NESTED SUBPROGRAMS

• Some non-C-based static-scoped languages (e.g., Fortran 95, Ada, Python, JavaScript, Ruby, and Lua) use stackdynamic local variables and allow subprograms to be nested

• All variables that can be non-locally accessed reside in some activation record instance in the stack

• The process of locating a non-local reference:

1. Find the correct activation record instance

2. Determine the correct offset within that activation record instance

Locating a Non-local Reference

• Finding the offset is easy

• Finding the correct activation record instance

– Static semantic rules guarantee that all non-local variables that can be referenced have been allocated in some activation record instance that is on the stack when the reference is made

Static Scoping

• A static chain is a chain of static links that connects certain activation record instances

• The static link in an activation record instance for subprogram A points to one of the activation record instances of A's static parent

• The static chain from an activation record instance connects it to all of its static ancestors

• Static_depth is an integer associated with a static scope whose value is the depth of nesting of that scope

• The chain_offset or nesting_depth of a nonlocal reference is the difference between the static_depth of the reference and that of the scope when it is declared

• A reference to a variable can be represented by the pair:

(chain_offset, local_offset),

where local_offset is the offset in the activation record of the variable being referenced

Example Ada Program

```
procedure Main_2 is
 X : Integer;
        procedure Bigsub is
                A, B, C : Integer;
                procedure Sub1 is
                        A, D : Integer;
                        begin -- of Sub1
                        A := B + C; <-----------------------1
                end; -- of Sub1
                procedure Sub2(X : Integer) is
                        B, E : Integer;
                        procedure Sub3 is
                                C, E : Integer;
                                begin -- of Sub3
                                Sub1;
                                E := B + A: <--------------------2
                                end; -- of Sub3
                        begin -- of Sub2
                        Sub3;
                        A := D + E; <-----------------------3
                        end; -- of Sub2 }
                begin -- of Bigsub
                Sub2(7);
                end; -- of Bigsub
        begin
        Bigsub;
```

end; of Main_2 }

• Call sequence for Main_2

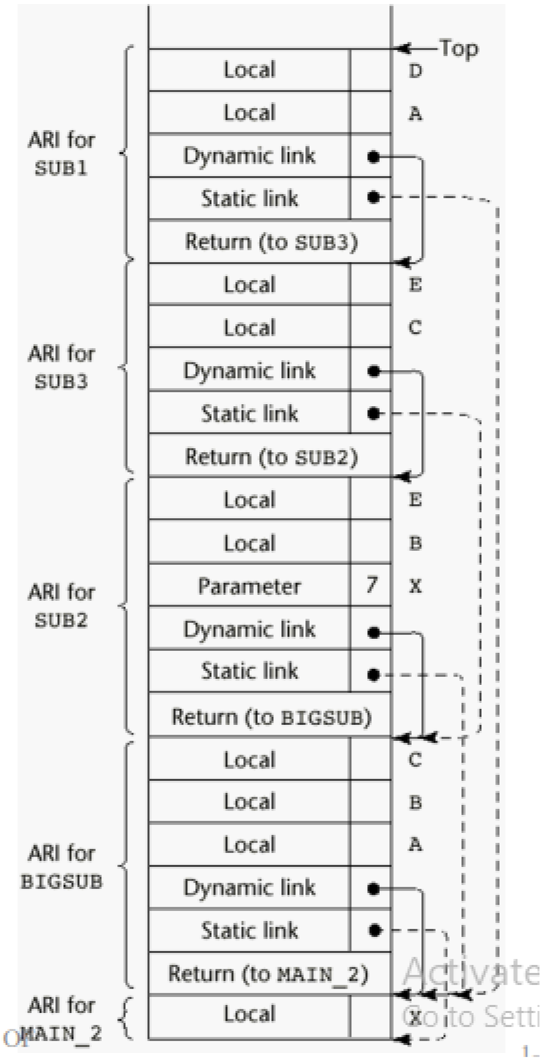
Main_2 calls Bigsub

Bigsub calls Sub2

Sub2 calls Sub3

Sub3 calls Sub1

Stack Contents at Position 1



Static Chain Maintenance

• At the call,

- The activation record instance must be built

- The dynamic link is just the old stack top pointer

- The static link must point to the most recent ari of the static parent

- Two methods:

1. Search the dynamic chain

2. Treat subprogram calls and definitions like variable references and definitions

Evaluation of Static Chains

• Problems:

1. A nonlocal areference is slow if the nesting depth is large

2. Time-critical code is difficult:

a. Costs of nonlocal references are difficult to determine

b. Code changes can change the nesting depth, and therefore the cost

Displays

• An alternative to static chains that solves the problems with that approach

• Static links are stored in a single array called a display

• The contents of the display at any given time is a list of addresses of the accessible activation record instances