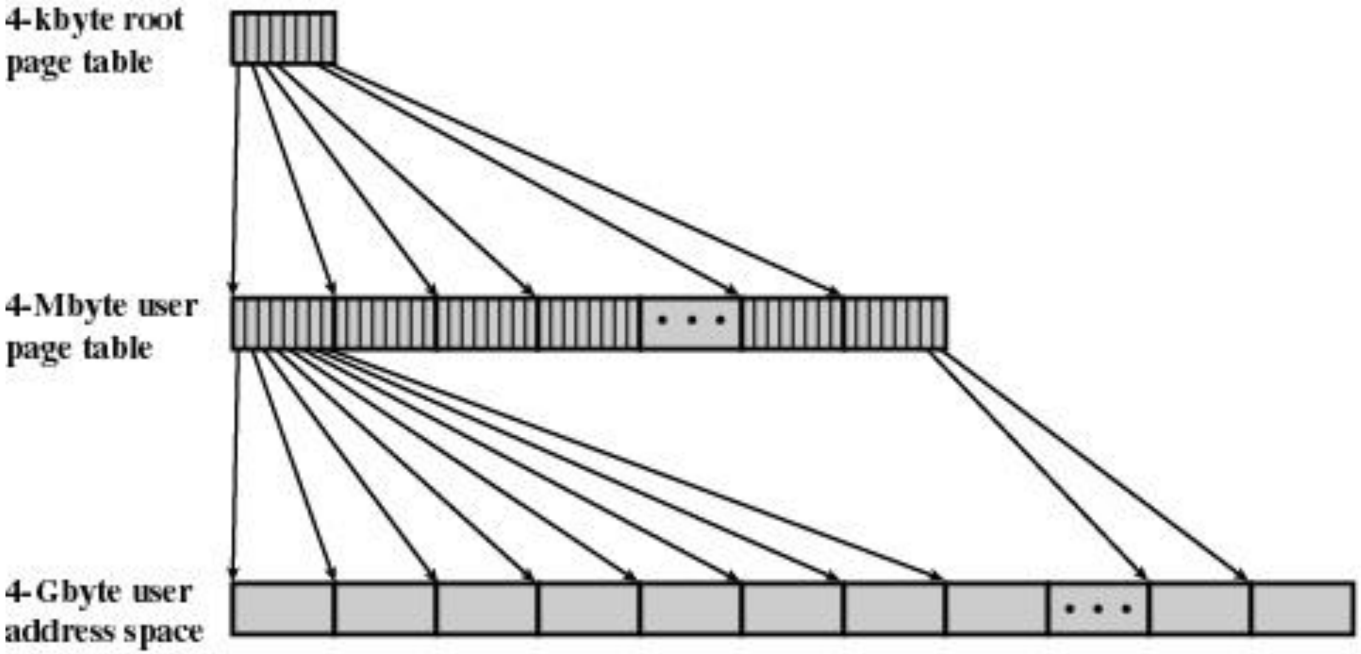
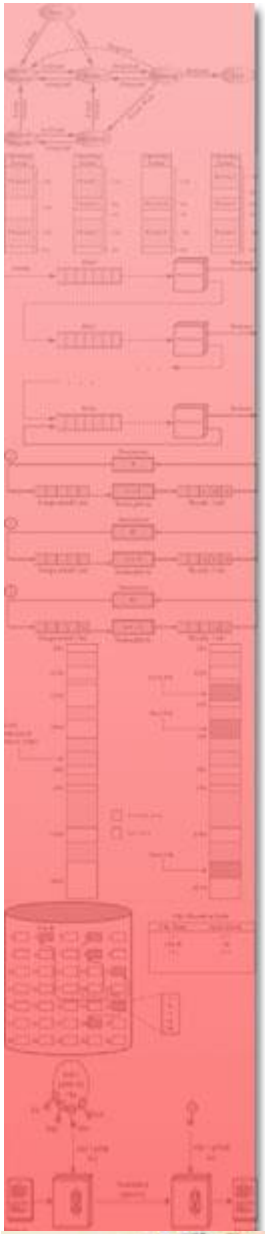


Figure 8.4 A Two-Level Hierarchical Page Table [JACO98a]



Page Tables

- The entire page table may take up too much main memory
- Page tables are also stored in virtual memory
- When a process is running, part of its page table is in main memory

Translation Lookaside Buffer

- Each virtual memory reference can cause two physical memory accesses
  - one to fetch the page table entry
  - one to fetch the data
- To overcome this problem a high-speed cache is set up for page table entries
  - called the TLB - Translation Lookaside Buffer

Translation Lookaside Buffer

- Contains page table entries that have been most recently used
- Functions same way as a memory cache

Translation Lookaside Buffer

- Given a virtual address, processor examines the TLB
- If page table entry is present (a hit), the frame number is retrieved and the real address is formed
- If page table entry is not found in the TLB (a miss), the page number is used to index the process page table

Translation Lookaside Buffer

- First checks if page is already in main memory
  - if not in main memory a page fault is issued
- The TLB is updated to include the new page entry

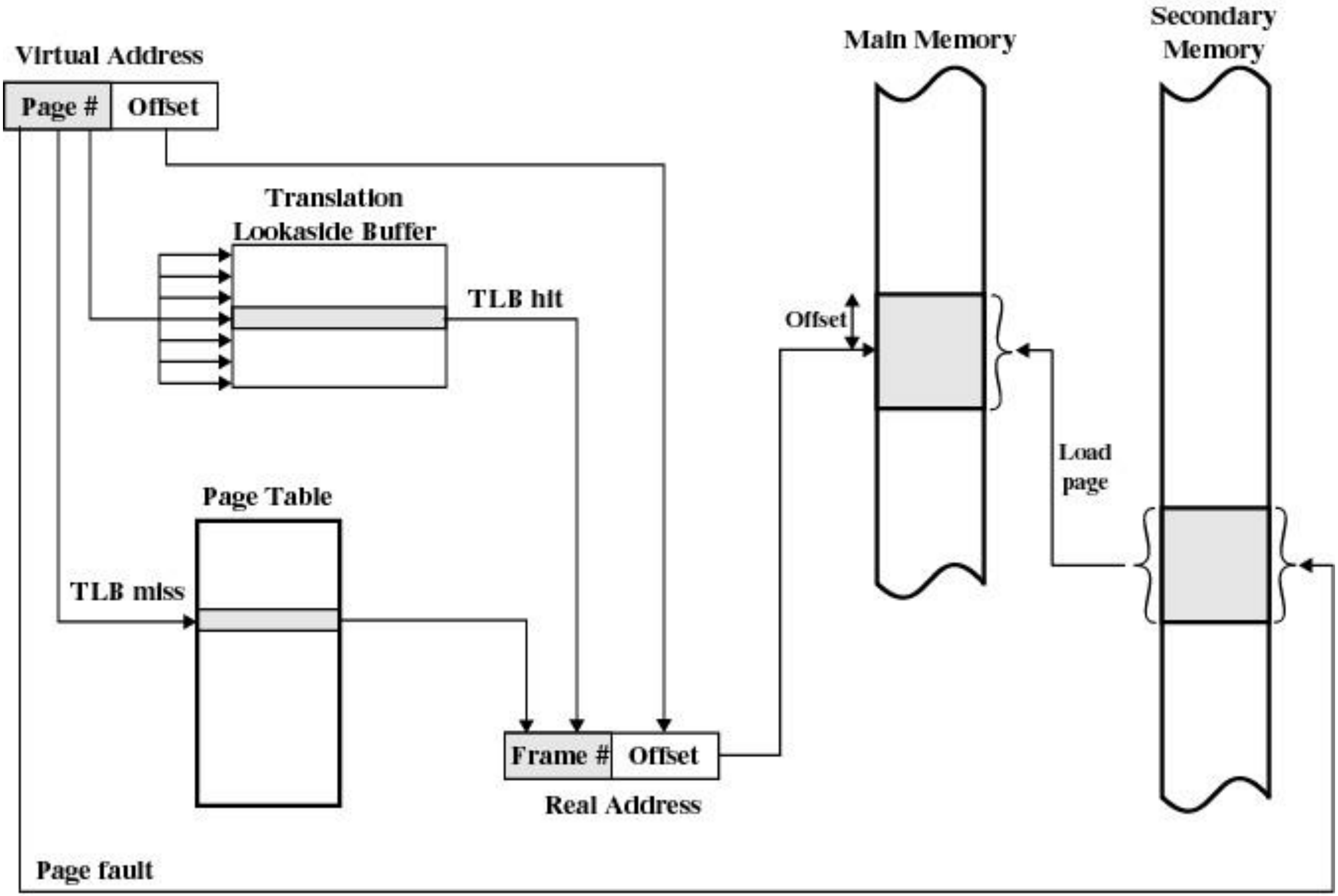


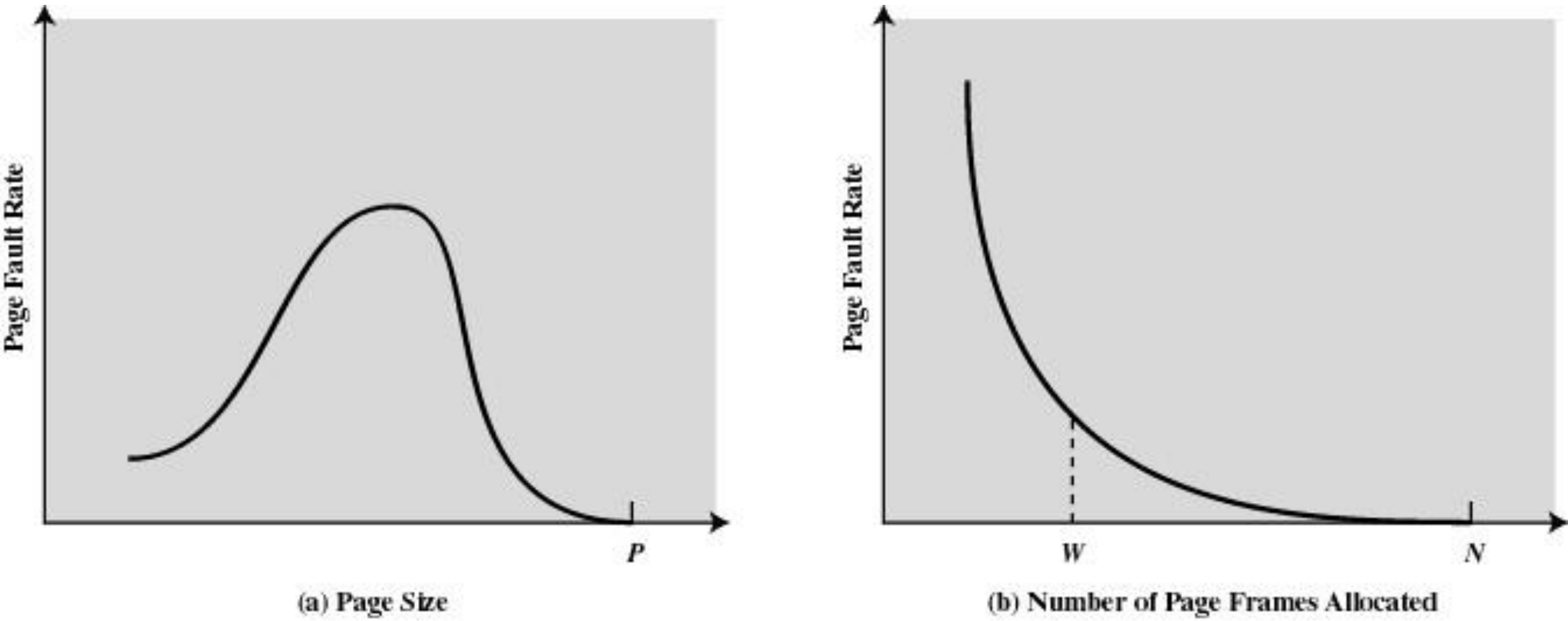
Figure 8.7 Use of a Translation Lookaside Buffer

Page Size

- Smaller page size, less amount of internal fragmentation
- Smaller page size, more pages required per process
- More pages per process means larger page tables
- Larger page tables means large portion of page tables in virtual memory
- Secondary memory is designed to efficiently transfer large blocks of data so a large page size is better

Page Size

- Small page size, large number of pages will be found in main memory
- As time goes on during execution, the pages in memory will all contain portions of the process near recent references. Page faults low.
- Increased page size causes pages to contain locations further from any recent reference. Page faults rise.



P = size of entire process
W = working set size
N = total number of pages in process

Figure 8.11 Typical Paging Behavior of a Program

Example Page Sizes

Table 8.2 Example Page Sizes

Computer	Page Size
Atlas	512 48-bit words
Honeywell-Multics	1024 36-bit word
IBM 370/XA and 370/ESA	4 Kbytes
VAX family	512 bytes
IBM AS/400	512 bytes
DEC Alpha	8 Kbytes
MIPS	4 kbyes to 16 Mbytes
UltraSPARC	8 Kbytes to 4 Mbytes
Pentium	4 Kbytes or 4 Mbytes
PowerPc	4 Kbytes

Segmentation

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

Segment Tables

- corresponding segment in main memory
- Each entry contains the length of the segment
- A bit is needed to determine if 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

Segment Table Entries

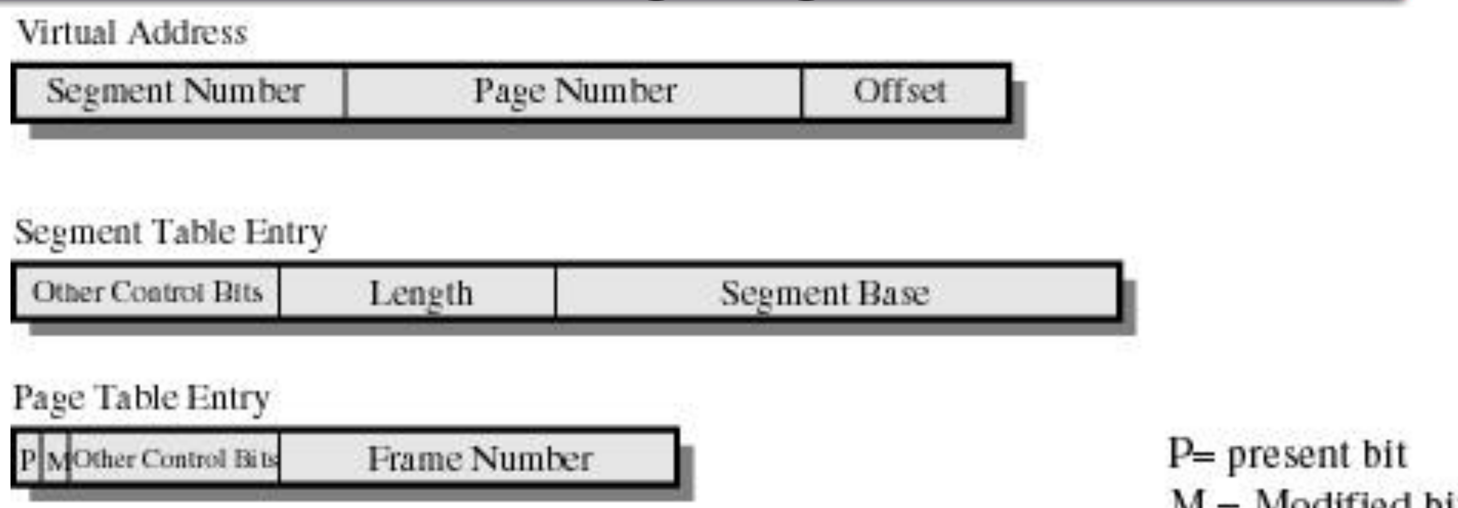
(b) Segmentation only

Figure 8.2 Typical Memory Management Formats

Combined Paging and Segmentation

- Paging is transparent to the programmer
- Paging eliminates external fragmentation
- Segmentation is visible to the programmer
- Segmentation allows for growing data structures, modularity, and support for sharing and protection
- Each segment is broken into fixed-size pages

Combined Segmentation and Paging



(c) Combined segmentation and paging

Figure 8.2 Typical Memory Management Formats

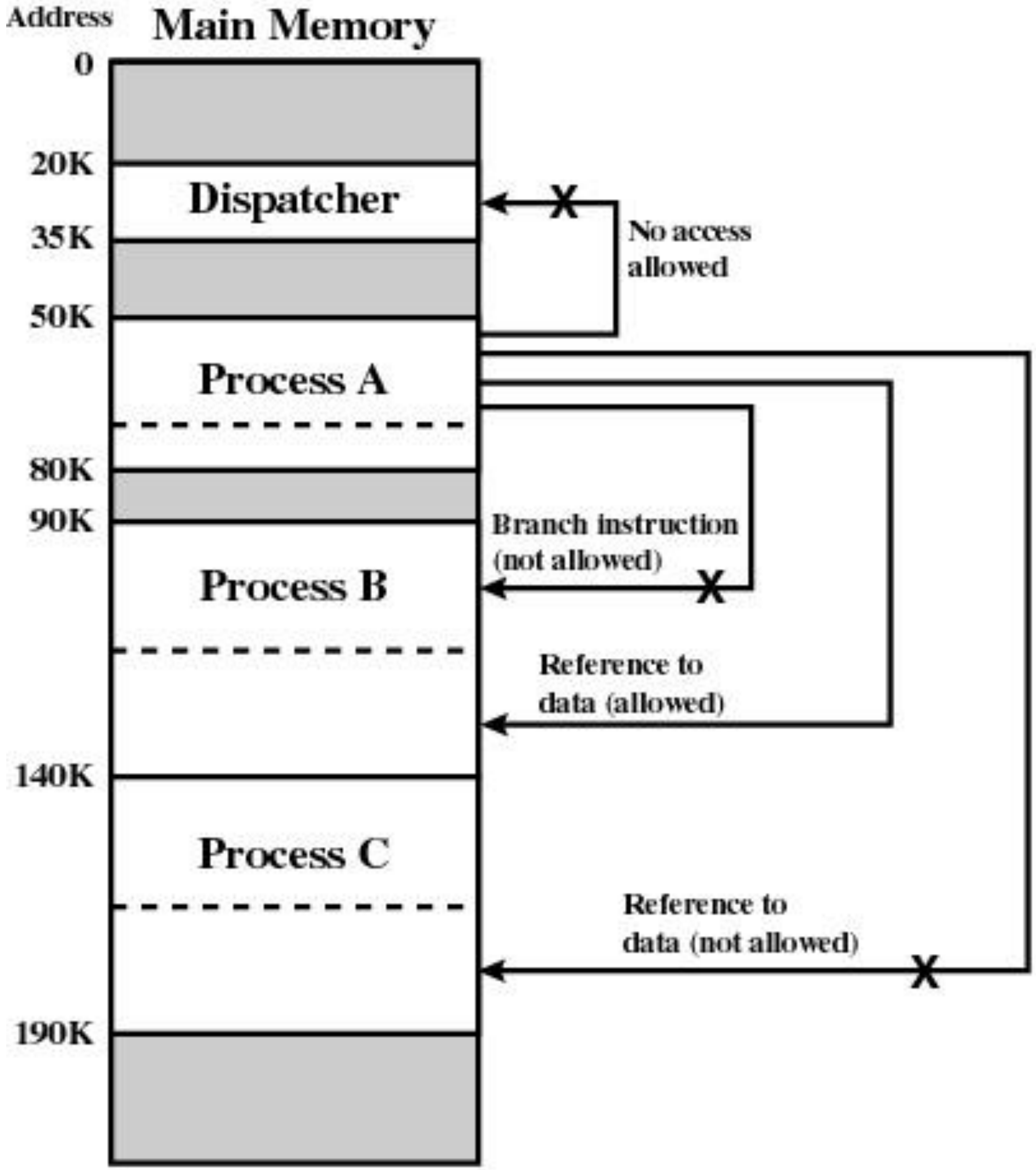


Figure 8.14 Protection Relationships Between Segments

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

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
  - Control structures
  - I/O buffers
  - Associate a lock bit with each frame

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

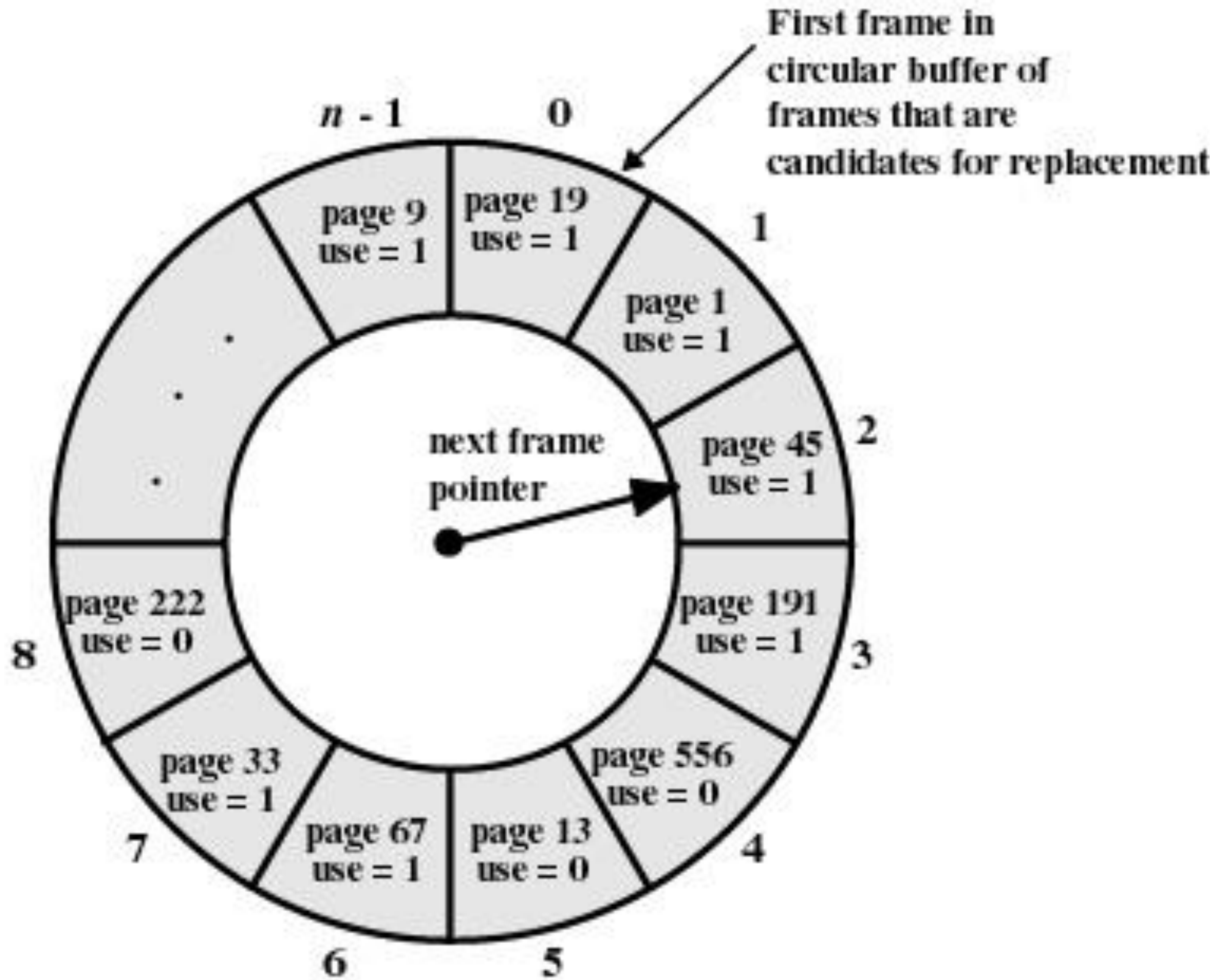
- Least Recently Used (LRU)
  - Replaces the page that has not been referenced for the longest time
  - By the principle of locality, this should be the page least likely to be referenced in the near future
  - Each page could be tagged with the time of last reference. This would require a great deal of overhead.

Basic Replacement Algorithms

- 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

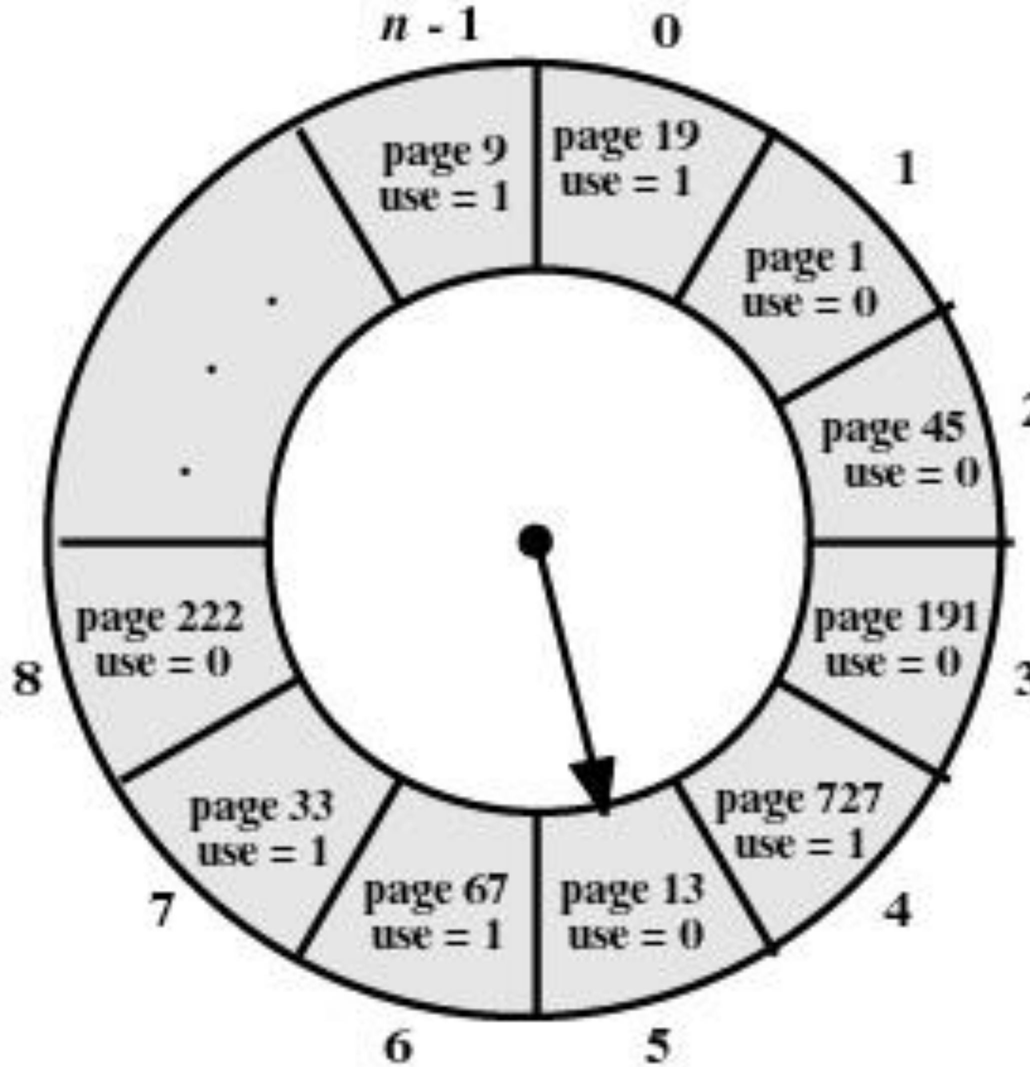
Basic Replacement Algorithms

- Clock Policy
  - Additional bit called a use bit
  - When a page is first loaded in memory, the use bit is set to 0
  - 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



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

Figure 8.16 Example of Clock Policy Operation



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

Figure 8.16 Example of Clock Policy Operation

Cleaning 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

Reducing Load by Process Suspension

- Lowest priority process is suspended
- Faulting process is suspended
  - 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

UNIX and Solaris Memory Management

- Paging System
  - Page table: one per process
  - Disk block descriptor: disk copy of a page
  - Page frame data table: frame-page mapping
  - Swap-use table: one per swap device

Data Structures

Page frame number	Age	Copy on write	Modify	Reference	Valid	Protect

(a) Page table entry

Swap device number	Device block number	Type of storage

(b) Disk block descriptor

Figure 8.22 UNIX SVR4 Memory Management Formats

Data Structures

Page state	Reference count	Logical device	Block number	Pfdata pointer

(c) Page frame data table entry

Reference count	Page/storage unit number

(d) Swap-use table entry

Figure 8.22 UNIX SVR4 Memory Management Formats

UNIX and Solaris Memory Management

- Page Replacement
  - refinement of the clock policy
  - First sweep sets use bit to 0
  - After some time, second sweep checks use bits, if still zero, this page can be replaced
- Kernel Memory Allocator
  - most blocks are smaller than a typical page size so buddy system is used and paging is not used