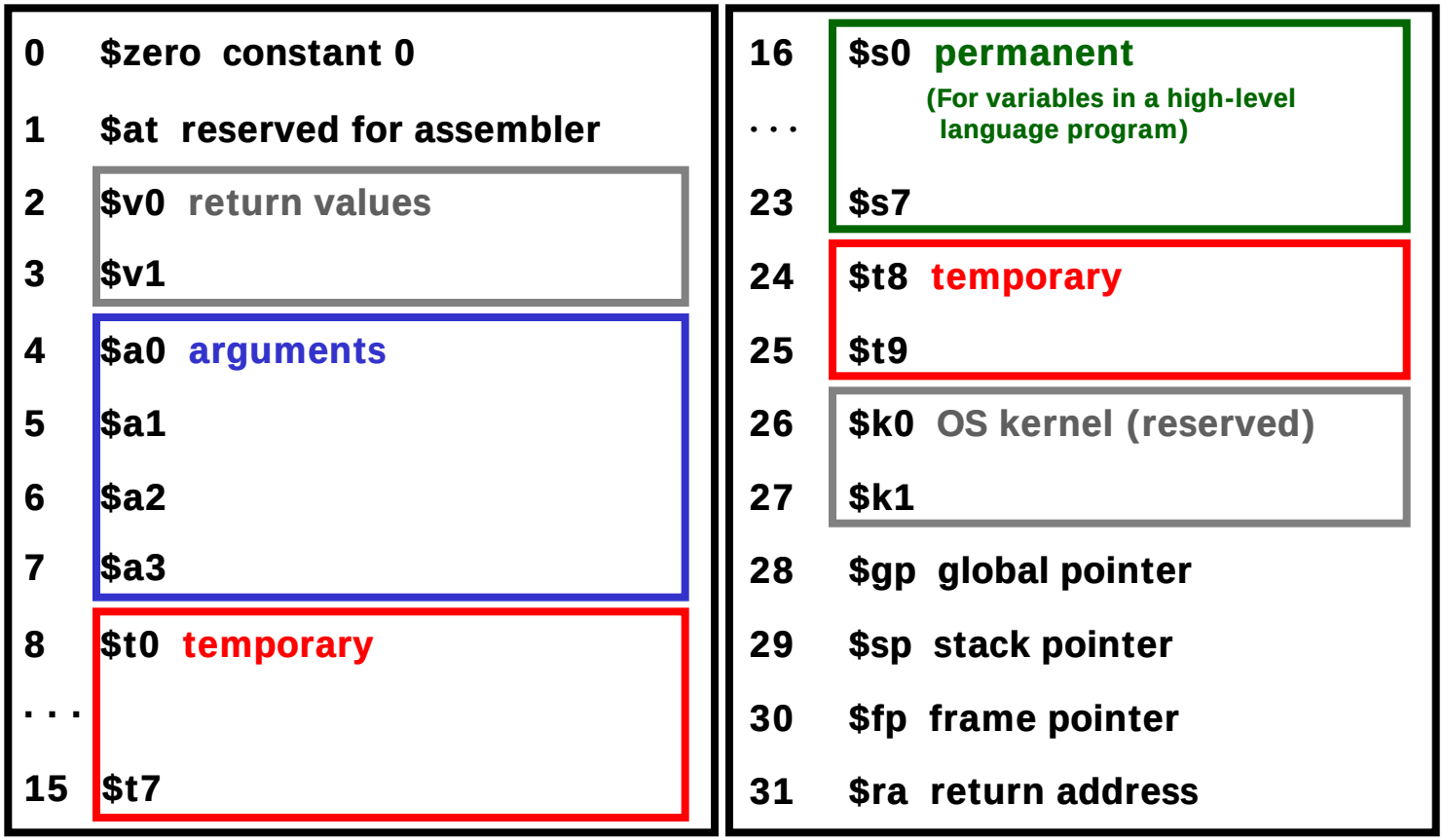
Computer Architecture

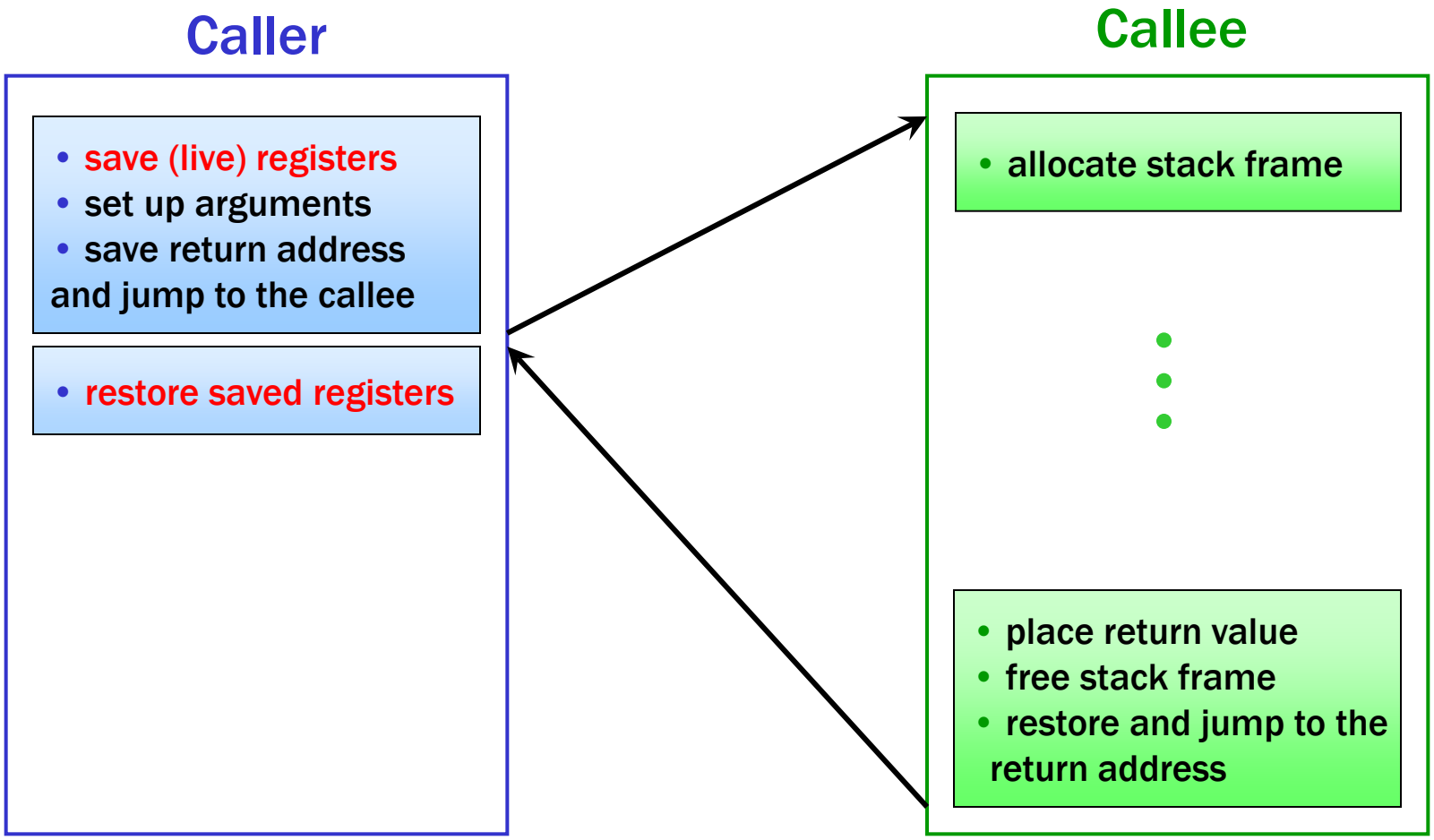
MIPS Instruction Set Architecture

(Supporting Procedures)

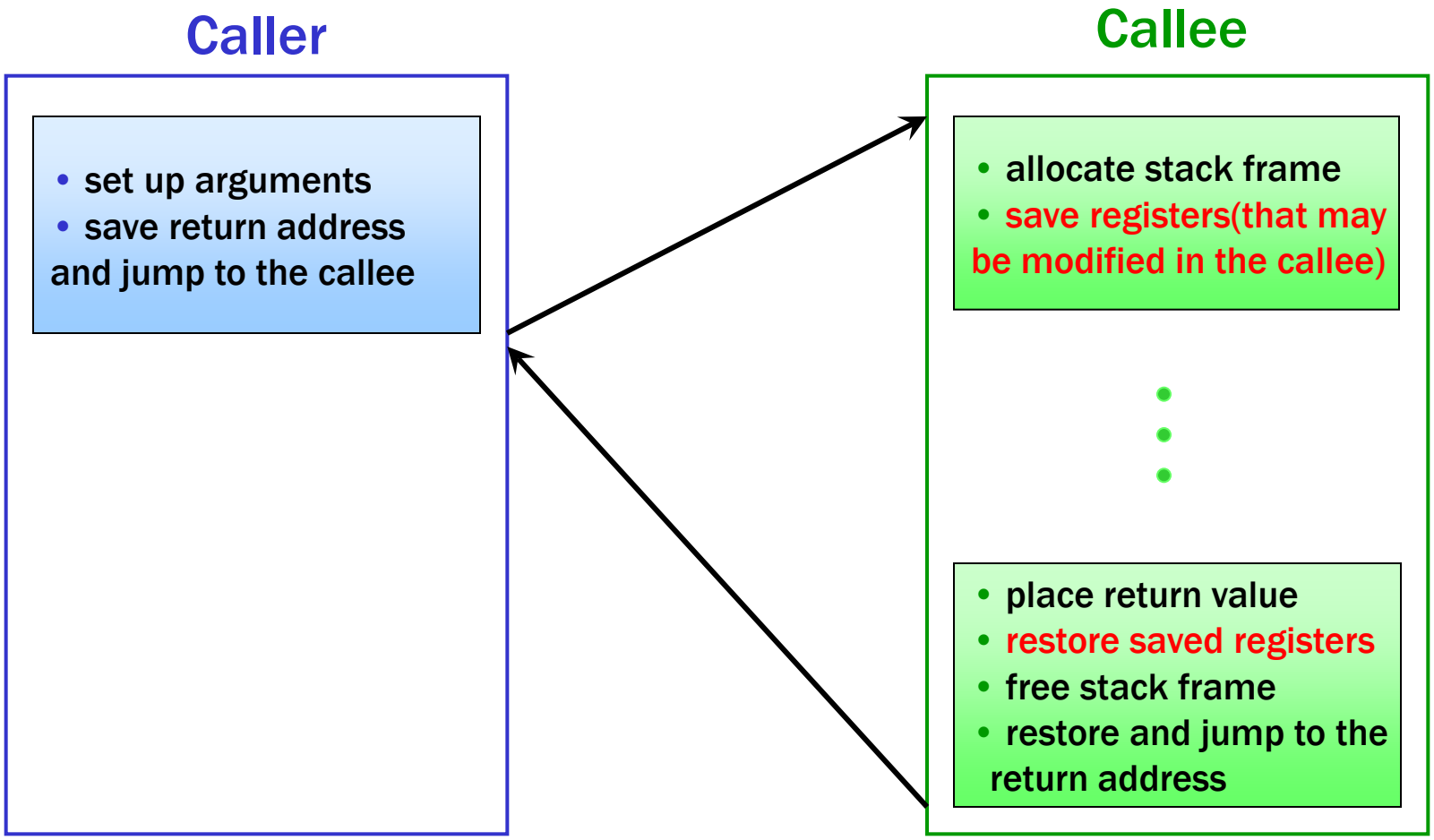
MIPS Register Naming



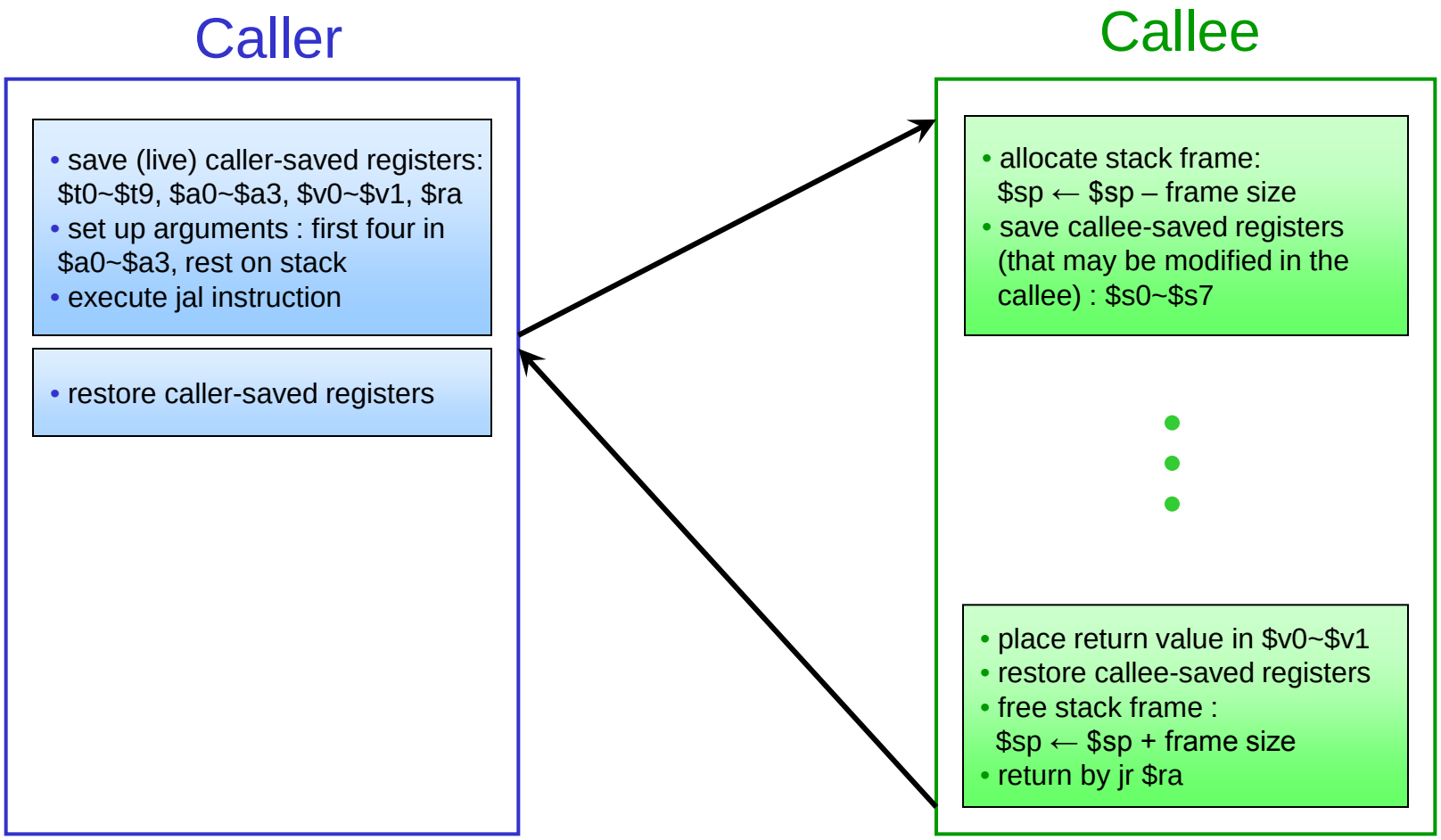
Calling Convention (Caller Save)



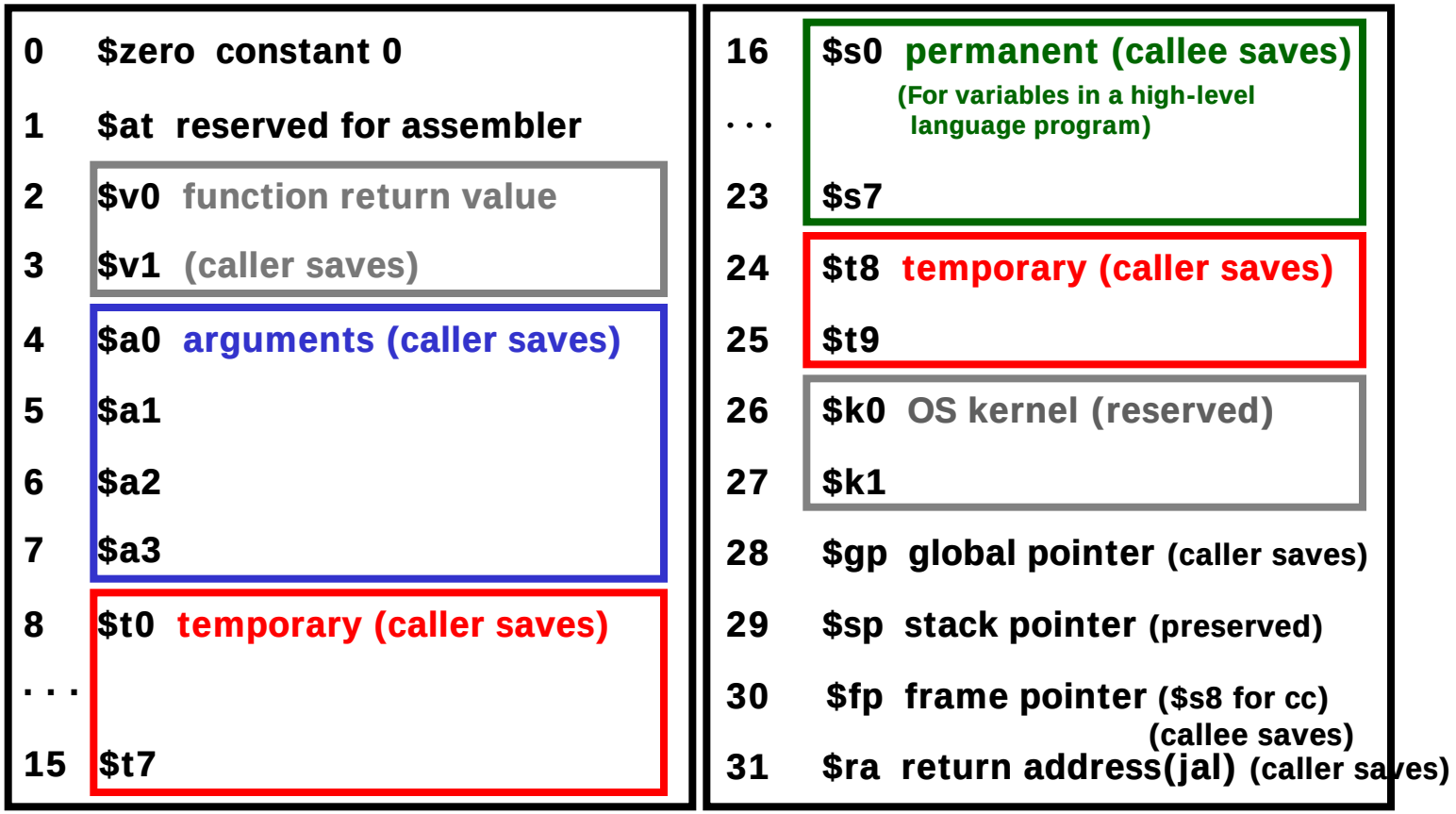
Calling Convention (Callee Save)



MIPS Calling Convention (Hybrid)



MIPS Register Convention



Compiling a Leaf Procedure

C

```
int leaf_example ( int g,
          int h, int, i, int, j )
 {
     int f ;

     f = ( g + h ) – ( i + j );
     return f;
  }
```

Actually
Not needed

Assembly

```
leaf_example :
        sub   $sp, $sp, 12
        sw    $t1,   8($sp)
        sw    $t0,   4($sp)
        sw    $s0,  0($sp)
        add   $t0, $a0, $a1
        add   $t1, $a2, $a3
        sub   $s0, $t0, $t1
        add   $v0, $s0, $zero
        lw    $s0,  0($sp)
        lw    $t0,   4($sp)
        lw    $t1,   8($sp)
        add   $sp, $sp,12
        jr      $ra
```

Stack Allocation

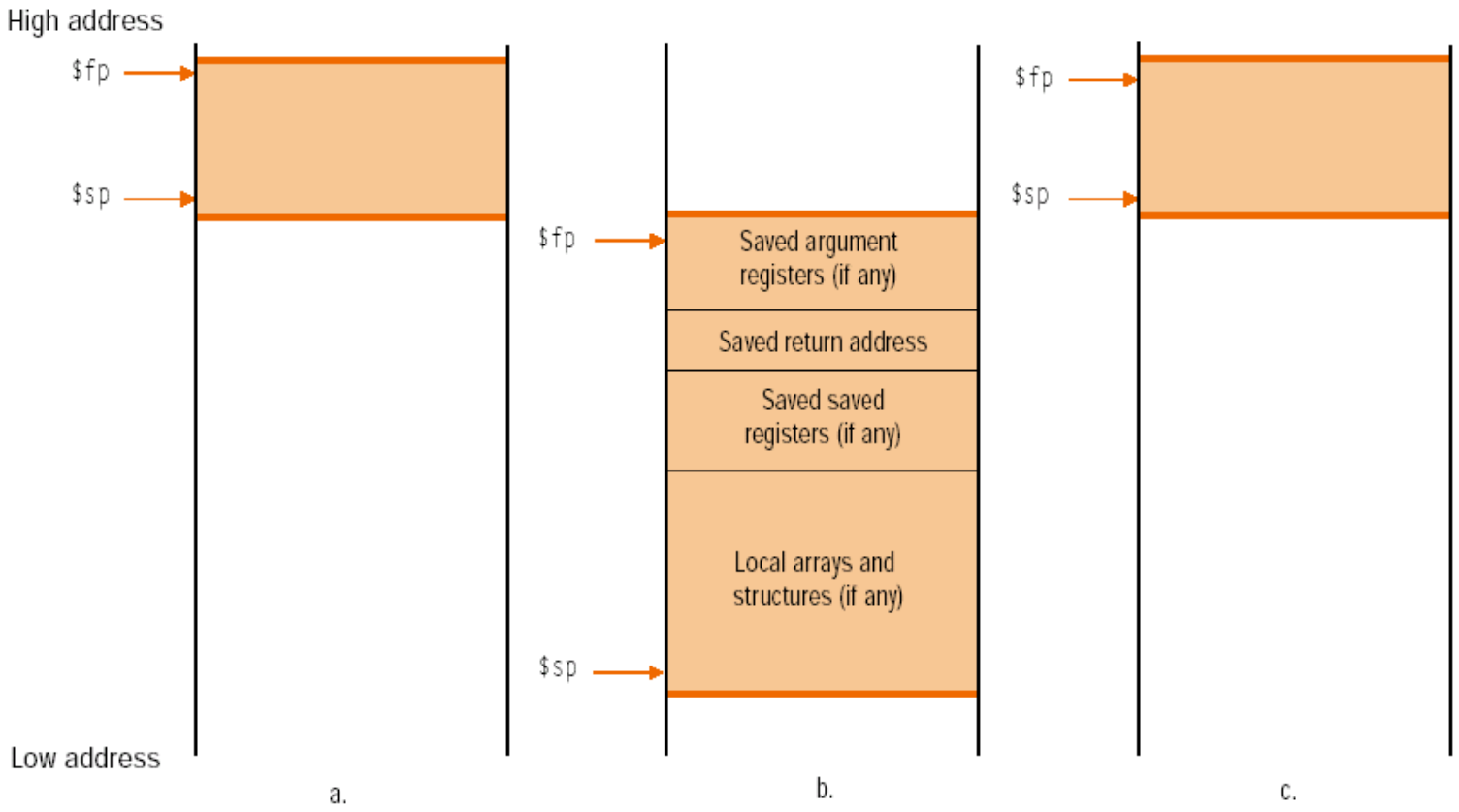


FIGURE 3.12 Illustration of the stack allocation (a)before, (b)during, and (c)after the procedure call

Compiling a Non-Leaf Procedure

C

```
int nonleaf_example (int arg)
{
    int result;
    result = arg +
          leaf_example(5,4,3,2);
    return result;
}
```

Assembly

```
nonleaf_example:
    sub $sp, $sp, 12
    sw  $s0, 8($sp)
    sw  $ra, 4($sp)
    sw  $a0, 0($sp)
    add $a0, $zero, 5
    add $a1, $zero, 4
    add $a2, $zero, 3
    add $a3, $zero, 2
    jal  leaf_example
    lw   $a0, 0($sp)
    lw   $ra, 4($sp)
    add $s0, $a0, $v0
    add $v0, $zero, $s0
    lw   $s0, 8($sp)
    add $sp, $sp, 12
    jr    $ra
```

Compiling a Recursive Procedure

C

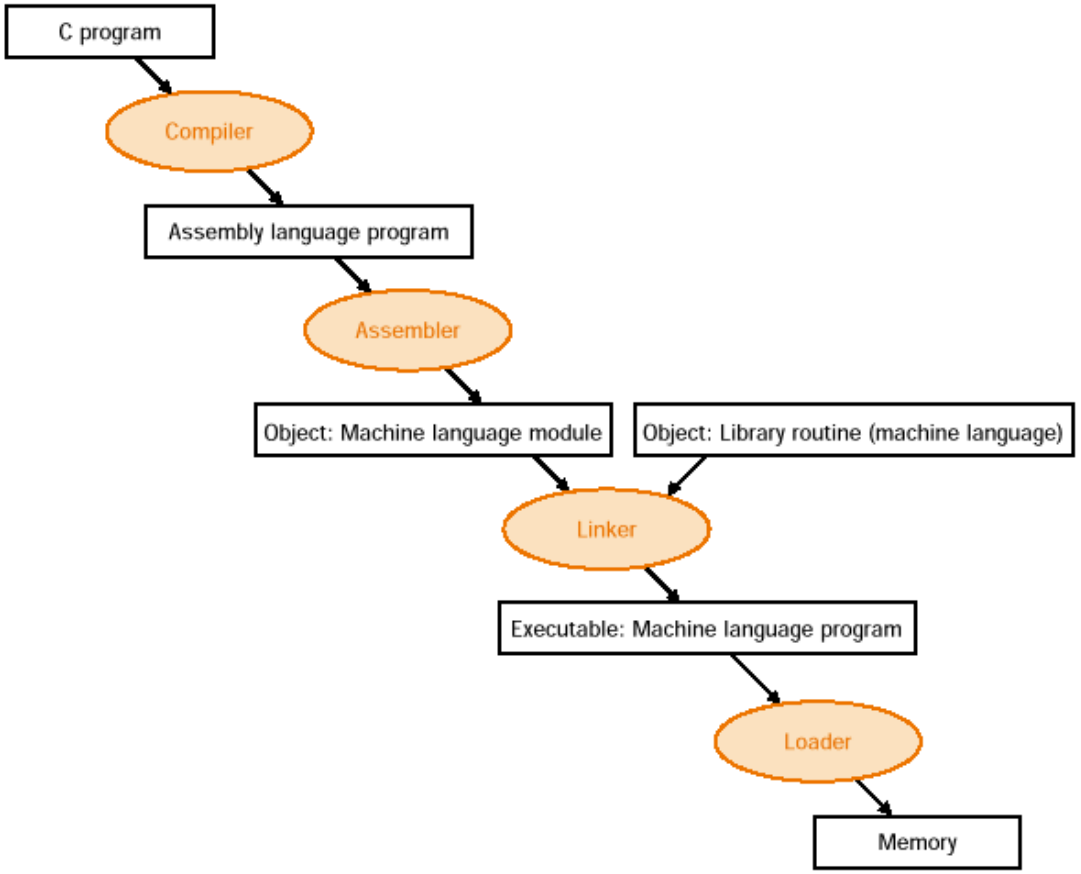
```
int fact ( int n )
 {
   if ( n < 1 ) return (1) ;
   else
     return ( n * fact ( n – 1) );

 }
```

Assembly

```
fact :
      sub   $sp, $sp, 8
      sw    $ra,  4($sp)
      sw    $a0, 0($sp)
      slt    $t0, $a0, 1
      beq   $t0, $zero, L1
      add   $v0, $zero, 1
      add   $sp, $sp, 8
      jr      $ra
L1 : sub   $a0, $a0, 1
      jal     fact
      lw     $a0,  0($sp)
      lw     $ra,  4($sp)
      add   $sp, $sp, 8
      mul   $v0, $a0, $v0
      jr      $ra
```

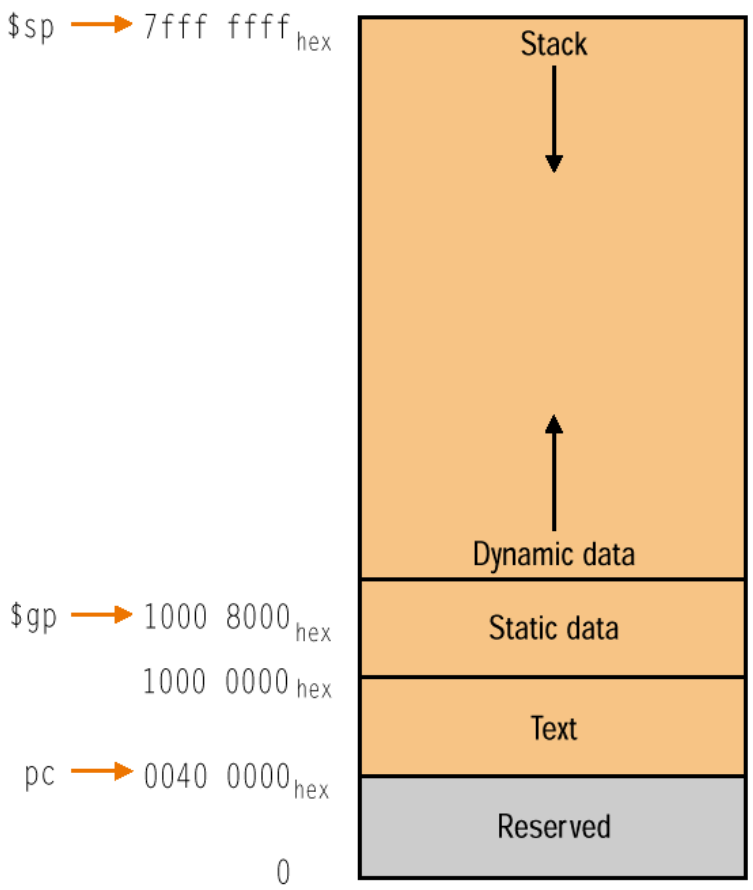
Compiling and Starting a Program



Object File Example

Object file header				**Object file header**			
	Name	Procedure A			Name	Procedure B	
	Text size	100_{hex}			Text size	200_{hex}	
	Data size	20_{hex}			Data size	30_{hex}	
Text segment	Address	Instruction		Text segment	Address	Instruction	
	0	lw $a0, 0($gp)			0	sw $a1, 0($gp)	
	4	jal 0			4	jal 0	
	…	…			…	…	
Data segment	0	(X)		Data segment	0	(Y)	
	…	…			…	…	
Relocation information	Address	Instruction type	Dependency	Relocation information	Address	Instruction type	Dependency
	0	lw	X		0	sw	Y
	4	jal	B		4	jal	A
Symbol table	Label	Address		Symbol table	Label	Address	
	X	-			Y	-	
	B	-			A	-	

MIPS Memory Allocation



Executable File Example

Executable file header		
	Text size	300_{hex}
	Data size	50_{hex}
Text segment	Address	Instruction
	$0040\ 0000_{hex}$	lw $a0, 8000_{hex}($gp)
	$0040\ 0004_{hex}$	jal 40 0100_{hex}
	…	…
	$0040\ 0100_{hex}$	sw $a1, 8020_{hex}($gp)
	$0000\ 0104_{hex}$	jal 40 0000_{hex}
	…	…
Data segment	Address	
	$1000\ 0000_{hex}$	(X)
	…	…
	$1000\ 0020_{hex}$	(Y)
	…	…