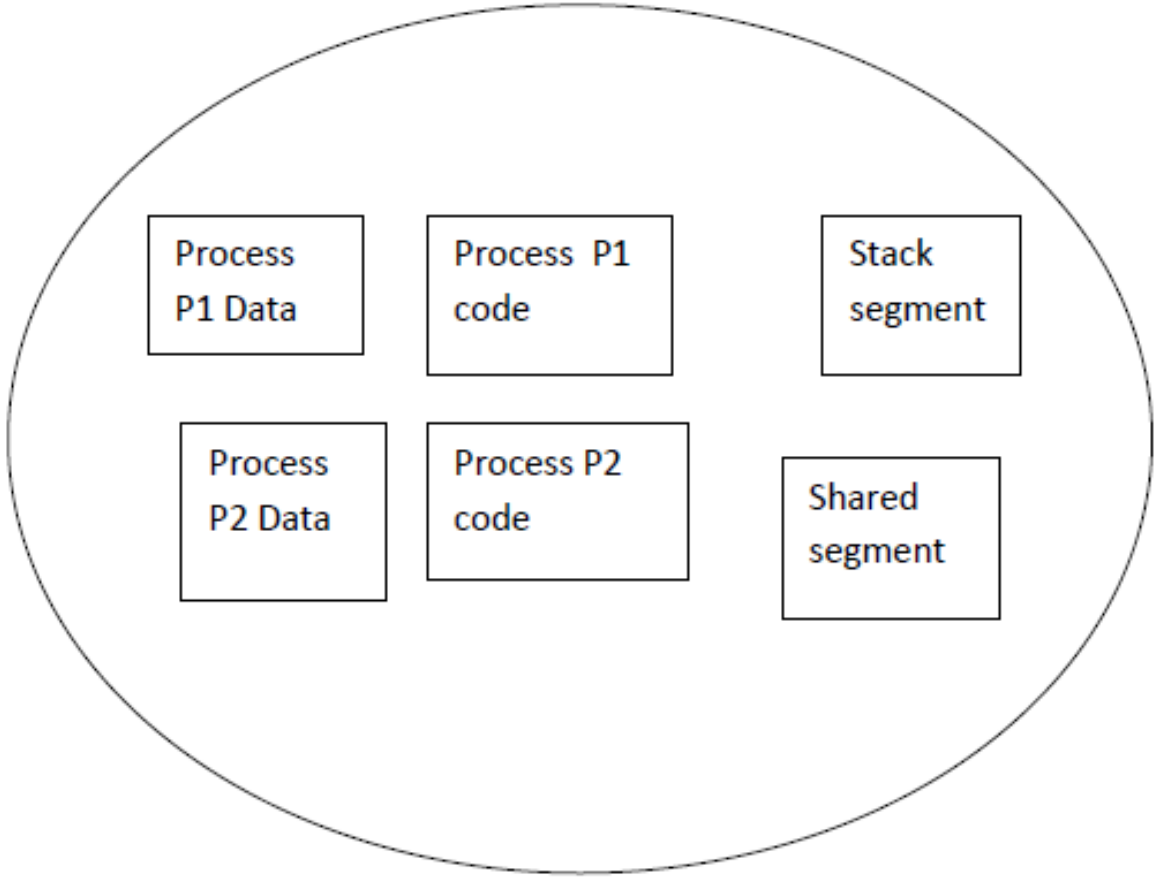
Segmentation Memory Management

Why Segmentation?

- A program can be visualized as a collection of methods, procedures or functions.
- It may also include various data structures such as objects, arrays, stacks, variables etc.
- Users view the program as a collection of modules and data.
- The extent of external and internal fragmentation and their negative impact on wasted memory should be reduced in systems where the average size of a request for allocation memory to segments is small.
- Segmentation is a way to reduce the average size of a request for memory by dividing the process's address space into blocks that may be placed into noncontiguous areas of memory.

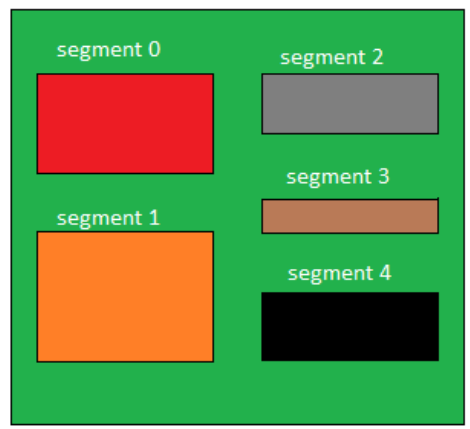
User’s View of a Program



- It is a technique to break memory into logical pieces where each piece represents a group of related information.
- These related information are called segments and are formed at program translation.
- The segments may be data segments, code segments, shared segments and stack segments.
- These segments may be placed in separate noncontiguous areas of physical memory, but the items belonging to a single segment must be placed in contiguous areas of physical memory.
- Thus segmentation possesses some properties of both contiguous (individual segments) and noncontiguous (address space of a process) schemes for memory management.

- These segments are of varying size and thus eliminates internal fragmentation.
- The elements within a segment are identified by their offset from the beginning of the segment. External fragmentation still exists but to lesser extent.

Logical View of Segmentation

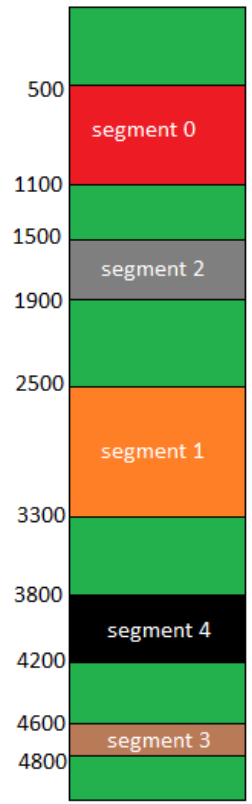


Logical Address Space

Segment Number

	base address	Limit
0	500	600
1	2500	800
2	1500	400
3	4600	200
4	3800	400

Segment Table



Physical Address Space

- Relocation.
  - dynamic
  - by segment table
- Allocation.
  - first fit/best fit
  - external fragmentation

- For relocation purposes, each segment is compiled to begin at its own virtual address 0.
- An individual item within a segment is then identifiable by its offset relative to the beginning of the enclosing segment.
- The unique designation of an item in a segmented address space requires the specification of both its segment and the relative offset.
- Address in segmented systems have two components:

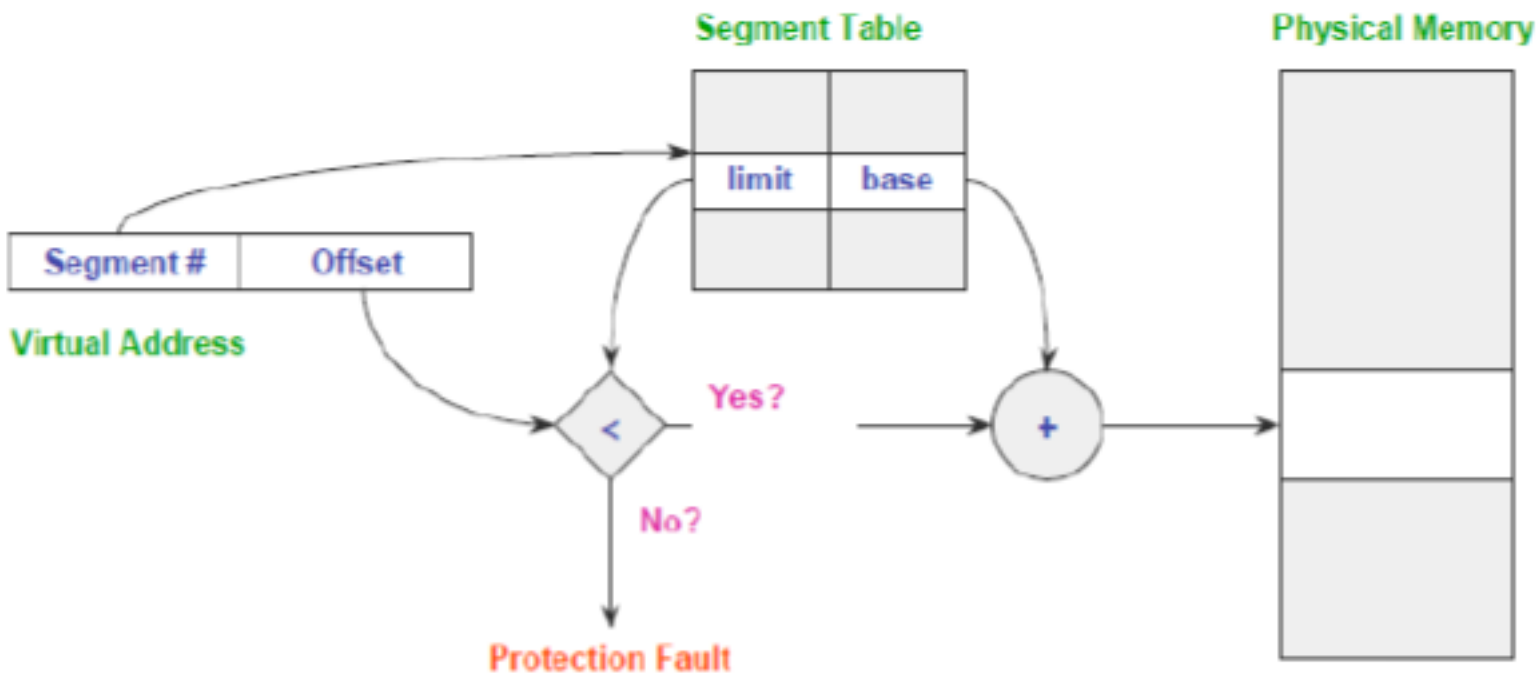
  Virtual Address : Segment name (number) and Offset within the segment.

  Segment number - used to index the SDT and to obtain the physical base address.

  Offset- used to produce physical address by adding with the base value

- When a segmented process is to be loaded onto the memory, OS attempts to allocate memory for all the segments of the process.
- It may create separate partitions to suit the needs of each segment and allocation of partition to segments is similar to dynamic partitioning.
- In segmented systems, an address translation scheme is used to convert a two dimensional virtual segment address into its single dimensional physical equivalent.
- The base (obtained during partition creation) and size (specified in the load module) of loaded segment are recorded as a tuple called the segment descriptor.

- All segment descriptors of a given process are collected in a table called the Segment Descriptor Table (SDT).
- The size of a SDT is related to the size of the virtual address space of a process.
- SDT is treated as a special type of segment. Two registers are used to access the SDT.
  - Segment Descriptor table base register (SDTBR) - points to the base of the running process's SDT
  - Segment Descriptor table limit register (SDTLR)- provided to mark the end of the SDT pointed to by the SDTBR.


Segment Table
Physical Memory
limit
base
Segment #
Offset
Virtual Address
<
Yes?
+
No?
Protection Fault

- Segmentation is multiple base limit version of dynamically partition memory. The price paid for segmenting the address space of a process is the overhead of storing and accessing SDTs.
- Mapping each virtual address requires two physical memory references for a single virtual (Program) reference.
  - memory reference to access the segment description in the SDT.
  - memory reference to access the target item in physical memory.

Segment Descriptor Caching:

- Hardware accelerators are used to speed the translation. Most frequently used segment descriptors are stored in registers.

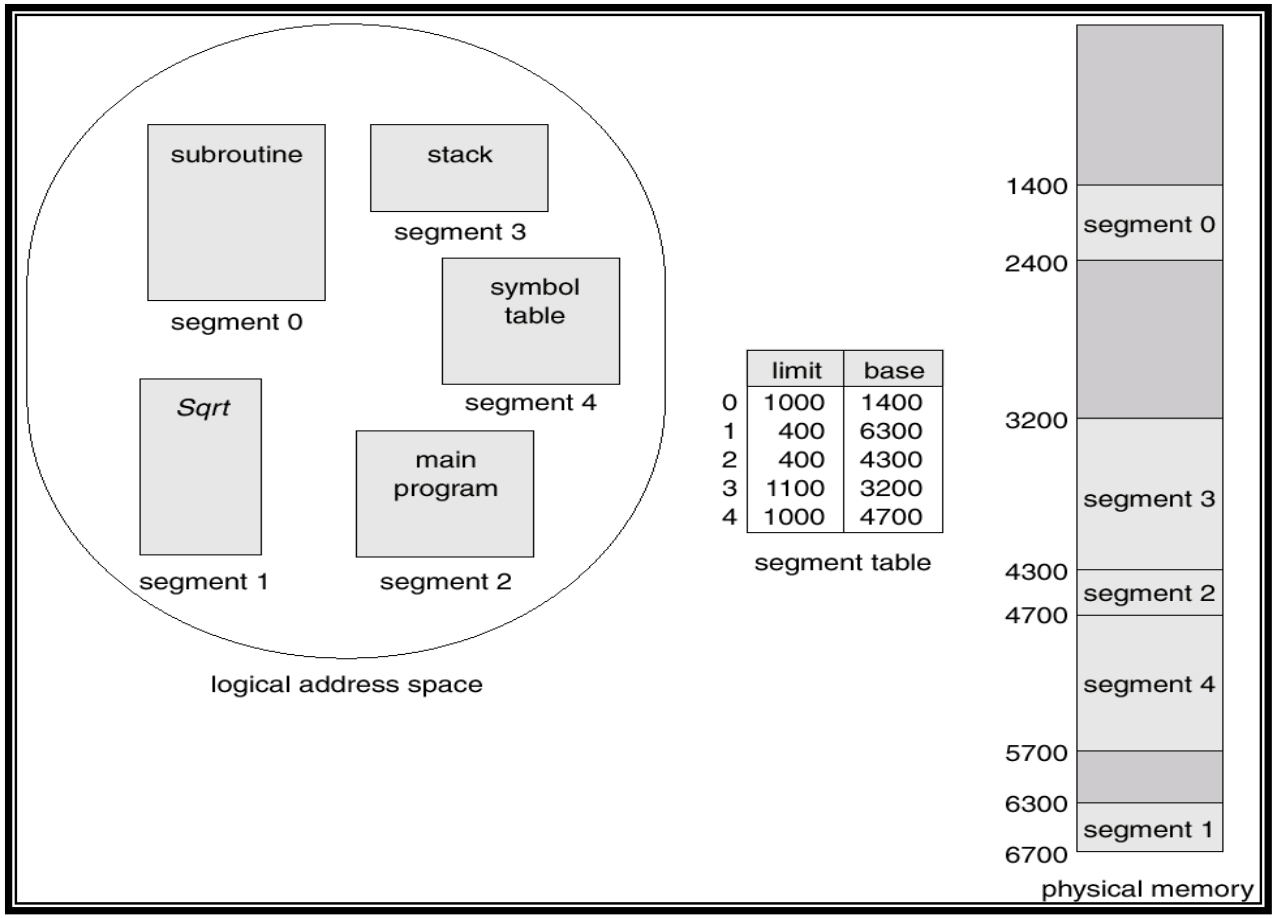
- Protection

The base limit form of protection is used. To provide protection within the address space of a single process, access rights such as read only, write only, execute only can be used depending on the information stored in the segments.

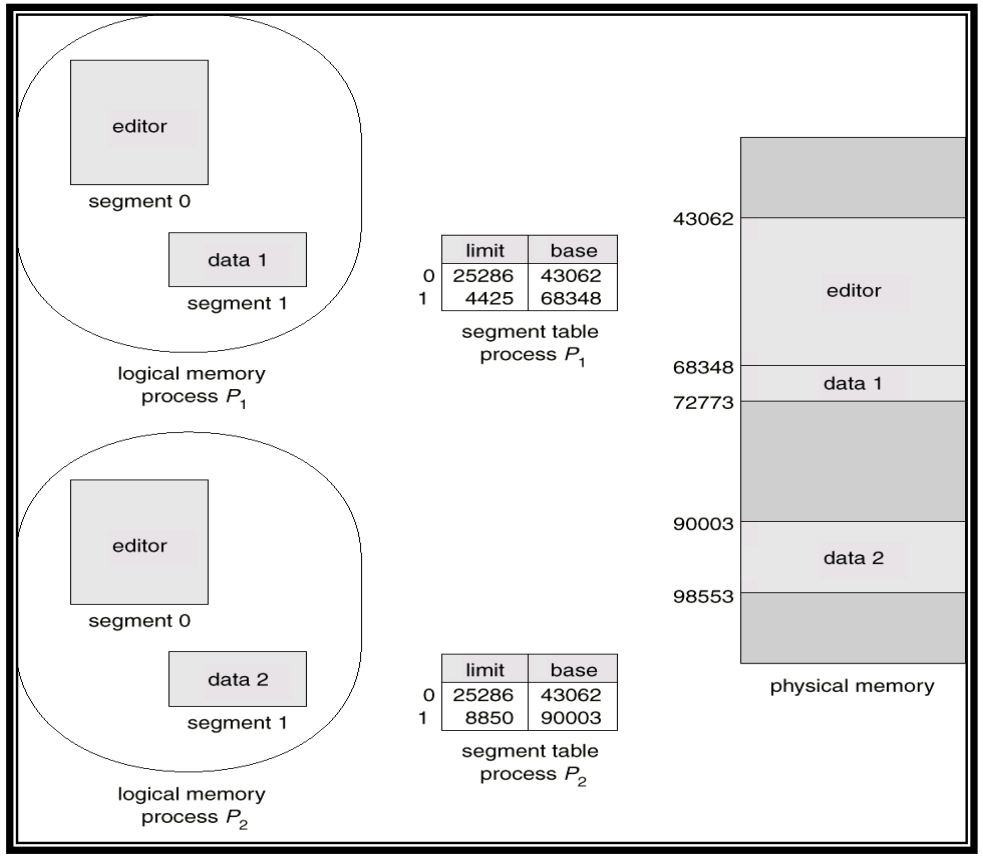
- Sharing

A shared segment may be mapped via the appropriate SDTs to the virtual address spaces of all processes that are authorized to reference it.

Example of Segmentation



Sharing of Segments



- Advantages:
  - Elimination of internal fragmentation
  - Support for dynamic growth of segments
  - Protection and sharing of segments
  - Modular program development
  - Dynamic linking and loading
- Disadvantages:
  - More complex strategies for compaction need to be employed.
  - The two step translation of virtual addresses to physical addresses has to be supported by dedicate hardware to avoid a drastic reduction in the effective memory bandwidth.
  - No single segment may be larger than the available physical memory.
  - Large data structures are split into several segments which results in run time overhead.