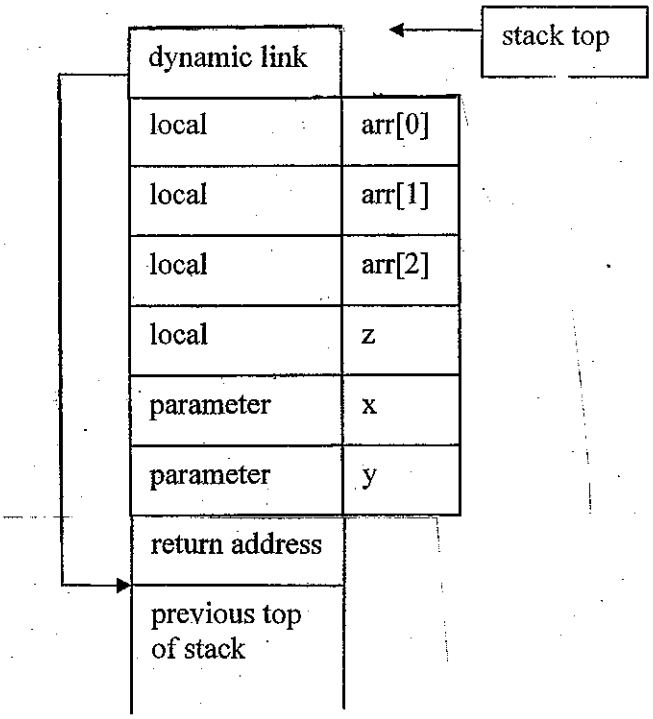
Subprograms: Implementation

Static Allocation (Fortran) : simple, no recursion

Dynamic Allocation – most block-structured languages

Activation Record (on stack)

Ex.
```
void fred (float x, int y) {
    int arr[3], z;
      - - -
}
```



fred's Activation Record Instance (ARI)

Recursive Procedure

Example. Factorial (n)

```
int fact (int n) {
    if (n ≤ 1) return 1;
    else return (n * fact(n – 1) );
}

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

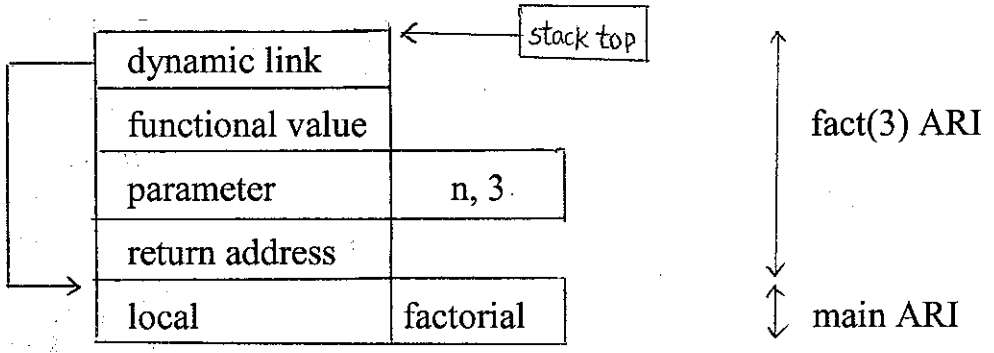
fact's activation record:

dynamic link	
functional value	
parameter	n
return address	

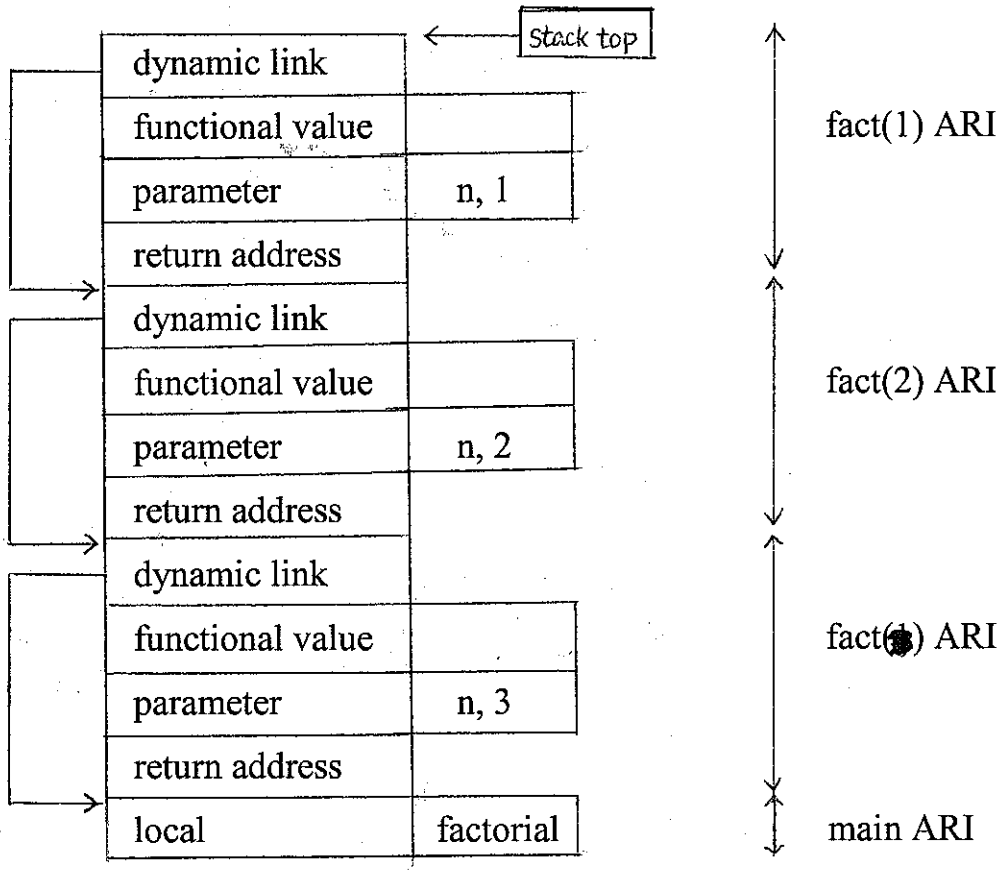
main's activation record:

local	factorial

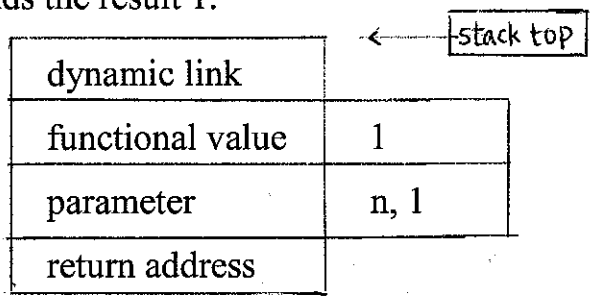
main calls fact(3):



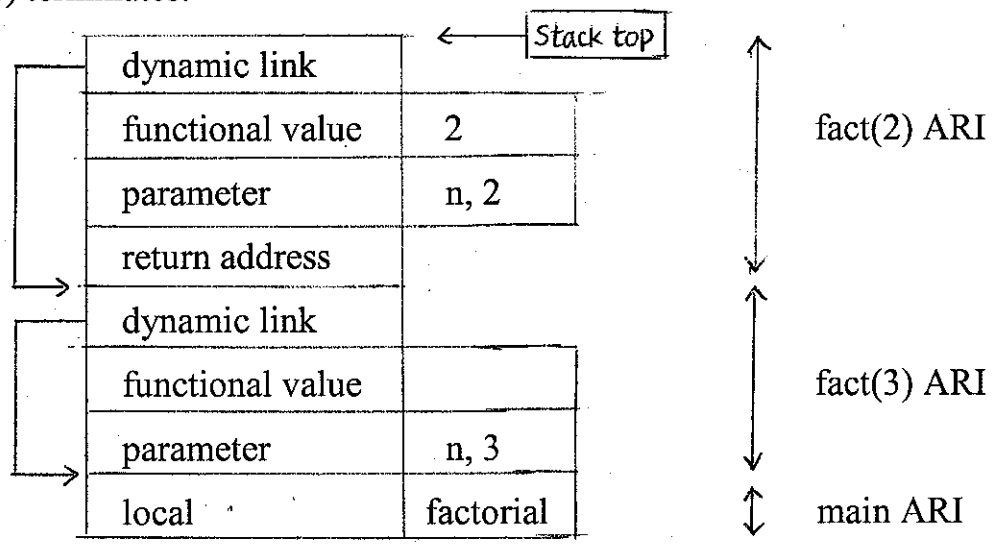
fact(3) calls fact(2) calls fact(1):



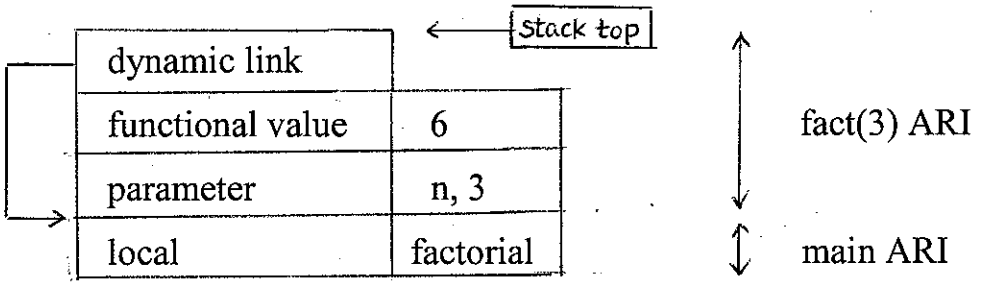
fact(1) finds the result 1.



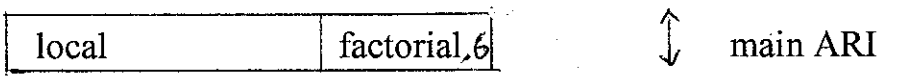
fact(1) terminates.



fact(2) terminates.



fact(3) terminates.



Nested Subprograms

Ex.

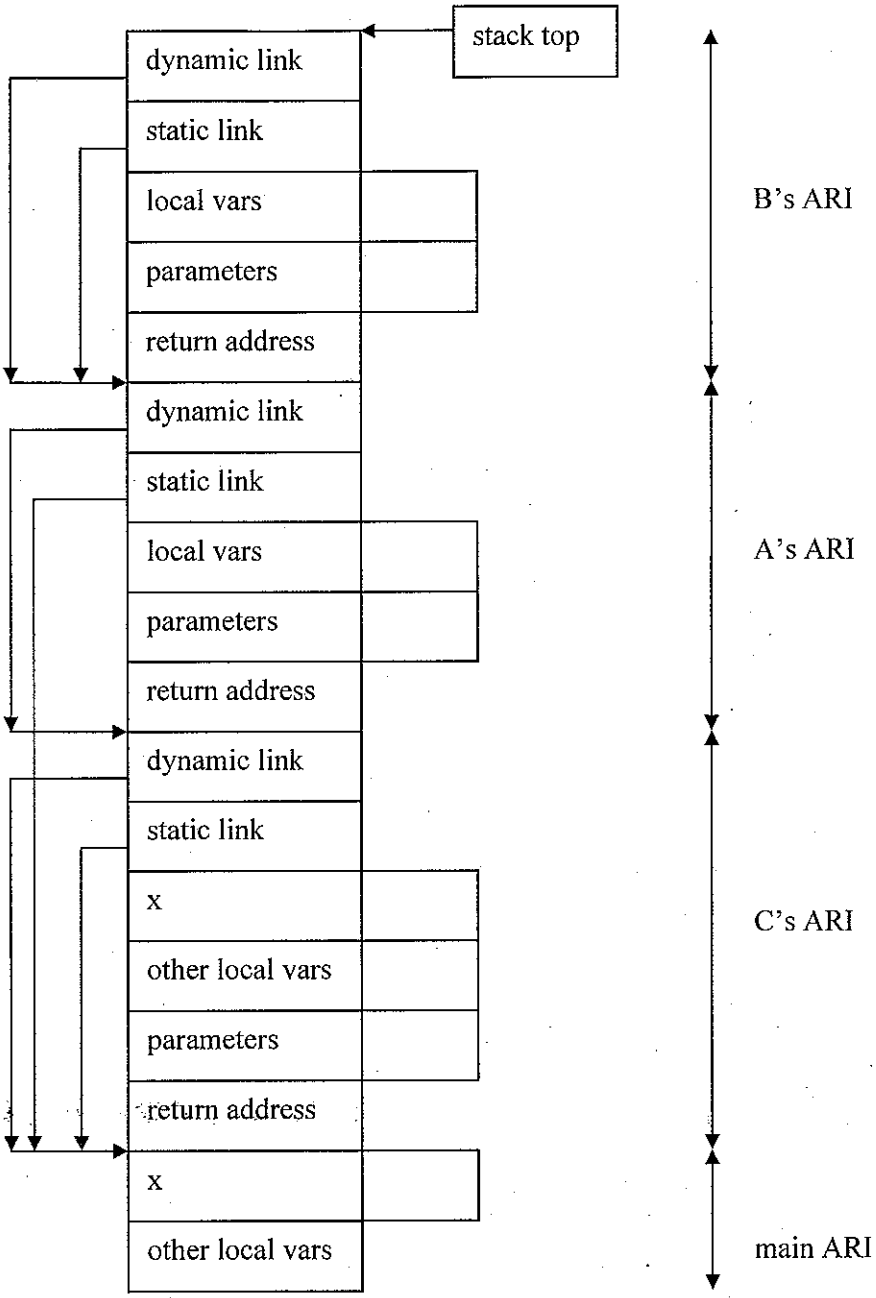
```
main
  var x
  subprogram A
    subprogram B
      x = 5      <---------- (*)

  subprogram C
    var x
```

main → C → A → B

Dynamic binding - dynamic link

Static binding - static link


stack top
dynamic link
static link
local vars
parameters
return address
dynamic link
static link
local vars
parameters
return address
dynamic link
static link
x
other local vars
parameters
return address
x
other local vars
B's ARI
A's ARI
C's ARI
main ARI

P456 Error Correction

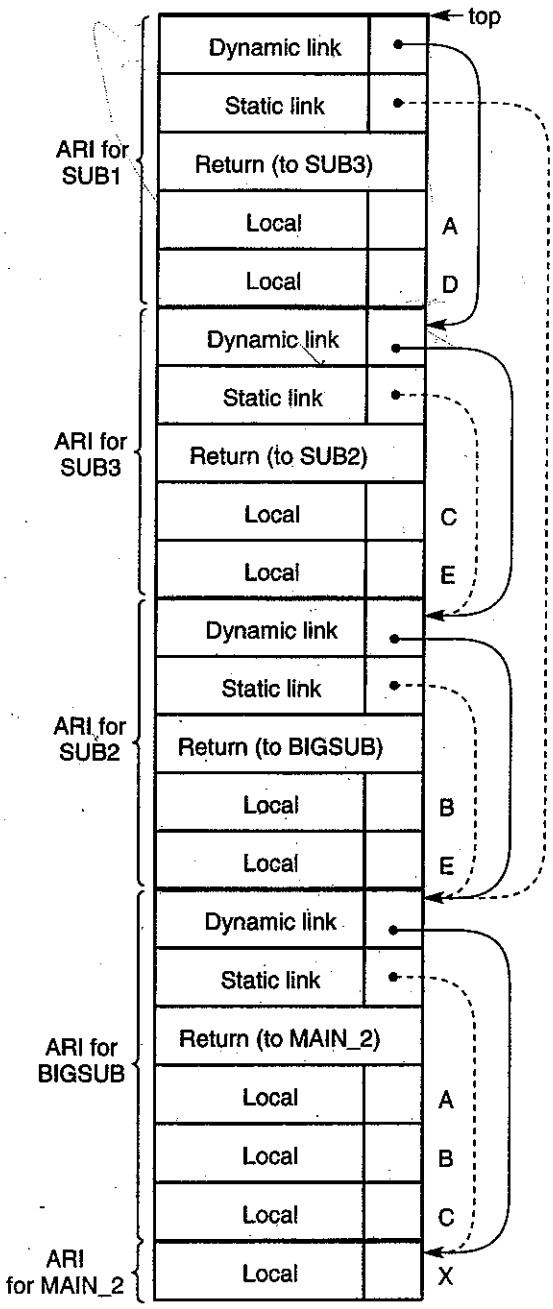


Figure 9.9
Stack contents at position 1 in the program MAIN_2