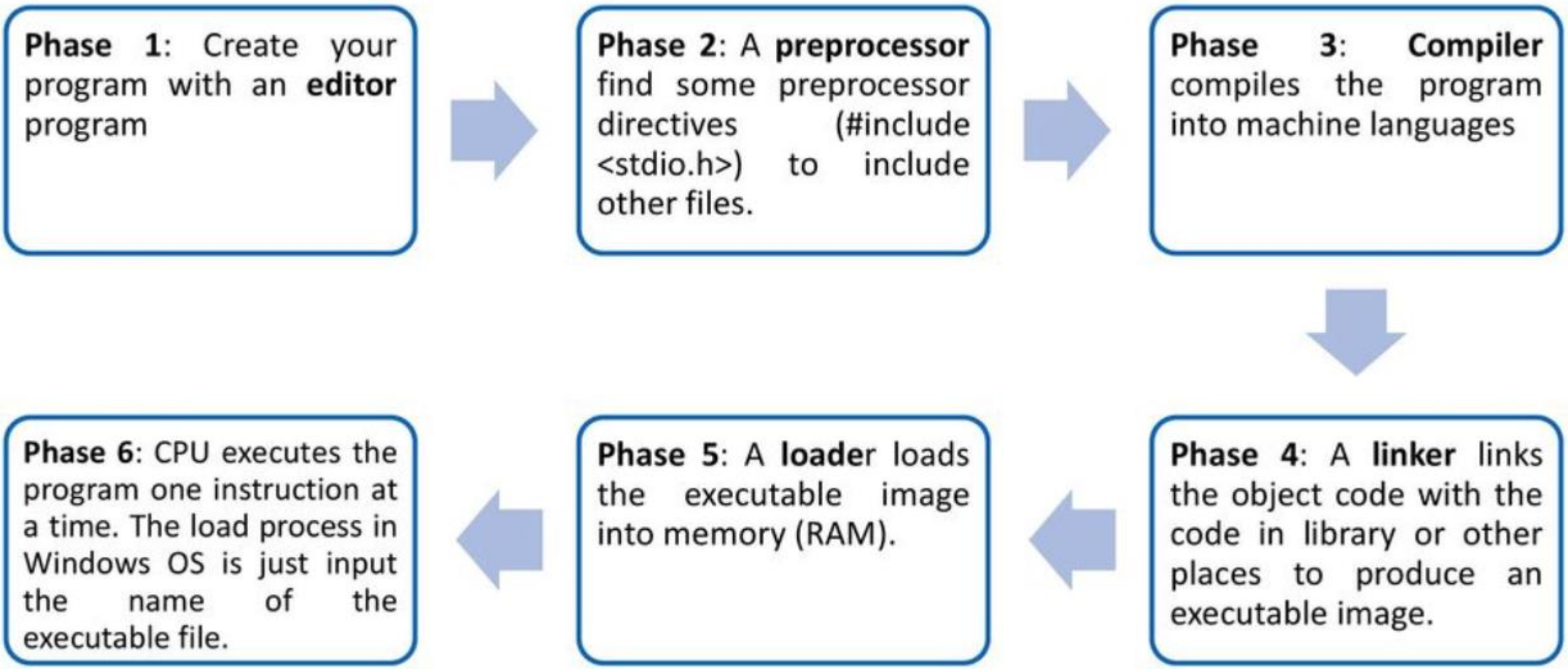
Unit 3.1
Run Time Storage Management

How storage organizations used to support the run-time environment of a program ?

What Data structures are used ?

C Program Development Environment



Run-time Environments

- A compiler must accurately implement the abstractions contained in the source language definition. These abstractions typically include the concepts such names, scope, bindings, data types, operators, procedures, parameters and flow-of-control constructs.
- The compiler must co-operate with operating system and other system software to support these abstractions on the target machine.
- To do so, the compiler creates and manages a run-time environment in which target code are being executed.
- 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.

Run-time Environment…

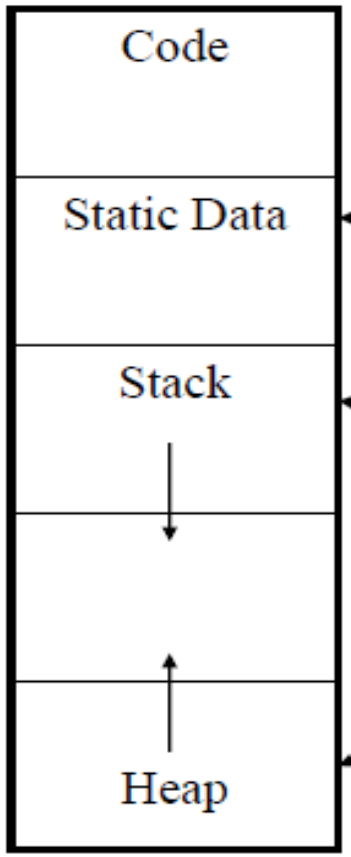
- Runtime support system is a package, mostly generated with the executable program itself and facilitates the process communication between the process and the runtime environment. It takes care of memory allocation and de-allocation while the program is being executed
- This environment deals with a number of issues such as
  - allocation of storage locations for the objects named in the source program,
  - the mechanisms used by the target program to access variables,
  - the linkages between procedures,
  - the mechanisms for passing parameters, and
  - interfaces to the operating system, input/output devices and other programs.
- That is,
  - Management of run-time resources
  - Correspondence between static (compile-time) and dynamic (run-time) structures
  - Storage organization/Allocation

Run-time Resources

- Execution of a program is initially under the control of the operating system (OS)
- When a program is invoked:
  - The OS allocates space for the program
  - The code is loaded into part of this space
  - The OS jumps to the entry point of the program

  (i.e., to the beginning of the “main” function)

Memory Layout: Storage Organization



Memory locations for code are determined at compile time. **.exe**

Locations of static data can also be determined at compile time. **Initialized global or static Data**

Data objects allocated at run-time. (Activation Records)

Other dynamically allocated data objects at run-time. (For example, malloc area in C). **DMA**

Uninitialized global and static data

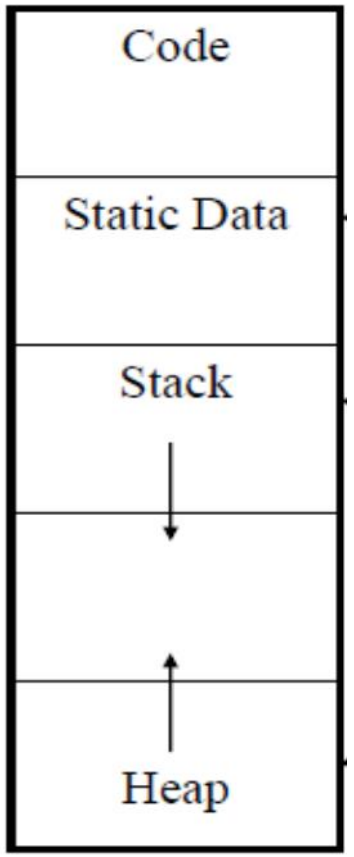
Run time Environment of C-Program

```
int i=10,j;

fun(int x)
{
    static int y=10,z;
    int a=10,b;

}

main()
{
    static int y=20, c;
    int d=10, e;
}
```



Correspondence between static and Dynamic structures

- Compiler must do the storage allocation and provide access to variables and data.
- At run time, we need a system to map NAMES (in the source program) to STORAGE on the machine.
- Allocation and de-allocation of memory is handled by a RUN-TIME SUPPORT SYSTEM typically linked and loaded along with the compiled target code.
- One of the primary responsibilities of the run-time system is to manage **ACTIVATIONS** of procedures.
- Procedure execution begins at the first statement of the procedure body.
- When a procedure returns, execution returns to the instruction immediately following the procedure call.

The Control Stack : Activation Record

- Function calls are often implemented using a stack of activation records (or stack frames).
- Calling a function pushes a new activation record onto the stack.
- Returning from a function pops the current activation record from the stack.

Activation and Activation Tree

- Every execution of a procedure is called an ACTIVATION.
- The LIFETIME of an activation of procedure P is the sequence of steps between the first and last steps of P's body, including any procedures called while P is running.
- Normally, when control flows from one activation to another, it must (eventually) return to the same activation.
- If a procedure is recursive, a new activation can begin before an earlier activation of the same procedure has ended.
- We can represent the activations of procedures during the running of an entire program by a tree, called an **activation tree.**

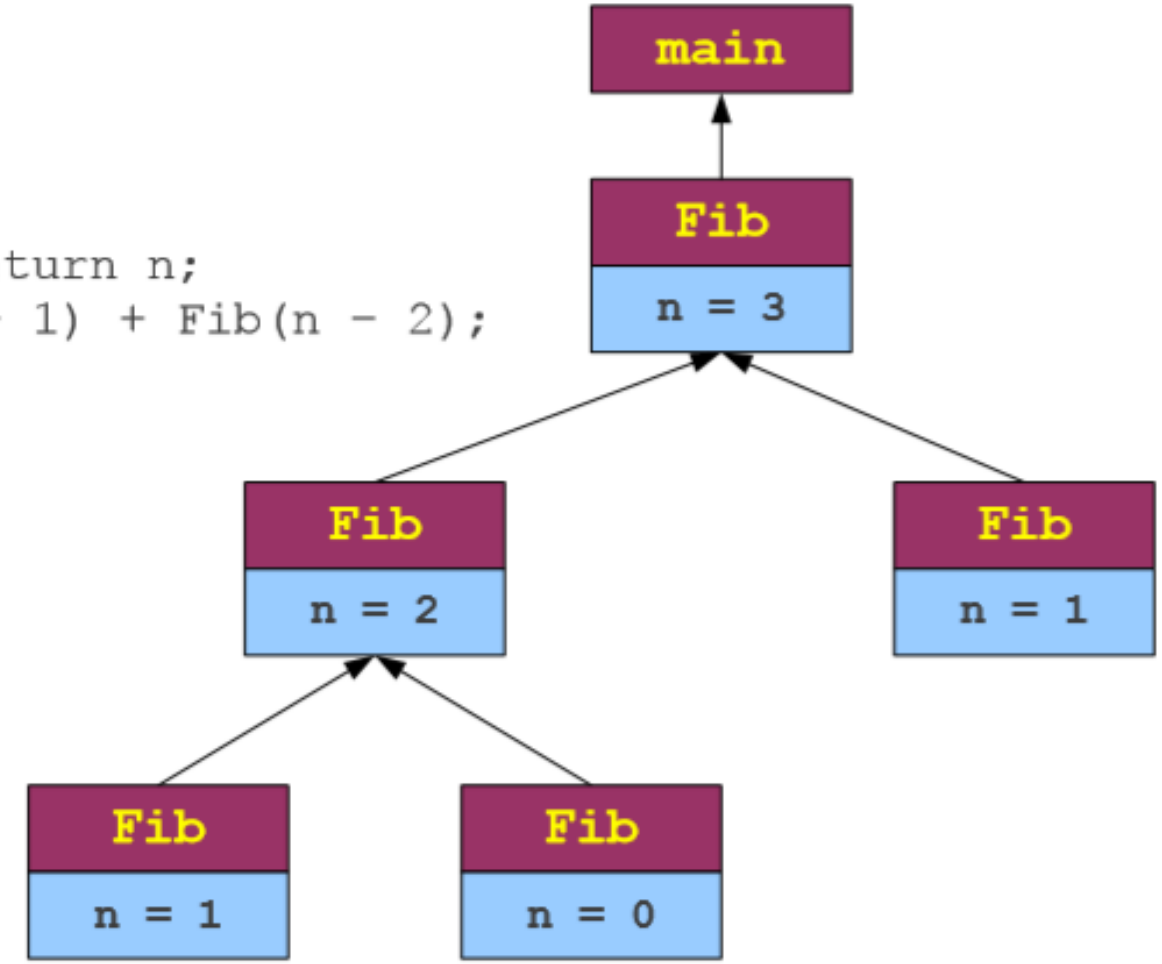
We can create a tree (known as activation tree) to show *the* way *control* enters and leaves activation. In an activation tree

- Each node represents an activation of a procedure.
- The root represents the activation of the main program.
- The node a is a parent of the node *b* if and only if *the control flows from a to b.*
- The node a is left to the node *b* if the lifetime of *a occurs before b.*

Activation Tree

```
int main() {
    Fib(3);
}

int Fib(int n) {
    if (n <= 1) return n;
    return Fib(n - 1) + Fib(n - 2);
}
```

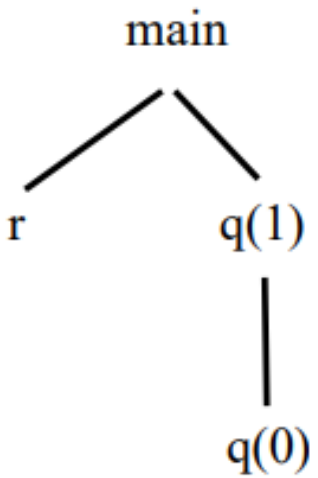


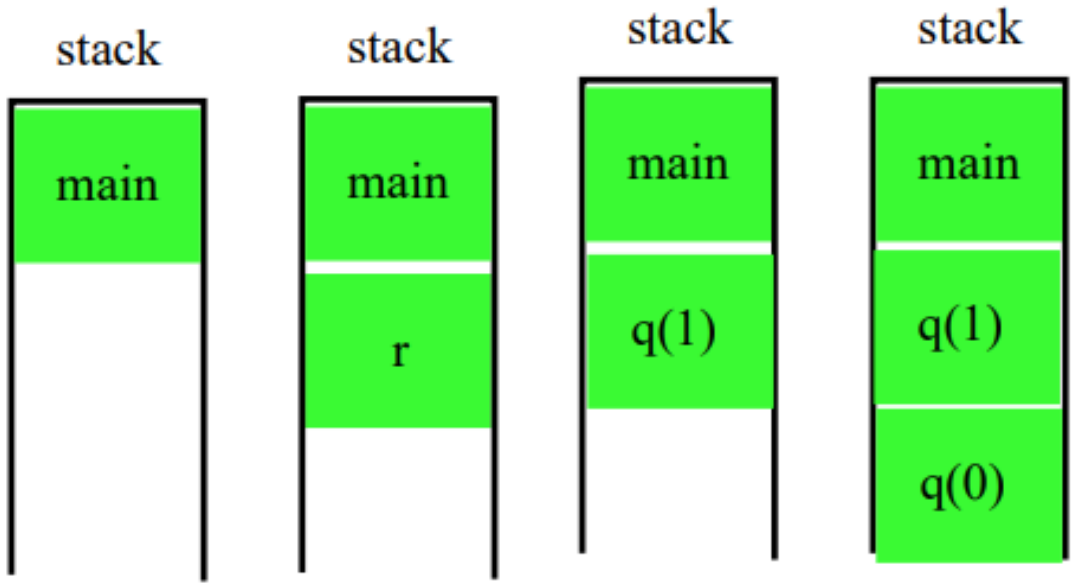
Example

```
main{
    r();
    q(1);
}

r{
...}

q(int i)
{
if(i>0) then q(i-1)
}
```





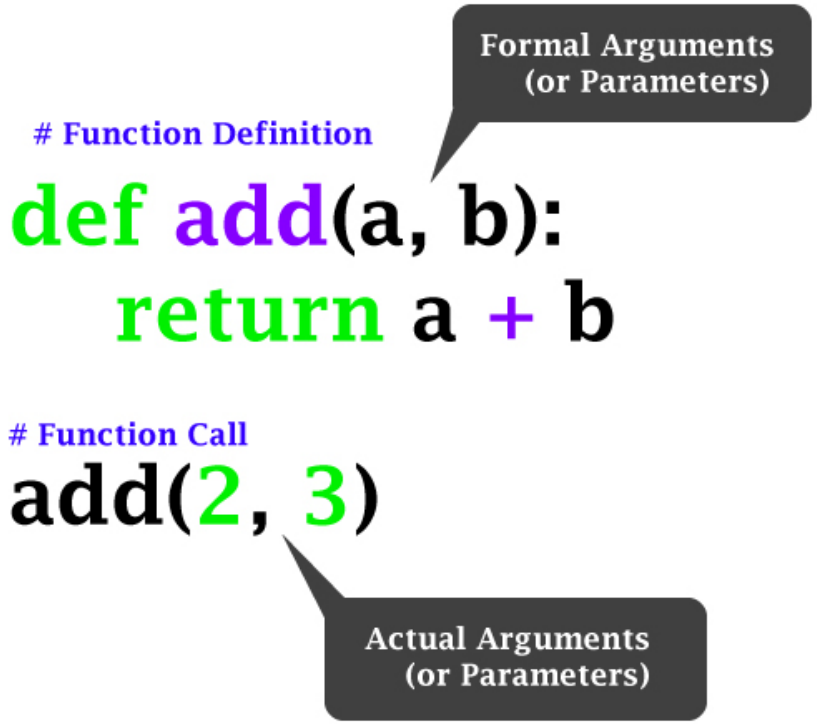
Terminology

- **Caller:** calling procedure (in this case, `main`)
- **Callee:** called procedure (in this case, `sum`)

High-level code example

```
void main()
{
  int y;
  y = sum(42, 7);
  ...
}

int sum(int a, int b)
{
  return (a + b);
}
```



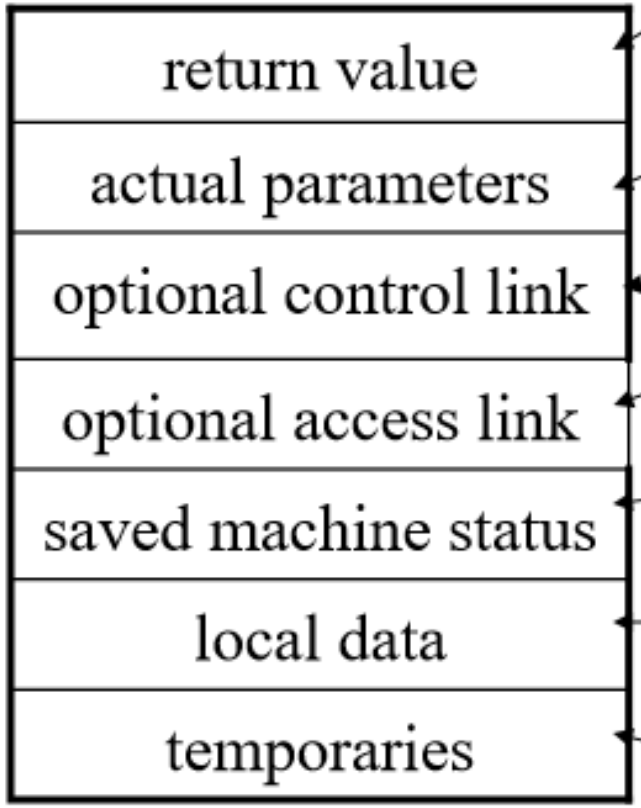
Access (or static) link: a pointer to places of non-local data,

Control (or dynamic) link: a pointer to the activation record of the caller.

Activation Record

- Information needed by a single execution of a procedure is managed using a contiguous block of storage called activation record.
- An activation record is allocated when a procedure is entered, and it is de-allocated when that procedure exited.

Activation Record



The returned value of the called procedure is returned in this field to the calling procedure. This value return in machine register for efficiency..

The field for actual parameters is used by the calling procedure to supply parameters to the called procedure.

The control link points to the activation record of the caller.

The access link is used to refer to nonlocal data held in other activation records.

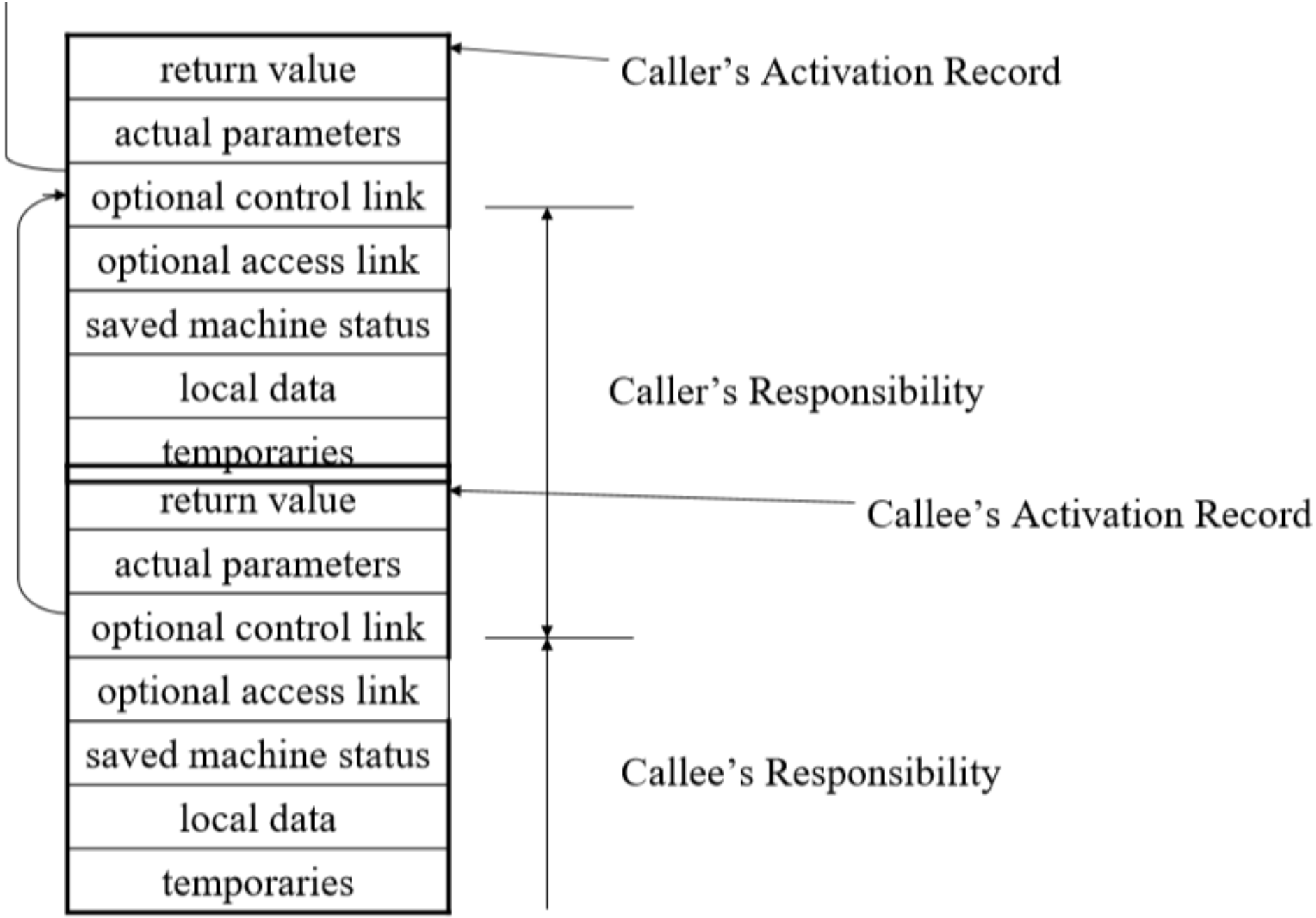
The field for saved machine status holds information about the state of the machine before the procedure is called.

The field of local data holds data that local to an execution of a procedure..

Temporary variables is stored in the field of temporaries.

Creation of An Activation Record

- Who allocates an activation record of a procedure?
  - ✓Some part of the activation record of a procedure is created by that procedure immediately after that procedure is entered.
  - ✓Some part is created by the caller of that procedure before that procedure is entered.


return value
actual parameters
optional control link
optional access link
saved machine status
local data
temporaries
return value
actual parameters
optional control link
optional access link
saved machine status
local data
temporaries
Caller's Activation Record
Caller's Responsibility
Callee's Activation Record
Callee's Responsibility

Sample calling sequence

- Caller evaluates the actual parameters and places them into the activation record of the callee.
- Caller stores a return address and old value for stack_top in the callee's activation record.
- Caller increments stack_top to the beginning of the temporaries and locals for the callee.
- Caller branches to the code for the callee.
- Callee saves all needed register values and status.
- Callee initializes its locals and begins execution.

Sample return sequence

- Callee places the return value at the correct location in the activation record (next to caller's activation record)
- Callee uses status information previously saved to restore stack_top and the other registers.
- Callee branches to the return address previously requested by the caller.
- [Optional] Caller copies the return value into its own activation record and uses it to evaluate an expression.

OLD Questions

1. What is activation record ? Discuss the different activities performed by caller and callee during procedure call and return.

2. Explain in details about the creation of an activation records.

- **Diagram**
- **Caller's and Callee's responsibilities**

3. Explain Run time Environment with a suitable example (C).

4.Explain the concept of Activation, Activation Records and Activation Tree.

5. How a language is processed ?

Preprocessing→ Compilation → Linking→ Loading→ OS/CA