Activation Records for “Simple” Programs

**Figure 10.1**
An activation record for simple subprograms

**Figure 10.2**
The code and activation records of a program with simple subprograms

**More Complex Activation Records.**
(Used with Stack-Dynamic Local Variables)

**Figure 10.3**
A typical activation record for a language with stack-dynamic local variables

**Figure 10.4**
The activation record for function `sub`
An Example without Recursion

```c
void fun1(float r) {
    int s, t;
    ...
    fun2(s);
    ...
}

void fun2(int x) {
    int y;
    ...
    fun3(y);
    ...
}

void fun3(int q) {
    ...
}

void main() {
    float p;
    fun1(p);
    ...
}
```

Figure 10.5
Stack contents for three points in a program
A Recursion Example

```c
int factorial(int n) {
    if (n <= 1)
        return 1;
    else return (n * factorial(n - 1));
}

void main() {
    int value;
    value = factorial(3);
}
```

---

### Figure 10.6
The activation record for factorial

- **n**
- Parameter
- Dynamic link
- Return address

---

### Figure 10.7
Stack contents at position 1 in factorial

---

### Figure 10.8
Stack contents during execution of main and factorial

---

e.g. calling with: `factorial(3)`
Nested Subprograms, with Non-Local Variables

procedure A is
    procedure B is
        procedure C is
            ...
            end; -- of C
            ...
            end; -- of B
            ...
            end; -- of A
end; -- of Main_2

procedure Main_2 is
    X : Integer;
    procedure Bigsub is
        A, B, C : Integer;
        procedure Sub1 is
            A, D : Integer;
            begin -- of Sub1
                A := E + C;
                ...
            end; -- of Sub1
        procedure Sub2(X : Integer) is
            B, E : Integer;
            procedure Sub3 is
                C, E : Integer;
                begin -- of Sub3
                    ...
                    Sub1;
                    ...
                    E := B + A;
                end; -- of Sub3
                begin -- of Sub2
                    ...
                    Sub3;
                    ...
                    A := D + E;
                end; -- of Sub2
                begin -- of Bigsub
                    ...
                    Sub2(?);
                    ...
                    end; -- of Bigsub
                begin -- of Main_2
                    ...
                    Bigsub;
                    ...
                end; -- of Main_2

Think of static chain lengths between various procedures.
Calculate below the (chainLength, offset) pairs for various variables.

Figure 10.9
Stack contents at position 1 in the program Main_2

ARI = activation record instance
Blocks as Parameterless Subprograms

```c
{ int temp;
  temp = list[upper];
  list[upper] = list[lower];
  list[lower] = temp;
}

void main() {
  int x, y, z;
  while ( ... ) {
    int a, b, c;
    ...
    while ( ... ) {
      int d, e;
      ...
    }
  }
  while ( ... ) {
    int f, g;
    ...
  }
  ...
}
```

**Figure 10.10**
Block variable storage when blocks are not treated as parameterless procedures
Dynamic Scoping

```c
void sub3() {
    int x, z;
    x = u + v;
    ...
}

void sub2() {
    int w, x;
    ...
}

void sub1() {
    int v, w;
    ...
}

void main() {
    int v, u;
    ...
}
```

**Figure 10.11**
Stack contents for a dynamic-scoped program

ARI = activation record instance