Consideration	Paging	Segmentation
Need the programmer be aware that this technique is being used?	No	Yes
How many linear address spaces are there?	1	Many
Can the total address space exceed the size of physical memory?	Yes	Yes
Can procedures and data be distinguished and separately protected?	No	Yes
Can tables whose size fluctuates be accommodated easily?	No	Yes
Is sharing of procedures between users facilitated?	No	Yes
Why was this technique invented?	To get a large linear address space without having to buy more physical memory	To allow programs and data to be broken up into logically independent address spaces and to aid sharing and protection

Figure 4-23. Comparison of paging and segmentation.

of a segmented memory has the illusion that all segments are in main memory all the time—that is, he can address them as though they were—he can protect each segment separately, without having to be concerned with the administration of overlaying them.

4.6.1 Implementation of Pure Segmentation

The implementation of segmentation differs from paging in an essential way: pages are fixed size and segments are not. Figure 4-24(a) shows an example of physical memory initially containing five segments. Now consider what happens if segment 1 is evicted and segment 7, which is smaller, is put in its place. We arrive at the memory configuration of Fig. 4-24(b). Between segment 7 and segment 2 is an unused area—that is, a hole. Then segment 4 is replaced by segment 5, as in Fig. 4-24(c), and segment 3 is replaced by segment 6, as in Fig. 4-24(d). After the system has been running for a while, memory will be divided up into a number of chunks, some containing segments and some containing holes. This phenomenon, called **checkerboarding** or **external fragmentation**, wastes memory in the holes. It can be dealt with by compaction, as shown in Fig. 4-24(e).

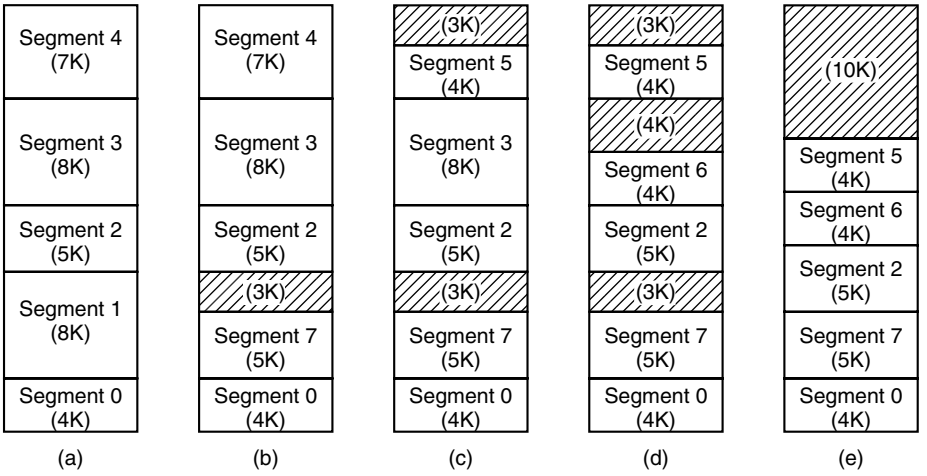

Figure 4-24. (a)-(d) Development of checkerboarding. (e) Removal of the checkerboarding by compaction.

4.6.2 Segmentation with Paging: The Intel Pentium

The Pentium supports up to 16K segments, each with up to 2^{32} bytes of virtual address space. The Pentium can be set up (by the operating system) to use only segmentation, only paging, or both. Most operating systems, including Windows XP and all flavors of UNIX, use the pure paging model, in which each process has a single segment of 2^{32} bytes. Since the Pentium is capable of providing processes with a much larger address space, and one operating system (OS/2) did actually use the full power of the addressing, we will describe how Pentium virtual memory works in its full generality.

The heart of the Pentium virtual memory consists of two tables, the **LDT** (**Local Descriptor Table**) and the **GDT** (**Global Descriptor Table**). Each program has its own LDT, but there is a single GDT, shared by all the programs on the computer. The LDT describes segments local to each program, including its code, data, stack, and so on, whereas the GDT describes system segments, including the operating system itself.

To access a segment, a Pentium program first loads a selector for that segment into one of the machine's six segment registers. During execution, the CS register holds the selector for the code segment and the DS register holds the selector for the data segment. The other segment registers are less important. Each selector is a 16-bit number, as shown in Fig. 4-25.

One of the selector bits tells whether the segment is local or global (i.e., whether it is in the LDT or GDT). Thirteen other bits specify the LDT or GDT entry number; thus tables are each restricted to holding 8K segment descriptors.