Chapter 12 -- Runtime Support (Storage)

- Static allocation (global)
  - Each variable gets a static location
  - name, address
  - (name, data area, offset ... fortran)

- Stack allocation (local variables)
  - Activation Record -- managing the stack
    
    ```plaintext
    Procedure P (a:integer)
    var b: boolean;
    c: array [1:10] of integer;
    begin
    ...
    end;
    ```

  - Parameters: a
  - Local Vars: b, c
  - Other Info: return addr, ...
Stack allocation / variable location

Access to variable in Activation Record

- Usually some reference point in AR, Frame Pointer (FP)
- Some kind of offset from the reference point

Specifics are machine dependent

VAX (A very old machine)  HC (For ATL/1)

- LOW  c
  - b
- FP --> Ret/Reg
- AP --> 1/ptr
  - a
  - HIGH

- a: special access via AP
- b: -4 relative to FP
- c: -44 relative to FP

- HIGH  c
  - b
- FP --> Ret/Reg
  - a
  - LOW

- a: -2
- b: 1
- c: 2
Dynamic arrays & Static variables

Dynamic Arrays

- Size known only after procedure entry
- Fixed entry in AR (usually pointer and size)
- Variable size entry pushed on AR at run time

Static / Own Variables

- Static (C), Own (algol)
- Allocate in global Data
Program P;
  Var a;

Procedure R;
  Var b;

Procedure S;
  Var c;
  begin
    a <-- b+c;
  end;

begin
  begin
  end

begin
  end

Local: c
Global: a
Not local, not global: b
Display: Draw picture
Static Chains: Draw picture
hc and display

- 16 display registers (Fixed)
- One for each static level

- load l, value  =>  load address d[l] + value

- call l, spaddr
  - push d[l]
  - push return-addr
  - d[l] <-- SP
  - PC <-- spaddr

- exit l, n
  - SP <-- d[l]
  - PC <-- Pop()
  - d[l] <-- Pop()
  - SP -= n
Static Chains

A non-local access method

☐ No display
☐ Each AR has pointer to enclosing static AR
☐ Access to non-local variables are via static pointers.
☐ Follow pointers to AR of variable.
☐ May save pointers to each AR on the way
☐ Advantages?
☐ Disadvantages?
Heap Allocation

- Dynamic memory allocation (not heap data structure)
- Not stored on the runtime stack.
- Explicit versions
  - Pascal, Ada, New
- Implicit
  - Snobol, Java

Total Memory Layout

- Traditional -- i386/ns32k/...
- HC