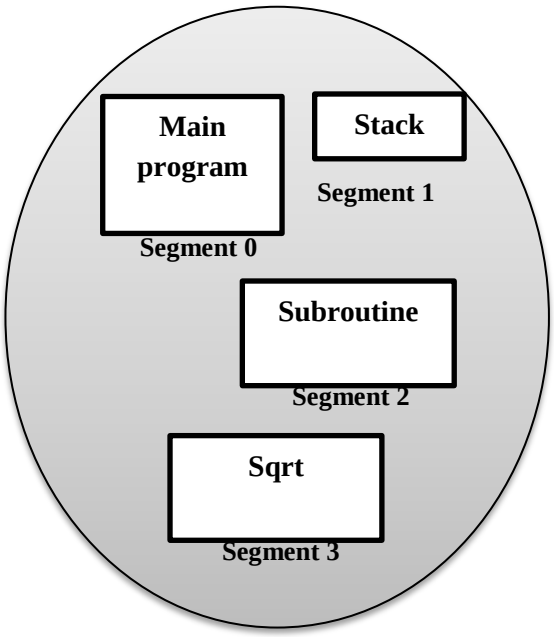
SEGMENTATION

Segmentation is a **memory management technique** that supports this user or programmer view of memory. A Logical address space is a collection of segments. Each segment has name and length.

Each Logical address consists of two tuple:

<segment-number, offset>



Logical Address Space

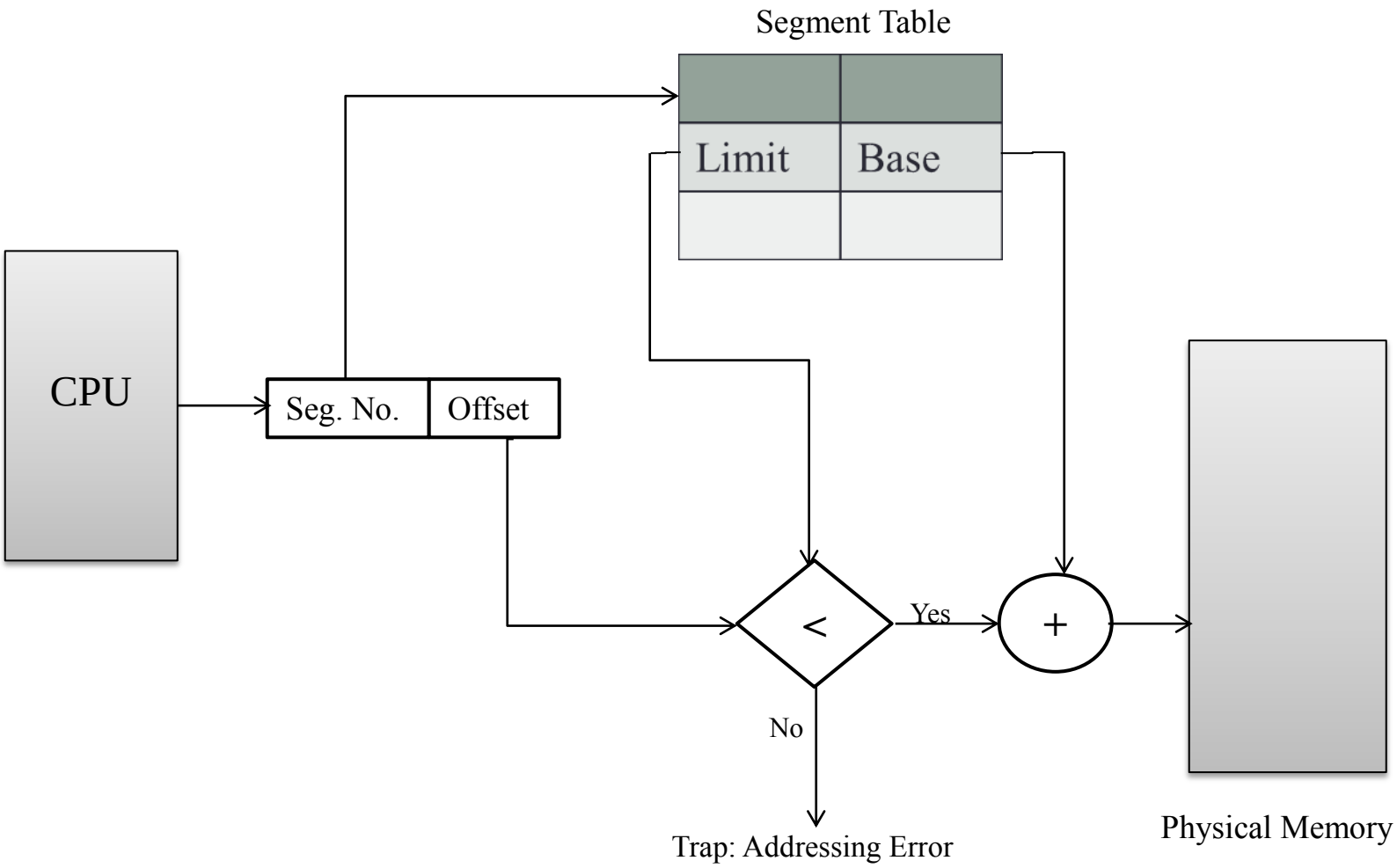
When the program is complied, the compiler automatically constructs the segments reflecting the input program.

A C Compiler might create separate segments for the following

1. The code.
2. Global variables.
3. The heap, from which memory is allocated.
4. The stacks.
5. The Standard C Library.

Libraries that are linked in during compile time might be assigned separate segments.

The loader would take all these segments and assign them segment numbers.



Hardware

- ✓ The mapping of two-dimensional programmer defined addresses into one-dimensional physical address is done by **Segment Table**.
- ✓ Each entry in the segment table has a **segment base** and **segment limit**.
- ✓ Segment base contains the starting physical address, where it resides in the memory.
- ✓ Segment limit specifies the length of the segment

Following steps are followed to translate or map logical address into physical address:

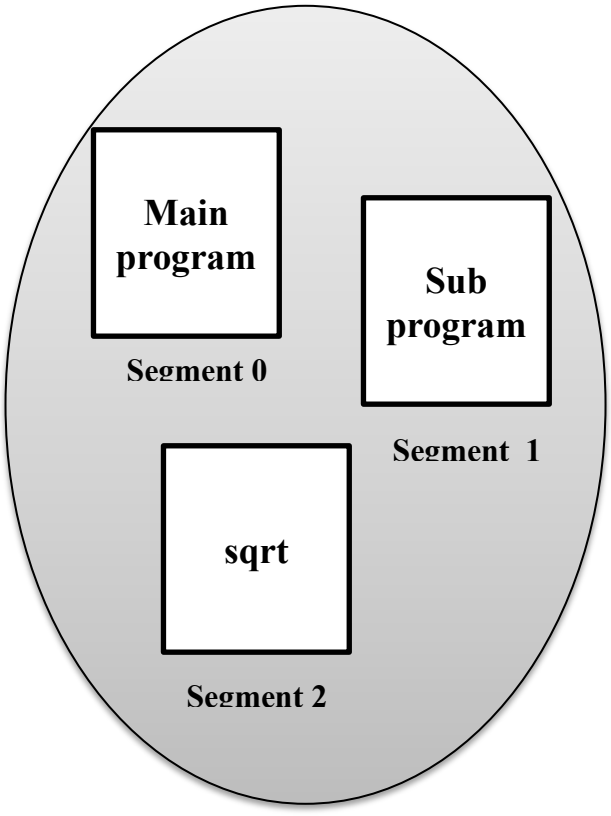
1. CPU generate the Logical address consists of two parts: Segment number and offset.
2. For the generated segment number, corresponding entry is located in the segment table.
3. segment offset is compared with the limit (size) of the segment.
   - ✓ **Case-01: offset >= Limit**

     If segment offset is found to be greater than or equal to the limit, a trap is generated.

   - ✓ **Case-02: offset < Limit**

     If segment offset is found to be smaller than the limit, then segment offset is added with the base address of the segment.

Example of Segmentation



Logical Address Space

Segment Table

	Limit	Base
Segment 0	100	1400
Segment 1	200	4000
Segment 2	1100	2000