Compiler Design

Samit Biswas

samit@cs.iiests.ac.in

Department of Computer Science and Technology,
Indian Institute of Engineering Science and Technology, Shibpur

October 11, 2018

Run Time Environments

Compiler must do the storage allocation and provide access to variables and data.

- The allocation and deallocation of data objects is managed by the **runtime support package**, consisting of routines loaded with the generated target code.
- Each execution of a procedure is referred to as an **activation** of the procedure.
- If the procedure is **recursive**, several of its activations may be alive at the same time.

Activation Trees

we make the following assumptions about the flow of control among procedures during execution of a program:

- Control flows sequentially; that is the execution of a program consists of a sequence of steps, with control being at some specific point in the program at each step.
- Each **execution** of a procedure starts at the beginning of the procedure body and eventually returns the control to the point immediately following the place where the procedure was called.

Lifetime : refers to a consecutive sequence of steps during the execution of a program.

Sketch of a Quicksort Programe

```
int a[11];
void readArray() { /* Reads 9 integers into a[1], ..., a[9]. */
    int i;
    ...
}
int partition(int m, int n) {
    /* Picks a separator value v, and partitions a[m..n] so that
       a[m..p-1] are less than v, a[p] = v, and a[p+1..n] are
       equal to or greater than v. Returns p. */
    ...
}
void quicksort(int m, int n) {
    int i;
    if (n > m) {
        i = partition(m, n);
        quicksort(m, i-1);
        quicksort(i+1, n);
    }
}
main() {
    readArray();
    a[0] = -9999;
    a[10] = 9999;
    quicksort(1,9);
}
```

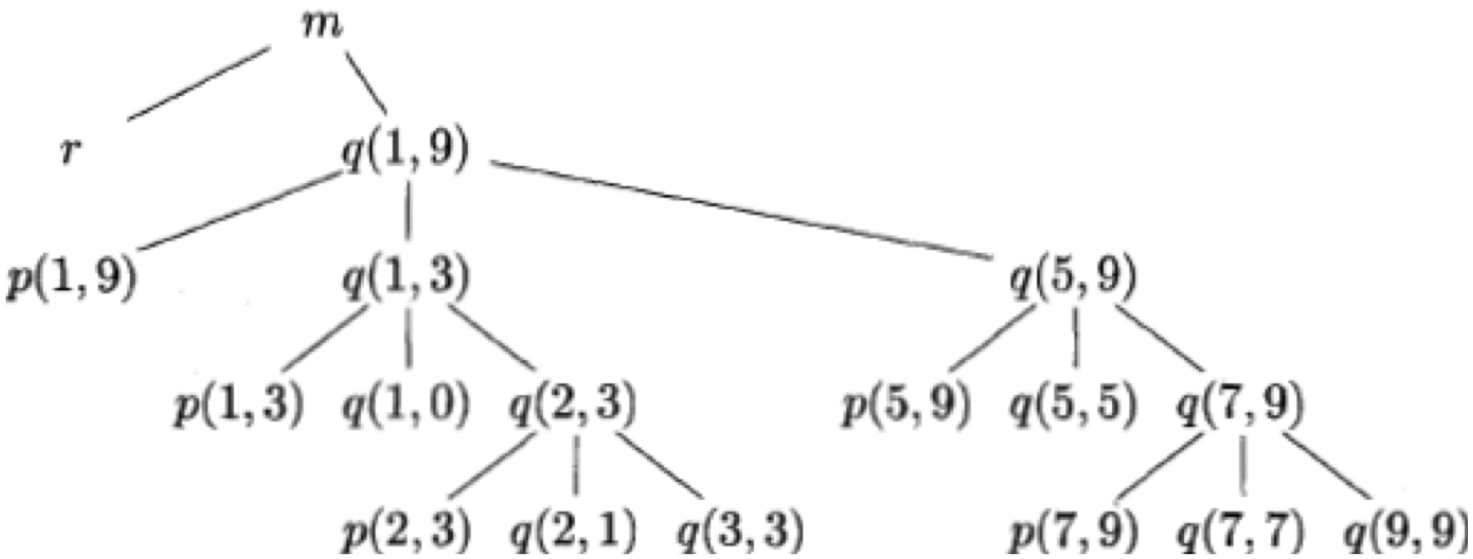
Activations for Quicksort

```
enter main()
    enter readArray()
    leave readArray()
    enter quicksort(1,9)
        enter partition(1,9)
        leave partition(1,9)
        enter quicksort(1,3)
            ...
        leave quicksort(1,3)
        enter quicksort(5,9)
            ...
        leave quicksort(5,9)
    leave quicksort(1,9)
leave main()
```

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 the parent of the node for *b* if and only if control flows from activation a to b;
- The node for a is to the left of the node for b if and only if the lifetime of a occurs before the lifetime of b.

Activation tree representing calls during an execution of quicksort

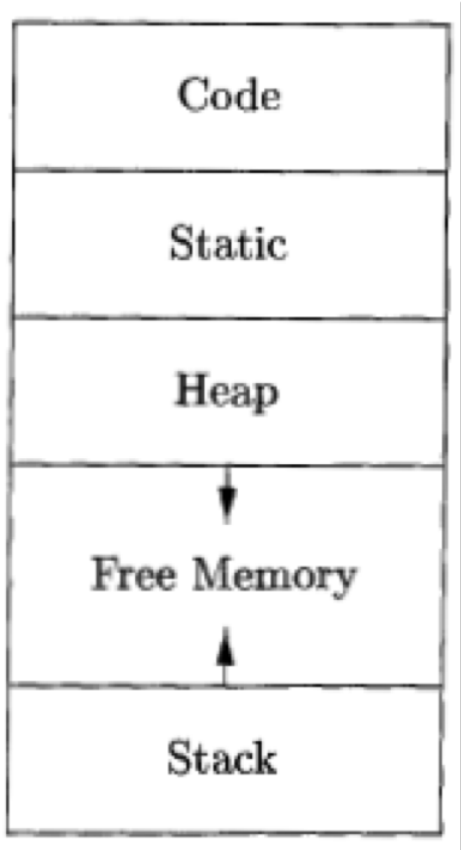


Storage Organization

Suppose that the compiler obtains a block of storage from the operating system for the compiled program to run in. Run time storage might be subdivided to hold:

- the generated target code,
- data objects, and
- a counter part of the control stack to keep track of procedure activation.

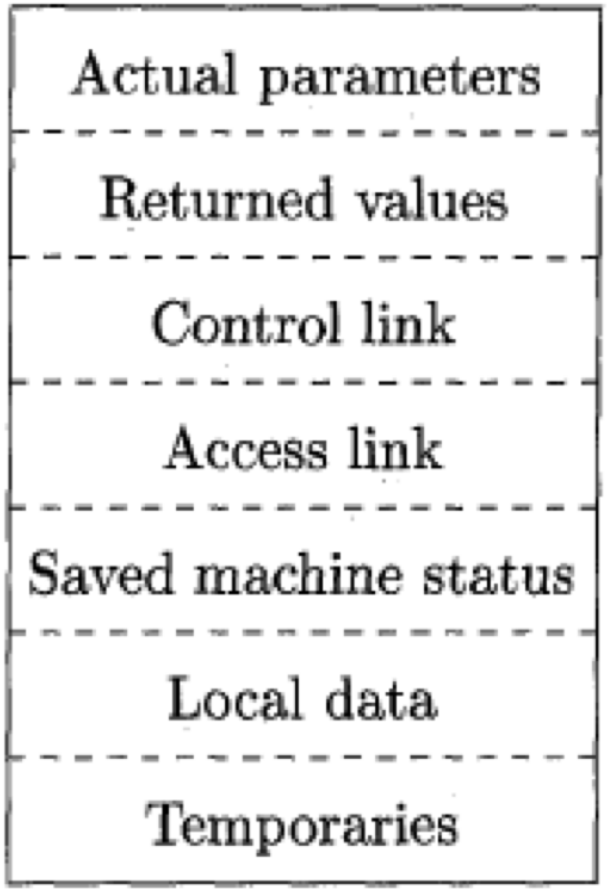
Typical subdivision of run-time memory into code and data areas



Activation Records

- Procedure calls and returns are usaully managed by a run-time stack called the control stack.
- Each live activation has an active record (sometimes called frame).
- The root of activation tree is at the bottom of the stack.
- The current execution path specifies the content of the stack with the last activation has record in the top of the stack.

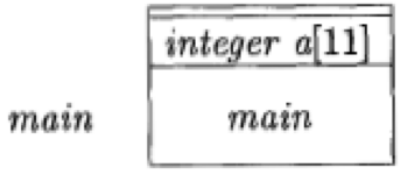
A general Activation Record



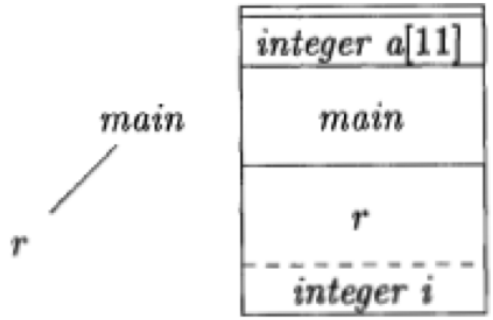
Activation Record

- **Temporary Values** - stores those which are arising in the evaluation of expression.
- **Local Data** - holds data that is local to an execution.
- **saved machine status** - holds information about the state of the machine just before the procedure call. This information typically includes the *return address*.
- **access link** - used to refer non-local data held in other activation records.
- **control link** - points to the activation record of the caller.
- Space for the **return value** of the called function.
- **Actual parameters** used by the calling procedure.

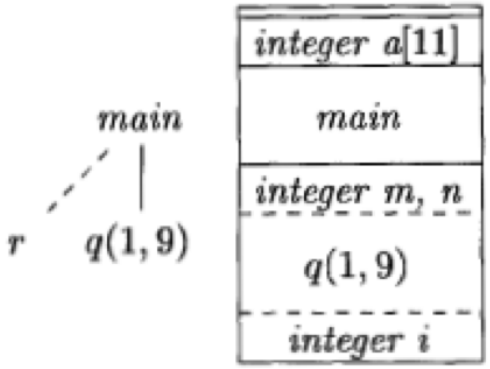
Downward growing stack of the activation records

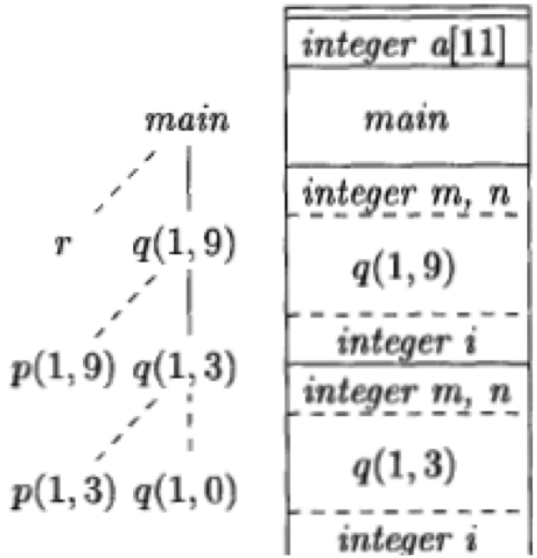


(a) Frame for *main*



(b) r is activated





Possible call Sequences

- The Caller Evaluates actuals.
- The Caller stores a return address and the old value of *top_sp* in to the callee's activation record. The caller then increments *top_sp* and moved past the caller's local data and temporaries and the callee's parameter and status field.
- The callee saves register values and other status information.
- The callee initializes its local data and begin execution.

A possible return sequences:

- The callee places a return value next to the activation record of the caller.
- using the information in the status field, the callee restores *top_sp* and other registers and branches to a return address in the caller's code.