

**Module 9: Memory Management**

- Background
- Logical versus Physical Address Space
- Swapping
- Contiguous Allocation
- Paging
- Segmentation
- Segmentation with Paging

9.1 Silberschatz, Galvin, and Gagne 1999

---

---

---

---

---

---

---

---

**Background**

- Program must be brought into memory and placed within a process for it to be executed.
- *Input queue* – collection of processes on the disk that are waiting to be brought into memory for execution.
- User programs go through several steps before being executed.

9.2 Silberschatz, Galvin, and Gagne 1999

---

---

---

---

---

---

---

---

**Binding of Instructions and Data to Memory**

Address binding of instructions and data to memory addresses can happen at three different stages.

- **Compile time:** If memory location known a priori, absolute code can be generated; must recompile code if starting location changes. Example - locations on Stack (address = offset from Stack Point).
- **Load time:** Must generate *relocatable* code if memory location is not known at compile time.
- **Execution time:** Binding delayed until run time if the process can be moved during its execution from one memory segment to another. Need hardware support for address maps (e.g., *base* and *limit registers*).

9.3 Silberschatz, Galvin, and Gagne 1999

---

---

---

---

---

---

---

---



**Overlays**

- Keep in memory only those instructions and data that are needed at any given time.
- Needed when process is larger than amount of memory allocated to it.
- Implemented by user, no special support needed from operating system, programming design of overlay structure is complex

7

9.7      Silberschatz, Galvin, and Gagne 1999

---

---

---

---

---

---

---

---

**Logical vs. Physical Address Space**

- The concept of a logical *address space* that is bound to a separate *physical address space* is central to proper memory management.
  - *Logical address* – generated by the CPU; also referred to as *virtual address*.
  - *Physical address* – address seen by the memory unit.
- Logical and physical addresses are the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme.

8

9.8      Silberschatz, Galvin, and Gagne 1999

---

---

---

---

---

---

---

---

**Memory-Management Unit (MMU)**

- Hardware device that maps virtual to physical address.
- In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory.
- The user program deals with *logical* addresses; it never sees the *real* physical addresses.

9

9.9      Silberschatz, Galvin, and Gagne 1999

---

---

---

---

---

---

---

---

### Swapping

- A process can be *swapped* temporarily out of memory to a *backing store*, and then brought back into memory for continued execution.
- Backing store – fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images.
- *Roll out, roll in* – swapping variant used for priority-based scheduling algorithms; lower-priority process is swapped out so higher-priority process can be loaded and executed.
- Major part of swap time is transfer time; total transfer time is directly proportional to the *amount* of memory swapped.
- Modified versions of swapping are found on many systems, i.e., UNIX and Microsoft Windows.

10

9.10 Silberschatz, Galvin, and Gagne 1999

---

---

---

---

---

---

---

---

### Schematic View of Swapping

11

9.11 Silberschatz, Galvin, and Gagne 1999

---

---

---

---

---

---

---

---

### Contiguous Allocation

- Main memory usually into two partitions:
  - Resident operating system, usually held in low memory with interrupt vector.
  - User processes then held in high memory.
- Single-partition allocation
  - Relocation-register scheme used to protect user processes from each other, and from changing operating-system code and data.
  - Relocation register contains value of smallest physical address; limit register contains range of logical addresses – each logical address must be less than the limit register.

12

9.12 Silberschatz, Galvin, and Gagne 1999

---

---

---

---

---

---

---

---

### Contiguous Allocation (Cont.)

- Multiple-partition allocation
  - Hole* – block of available memory; holes of various size are scattered throughout memory.
  - When a process arrives, it is allocated memory from a hole large enough to accommodate it.
  - Operating system maintains information about:
    - allocated partitions
    - free partitions (hole)

13

9.13 Silberschatz, Galvin, and Gagne 1999

---

---

---

---

---

---

---

---

### Dynamic Storage-Allocation Problem

How to satisfy a request of size  $n$  from a list of free holes.

- First-fit:** Allocate the *first* hole that is big enough.
- Best-fit:** Allocate the *smallest* hole that is big enough; must search entire list, unless ordered by size. Produces the smallest leftover hole.
- Worst-fit:** Allocate the *largest* hole; must also search entire list. Produces the largest leftover hole.

First-fit and best-fit better than worst-fit in terms of speed and storage utilization.

14

9.14 Silberschatz, Galvin, and Gagne 1999

---

---

---

---

---

---

---

---

### Fragmentation

- External fragmentation – total memory space exists to satisfy a request, but it is not contiguous.
- Internal fragmentation – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used.
- Reduce external fragmentation by compaction
  - Shuffle memory contents to place all free memory together in one large block.
  - Compaction is possible *only* if relocation is dynamic, and is done at execution time.
  - I/O problem
    - \* Latch job in memory while it is involved in I/O.
    - \* Do I/O only into OS buffers.

15

9.15 Silberschatz, Galvin, and Gagne 1999

---

---

---

---

---

---

---

---

## Paging

- Logical address space of a process can be noncontiguous; process is allocated physical memory whenever the latter is available.
- Divide physical memory into fixed-sized blocks called frames (size is power of 2, between 512 bytes and 8192 bytes).
- Divide logical memory into blocks of same size called pages.
- Keep track of all free frames.
- To run a program of size  $n$  pages, need to find  $n$  free frames and load program.
- Set up a page table to translate logical to physical addresses.
- Internal fragmentation.

9.16

Silberschatz, Galvin, and Gagne 1999

16

---

---

---

---

---

---

---

---

## Address Translation Scheme

- Address generated by CPU is divided into:
  - Page number ( $p$ ) – used as an index into a page table which contains base address of each page in physical memory.
  - Page offset ( $d$ ) – combined with base address to define the physical memory address that is sent to the memory unit.

9.17

Silberschatz, Galvin, and Gagne 1999

17

---

---

---

---

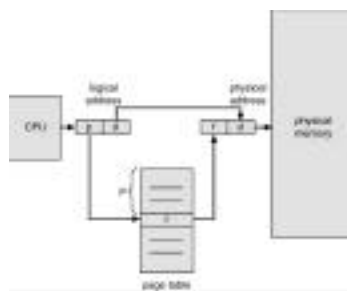
---

---

---

---

## Address Translation Architecture



9.18

Silberschatz, Galvin, and Gagne 1999

18

---

---

---

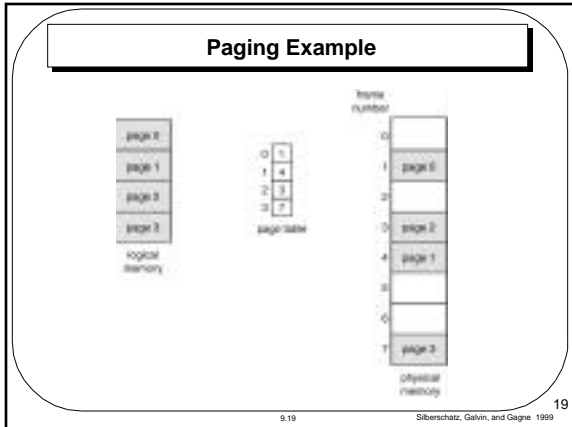
---

---

---

---

---




---

---

---

---

---

---

---

---

---

---

- ### Implementation of Page Table
- Page table is kept in main memory.
  - *Page-table base register (PTBR)* points to the page table.
  - *Page-table length register (PRLR)* indicates size of the page table.
  - In this scheme every data/instruction access requires two memory accesses. One for the page table and one for the data/instruction.
  - The two memory access problem can be solved by the use of a special fast-lookup hardware cache called *associative registers* or *translation look-aside buffers (TLBs)*
- 9.20 Silberschatz, Galvin, and Gagne 1999 20

---

---

---

---

---

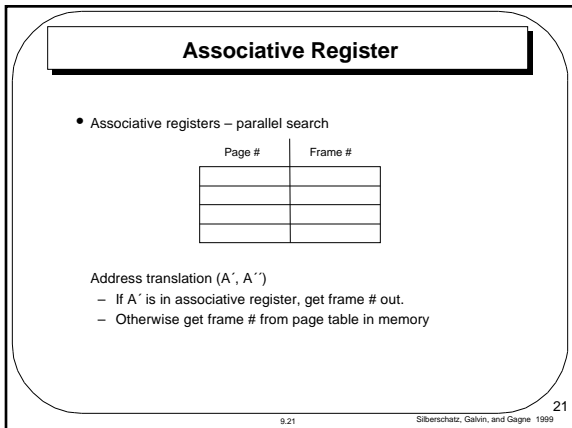
---

---

---

---

---




---

---

---

---

---

---

---

---

---

---

### Effective Access Time

- Associative Lookup = time unit
- Assume memory cycle time is 1 microsecond
- Hit ration – percentage of times that a page number is found in the associative registers; ration related to number of associative registers.
- Hit ratio =
- Effective Access Time (EAT)

$$EAT = (1 + ) + (2 + )(1 - )$$

$$= 2 + -$$

22

---

---

---

---

---

---

---

---

### Memory 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.

23

---

---

---

---

---

---

---

---

### Two-Level Page-Table Scheme

24

---

---

---

---

---

---

---

---



## 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.

9.28

Silberschatz, Galvin, and Gagne 1999

28

---

---

---

---

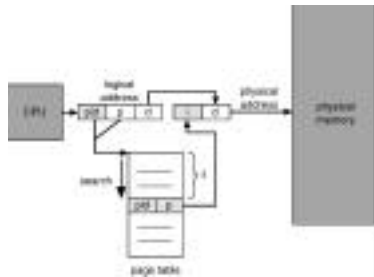
---

---

---

---

## Inverted Page Table Architecture



9.29

Silberschatz, Galvin, and Gagne 1999

29

---

---

---

---

---

---

---

---

## Shared Pages

- Shared code
  - One copy of read-only (reentrant) code shared among processes (i.e., text editors, compilers, window systems).
  - Shared code must appear in same location in the logical address space of all processes.
- Private code and data
  - Each process keeps a separate copy of the code and data.
  - The pages for the private code and data can appear anywhere in the logical address space.

9.30

Silberschatz, Galvin, and Gagne 1999

30

---

---

---

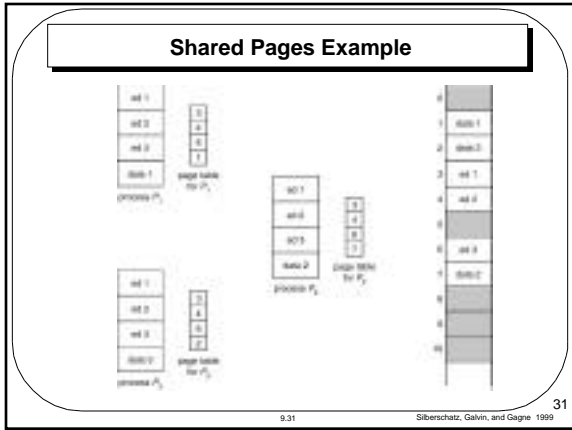
---

---

---

---

---




---

---

---

---

---

---

---

---

---

---

### Segmentation

- Memory-management scheme that supports user view of memory.
- A program is a collection of segments. A segment is a logical unit such as:
  - main program,
  - procedure,
  - function,
  - local variables, global variables,
  - common block,
  - stack,
  - symbol table, arrays

9.32 Silberschatz, Galvin, and Gagne 1999 32

---

---

---

---

---

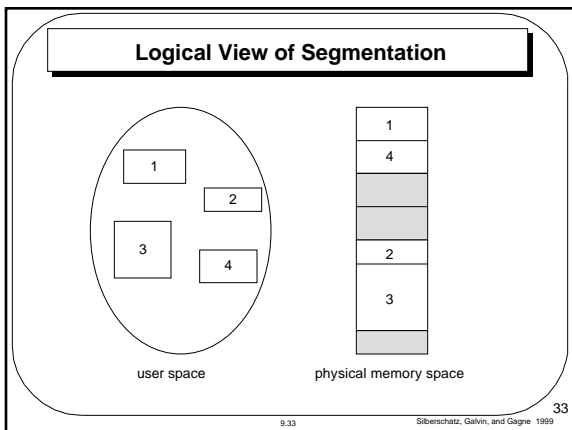
---

---

---

---

---




---

---

---

---

---

---

---

---

---

---

### Segmentation Architecture

- 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-table base register (STBR)* points to the segment table's location in memory.
- *Segment-table length register (STLR)* indicates number of segments used by a program;  
     segment number  $s$  is legal if  $s < STLR$ .

34

---

---

---

---

---

---

---

---

### Segmentation Architecture (Cont.)

- Relocation.
  - dynamic
  - by segment table
- Sharing.
  - shared segments
  - same segment number
- Allocation.
  - first fit/best fit
  - external fragmentation

35

---

---

---

---

---

---

---

---

### Segmentation Architecture (Cont.)

- Protection. With each entry in segment table associate:
  - validation bit = 0    illegal segment
  - read/write/execute privileges
- Protection bits associated with segments; code sharing occurs at segment level.
- Since segments vary in length, memory allocation is a dynamic storage-allocation problem.
- A segmentation example is shown in the following diagram

36

---

---

---

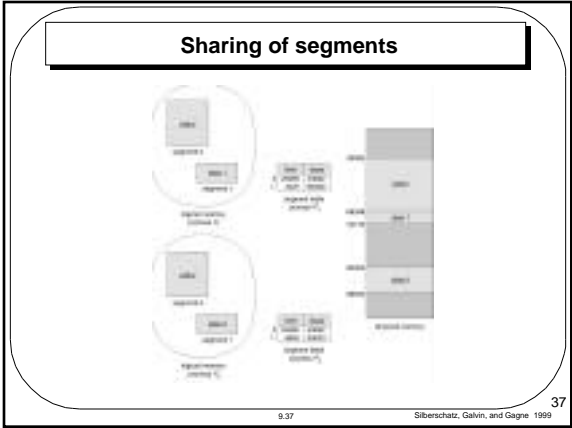
---

---

---

---

---




---

---

---

---

---

---

---

---

### Segmentation with Paging – MULTICS

- The MULTICS system solved problems of external fragmentation and lengthy search times by paging the segments.
- Solution differs from pure segmentation in that the segment-table entry contains not the base address of the segment, but rather the base address of a *page table* for this segment.

38

9.38 Silberschatz, Galvin, and Gagne 1999

---

---

---

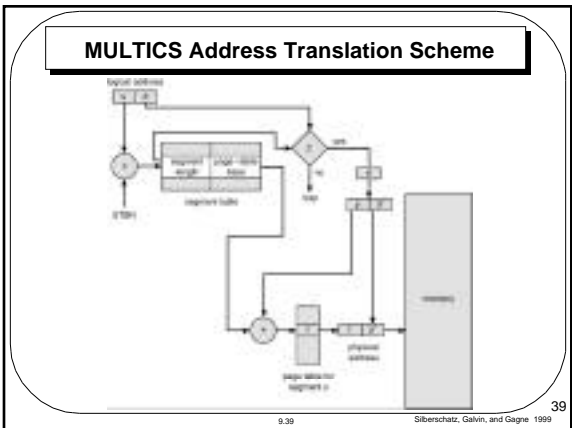
---

---

---

---

---




---

---

---

---

---

---

---

---

### Segmentation with Paging – Intel 386

- As shown in the following diagram, the Intel 386 uses segmentation with paging for memory management with a two-level paging scheme.

9.40

Silberschatz, Galvin, and Gagne 1999

40

---

---

---

---

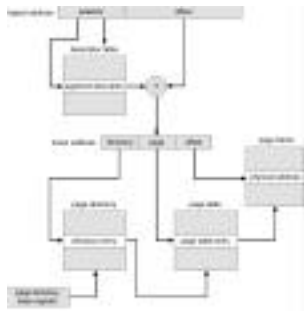
---

---

---

---

### Intel 30386 address translation



9.41

Silberschatz, Galvin, and Gagne 1999

41

---

---

---

---

---

---

---

---

### Comparing Memory-Management Strategies

- Hardware support
- Performance
- Fragmentation
- Relocation
- Swapping
- Sharing
- Protection

9.42

Silberschatz, Galvin, and Gagne 1999

42

---

---

---

---

---

---

---

---