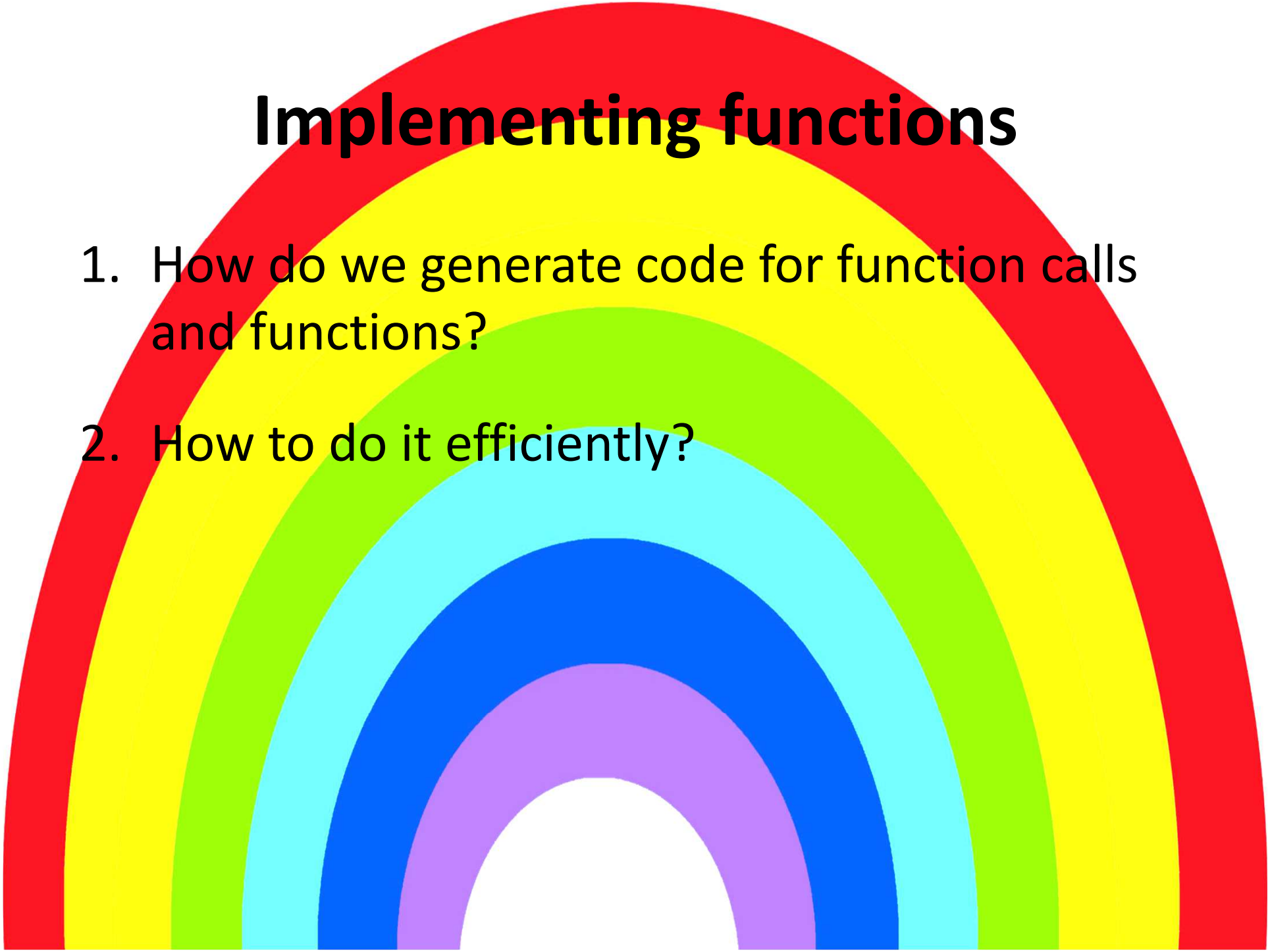
Implementing functions



1. How do we generate code for function calls and functions?
2. How to do it efficiently?

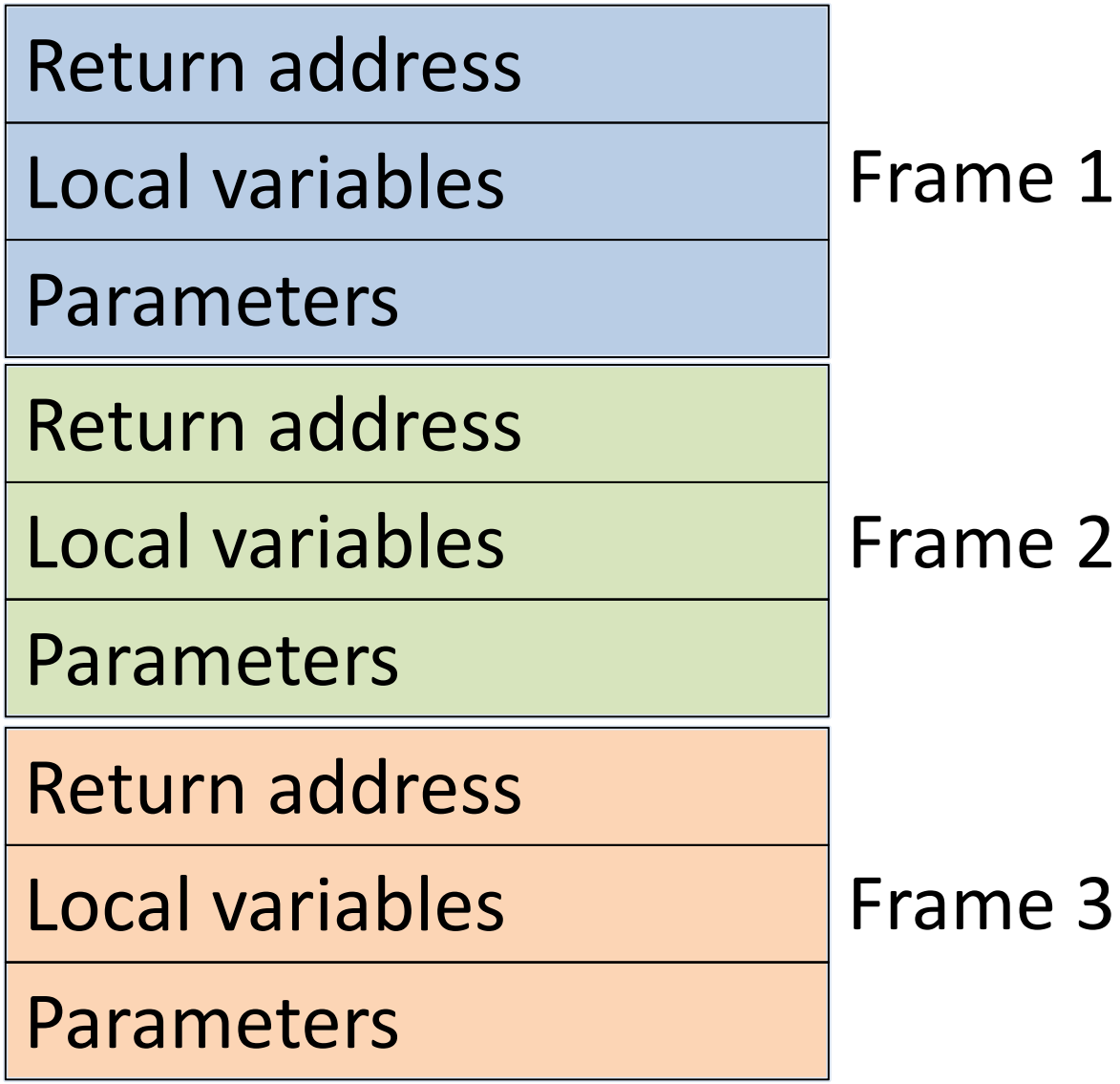
Implementing functions

1. How do we generate code for function calls and functions?
   - Jump to function and return
   - Pass parameters to function
   - Get return value from function
   - Allocate local variables for use in function
2. How to do it efficiently?

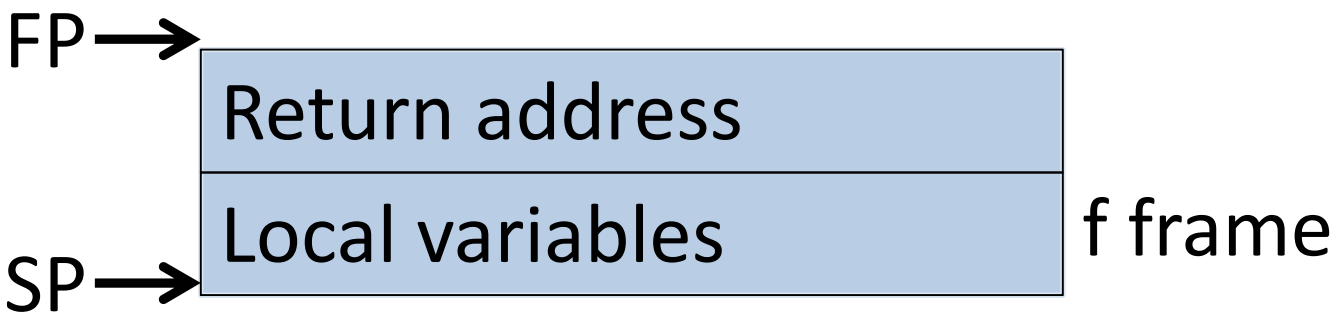
Implementing functions

- Need a stack, to allow recursive function calls
- One stack frame for each function invocation
- FP (frame pointer) points to beginning of current stack frame
- SP (stack pointer) points to current top of stack
  - Not the same as frame pointer

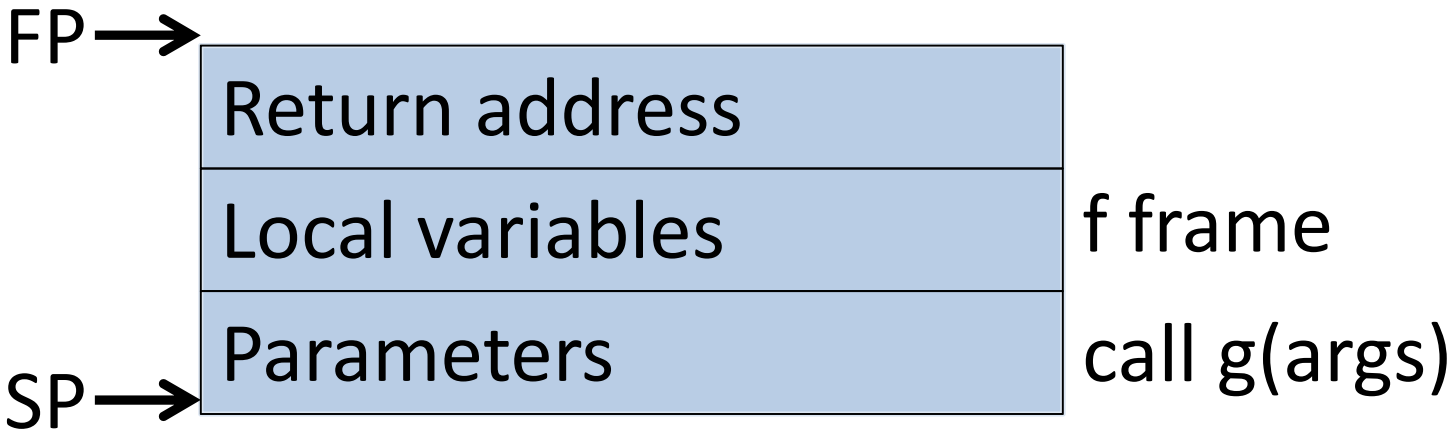
Implementing functions: simple



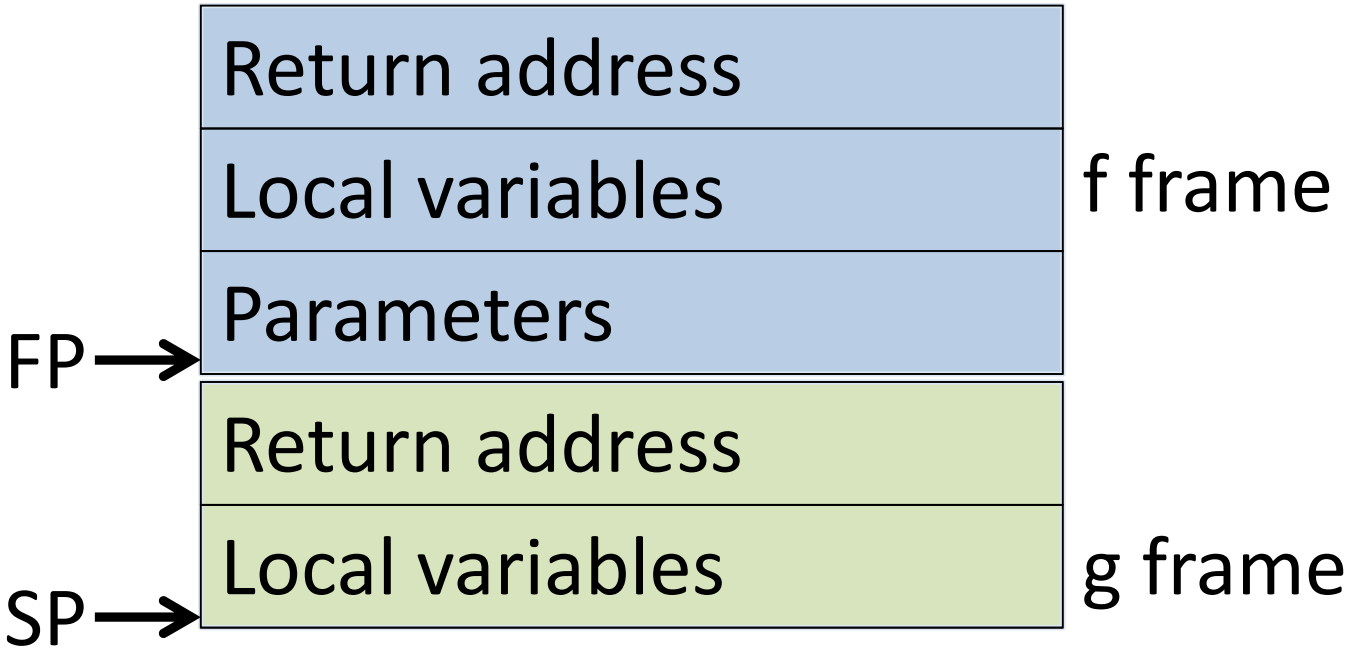
Implementing functions: simple



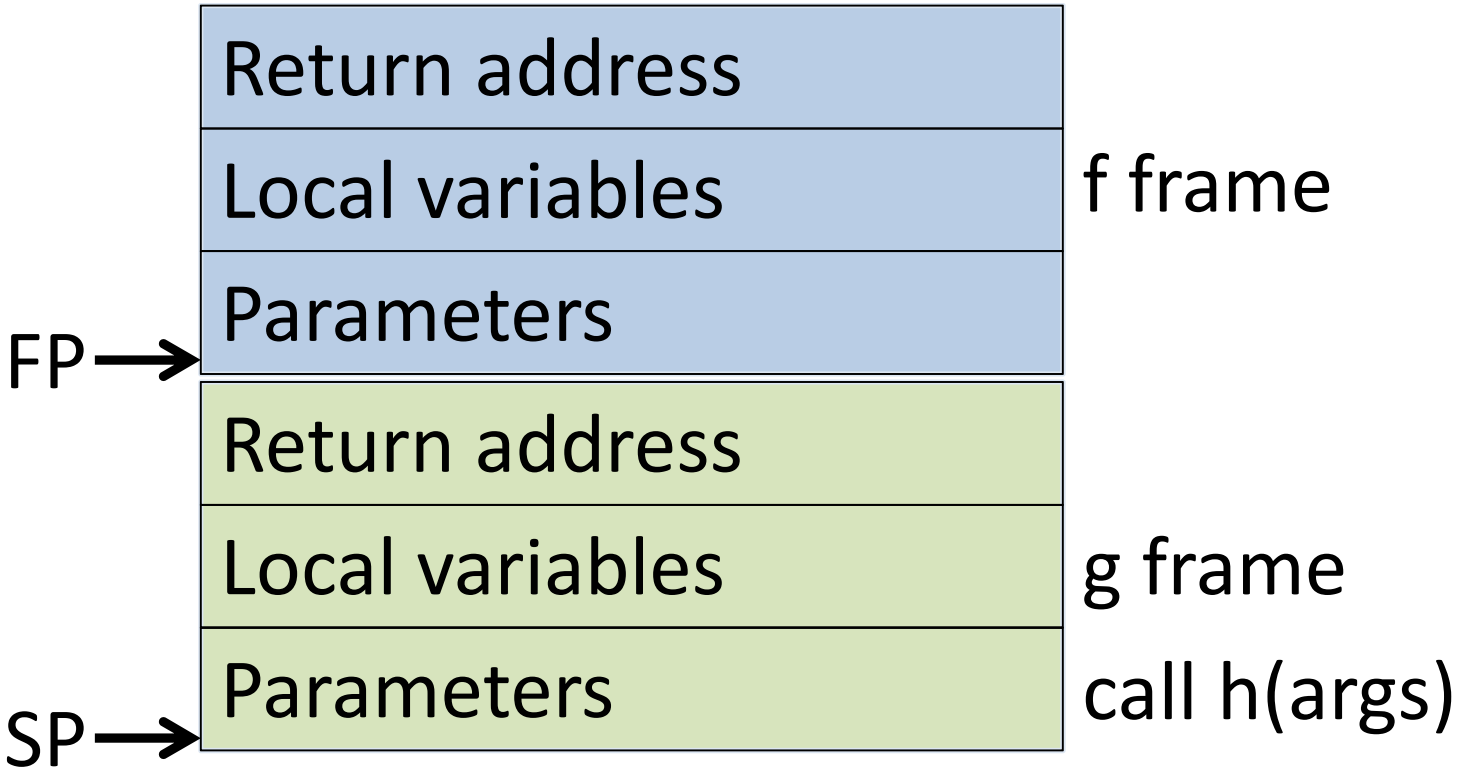
Implementing functions: simple



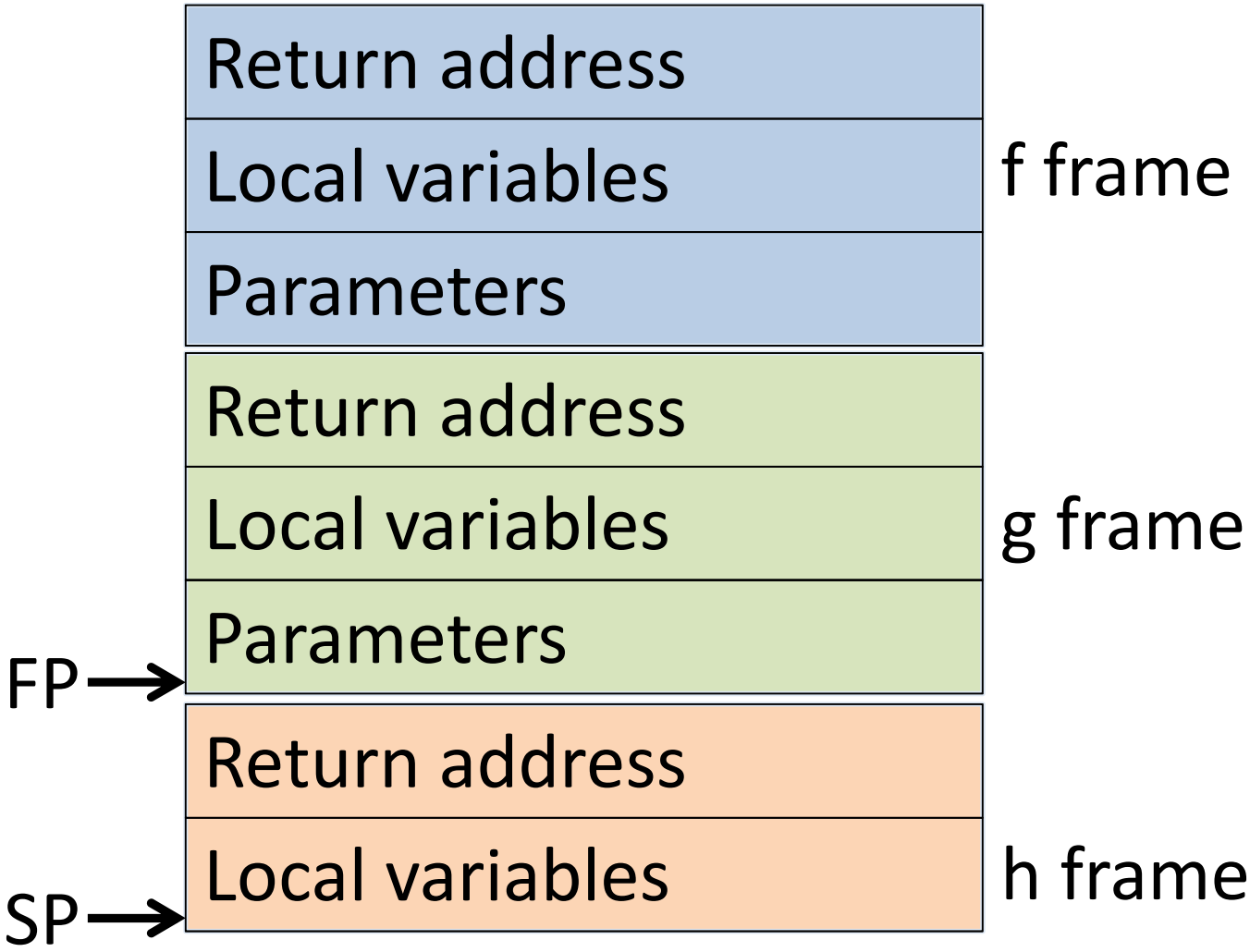
Implementing functions: simple



Implementing functions: simple



Implementing functions: simple



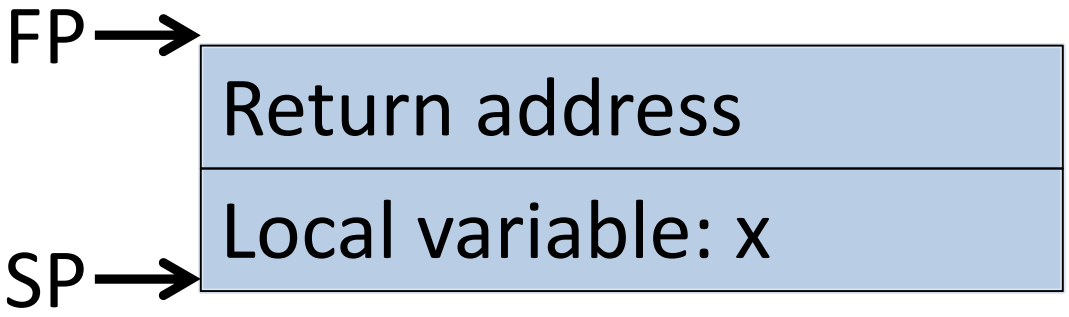
Implementing functions: simple

Example:

```
x = sqr(9)


function sqr(v)
r = v * v
return r
```

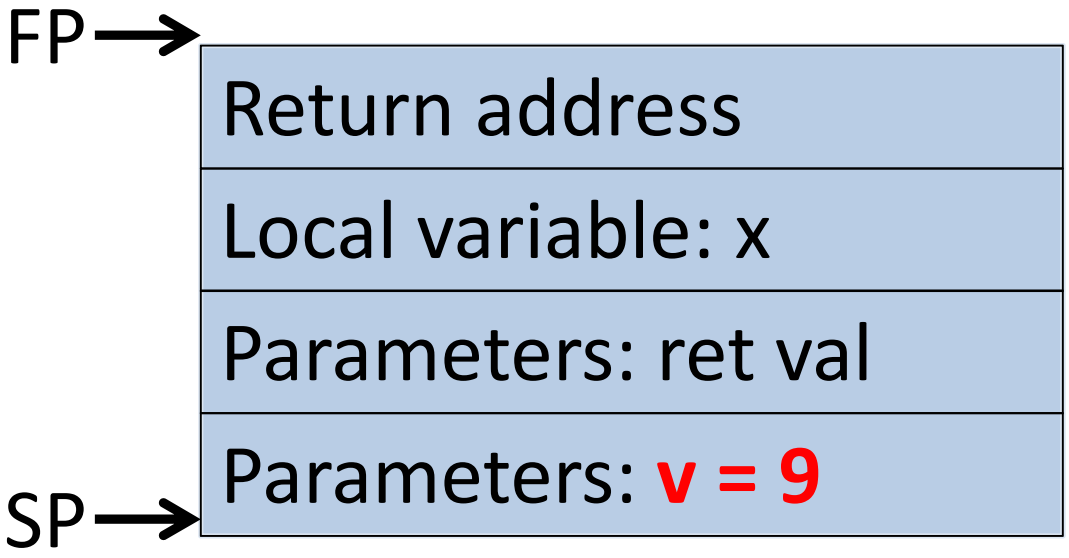
Implementing functions: simple




```
      ADDI R1,R0,9
      STORE FP,-16,R1
      STORE FP,-20,L1
      JMP SQ
L1:   LOAD R1,FP,-12
      STORE FP,-8,R1

SQ:   ADDI FP,SP,4
      LOAD R1,FP,0
      LOAD R2,FP,0
      MUL R1,R1,R2
      STORE FP,-8,R1
      LOAD R1,FP,-8
      STORE FP,4,R1
      LOAD R1,FP,-4
      ADDI FP,FP,16
      JUMP R1
```

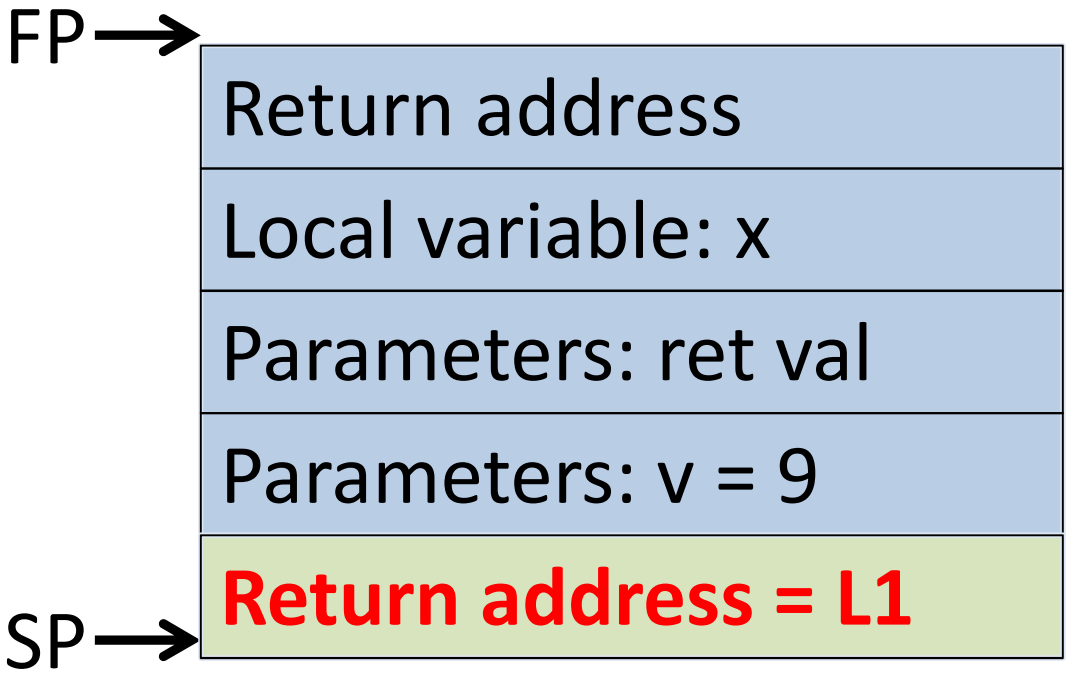
Implementing functions: simple




```
     ADDI R1,R0,9
     STORE FP,-16,R1
     STORE FP,-20,L1
     JMP SQ
L1:  LOAD R1,FP,-12
     STORE FP,-8,R1

SQ:  ADDI FP,SP,4
     LOAD R1,FP,0
     LOAD R2,FP,0
     MUL R1,R1,R2
     STORE FP,-8,R1
     LOAD R1,FP,-8
     STORE FP,4,R1
     LOAD R1,FP,-4
     ADDI FP,FP,16
     JUMP R1
```

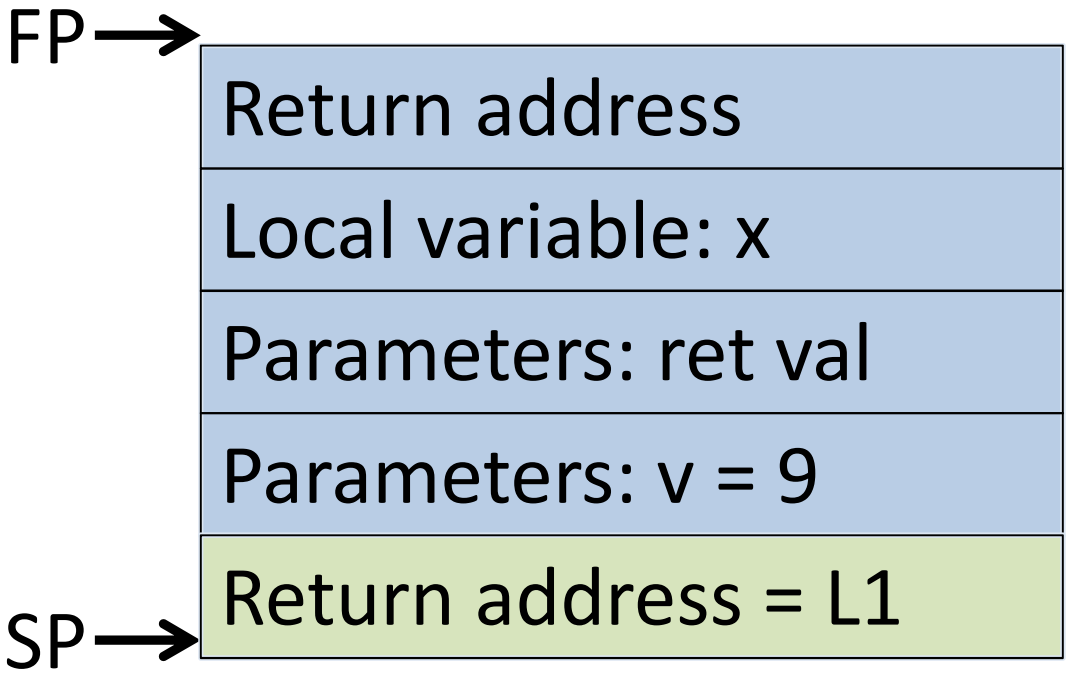
Implementing functions: simple




```
      ADDI R1,R0,9
      STORE FP,-16,R1
      STORE FP,-20,L1
      JMP SQ
L1:   LOAD R1,FP,-12
      STORE FP,-8,R1

SQ:   ADDI FP,SP,4
      LOAD R1,FP,0
      LOAD R2,FP,0
      MUL R1,R1,R2
      STORE FP,-8,R1
      LOAD R1,FP,-8
      STORE FP,4,R1
      LOAD R1,FP,-4
      ADDI FP,FP,16
      JUMP R1
```

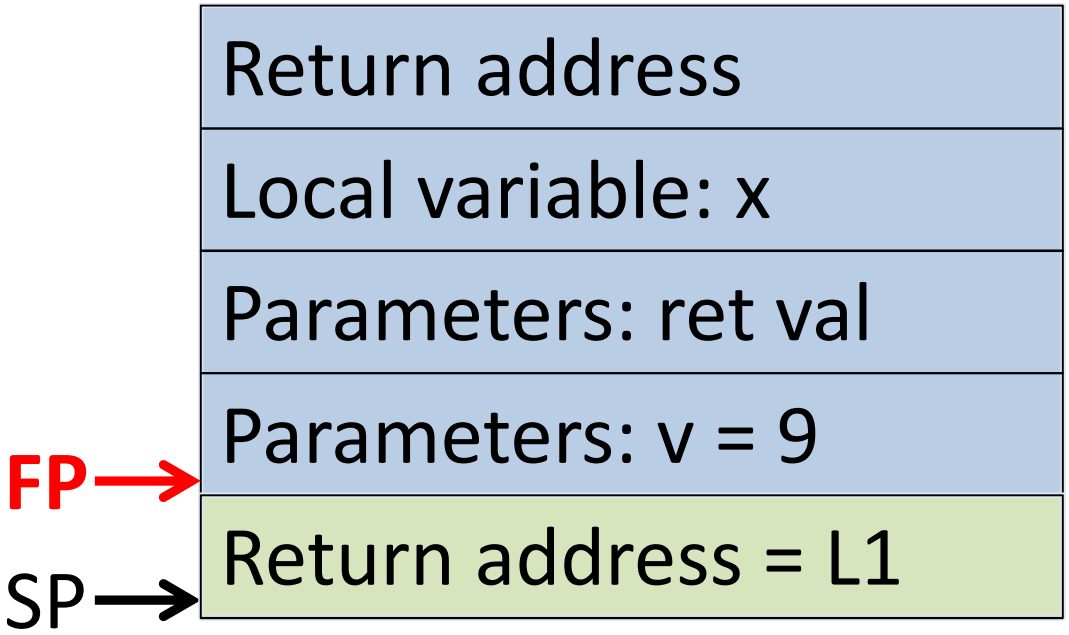
Implementing functions: simple




```
      ADDI R1,R0,9
      STORE FP,-16,R1
      STORE FP,-20,L1
      JMP SQ
L1:   LOAD R1,FP,-12
      STORE FP,-8,R1

SQ:   ADDI FP,SP,4
      LOAD R1,FP,0
      LOAD R2,FP,0
      MUL R1,R1,R2
      STORE FP,-8,R1
      LOAD R1,FP,-8
      STORE FP,4,R1
      LOAD R1,FP,-4
      ADDI FP,FP,16
      JUMP R1
```

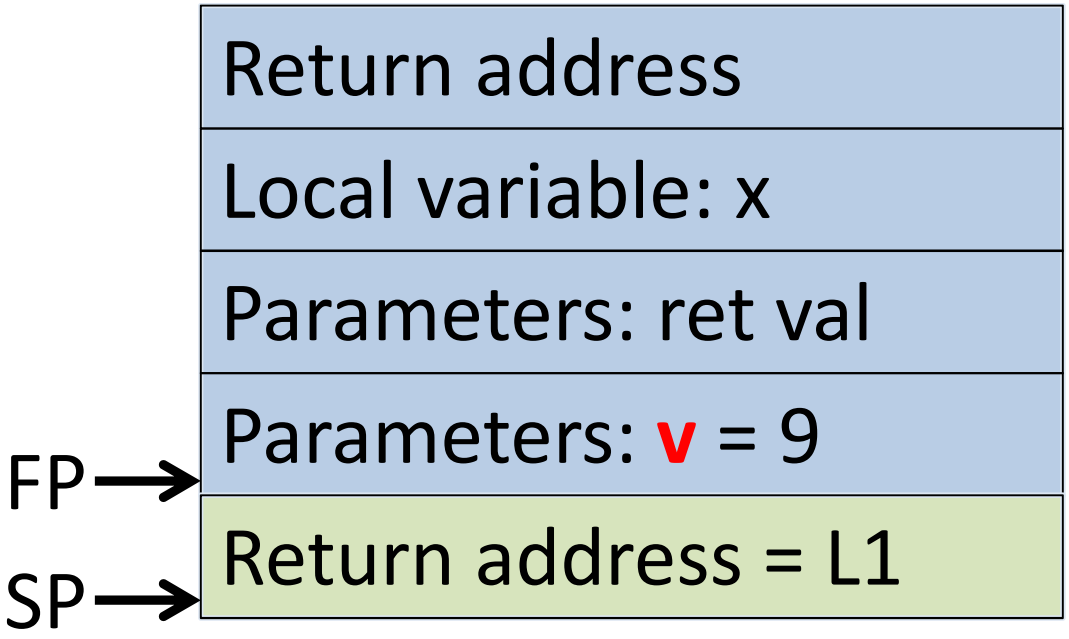
Implementing functions: simple




```
      ADDI R1,R0,9
      STORE FP,-16,R1
      STORE FP,-20,L1
      JMP SQ
L1:   LOAD R1,FP,-12
      STORE FP,-8,R1

SQ:   ADDI FP,SP,4
      LOAD R1,FP,0
      LOAD R2,FP,0
      MUL R1,R1,R2
      STORE FP,-8,R1
      LOAD R1,FP,-8
      STORE FP,4,R1
      LOAD R1,FP,-4
      ADDI FP,FP,16
      JUMP R1
```

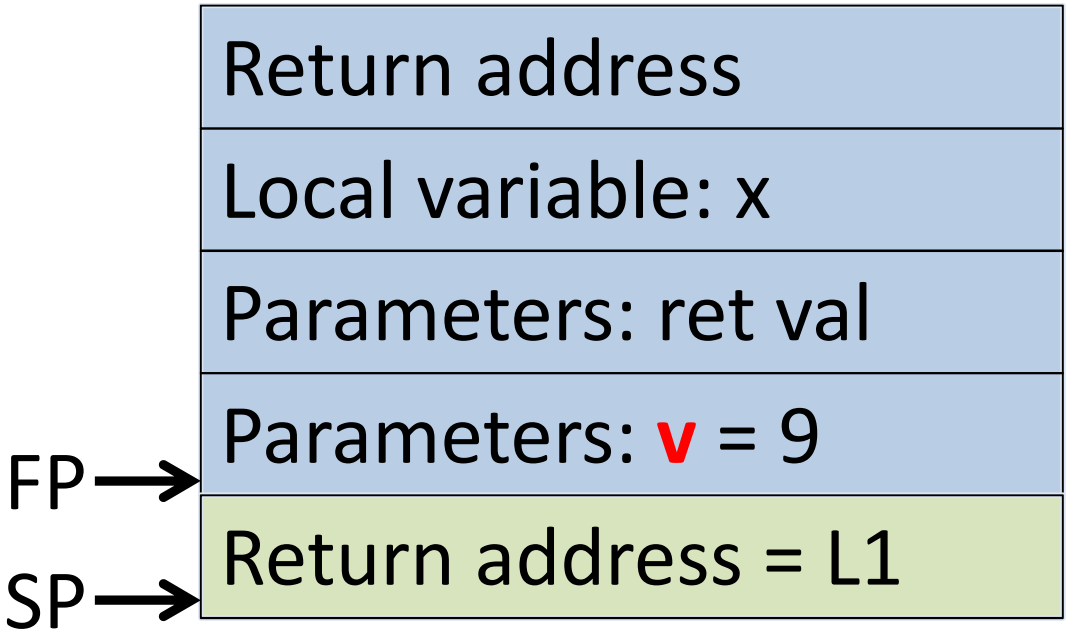
Implementing functions: simple




```
      ADDI R1,R0,9
      STORE FP,-16,R1
      STORE FP,-20,L1
      JMP SQ
L1:   LOAD R1,FP,-12
      STORE FP,-8,R1

SQ:   ADDI FP,SP,4
      LOAD R1,FP,0
      LOAD R2,FP,0
      MUL R1,R1,R2
      STORE FP,-8,R1
      LOAD R1,FP,-8
      STORE FP,4,R1
      LOAD R1,FP,-4
      ADDI FP,FP,16
      JUMP R1
```

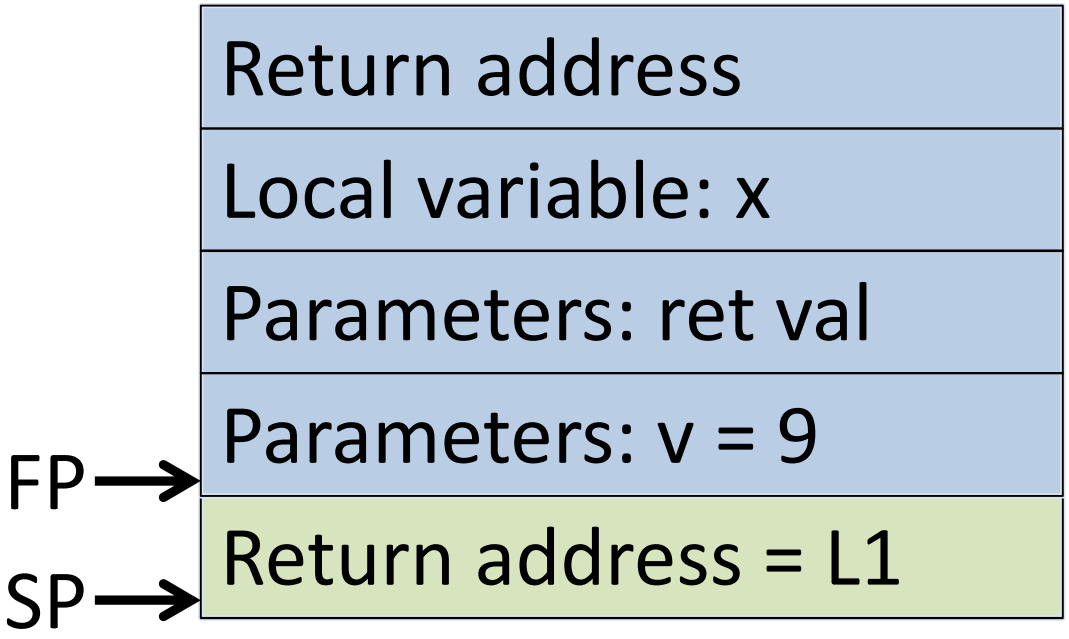
Implementing functions: simple




```
      ADDI R1,R0,9
      STORE FP,-16,R1
      STORE FP,-20,L1
      JMP SQ
L1:   LOAD R1,FP,-12
      STORE FP,-8,R1

SQ:   ADDI FP,SP,4
      LOAD R1,FP,0
      LOAD R2,FP,0
      MUL R1,R1,R2
      STORE FP,-8,R1
      LOAD R1,FP,-8
      STORE FP,4,R1
      LOAD R1,FP,-4
      ADDI FP,FP,16
      JUMP R1
```

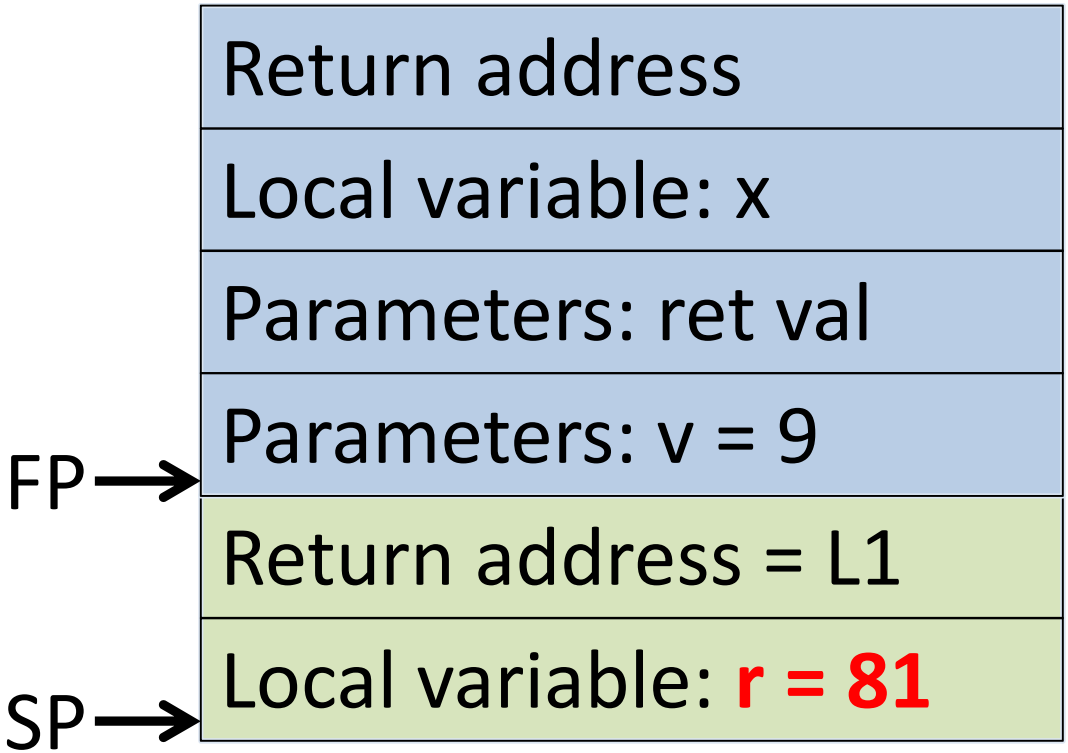
Implementing functions: simple




```
      ADDI R1,R0,9
      STORE FP,-16,R1
      STORE FP,-20,L1
      JMP SQ
L1:   LOAD R1,FP,-12
      STORE FP,-8,R1

SQ:   ADDI FP,SP,4
      LOAD R1,FP,0
      LOAD R2,FP,0
      MUL R1,R1,R2
      STORE FP,-8,R1
      LOAD R1,FP,-8
      STORE FP,4,R1
      LOAD R1,FP,-4
      ADDI FP,FP,16
      JUMP R1
```

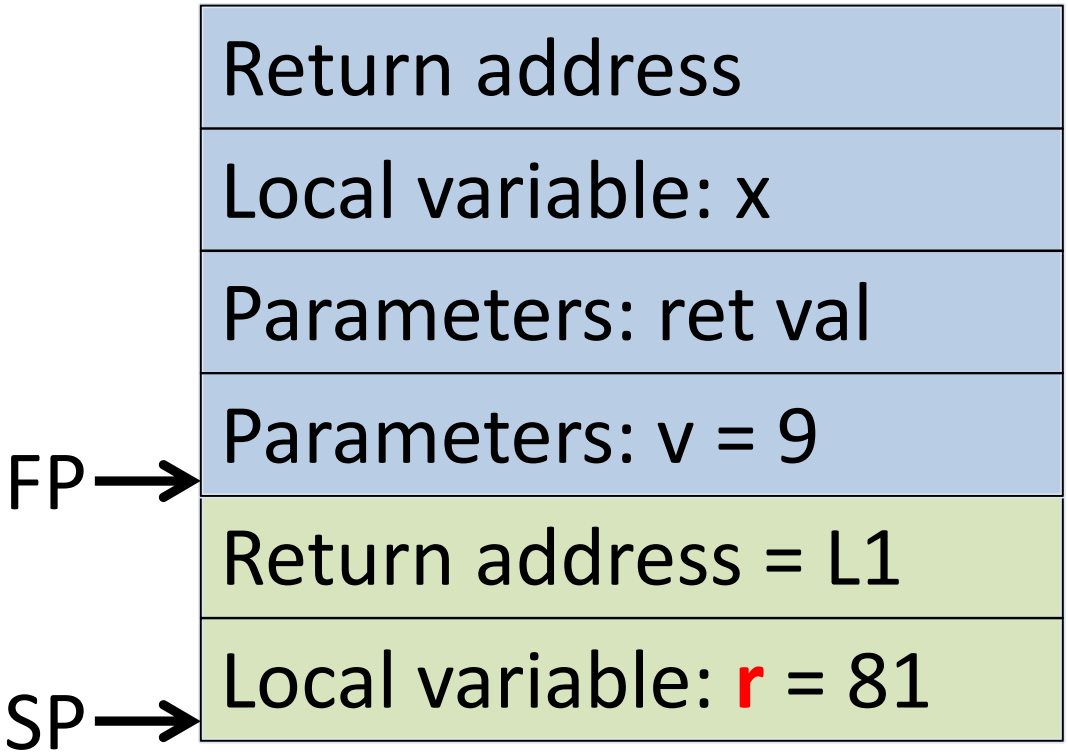
Implementing functions: simple




```
     ADDI R1,R0,9
     STORE FP,-16,R1
     STORE FP,-20,L1
     JMP SQ
L1:  LOAD R1,FP,-12
     STORE FP,-8,R1

SQ:  ADDI FP,SP,4
     LOAD R1,FP,0
     LOAD R2,FP,0
     MUL R1,R1,R2
     STORE FP,-8,R1
     LOAD R1,FP,-8
     STORE FP,4,R1
     LOAD R1,FP,-4
     ADDI FP,FP,16
     JUMP R1
```

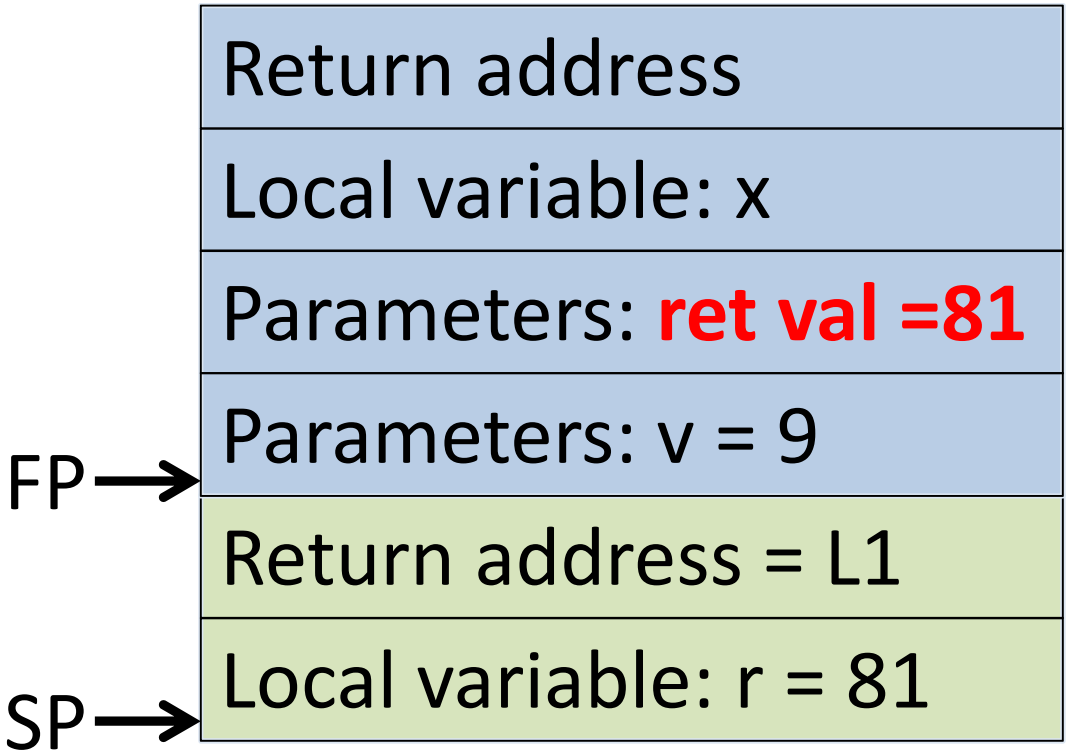
Implementing functions: simple




```
      ADDI R1,R0,9
      STORE FP,-16,R1
      STORE FP,-20,L1
      JMP SQ
L1:   LOAD R1,FP,-12
      STORE FP,-8,R1

SQ:   ADDI FP,SP,4
      LOAD R1,FP,0
      LOAD R2,FP,0
      MUL R1,R1,R2
      STORE FP,-8,R1
      LOAD R1,FP,-8
      STORE FP,4,R1
      LOAD R1,FP,-4
      ADDI FP,FP,16
      JUMP R1
```

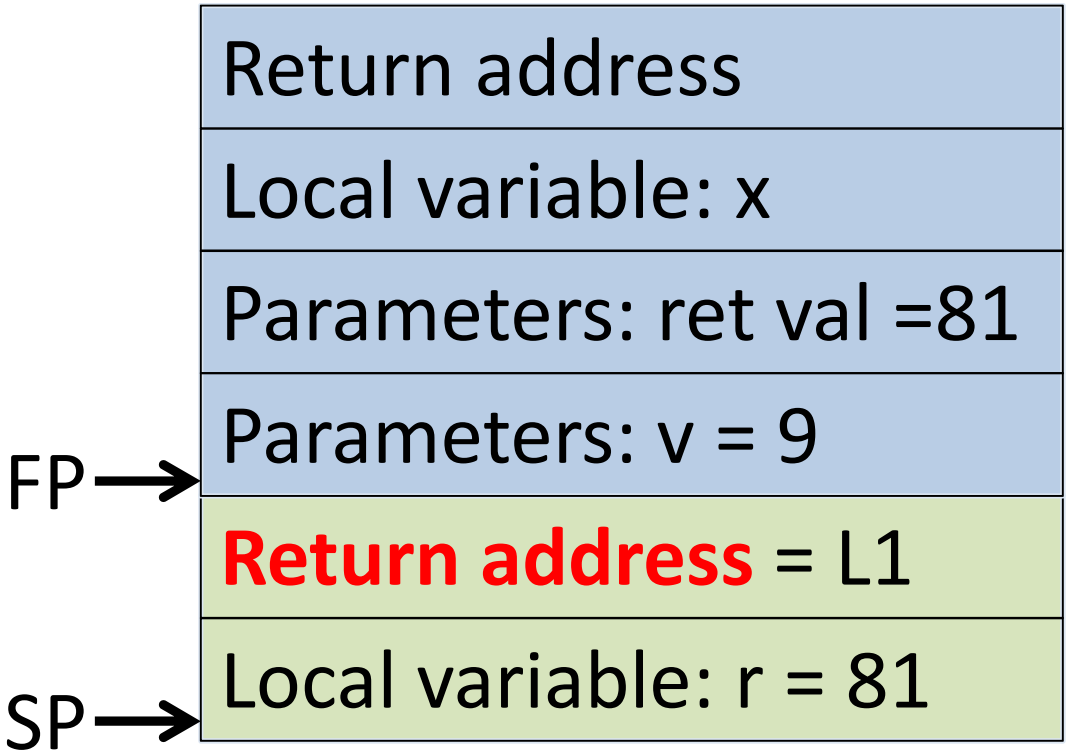
Implementing functions: simple




```
      ADDI R1,R0,9
      STORE FP,-16,R1
      STORE FP,-20,L1
      JMP SQ
L1:   LOAD R1,FP,-12
      STORE FP,-8,R1

SQ:   ADDI FP,SP,4
      LOAD R1,FP,0
      LOAD R2,FP,0
      MUL R1,R1,R2
      STORE FP,-8,R1
      LOAD R1,FP,-8
      STORE FP,4,R1
      LOAD R1,FP,-4
      ADDI FP,FP,16
      JUMP R1
```

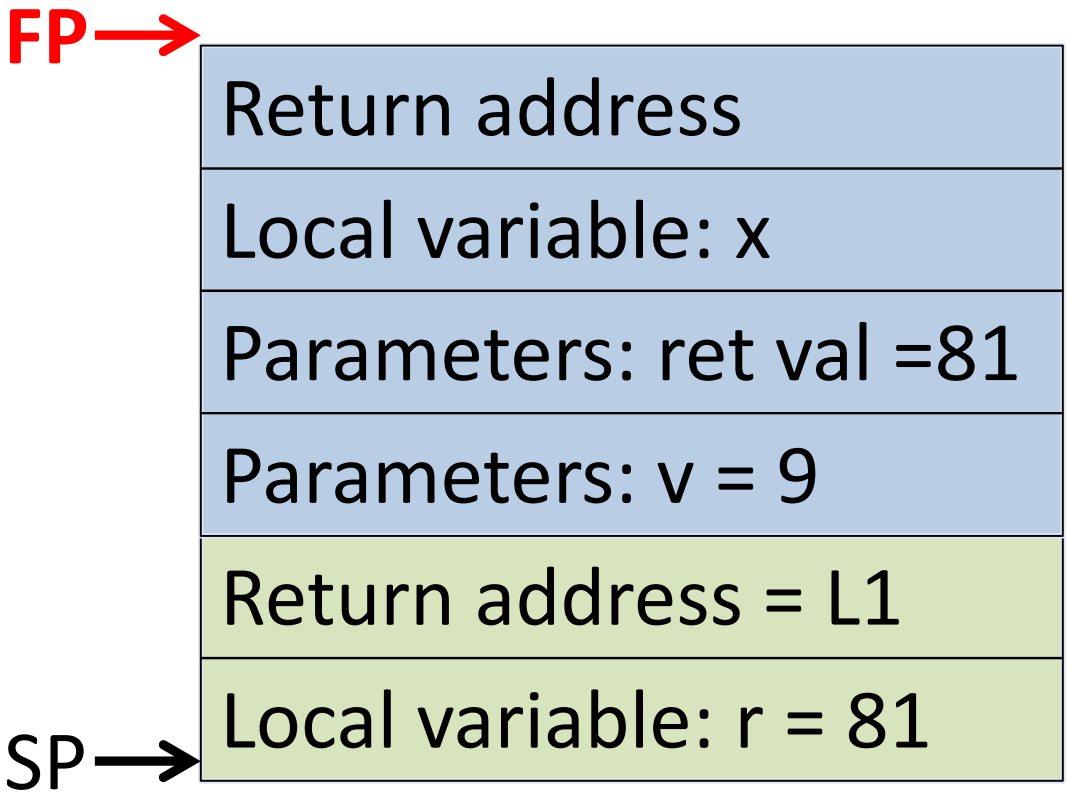
Implementing functions: simple




```
      ADDI R1,R0,9
      STORE FP,-16,R1
      STORE FP,-20,L1
      JMP SQ
L1:   LOAD R1,FP,-12
      STORE FP,-8,R1

SQ:   ADDI FP,SP,4
      LOAD R1,FP,0
      LOAD R2,FP,0
      MUL R1,R1,R2
      STORE FP,-8,R1
      LOAD R1,FP,-8
      STORE FP,4,R1
      LOAD R1,FP,-4
      ADDI FP,FP,16
      JUMP R1
```

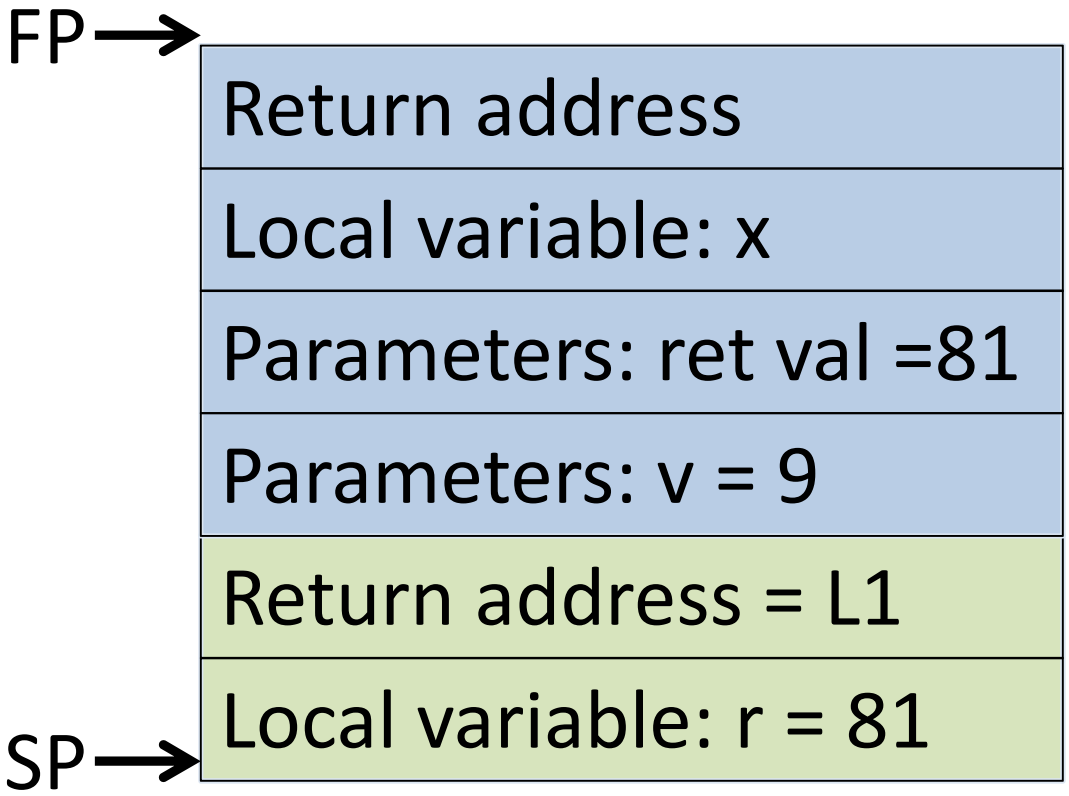
Implementing functions: simple




```
      ADDI R1,R0,9
      STORE FP,-16,R1
      STORE FP,-20,L1
      JMP SQ
L1:   LOAD R1,FP,-12
      STORE FP,-8,R1

SQ:   ADDI FP,SP,4
      LOAD R1,FP,0
      LOAD R2,FP,0
      MUL R1,R1,R2
      STORE FP,-8,R1
      LOAD R1,FP,-8
      STORE FP,4,R1
      LOAD R1,FP,-4
      ADDI FP,FP,16
      JUMP R1
```

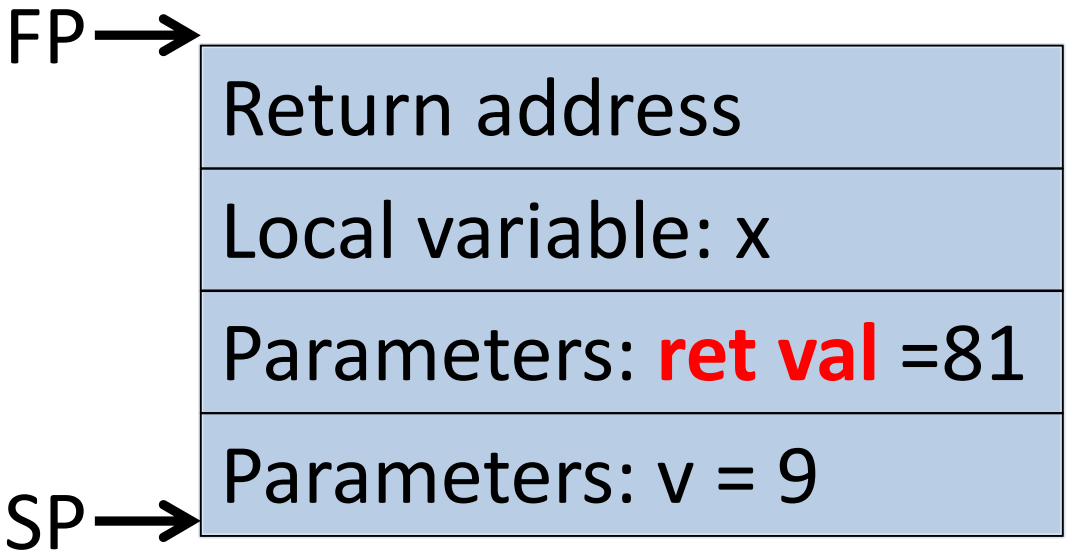
Implementing functions: simple




```
      ADDI R1,R0,9
      STORE FP,-16,R1
      STORE FP,-20,L1
      JMP SQ
L1:   LOAD R1,FP,-12
      STORE FP,-8,R1

SQ:   ADDI FP,SP,4
      LOAD R1,FP,0
      LOAD R2,FP,0
      MUL R1,R1,R2
      STORE FP,-8,R1
      LOAD R1,FP,-8
      STORE FP,4,R1
      LOAD R1,FP,-4
      ADDI FP,FP,16
      JUMP R1
```

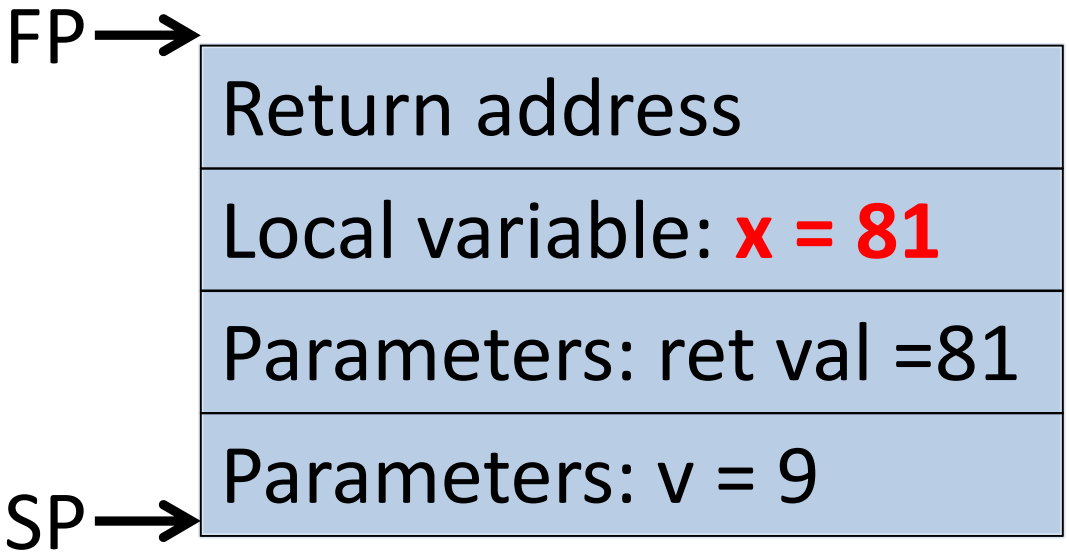
Implementing functions: simple




```
     ADDI R1,R0,9
     STORE FP,-16,R1
     STORE FP,-20,L1
     JMP SQ
L1:  LOAD R1,FP,-12
     STORE FP,-8,R1

SQ:  ADDI FP,SP,4
     LOAD R1,FP,0
     LOAD R2,FP,0
     MUL R1,R1,R2
     STORE FP,-8,R1
     LOAD R1,FP,-8
     STORE FP,4,R1
     LOAD R1,FP,-4
     ADDI FP,FP,16
     JUMP R1
```

Implementing functions: simple




```
      ADDI R1,R0,9
      STORE FP,-16,R1
      STORE FP,-20,L1
      JMP SQ
L1:   LOAD R1,FP,-12
      STORE FP,-8,R1

SQ:   ADDI FP,SP,4
      LOAD R1,FP,0
      LOAD R2,FP,0
      MUL R1,R1,R2
      STORE FP,-8,R1
      LOAD R1,FP,-8
      STORE FP,4,R1
      LOAD R1,FP,-4
      ADDI FP,FP,16
      JUMP R1
```

Implementing functions: simple

Problems?

Implementing functions: simple

Problems:

1. Slow: lot of memory accesses.

Implementing functions: simple

Problems:

1. Slow: lot of memory accesses.
2. Interferes with use of registers.

Implementing functions: registers

Example:

x = **1 + sqr(9)**

function sqr(v)
r = v * v
return r

Register could be overwritten by called function

```
      ADDI R1,R0,1
      ADDI R2,R0,9
      STORE FP,-16,R2
      STORE FP,-20,L1
      JMP SQ
L1:   LOAD R2,FP,-12
      ADD R1,R1,R2
      STORE FP,-8,R1

SQ:   ADDI FP,SP,4
      LOAD R1,FP,0
      LOAD R2,FP,0
      MUL R1,R1,R2
      ...
```

Implementing functions: registers

Solution: Caller-save

Register value saved on stack by caller

```
     ADDI R1,R0,1
     ADDI R2,R0,9
     STORE FP,-16,R2
     STORE FP,-20,L1
     STORE FP,-??,R1
     JMP SQ
L1:  LOAD R1,FP,-??
     LOAD R2,FP,-12
     ADD R1,R1,R2
     STORE FP,-8,R1

SQ:  ADDI FP,SP,4
     LOAD R1,FP,0
     LOAD R2,FP,0
     MUL R1,R1,R2
     ...
```

Implementing functions: registers

Solution: Caller-save

Register value saved on stack by caller

Useful if called function uses these registers

```
     ADDI R1,R0,1
     ADDI R2,R0,9
     STORE FP,-16,R2
     STORE FP,-20,L1
     STORE FP,-??,R1
     JMP SQ
L1:  LOAD R1,FP,-??
     LOAD R2,FP,-12
     ADD R1,R1,R2
     STORE FP,-8,R1

SQ:  ADDI FP,SP,4
     LOAD R1,FP,0
     LOAD R2,FP,0
     MUL R1,R1,R2
     ...
```

Implementing functions: registers

Solution: Callee-save

Register value saved on stack by called function

```
     ADDI R1,R0,1
     ADDI R2,R0,9
     STORE FP,-16,R2
     STORE FP,-20,L1
     JMP SQ
L1:  LOAD R2,FP,-12
     ADD R1,R1,R2
     STORE FP,-8,R1

SQ:  ADDI FP,SP,4
     STORE FP,-??,R1
     LOAD R1,FP,0
     LOAD R2,FP,0
     MUL R1,R1,R2
     ...
     LOAD R1,FP,-??
```

Implementing functions: registers

Solution: Callee-save

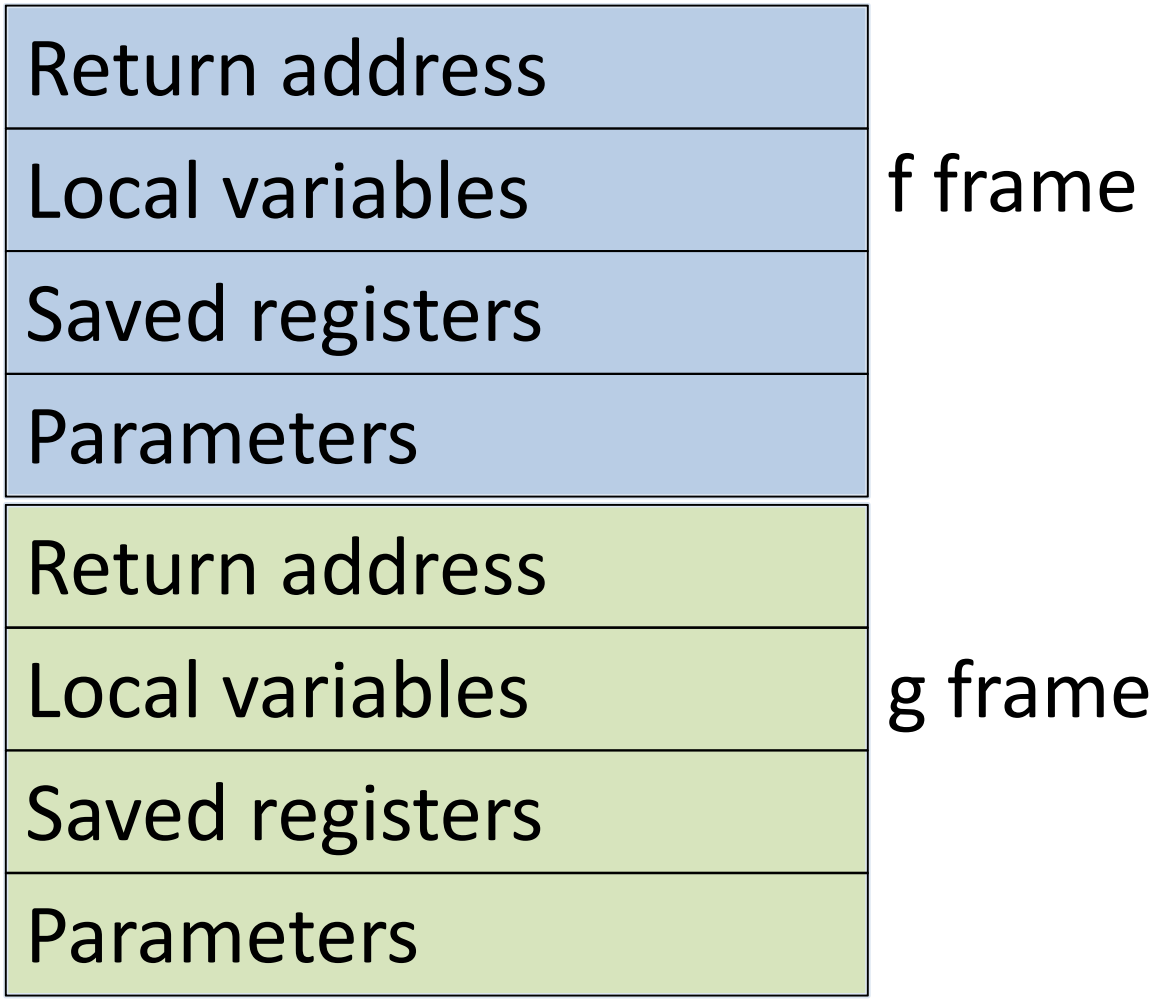
Register value saved on stack by called function

Useful if calling function is currently using these registers (they are live)

```
     ADDI R1,R0,1
     ADDI R2,R0,9
     STORE FP,-16,R2
     STORE FP,-20,L1
     JMP SQ
L1:  LOAD R2,FP,-12
     ADD R1,R1,R2
     STORE FP,-8,R1

SQ:  ADDI FP,SP,4
     STORE FP,-??,R1
     LOAD R1,FP,0
     LOAD R2,FP,0
     MUL R1,R1,R2
     ...
     LOAD R1,FP,-??
```

Implementing functions: registers



Implementing functions: registers

Caller-save and callee-save registers

- Machine-specific: different architectures have different conventions.
- Can be implemented by combining with register allocation.

Implementing functions: registers

1. How to implement caller-save?

Combine with register allocation.

Make CALL interfere with all caller-save registers.

- If variable is not live across a function call, it can be allocated to a caller-save register.
- No chance of function unnecessarily saving it.

Implementing functions: registers

2. How to implement callee-save?

Combine with register allocation.

Make function move every callee-save register (r) to new temporary variable (t) during whole function body.

- Variable (v) will be allocated to non-callee-save registers if possible.
- If necessary, v will be allocated to r by spilling t.
- Spilling t = saving callee-save register.

Implementing functions: optimizing

Problem: Slow: lot of memory accesses.

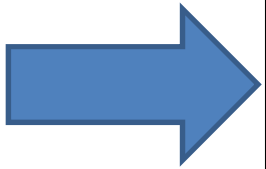
Solution: Use more registers!

- Use register for function return value.
  - Need to save value after return.
  - Canonical IR trees.
- Use registers for some of the parameters.
- Use register for return address.

Implementing functions: optimizing

```
     ADDI R1,R0,9
     STORE FP,-16,R1
     STORE FP,-20,L1
     JMP SQ
L1:  LOAD R1,FP,-12
     STORE FP,-8,R1

SQ:  ADDI FP,SP,4
     LOAD R1,FP,0
     LOAD R2,FP,0
     MUL R1,R1,R2
     STORE FP,-8,R1
     LOAD R1,FP,-8
     STORE FP,4,R1
     LOAD R1,FP,-4
     ADDI FP,FP,16
     JUMP R1
```



```
     ADDI R7,R0,9
     IADDR R8,L1
     JMP SQ
L1:  ADDI R1,R9,0
     STORE FP,-4,R1


SQ:  ADDI FP,SP,4
MUL  R1,R7,R7
     STORE FP,-4,R1
     LOAD R1,FP,-4
     ADDI R9,R1,0
     ADDI FP,FP,4
     JUMP R8
```

Implementing functions: optimizing

- Use registers for some of the parameters.
- Use register for return address.

What happens if function calls another function?

- Callee saves parameters on stack.
- Callee saves return address on stack.

Implementing functions: optimizing

What is the point of optimizing?

- Most functions (f) are ***leaf*** functions: they don't call other functions (g).
- Even if f calls g, f's parameter p may not be live at time of call to g. No need to save p on stack.