Assignment

How does a function execute in the backend of a machine?

Ans
Here we will try to understand the execution of a function in the backend with respective to a procedural or imperative programming such as c#.

Example
```
Using System;

namespace name1{
        Internal class n1{
                Public static int add(int a, int b) {
                    int result = a + b;
                    return result;
                }

                Public static void main(String[] args) {
                    int x = 5;
                    int y = 7;
                    int sum = add(x, y);
                    Console.Writeline(“this is the sum “, sum);
                    return 0;
                }
        }
}
```

As C# supports procedural or imperative programming paradigm Main function will be considered as the starting/entry point of the program.

Here as the program encounters a function call Main(entry point), it needs to record the current state of execution, including the memory location of the next instruction to be executed.
Program record :101 | 105

Once the program encounters a function it allocates a stack memory to it and then jumps to the memory location of the function and starts executing the first statement inside the function.

The function might allocate memory on the stack or in registers for local variables. This memory is used to store data that is specific to the function.

The statements inside the function are executed in order. Control flows through conditional statements (if-else), loops, and other constructs as specified in the function's code. Local variables are read from and written to during this process.

Program record :105 | 110

```
 Address          +---------------------+
                  |  Stack Frame (main function) |
                  |---------------------|
             101 |       x = 5              |
                  |---------------------|
             105 |        y = 7              |
                  |---------------------|
             110 | sum = add(5,7) => 12 |
                  |---------------------|
             115 | Console.Writeline("this is the sum ", sum); |
                  +---------------------+
```

If the function makes calls to other functions, the process of recording the state, entering the new function, and executing it stepwise is repeated. This can lead to a call stack of functions being executed, with the innermost function executing first and the outer functions waiting for the inner ones to complete.
Program record 110 | 116

```
 Address +---------------------+
          |    Stack Frame (add)   |
          |---------------------|
          |     Parameters        |
          |---------------------|
     116  |        a = 5              |
          |---------------------|
     121  |        b = 7              |
          |---------------------|
     126  |   return result = 12 |
          +---------------------+
```

When the function reaches a return statement, it typically returns a value to the caller. The return value is often placed in a designated register or memory location. The function's local variables may also be deallocated at this point.

The program returns to the calling function, typically by popping the state information from the call stack and restoring the previous instruction pointer and data. The return value from the function call is used as needed in the calling function.

The program continues executing the instructions following the original function call, using the returned value or continuing with other operations.

Process record 110 | 115

//Output : this is the sum 12.