A program as a source code is merely a collection of text (code, statements etc.) and to make it alive, it requires actions to be performed on the target machine.

A program needs memory resources to execute instructions.

A program contains names for procedures, identifiers etc., that require mapping with the actual memory location at runtime.

Run time environment

By runtime, we mean a program in execution.

Runtime environment is a state of the target machine, which may include software libraries, environment variables, etc., to provide services to the processes running in the system.

Activation Trees

1. A program is a sequence of instructions combined into a number of procedures.
2. Instructions in a procedure are executed sequentially.
3. A procedure has a start and an end delimiter and everything inside it is called the body of the procedure.
4. The procedure identifier and the sequence of finite instructions inside it make up the body of the procedure.
5. The execution of a procedure is called its activation.
6. An activation record contains all the necessary information required to call a procedure.
7. An activation record may contain the following units (depending upon the source language used).

Temporaries	Stores temporary and intermediate values of an expression.
Local Data	Stores local data of the called procedure.
Machine Status	Stores machine status such as Registers, Program Counter etc., before the procedure is called.
Control Link	Stores the address of activation record of the caller procedure.
Access Link	Stores the information of data which is outside the local scope.
Actual Parameters	Stores actual parameters, i.e., parameters which are used to send input to the called procedure.
Return Value	Stores return values.

8. When a procedure calls another procedure, the execution of the caller is suspended until the called procedure finishes execution.

9. At this time, the activation record of the called procedure is stored on the stack.

10 We assume that the program control flows in a sequential manner and when a procedure is called, its control is transferred to the called procedure.

11. When a called procedure is executed, it returns the control back to the caller.

12. This type of control flow makes it easier to represent a series of activations in the form of a tree, known as the activation tree.

Storage Allocation

Runtime environment manages runtime memory requirements for the following entities:

- Code : It is known as the text part of a program that does not change at runtime. Its memory requirements are known at the compile time.
- Procedures : Their text part is static but they are called in a random manner. That is why, stack storage is used to manage procedure calls and activations.
- Variables : Variables are known at the runtime only, unless they are global or constant. Heap memory allocation scheme is used for managing allocation and de-allocation of memory for variables in runtime.

Static Allocation

In this allocation scheme, the compilation data is bound to a fixed location in the memory and it does not change when the program executes.

As the memory requirement and storage locations are known in advance, runtime support package for memory allocation and de-allocation is not required.

Stack Allocation

Procedure calls and their activations are managed by means of stack memory allocation. It works in last-in-first-out (LIFO) method and this allocation strategy is very useful for recursive procedure calls.

Heap Allocation

Variables local to a procedure are allocated and de-allocated only at runtime.

Heap allocation is used to dynamically allocate memory to the variables and claim it back when the variables are no more required.

Except statically allocated memory area, both stack and heap memory can grow and shrink dynamically and unexpectedly.

Therefore, they cannot be provided with a fixed amount of memory in the system.