OS View of Main Memory

- 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.

Binding of Instructions and Data

- Binding: A mapping from one address space to another
- Example:
  - Binding a symbolic address to a relocatable address
  - Binding a relocatable address to an absolute address
- 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; If starting address changes, need to recompile.
  - Load time: Compiler generates relocatable code if memory location is not known at compile time. Address assigned as loaded. If starting address

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.
- Virtual and physical addresses are the same in compile-time and load-time address-binding schemes.
- Virtual and physical addresses differ in execution-time address-binding scheme.

Memory-Management Unit (MMU)

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

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.
- 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
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.

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.

Dynamic Storage-Allocation Problem

- 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.

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.

Paging

- 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.
- Logical address space of a process can be noncontiguous; process is allocated physical memory whenever the latter is available.
- Keep track of all free frames.
- To run a program of size n pages, need to find n page frames and load memory.

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.
Address Translation Architecture

Paging Example

Implementation of Page Table

• Page table is kept in main memory.
• Page-table base register (PTBR) points to 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).

Associative Register or TLB

• Associative registers – parallel search

- Address translation (A', A'')
  - If A' is in associative register, get frame # out.
  - Otherwise get frame # from page table in memory

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.
• For multiple processes, associate a PID
  - ASN (address space number) on Alpha 21264

Alpha 21264 Data TLB
Two-Level Paging Example

- A logical address (on a 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:

Two-Level Page-Table Scheme

P₁ gets the entry in the outer-page table

P₂ is the displacement within the page selected by P₁

Inverted Page Table

- One entry for each physical 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.

Inverted Page Table Architecture

Virtual Memory

- 4Qs for VM?
  - Q1: Where can a block be placed in the upper level?
    Fully Associative, Set Associative, Direct Mapped
  - Q2: How is a block found if it is in the upper level?
    Tag/Block
  - Q3: Which block should be replaced on a miss?
    Random, LRU
  - Q4: What happens on a write?
Selecting a Page Size

- Reasons for larger page size
  - Page table size is inversely proportional to the page size; therefore memory saved
  - Fast cache hit time easy when cache page size (VA caches); bigger page makes it feasible as cache size grows
  - Transferring larger pages to or from secondary storage, possibly over a network, is more efficient
  - Number of TLB entries are restricted by clock cycle time, as a larger page size maps more memory, thereby reducing TLB misses

- Reasons for a smaller page size
  - Fragmentation: don’t waste storage; data must be contiguous within page
  - Quicker process start for small processes

- Hybrid solution: multiple page sizes