The Call Stack

Introduction

- Storage Mechanisms
- Stack Frame (Activation Record)
- Links
  - Static Links
  - Dynamic Links
- Closures
- Recursive Thinking

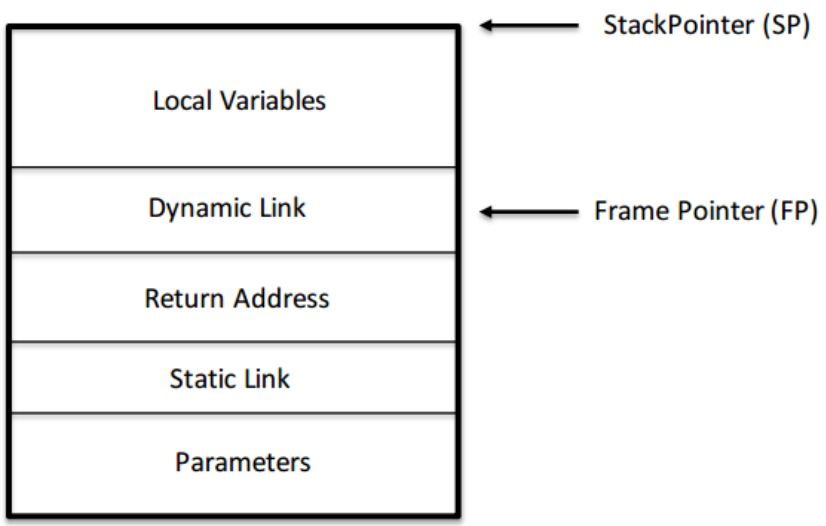
Storage Mechanisms

- *Static* objects are given an absolute address that is retained throughout the program's execution.
- *Stack* objects are allocated and deallocated in last-in, first-out order, usually in conjunction with subroutine calls and returns.
- *Heap* objects may be allocated and deallocated at arbitrary times. They require a more general (and expensive) storage management algorithm

Stack Frame or Activation Record

- Each routine, as it is called, is given a new *stack frame*, or *activation record*, at the top of the stack
- When a subroutine returns, its frame is popped from the stack.
- There are 2 dedicated registers associated with stack frame:
  - Frame Pointer: Points to a fixed place within the frame.
  - Stack pointer: Contains the address of the top of the stack

Layout of the stack frame



Static Links

- In a language with nested subroutines and static scoping objects that lie in surrounding subroutines and thus neither local nor global, can be found by maintaining a *static chain.*
- Each Stack frame contains a reference to the frame of the lexically surrounding subroutine and is called static link.

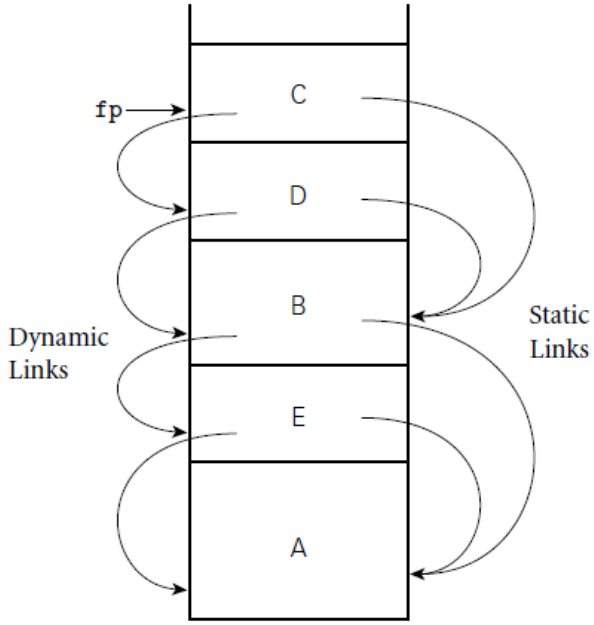
Dynamic Link

- Dynamic link points to the top of the caller
- The saved value of the frame pointer which will be restored on subroutine return, is called the dynamic link

Example

```
procedure A
    procedure B(..)
            procedure C(..)
            begin
            ..
            end
            procedure D(..)
            begin –procedure D
            C(..);
            end
    begin –procedure B
        D(..)
    end

    procedure E(..)
    begin –procedure E
      B(..);
    end
begin –procedure A
E(..);
end
```



Example

```
procedure main
    g : integer
    procedure B(a : integer)
        x : integer
        procedure A(n : integer)
            g := n
        procedure R(m : integer)
            write integer(x)
            x /:= 2 – – integer division
            if x > 1
                R(m + 1)
            else
                A(m)
        – – body of B
        x := a * a
        R(1)
    – – body of main
    B(3)
    write integer(g)
```

Functional Programming

- Model of computation is the evaluation of functions analogous to mathematical functions
- Use declarations and expressions rather than statements and instructions
- Generally avoids mutable data – variables represent values not memory locations
- Heavy use of recursion instead of loops
- Functions are first-class values

First Class Functions

- Necessity of Functional Languages.
- Functions can be passed as arguments to other functions.
- Return functions as output of other functions.
- Assign them to variables or store them in any data structure.

Closures

- Functions passed as parameters are passed using closures.
- Closures is a combination of a code pointer and an environment pointer

- The environment pointer (EP) is a static link and points to the procedure where the parameter function is defined.
- All closures go on heap.

Example

```
program main
    procedure A(I : integer; procedure P);
        procedure B;
        begin
            writeln(I);
        end;
    begin (* A *)
        if I > 1 then
            P
        else
            A(2, B);
    end;
    procedure C; begin end;
begin (* main *)
    A(1, C);
end.
```

Recursive thinking

- Step1: Handle Base Case
- Step2: Assume the function works on input of size n-1
- Step3: Create the solution for size n under above assumption.

Examples

- Fibonacci series
- Factorial
- Length of a list
- Print a list
- Append 2 lists
- Merge function for 2 lists
- Reversing a list
- Insert element at the rear of the list
- Search element in a list
- Max of a list

Installing Racket (R5RS)

To install Racket:

- Go to http://racket-lang.org and click "Download" to download and install the Racket environment for your computer.
- The IDE application is called "Dr. Racket". The first time you run it, you must choose the right language, i.e. R5RS Scheme, for it to support. To do so, click on the drop-down menu at the very bottom left of the Dr. Racket window. Click "Choose Language > Other Languages > R5RS", and then "OK". At the bottom left of the Dr. Racket window, you should now see "R5RS".