Chapter 8

- The main purpose of a computer system is to execute programs. These programs, together with the data they access, must be in ***main memory*** (at least partially) during execution.
- The CPU fetches instructions from memory according to the value of the ***program counter.***
- A typical instruction-execution cycle, first fetches an instruction from ***memory***. The instruction is then ***decoded*** and may cause operands to be ***fetched*** from ***memory***. After the instruction has been ***executed*** on the operands, results may be ***stored*** back in ***memory***.
- The base and limit registers can be loaded only by the operating system, which uses a special ***privileged*** instruction.
- ***Privileged*** instructions can be executed only in kernel mode, and since only the operating system executes in kernel mode, only the operating system can load the memory base and limit registers.
- An address generated by the CPU is commonly referred to as a ***logical*** address.
- The address loaded into the memory-address register of the memory is commonly referred to as a ***physical*** address,
- In ***compile-time*** and ***load-time*** address-binding schemes, logical and physical addresses are the ***same***.
- In ***execution-time*** address-binding scheme, logical (virtual) and physical addresses ***differ***.
- We can place the operating system in either low memory or high memory. The major factor affecting this decision is the location of the ***interrupt vector.***
- In the contiguous memory allocation, each process is contained in a ***single contiguous*** section of memory.
- The memory allocated to a process may be slightly larger than the requested memory. The difference between these two numbers is ***internal fragmentation***.
- ***Internal fragmentation*** refers to the memory that is internal to a partition but is not being used.
- ***External fragmentation*** exists when there is enough total memory space to satisfy a request, but the available spaces are not contiguous; storage is fragmented into a large number of small holes.
- A 32-bit page-table entry can point to one of 2^{32} physical page frames. If frame size is 4 KB, then a system with 4-byte entries can address ***2^{44}*** bytes (or ***16 TB***) of physical memory.
- Instructions to load or modify the page-table registers are ***privileged***, so that only the operating system can change the memory map.
- ***Segmentation*** is a memory-management scheme that supports this user view of memory.
- In ***segmentation*** scheme, a logical address consists of two parts: a ***segment number***, *s,* and an ***offset*** into that segment, *d.* The segment number is used as an index to the ***segment table.*** The offset *d* of the logical address must be between 0 and the ***segment limit.***
- In segmentation scheme, the user specifies each address by two quantities: a segment name and an offset. Contrast this scheme with the ***paging*** scheme, in which the user specifies only a single address, which is ***partitioned*** by the ***hardware*** into a page number and an offset, all invisible to the programmer.
- Systems with fixed-sized allocation units, such as the single-partition scheme and paging, suffer from ***internal*** fragmentation.

- Systems with variable-sized allocation units, such as the multiple-partition scheme and segmentation, suffer from ***external*** fragmentation.

Chapter 9

- ***Virtual*** memory is a technique that allows the execution of processes that are not completely in memory.
- Virtual address spaces that include holes are known as ***sparse*** address spaces.
- In addition to separating logical memory from physical memory, virtual memory also allows files and memory to be ***shared*** by two or more processes through page sharing.
- Consider how an executable program might be loaded from disk into memory.
    1. One option is to load the entire program in physical memory at program execution time.
    2. An alternative strategy is to initially load pages only as they are needed.
       This technique is known as ***demand paging*** and is commonly used in virtual memory systems.
- Virtual memory is commonly implemented by ***demand paging***.
- A demand-paging system is similar to a paging system with swapping where processes reside in secondary memory. When we want to execute a process, we swap it into memory. Rather than swapping the entire process into memory, however, we use a ***lazy swapper.***
- lazy swapper ***never*** swaps a page into memory unless that page will be needed.
- Since we are now viewing a process as a sequence of ***pages***, rather than as one large ***contiguous*** address space.
- A ***swapper*** manipulates ***entire*** processes, whereas a ***pager*** is concerned with the individual ***pages*** of a process.
- What happens if the process tries to access a page that was not brought into memory?
    1. Access to a page marked invalid causes a ***page-fault trap***.
    2. The paging hardware, in ***translating*** the address through the ***page table***, will notice that the ***invalid*** bit is ***set,*** causing a trap to the operating system.
- The ***effective access time*** is directly ***proportional*** to the ***page-fault rate***.
- It is important to keep the page-fault rate ***low*** in a demand-paging system. Otherwise, the effective access time increases, ***slowing*** process execution dramatically.
- Disk I/O to ***swap*** space is generally ***faster*** than that to the ***file system***, because swap space is allocated in much larger blocks, and file lookups and indirect allocation methods are not used .
- The system can therefore gain better paging throughput by
    1. copying an entire file image into the swap space at process ***startup*** and then performing demand paging from the swap space.
    2. to read demand pages from the file system initially but to write the pages to swap space as they are replaced. This approach will ensure that only ***needed*** pages are read from the file system but that all subsequent paging is done from swap space.
- We evaluate an algorithm by running it on a particular string of memory references and computing the number of page faults. The string of memory references is called a ***reference*** string.
- The key distinction between the FIFO and OPT algorithms is that the FIFO algorithm uses the time when a page was ***brought into*** memory, whereas the OPT algorithm uses the time when a

page is to be ***used***.

- If we use the recent past as an approximation of the near future, then we can replace the page that ***has not been used*** for the ***longest*** period of time. This approach is the ***least-recently-used (LRU)*** algorithm**.**
- The ***reference*** bit for a page is set by the ***hardware*** whenever that page is referenced (either a read or a write to any byte in the page).
- Reference bits are associated with each ***entry*** in the page table.
- Second-chance replacement degenerates to ***FIFO*** replacement if all bits are set.
- This high paging activity is called ***thrashing***.
- A process is ***thrashing*** if it is spending more time paging than executing.
- A ***locality*** is a set of pages that are actively used together, which is defined by the program structure and its data structures.
- If the total demand for frames is greater than the total number of available frames, ***thrashing*** will occur, because some processes will not have enough frames.
- the ***working-set model*** is based on the assumption of ***locality***， which are defined by the program structure and its data structures.
- Thrashing has a ***high*** page-fault rate.
- If a process does not have enough memory for its working set, it will ***thrash***.
- ***Memory mapping*** a file allows a part of the virtual address space to be logically associated with the file.
- Memory mapping a file is accomplished by mapping a disk block to a ***page*** in memory.
- To provide memory-mapped I/O, ranges of memory addresses are set aside and are mapped to the device ***registers*****,** called an I/O port**.**
- The ***buddy*** system allocates memory from a fixed-size segment consisting of physically contiguous pages by using a ***power-of-2*** allocator, which satisfies requests in units sized as a power of 2.
- For ***slab*** allocation,
    1. A slab is made up of one or more physically contiguous pages.
    2. A ***cache*** consists of one or more ***slabs***.
    3. There is a single cache for each unique kernel data structure.
- The buddy system allocates memory to kernel processes in units sized according to a power of ***2***, which often results in ***fragmentation***.
- Slab allocators assign kernel data structures to caches associated with slabs, which are made up of one or more physically ***contiguous*** pages. With slab allocation, no memory is wasted due to ***fragmentation***,
- To minimize internal fragmentation, we need a ***small*** page size.
- A desire to minimize I/O time argues for a ***larger*** page size.
- A ***smaller*** page size allows each page to match program locality more accurately.
- To minimize the number of page faults, we need to have a ***large*** page size.
- The TLB reach refers to the amount of memory accessible from the TLB and is simply the number of entries multiplied by the ***page size***.