Multithreading

Before getting to know what is multithreading, let us first understand why multithreading?

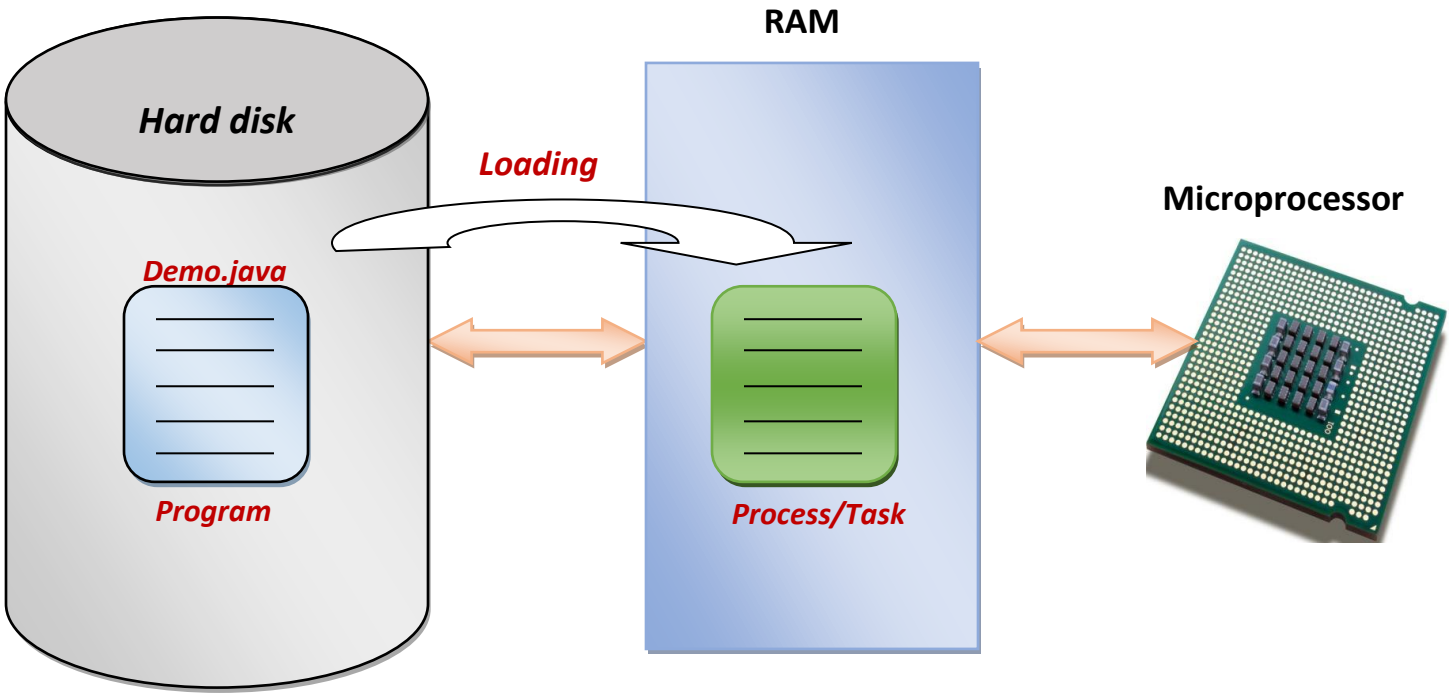
Why Multithreading??

Any developer should understand the concept of Multithreading to make their program efficient and for faster execution.

Before going into the depth of Multithreading let us understand what does process/task means?

Let us assume we have a java program named ***Demo.java*** saved on the hard disk. The execution doesn't happens in Hard disk so for the program to get executed it should be placed on the RAM. The process of taking the program and placing it on the RAM is called as ***Loading***. Once it is placed on RAM it is called as ***process*** as it is available to the microprocessor for execution. Process provides an execution environment for our program.

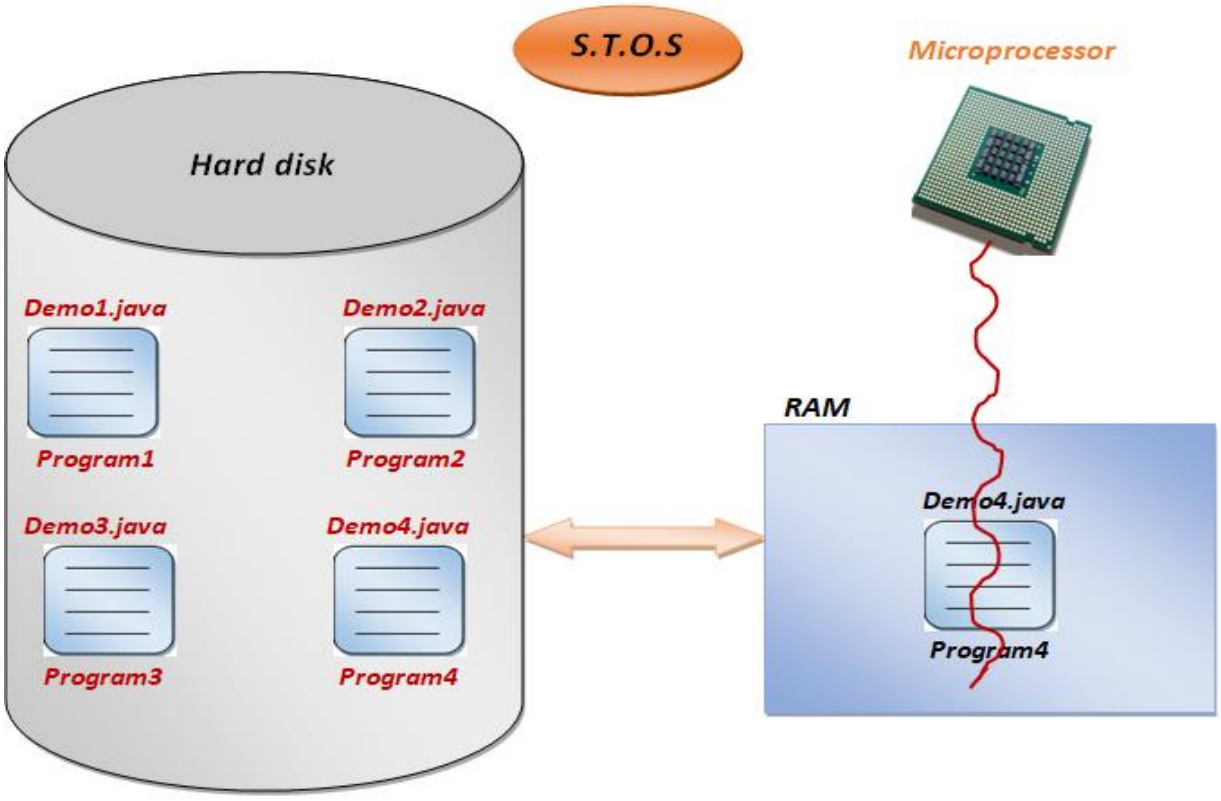
Therefore, a program under execution is technically called as Process.



Now Let us understand what happens with the Operating System.

Let us assume that the Operating System we used was **Single Tasking Operating System (S.T.O.S)**. STOS as the name implies, per processor it gives only one task. It is designed to manage the computer so that one user can effectively do one thing at a time.

Let us imagine there are multiple programs on the Hard disk but since we are working with STOS with use of single microprocessor, only one program gets loaded onto the RAM. And now there is one process on the RAM and microprocessor starts executing.
If the other program's has to get executed then the process which is under execution in the RAM should complete its execution and it is removed from the RAM. Then the next program gets loaded onto the RAM and execution continues in this way. This style of execution is called as Sequential Execution.



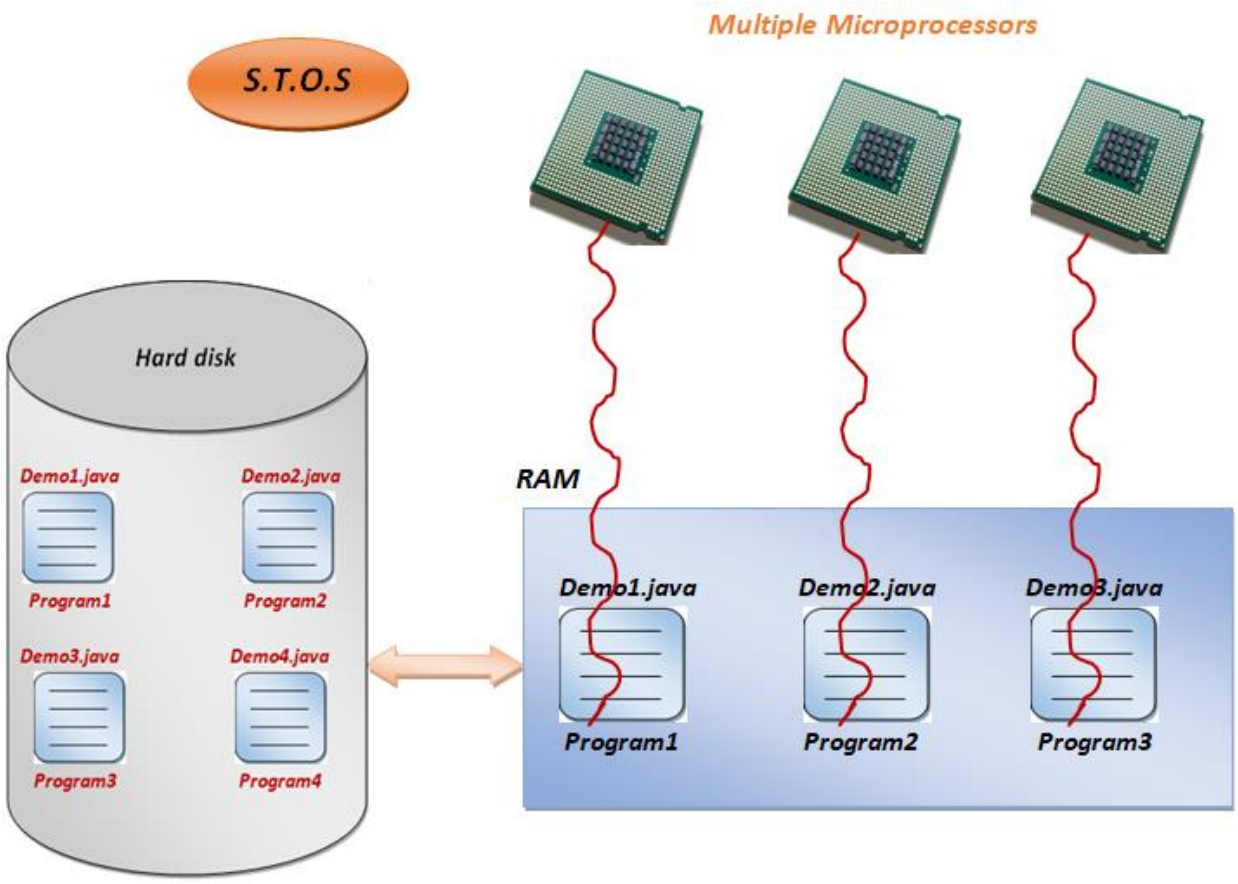
Single task is achieved using single processor with the use of S.T.O.S

Disadvantage of STOS with single processor approach:

- **Tasks take longer time to complete:** As we know if no multiple tasks run at a time then many tasks are waiting for the CPU. This will make system slow.

Now if we want multiple processes to work parallel what should be done??

Let us consider the same **Single Tasking Operating System (S.T.O.S).**
As we can see there are multiple programs on the Hard disk. But now let us consider multiple microprocessors which means multiple processes can be loaded onto the RAM. Since there are multiple processors each processors started executing one task. Therefore, multiple task is been done.



Multitasking is achieved using multiple processors with the use of S.T.O.S

Advantage of STOS with multiple processor approach:

- *Execution is fast as multiple programs are executing at the same time. This style of execution is called as* ***Parallel Execution****.*

Disadvantage of STOS with multiple processor approach:

- *Extremely expensive as it requires multiple processors.*

Now we should perform multitasking but with overcoming the disadvantage of using multiple processors which makes the cost expensive, but how???

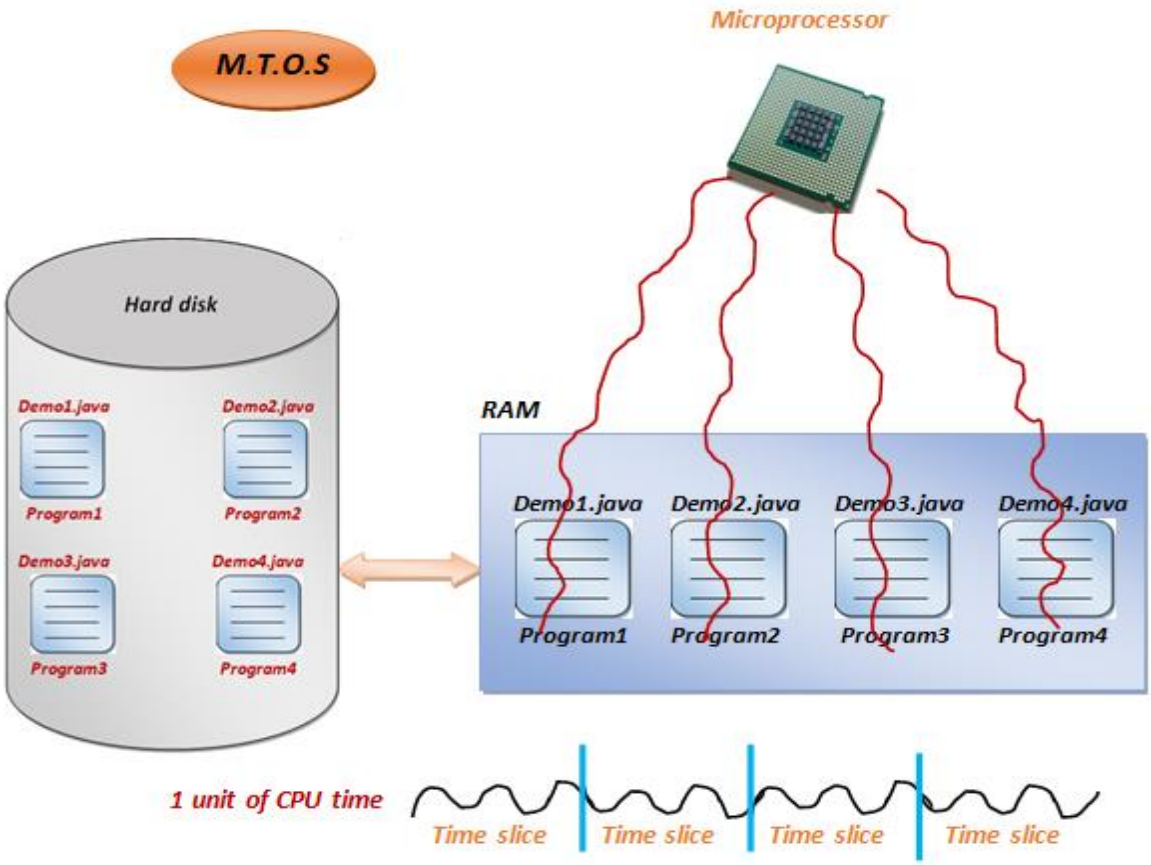
The solution for this was they thought of changing the software.
That is they replaced the S.T.O.S with **M.T.O.S (Multi Tasking Operating System)**.
Multitasking Operating System provides the interface for executing the multiple program tasks by single user at a same time on the one computer system.
This operating system is able to keep track of current task and can go from one to the other without losing information.
Example: user can open Gmail and Power Point same time.

The unit of the CPU depends upon the microprocessor. It can be nanoseconds, milliseconds, microseconds.
Here, the Operating System takes one unit of CPU time and divides into equal multiple slices.
Then multiple processes are loaded onto the RAM from the hard disk as it is working with M.T.O.S. Now there is a single processor with multiple tasks.

How does that work: 

The first slice of the CPU time is given to processor to execute one task, the moment the time is done then the control is taken away from the first task and the next slice of CPU time is given to the processor to execute the second task for that period of time. This switching continues with all the processes by giving time slice of CPU time till the task of all the processes it done.

This switching creates an illusion that multiple processes are executing at the same time. This kind of working is called ***Time-sharing Operating System or Time-slicing OS.*** *This style of execution is called as* ***concurrent execution****.*



Multitasking is achieved using single processor with the use of M.T.O.S

Advantage of M.T.O.S:

- Since multiple processors are not used cost is inexpensive.
- Concurrent execution that is time sharable in which, all tasks are allocated specific piece of time, so they do not need for waiting time for CPU.

Let us see whether our Java program efficiently utilise the CPU time.

In this example, let us perform **three activities.**

The first activity is **adding two numbers**, second is **printing characters** and the **third activity is printing numbers from 1 to 10**. All these activities are independent of each other.

Let us see which type of execution happens with this program.

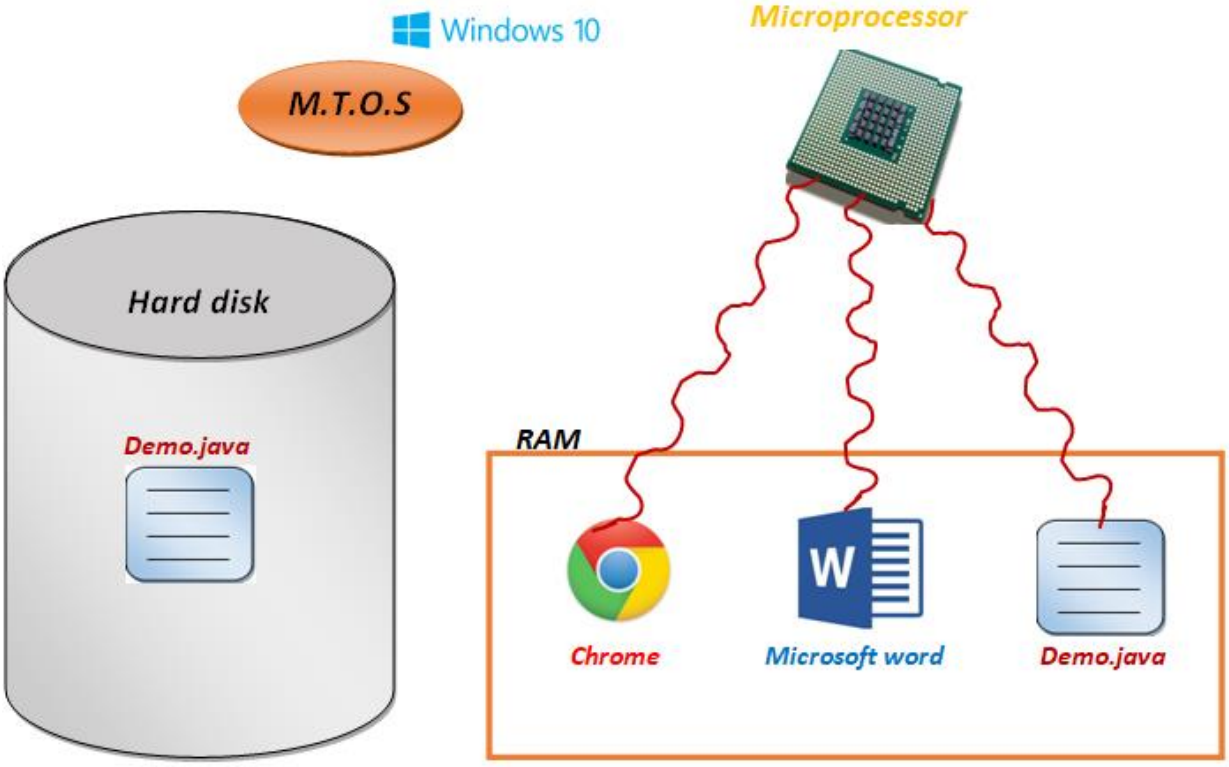
```
import java.util.Scanner;
class Demo
{
    public static void main(String[] args) throws Exception
    {
        System.out.println("Addition task started");
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter the first number");
        int a = scan.nextInt();
        System.out.println("Enter the second number");
        int b = scan.nextInt();
        int c = a+b;
        System.out.println(c);
        System.out.println("Addition task completed");

        System.out.println("Character priniting task started");
        for(int i=65;i<=75;i++)
        {
            System.out.println((char)i);
            Thread.sleep(4000);
        }
        System.out.println("Character priniting task completed");

        System.out.println("Number priniting task started");
        for(int i=1;i<=10;i++)
        {
            System.out.println(i);
            Thread.sleep(4000);
        }
        System.out.println("Number priniting task completed");
    }
}
```

In this program, when it enters the main method it get started with the first activity of addition of two numbers.
But here according to the program since we have given Scanner input, the execution doesn't go forward to the next activities until and unless the input is given for the first activity of addition and this gets complete.
So don't you think there is loss of CPU time? Therefore, with this analysis we can understand that the type of execution that takes places here **is Sequential execution which makes the program slow and inefficient.**
This is the disadvantage of single threaded approach.

Output:

```
Addition task started
Enter the first number
10
Enter the second number
20
30
Addition task completed
Character printing started
A
B
C
D
E
F
G
H
I
J
K
Character printing completed
Number printing started
1
2
3
4
5
6
7
8
9
10
Number printing Completed
```

Consider the example shown below:



Let us assume our Java program is saved in the file Demo.java which is stored on the hard disk. The Operating System which we are using M.T.O.S which may be Windows 10 OS. Now our program is loaded onto the RAM and now it's called as a process. But in our computer we might have multiple processes on the RAM which are shown here as Google chrome, Microsoft Word. But all these have a single processor for execution. With the use of Time slicing method, one slice of time is given to chrome and when the time completes the control is given to next task i.e., Microsoft word and in this way it continues. This is **concurrent execution.** Now the java program must efficiently utilize the CPU time given to it but in the above code the execution not concurrent rather it was sequential and hence CPU time was getting wasted.