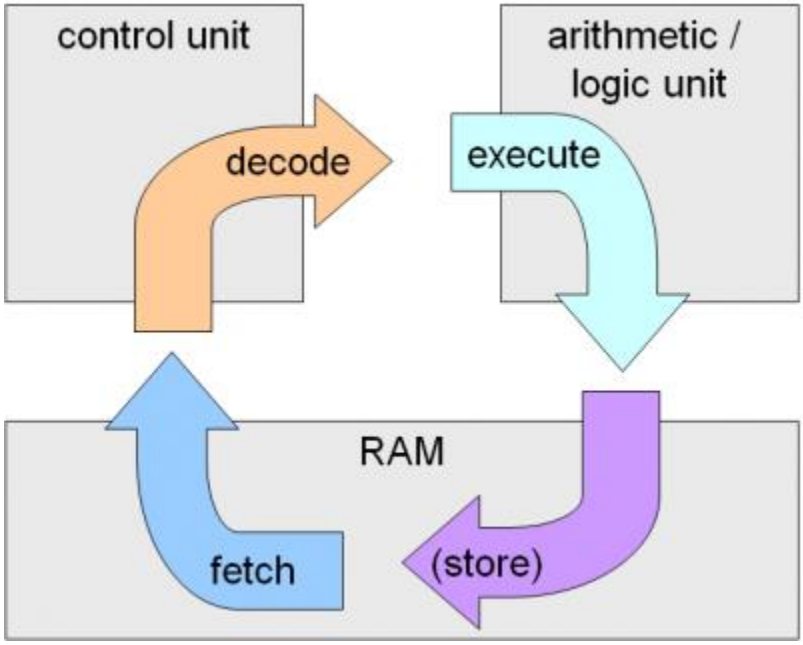
Instruction Cycle (How a program is executed?)



Process: A process is basically a program in execution.

A process is defined as an entity which represents the basic unit of work to be implemented in the system. The execution of a process must progress in a sequential fashion.

When the program runs, the processing is performed in two spaces called **Kernel Space** and **User Space** on the system. The two processing spaces implicitly interfere with each other and the processing of the program proceeds.

- **Kernel Space**

The kernel space can be accessed by user processes only using system calls that are requests in a Unix-like operating system such as input/output (I/O) or process creation.

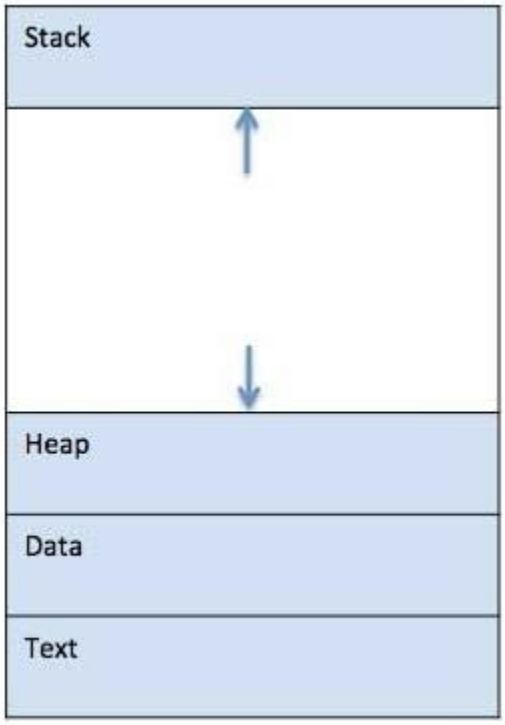
- **User Space**

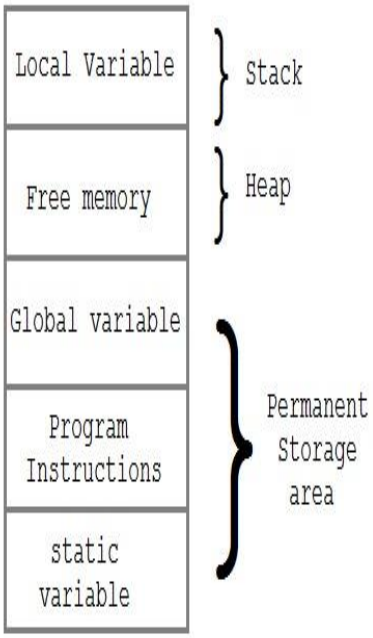
The user space is a computational resource allocated to a user, and it is a resource that the executing program can directly access. This space can be categorized into some segments.

To put it in simple terms, we write our computer programs in a text file and when we execute this program, it becomes a process which performs all the tasks mentioned in the program.

When a program is loaded into the memory and it becomes a process, it can be divided into four sections – **stack, heap, text and data**.

Memory Layout in program





Stack

The stack is LIFO (last-in-first-out) data structure.

Two principal operations:

- **push**, which adds an element to the collection, and
- **pop**, which removes the most recently added element that was not yet removed.

The process Stack contains the temporary data such as method/function parameters, return address and local variables.

Heap

This is dynamically allocated memory to a process during its run time.

Data

This section contains the global and static variables.

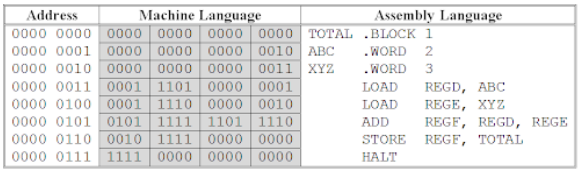
Text

A segment in which a machine language instruction is stored. This segment is a read-only space.

```
// Program to calculate the sum of n numbers entered by the user
#include <stdio.h>
#include <stdlib.h>
  int n, i, //Global variable
int main()
{  *ptr, //Local Variable
sum = 0; //Local Variable
    printf("Enter number of elements: ");
   scanf("%d", &n);
    ptr = (int*) malloc(n * sizeof(int));     // Dynamic memory allocation
    if(ptr == NULL)
                    {       printf("Error! memory not allocated.");
       exit(0);
    }
    printf("Enter elements: ");
```

```
   for(i = 0; i < n; ++i)
  {
     scanf("%d", ptr + i);
      sum += *(ptr + i);
   }
   printf("Sum = %d", sum);
   free(ptr);
   return 0;
}
```

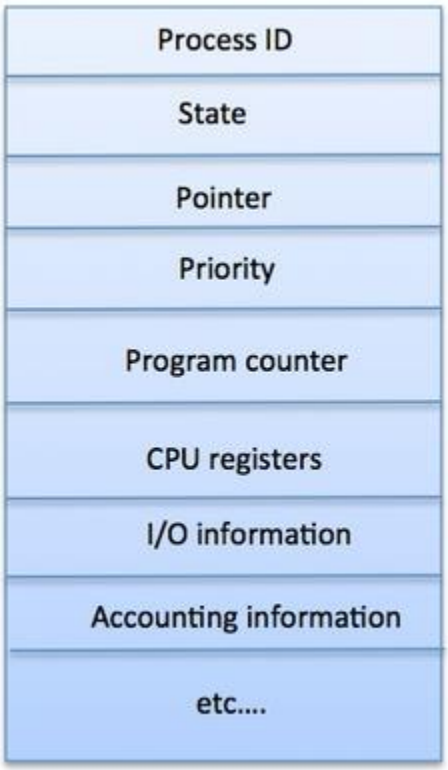
How the program is stored into memory:

STACK	ptr, sum, return address etc.
HEAP	malloc(), free()
DATA:	INT n, I
TEXT	 machine language instructions

Address	Machine Language				Assembly Language			
0000 0000	0000	0000	0000	0000	TOTAL	.BLOCK	1	
0000 0001	0000	0000	0000	0010	ABC	.WORD	2	
0000 0010	0000	0000	0000	0011	XYZ	.WORD	3	
0000 0011	0001	1101	0000	0001		LOAD	REGD, ABC	
0000 0100	0001	1110	0000	0010		LOAD	REGE, XYZ	
0000 0101	0101	1111	1101	1110		ADD	REGF, REGD, REGE	
0000 0110	0010	1111	0000	0000		STORE	REGF, TOTAL	
0000 0111	1111	0000	0000	0000		HALT		

Process Control Block (PCB): **A Process Control Block is a data structure maintained by the Operating System for every process. The PCB is identified by an integer process ID (PID). A PCB keeps all the information needed to keep track of a process.**

The architecture of a PCB is completely dependent on Operating System and may contain different information in different operating systems.



The various information maintained by PCB

Information & Description
Process State The current state of the process i.e., whether it is ready, running, waiting, or whatever.
Process privileges This is required to allow/disallow access to system resources.
Process ID Unique identification for each of the process in the operating system.
Pointer A pointer to parent process.
Program Counter (PC) Program Counter is a pointer to the address of the next instruction to be executed for this process.
CPU registers Various CPU registers where process need to be stored for execution for running state.
CPU Scheduling Information Process priority and other scheduling information which is required to schedule the process.
Memory management information This includes the information of page table, memory limits, Segment table depending on memory used by the operating system.
Accounting information This includes the amount of CPU used for process execution, time limits, execution ID etc.
IO status information This includes a list of I/O devices allocated to the process.

Program Counter(PC) : PC is a 16-bit register. It contains a memory address. PC contains that very memory address from where the next instruction is to be fetched for execution. Suppose the PC contents are 8000H, then it means that the processor desires to fetch the instruction Byte at 8000H. After fetching the Byte at 8000H, the PC is automatically incremented by 1.