Memory Allocation inside CPU

■ Code Segment

- The code segment, also referred as the text segment, is the area of memory which contains the frequently executed code.

- The code segment is often read-only to avoid risk of getting overridden by programming bugs like buffer-overflow, etc.

- The code segment does not contain program variables like local variable (*also called as automatic variables in C*), global variables, etc.

- Based on the C implementation, the code segment can also contain read-only string literals. For example, when we do printf("Hello, world") then string "Hello, world" gets created in the code/text segment. we can verify this using size command in Linux OS.

■ Data Segment

- The data segment is divided in the below two parts and typically lies below the heaparea or in some implementations above the stack, but the data segment never lies between the heap and stack area.

■ Uninitialized data segment

- This segment is also known as bss.
- This is the portion of memory which contains:

1. **Uninitialized global variables (including pointer variables)**
2. **Uninitialized constant global variables.**
3. **Uninitialized local static variables.**

- Any global or static local variable which is not initialized will be stored in the uninitialized data segment
- For example: global variable int globalVar; or static local variable static int localStatic; will be stored in the uninitialized data segment.

- If we declare a global variable and initialize it as 0 or NULL then still it would go to uninitialized data segment or bss.

■ Initialized data segment

- This segment stores:

- Initialized global variables (including pointer variables)
- Initialized constant global variables.
- Initialized local static variables.
- For example: global variable int globalVar = 1; or static local variable static int localStatic = 1; will be stored in initialized data segment.
- This segment can be further classified into initialized read-only area and initialized read-write area. Initialized constant global variables will go in the initialized read-only area while variables whose values can be modified at runtime will go in the initialized read-write area.
- The size of this segment is determined by the size of the values in the program's source code, and does not change at run time,

■ Stack Segment

- Stack segment is used to store variables which are created inside functions (function could be main function or user-defined function), variable like

    1. **Local variables** of the function ***(including pointer variables)***
    2. **Arguments passed to function**
    3. **Return address**

- Variables stored in the stack will be removed as soon as the function execution finishes.

■ Heap Segment

- This segment is to support dynamic memory allocation. If the programmer wants to allocate some memory dynamically then in C it is done using the malloc, calloc, or realloc methods.

- For example, when int* prt = malloc(sizeof(int) * 2) then eight bytes will be allocated in heap and memory address of that location will be returned and stored in ptr variable. The ptr variable will be on either the stack or data segment depending on the way it is declared/used.