STORAGE ORGANIZATION

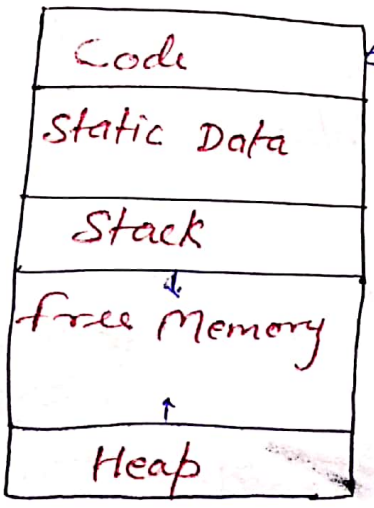
⊛ Storage Organisation :-

⇒ The executing target program runs in its own logical address space in which each program value has a location

⇒ The management and organisation of this logical address space is shared between the compiler, Operating System and target machine.

⇒ The operating system maps the logical address into physical address, which are usally spread throughout memory.

⊛ Sub division of Run time memory



- Memory location code are determined at compile time.
- Location of static data can be alsobe determine at Compile time.
- Data objects allocated at run-time (Activation Records)
- Other Dynamically Allocated data objects at Run-time (for eg:- Malloc area in C).

⇒ Runtime storage comes into blocks, where a byte is used to show the smallest unit of addressable memory using the four bytes a machine word can form.

⇒ Object of multibyte is stored in consecutive bytes and gives the first byte address.

→ Runtime storage can be subdivide to hold the different components of an executing program.

1- Generated executable code
2- static data objects
3- Dynamic data objects - heap
4- Automatic data objects - stack
5-

⊛ Storage Allocation :-

The different ways to allocate memory are:-

1. Static Storage Allocation.
2. Stack storage Allocation.
3. Heap Storage Allocation.

Static Storage Allocation:-

→ In static allocation, names are bound to storage locations.

→ If memory is created at compile time then the memory will be created in static area and only once.

→ Static allocation supports the dynamic data structure that means, memory is created only at compile time and deallocated after program completion.

→ The drawback with static storage allocation is that the size and position of data objects should be known at compile time.

→ Another drawback is restriction of the recursion procedure.

⊛ Stack Storage Allocation :-

- Storage is organized as a stack.
- Activation records are pushed and popped.
- Activation record contains the locals so that they are bound to fresh storage in each activation record.
- The value of locals is deleted when the activation ends.
- it works on the basis of LIFO and this allocation supports the recursion process.

⊛ Heap Storage Allocation :-

(dynamically)

→ it is the most flexible allocation scheme.

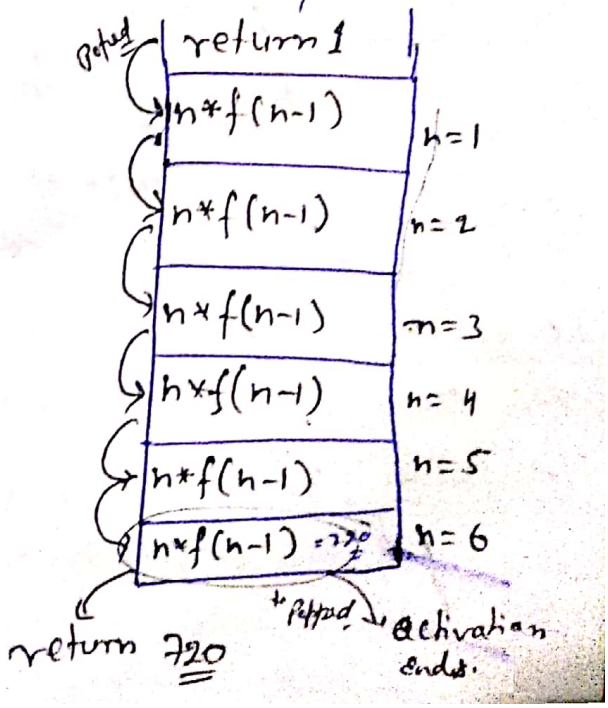
↦ Allocation and deallocation of memory can be done at any time and at any place depending upon the user's requirement.

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

→ Heap storage allocation supports the recursion process.

Example :-

```
fact (int n)
{
 if (n<=1)
 return 1;
 else
 return (n*fact(n-1));
}
 fact (6)
```



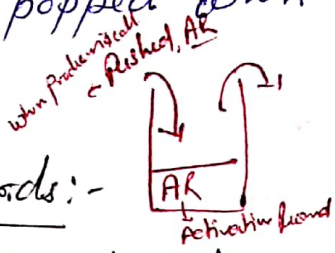
⊛ Activation Records:-

⇒ Control Stack is a runtime stack which is used to keep track of the live procedure activations, i.e:- it is used to find out the procedure whose execution have not been completed.

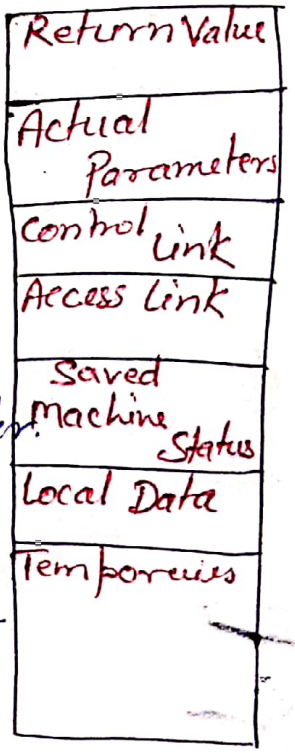
⇒ When the activation begins then the procedure name will push on to the Stack and when returns (activation ends) then it will popped.

⇒ Activation record is used to manage the information needed by a single execution of a procedure.

⇒ An activation record is pushed into the stack when a procedure is called, and is popped when the control return to the caller function.



⊛ Contents of Activation Records:-



Return value:- it is used by call procedure to return a value to calling procedure.

Actual Parameters:- it is used by calling procedure to supply parameters to called procedure.

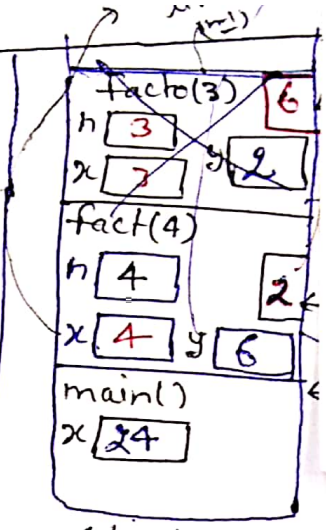
Control link:- it points to activation record of the caller.

Access link:- it is used to refer to non local data

Saved Machine Status:- it holds the information about status of machine before the procedure is called.

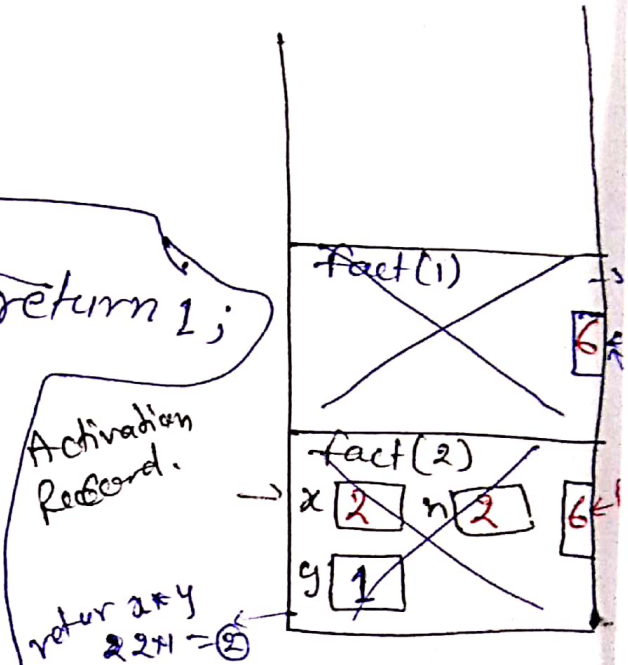
Local Data:- It holds the data that is local to the execution of the procedure.

Tempories:- it stores the value that arises in the evaluation of an expression.



Stack.

```
Void main()
 {
1- int x;
2→ x = fact(4);
3→ Print (x);
 }
Void fact (int n)
 {
4→ if (n==0) || (n==1) return 1;
5→ int x = n;
6→ int y = fact(n-1);
7→ return x*y;
 }
```



Activation Record.

return 2*1 = 2

Execute this program we need to Compiler

Compiler

Logic + Extra Code.

This is called Stack of Activation Record.