Segmentation: Non-Contiguous Memory Allocation

Programmers view memory as a collection of variable-sized segments, with no necessary ordering among the segments. When writing a program, a programmer thinks of it as a main program with a set of methods, procedures, or functions. Programs are logical entities and program size varies, not necessarily in power of 2 as shown in Figure-1.

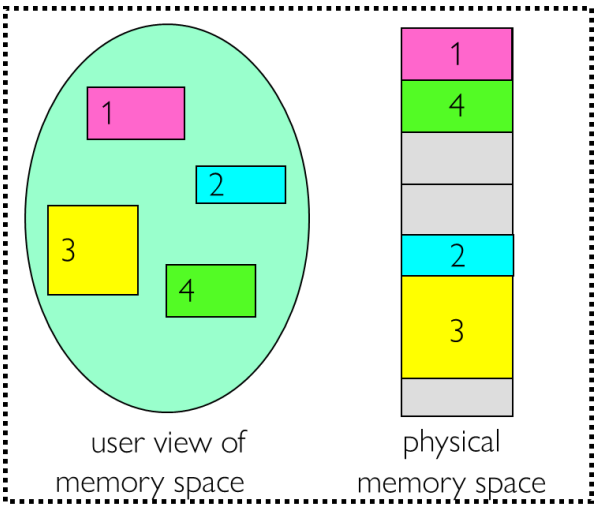


Figure-1: User's view of logical and physical address space

To facilitate programs sharing, and placement in memory the segmentation technique of memory management is supported by certain computer systems. Segmentation is an extension of variable partitions allocation technique, where more than one partitions (of variable size) is allocated to a process. Segmentation is a technique to split program into logical pieces where each piece represents a group of related information. These logical pieces (of variable sizes) are called segments. In variable partitions, a list of free partitions is maintained for allocation, in segmentation also a list of partitions is maintained. Segments are placed in partitions, where each segment is placed in a separate partition. A segment table is maintained which is indexed on segment-number and have based address and size of each segment. Each entry in the segment table has a segment base and a segment limit. Logical address space is split into segment and displacement within segment. The segment base contains the starting physical address of the partition the segment is placed, and the segment limit specifies the length/size of the segment. Logical to physical address translation using segment table is illustrated in Figure-2.

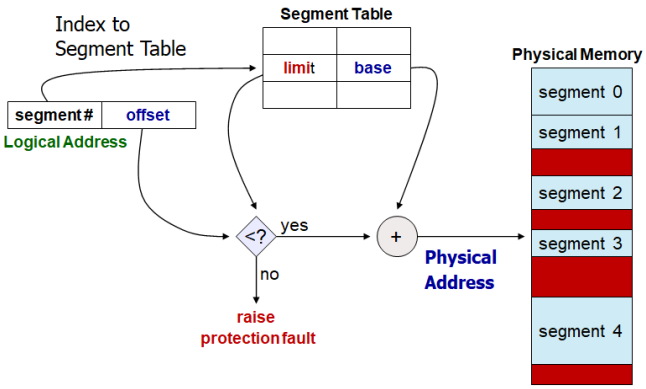


Figure-2: Logical to Physical address translation using segment table

Since address translation is performed by interpreting address space bits, maximum size of a segment is fixed at system generation time. As described at the beginning of the section, segmentation is an extension of variable partitions (one program, a pair of base and limit registers used). In segmentation a program is divided into segments, and for each segment, an entry of base-limit values in segment table is maintained. In variable partitions, address translation is performed on program (base and displacement) and in segmentation address translation is performed on segment (base and displacement of a segment in segment table). Just like in variable partitions, different sizes of partitions are available at any given time, different allocation approaches (First/Next/Best/Worst fit) are implemented. In variable partitions, external fragmentation occurs, likewise in segmentation, external fragmentation occurs. To place a segment, in such a situation, system may perform compaction. We can describe logical to physical address translation using S, D bits and segment table through a simple example of a process with two segments (segment number 0 and 1) as shown in Figure-3.

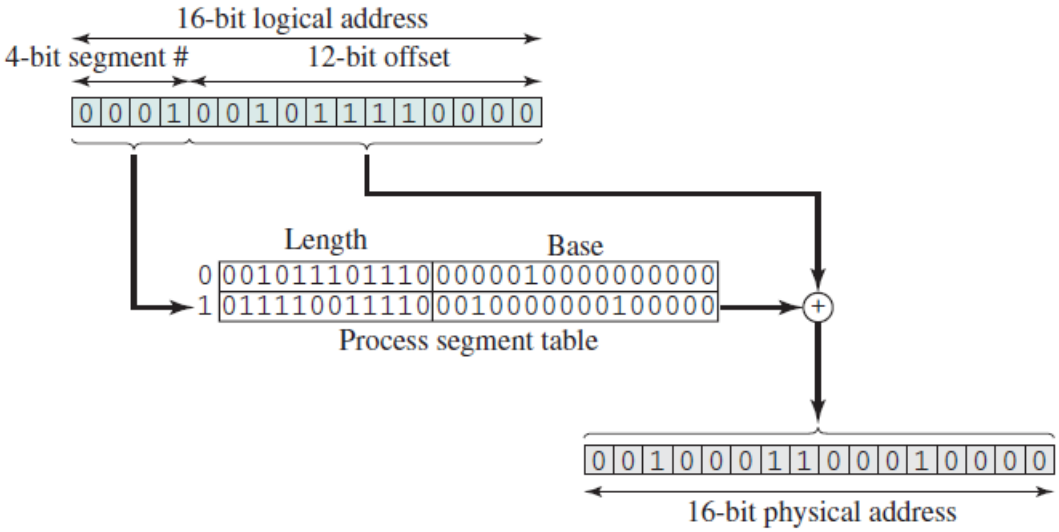


Figure-3: Logical to Physical address translation manipulating S and D bits

Logical address points towards segment-1's entry in segment table, the offset bits are added to the base bits and physical address is computed. Sharing logical units of program is easier, only shared segments base and limit register values will be copied in segment tables of all those processes sharing the segments. Like paging, protection bit is added in segment table to make sure that memory area is protected.

Multi-Level Paging

Most modern computer systems support a large logical address space (2^{32} to 2^{64}) and the page table itself becomes excessively large. For example, consider a system with a 32-bit logical address space. If the page size in such a system is 4KB (2^{12}), then a page table may consist of up to 1 million entries (2^{20}). If each entry consists of 4 bytes, each process may need up to 4MB of physical address space for the page table alone. One simple solution to this problem is to divide the page table into smaller pieces.

One way is to use a two-level paging algorithm, in which the page table itself is also paged as shown in Figure-4. For example, consider again the system with a 32-bit logical address space and a page size of 4KB. A logical address is divided into a page number consisting of 20 bits and a page offset consisting of 12 bits. Because we page the page table,

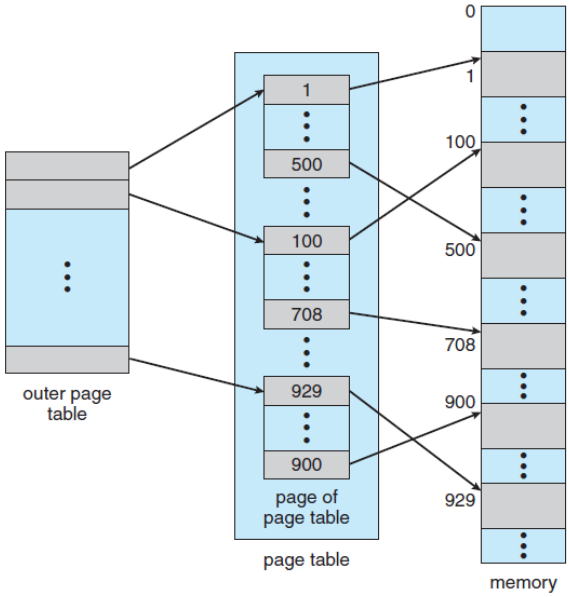
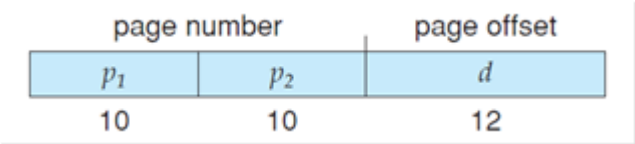


Figure-4: Two level page table

the page number is further divided into a 10-bit page number and 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 inner page table. The address-translation method for this architecture is shown in Figure-5. Because address translation works from the outer page table inward, this scheme is also known as a forward-mapped page table.

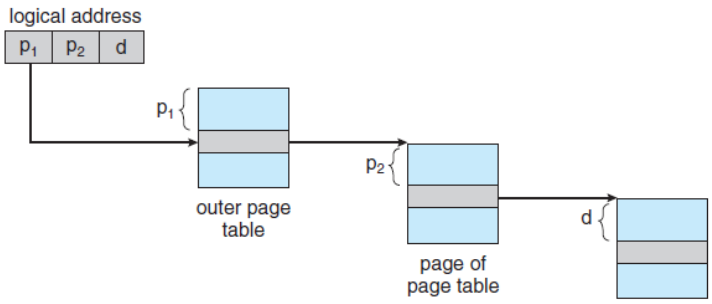
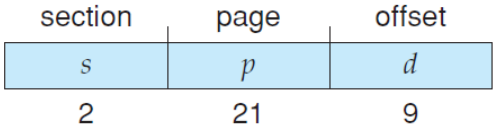


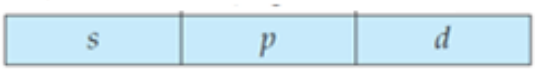
Figure-5: Address translation of two-level paging scheme

The VAX architecture supported a variation of two-level paging. The VAX is a 32-bit machine with a page size of 512 bytes. The logical address space of a process is divided into four equal sections, each of which consists of 230 bytes. Each section represents a different part of the logical address space of a process. The first 2 high-order bits of the logical address designate the appropriate section. The next 21 bits represent the logical page number of that section, and the final 9 bits represent an offset in the desired page. By partitioning the page table in this manner, the operating system can leave partitions unused until a process needs them. Entire sections of virtual address space are frequently unused, and multilevel page tables have no entries for these spaces, greatly decreasing the amount of memory needed to store virtual memory data structures.
An address on the VAX architecture is as follows:



where s designates the section number, p is an index into the page table, and d is the displacement within the page. Even when this scheme is used, the size of a one-level page table for a VAX process using one section is 221 bits ∗ 4 bytes per entry = 8 MB. To further reduce main-memory use, the VAX pages the user-process page tables.

Combining Segmentation and Paging
In main frame systems, to utilize memory in a better way by reducing internal fragmentation page size was od small size. For sharing code segments were shares. In such systems, the concept of paged-segmentation was used. Memory is divided into fixed size blocks (frames) and logical address space is divide into segments. Rather than placing a segment contiguously, segment is placed in different frames and page table for each segment is maintained. Logical address is as follows:



Where s is segment number, p is page number within segment, and d is displacement within page.

For larger address space, it is possible to combine segmentation and paging. Virtual address is consisting of segments, which is further split into pages. By using this two-level addressing, for a process it is possible to have small number of page table entries in memory.

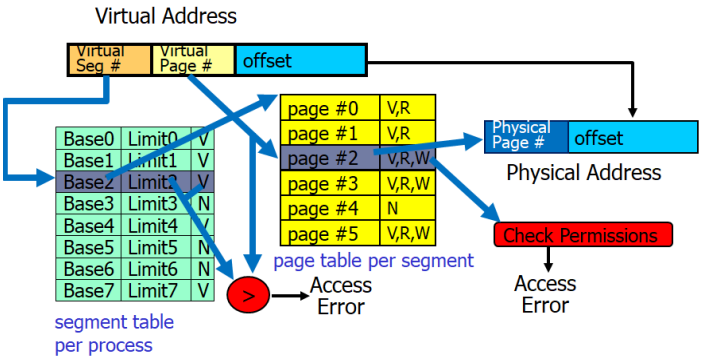


Figure-6: Virtual address to physical address translation

Virtual address space and address translation is depicted in Figure-6. Sharing of segments in this scheme is simple and easy.