Virtual Memory

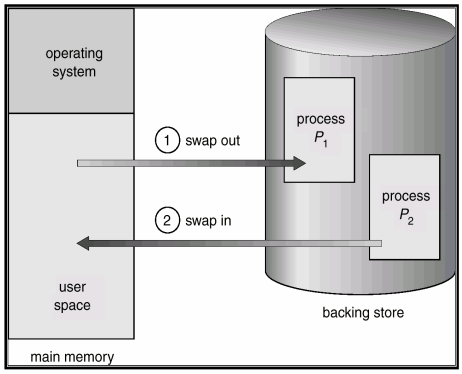
Chapter 8 do William Stallings
Chapter 10 do Silberschatz

SISTEMAS OPERATIVOS, 1º Semestre, 2004-2005

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.

Swap-in/Swap-out



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.
- If a process tries to access a logical address that is not in main memory -> **interrupt (memory access fault).**
- Operating system places the process in a blocking state.

Execution of a Program (cont)

- 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.

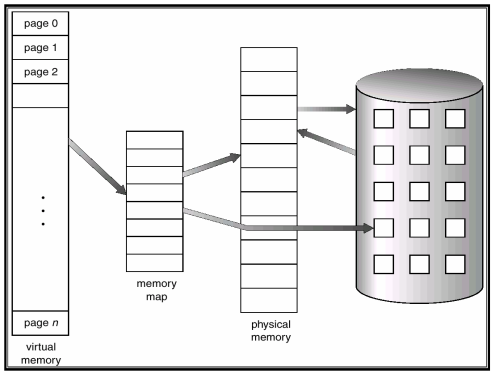
Advantages of Breaking up a Process

1. **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.
2. **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.

Memória Virtual



Thrashing

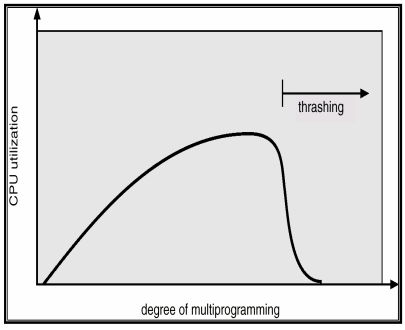
- When the processor spends most of its time swapping pieces rather than executing user instructions …

this is **thrashing.**

Thashing

- Se um processo não tem páginas "suficientes", o ritmo de faltas de página aumenta muito. Isto é caracterizado pelo comando *vmstat*:
  - Baixa taxa de ocupação do CPU.
  - Grande número de operações de I/O sobre o disco de paginação.
  - Poucas "frames"livres.
- **Thrashing** ≡ o processo está praticamente sempre à espera que o SO carregue páginas (page-in) e a transferir páginas (page-out) de/para o disco.

Thrashing



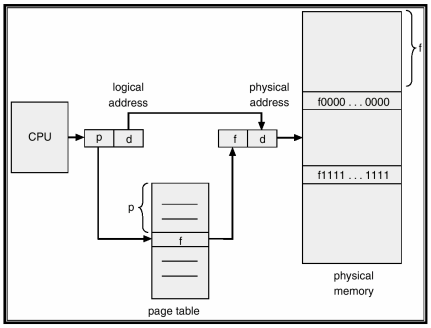
Principle of Locality

- Data references within a process tend to be accessed in a cluster of **locality**.
- 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.

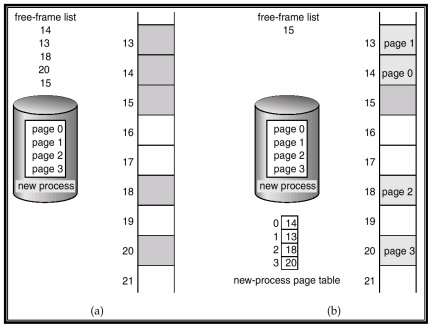
Support for Virtual Memory

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

Paging



Paging: free pages



Paging: Page Table Entries

- Each process has its own page table.
- Each page table entry (PTE) 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 (presence bit).

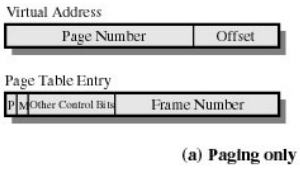
Protection

- Memory protection implemented by associating protection bit with each frame.
- **Valid-invalid bit** attached to each entry in the page table:
  - "valid" indicates that the associated page is in the process' logical address space, and is thus a legal page.
  - "invalid" indicates that the page is not in the process' logical address space.

Modify Bit in Page Table

- Another bit is needed to indicate if the page has been altered since it was last loaded into main memory (modify bit).
- 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



(a) Paging only

Figure 8.2 Typical Memory Management Formats

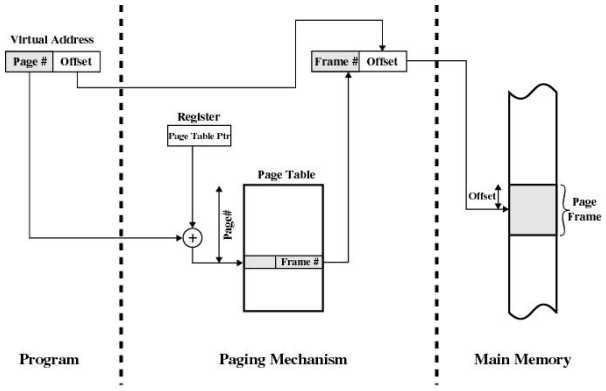
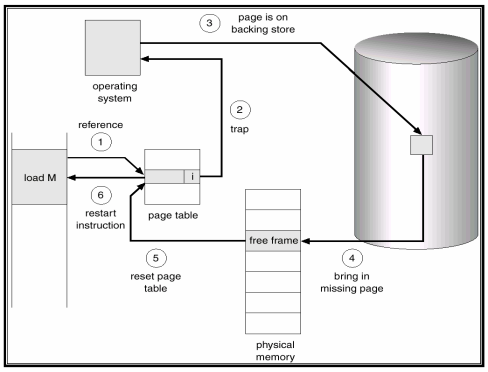


Figure 8.3 Address Translation in a Paging System

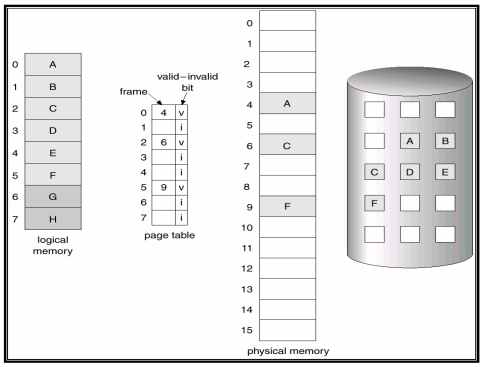
Page Fault



…Page Fault

1. Trap to OS
2. Save user registers and process state
3. Determine that interrupt was page fault
4. Check that page reference was legal, determine location of page on disk
5. Issue read from disk to free (physical) frame:
   - Wait in queue for device to service read request
   - Wait for device seek / rotational latency
   - Wait for transfer
6. [While waiting for disk: allocate CPU to another process]
7. Interrupt from disk…
8. [Save registers and process state for other process]
9. Determine that interrupt was from disk
10. Update page / OS tables (incl. valid bit) to show page is in memory
11. [Wait for CPU to be allocated to this process]
12. Restore registers, process state, page table
13. Restart instruction

Paging and Virtual Memory



Go to QUIZ_5

Page Table Structures:

A- Hierarchical Paging
B- Hashed PageTables
C- Inverted Page Tables

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.

Exemplo: VAX

- 2^{31} = 2 Gbytes of virtual memory
- Size of page= 512 bytes (2^9)
- Nº page table entries per process= 2^{22}.

- The Page Table is stored in virtual memory and subject to paging...

(A) Two-Level Scheme for 32-bit Address

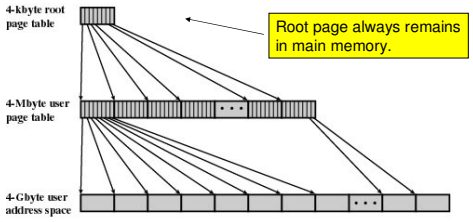


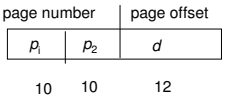
Figure 8.4 A Two-Level Hierarchical Page Table [JACO98a]

(A)- Two-Level Paging System

- 32-bit address: Address Space = 2^{32} (4 Gbytes)
- 4 Kbytes pages: 2^{12}
- 2^{20} pages
- If a page table entry used 4 bytes then the Page Table requires: 2^{22} bytes.
- This corresponds to 2^{10} pages...
- These 2^{10} pages can be kept in virtual memory and mapped by a Root Page Table with 2^{10} page table entries, occupying 4Kbytes (2^{12}) of main memory.

Two-Level Paging Example

- A logical address (on 32-bit machine with 4K page size) is divided into:
  - a page number consisting of 20 bits.
  - a page offset consisting of 12 bits.
- Since the page table is paged, the page number is further divided into:
  - a 10-bit page number.
  - a 10-bit page offset.
- Thus, a logical address is as follows:



- where p_1 is an index into the outer page table, and p_2 is the displacement within the page of the outer page table.

(A)- Address Translation: 2-level Paging System

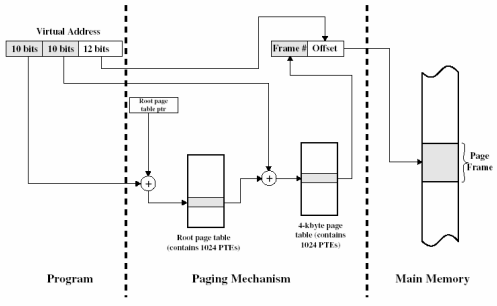
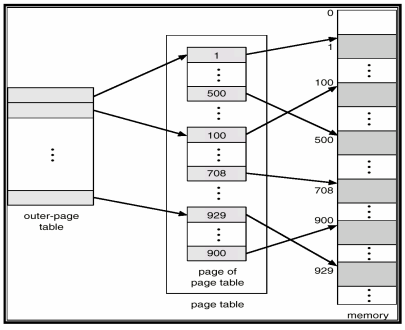


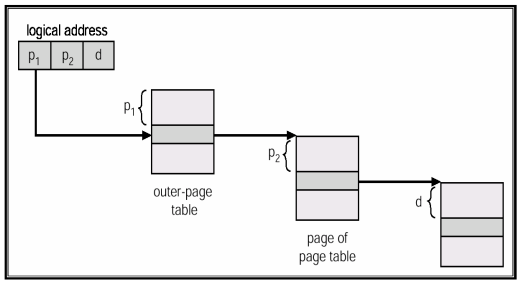
Figure 8.5 Address Translation in a Two-Level Paging System

(A)- Two-level Page-Table



(A)- Address Translation ...

- Address translation scheme for a two-level 32-bit paging architecture



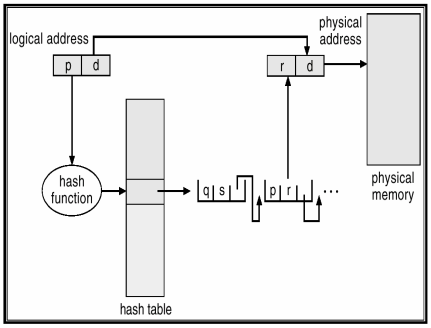
N-Level Paging

- 64-bit UltraSparc would require 7 levels of paging ... Not appropriate.
- Use other schemes (Hashed Page Tables; Inverted Page Table) and a TLB.

(B) Hashed Page Tables

- Common in address spaces > 32 bits.
- The virtual page number is hashed into a page table. This page table contains a chain of elements hashing to the same location.
- Virtual page numbers are compared in this chain searching for a match. If a match is found, the corresponding physical frame is extracted.

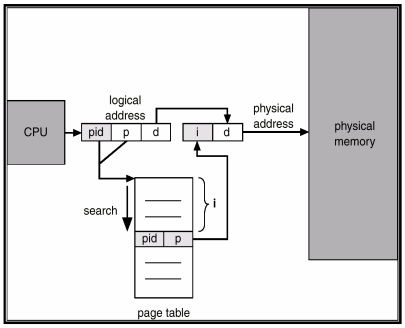
(B)-Hashed Page Table



(C) Inverted Page Table

- **One entry for each real page of memory**.
- Entry consists of the virtual address of the page stored in that real memory location, with information about the process that owns that page.
- Decreases memory needed to store each page table, but increases time needed to search the table when a page reference occurs.
- Use hash table to limit the search to one — or at most a few — page-table entries.

(C)-Inverted Page Table

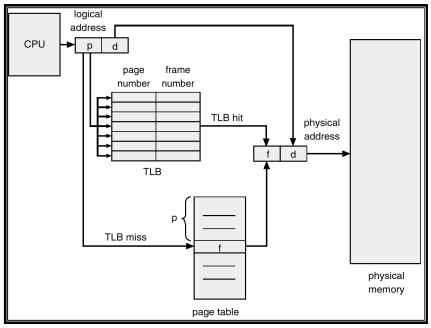


Translation Lookaside Buffer (TLB)

Translation Lookaside Buffer

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

Paging Hardware with a TLB



Translation Lookaside Buffer

- Contains page table entries that have been most recently used.
- Works in a same way as a memory cache.
- Usually has between 64 and 1024 entries.

Translation Lookaside Buffer

- Given a virtual address, the 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.

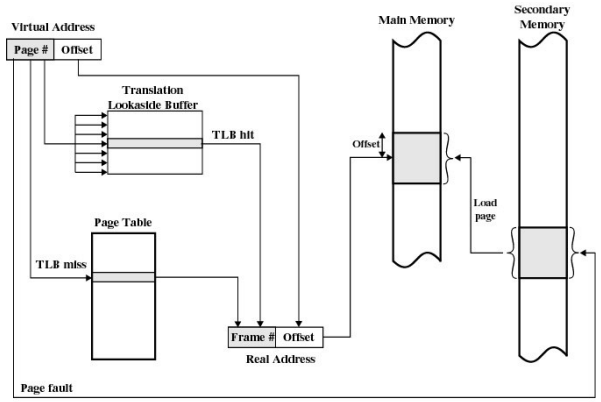


Figure 8.7 Use of a Translation Lookaside Buffer

TLB and the Principle of Locality

- By the **principle of locality** most virtual memory references will be to locations in recently used pages.
- Hopefully...

Design Decision: 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 → low number of page faults.
- Increased page size causes pages to contain locations further from any recent reference → rise in page faults.

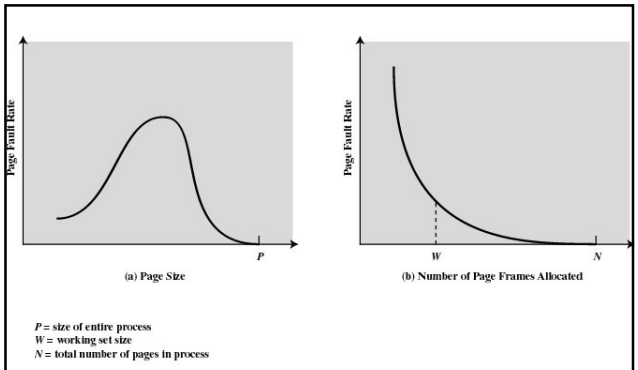


Figure 8.11 Typical Paging Behavior of a Program

Page Size

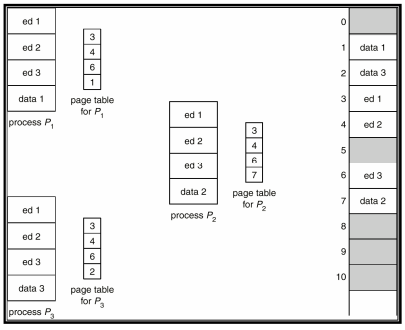
- Multiple page sizes provide the flexibility needed to effectively use a TLB.
- Large pages can be used for program instructions.
- Small pages can be used for threads.
- Most operating system support only one page size.

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 kbytes to 16 Mbytes
UltraSPARC	8 Kbytes to 4 Mbytes
Pentium	4 Kbytes or 4 Mbytes
PowerPc	4 Kbytes

Shared Pages



Segmentation

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.

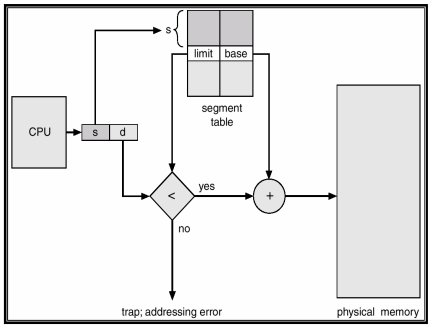
Addressing in Segmentation

- Logical address consists of a two tuple:
  <segment-number, offset>,
- **Segment table** – maps two-dimensional physical addresses; each table entry has:
  - **base** – contains the starting physical address where the segments reside in memory.
  - ***limit*** – specifies the length of the segment.

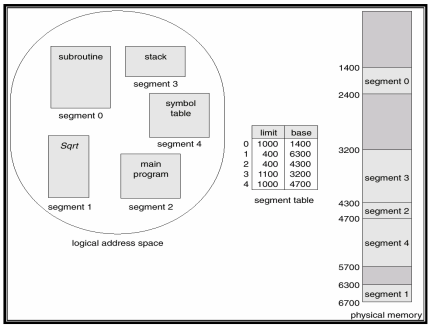
Segment Tables

- 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.

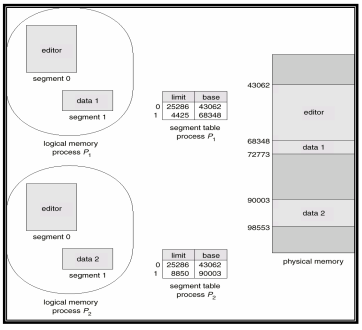
Segmentation Hardware



Example of Segmentation



Segmentation: Sharing



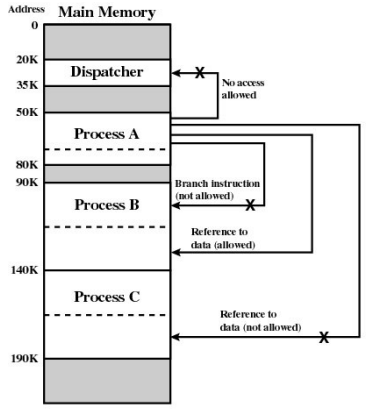


Figure 8.14 Protection Relationships Between Segments

Segment Table Entries

(b) Segmentation only

Figure 8.2 Typical Memory Management Formats

Address Translation in a Segmentation System

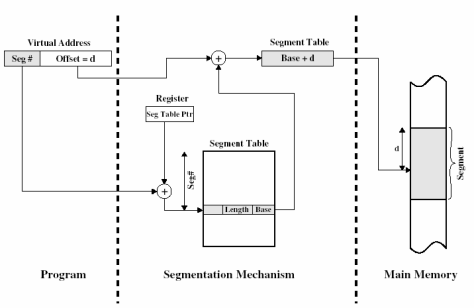


Figure 8.12 Address Translation in a Segmentation System

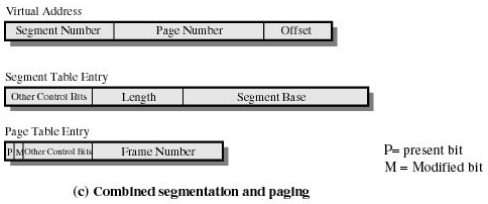
Paging and Segmentation

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.

- Hybrid scheme: each **segment** is broken into **fixed-size pages**.

Combined Segmentation and Paging



(c) Combined segmentation and paging

Figure 8.2 Typical Memory Management Formats

Address Translation in a Segmentation/Paging System

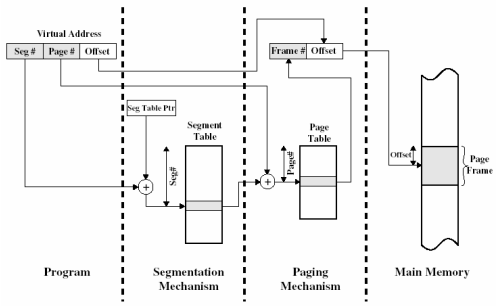
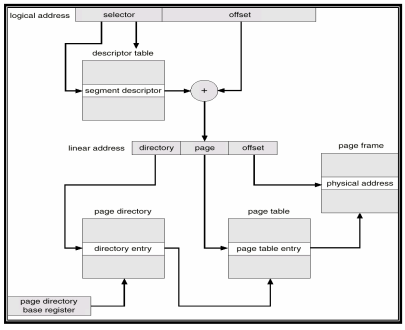


Figure 8.13 Address Translation in a Segmentation/Paging System

Intel 80386 Address Translation



- The Intel 386 uses segmentation with paging for memory management with a two-level paging scheme.