

---

# Computer Architecture

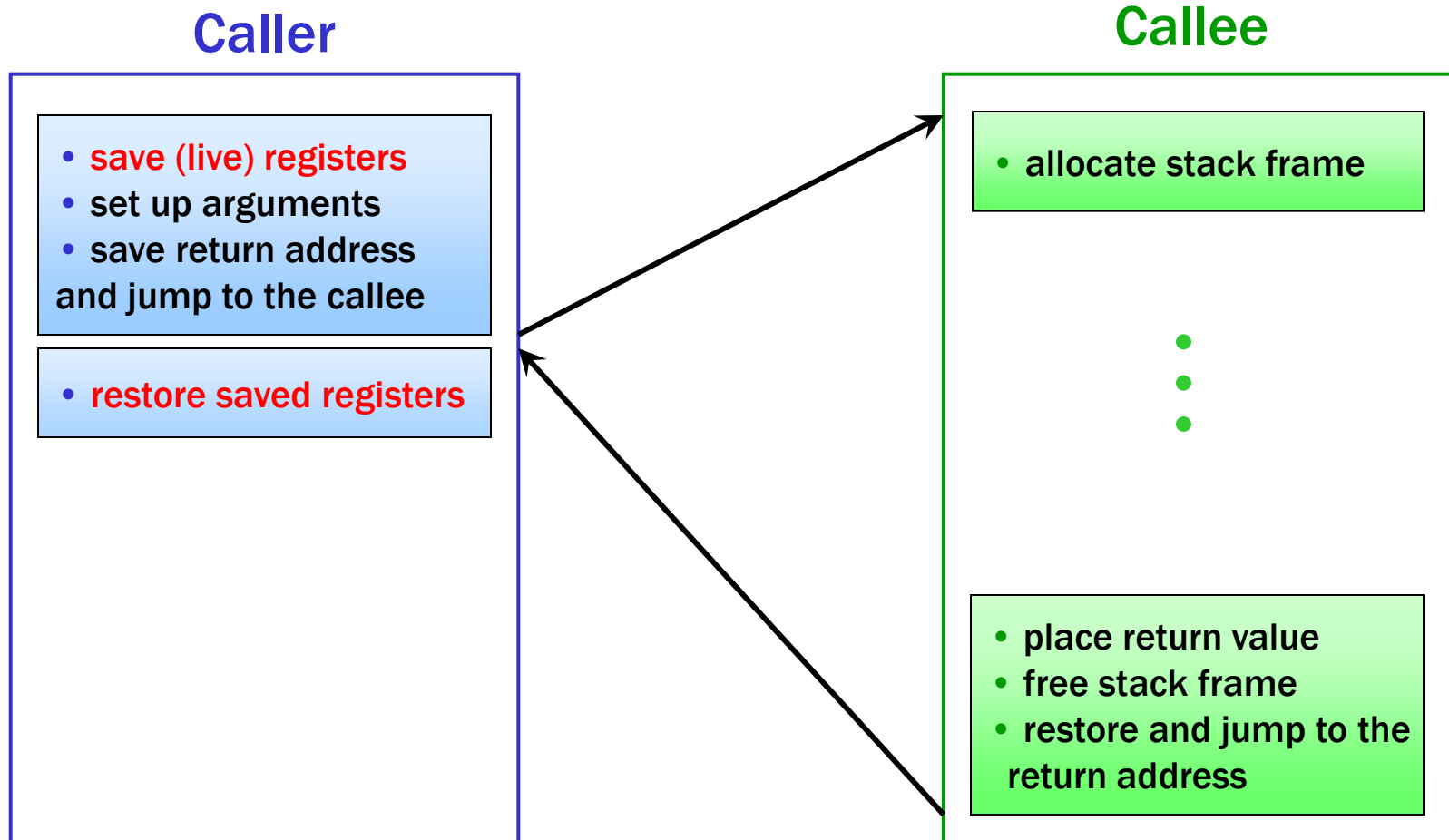
MIPS Instruction Set Architecture  
( Supporting Procedures )

---

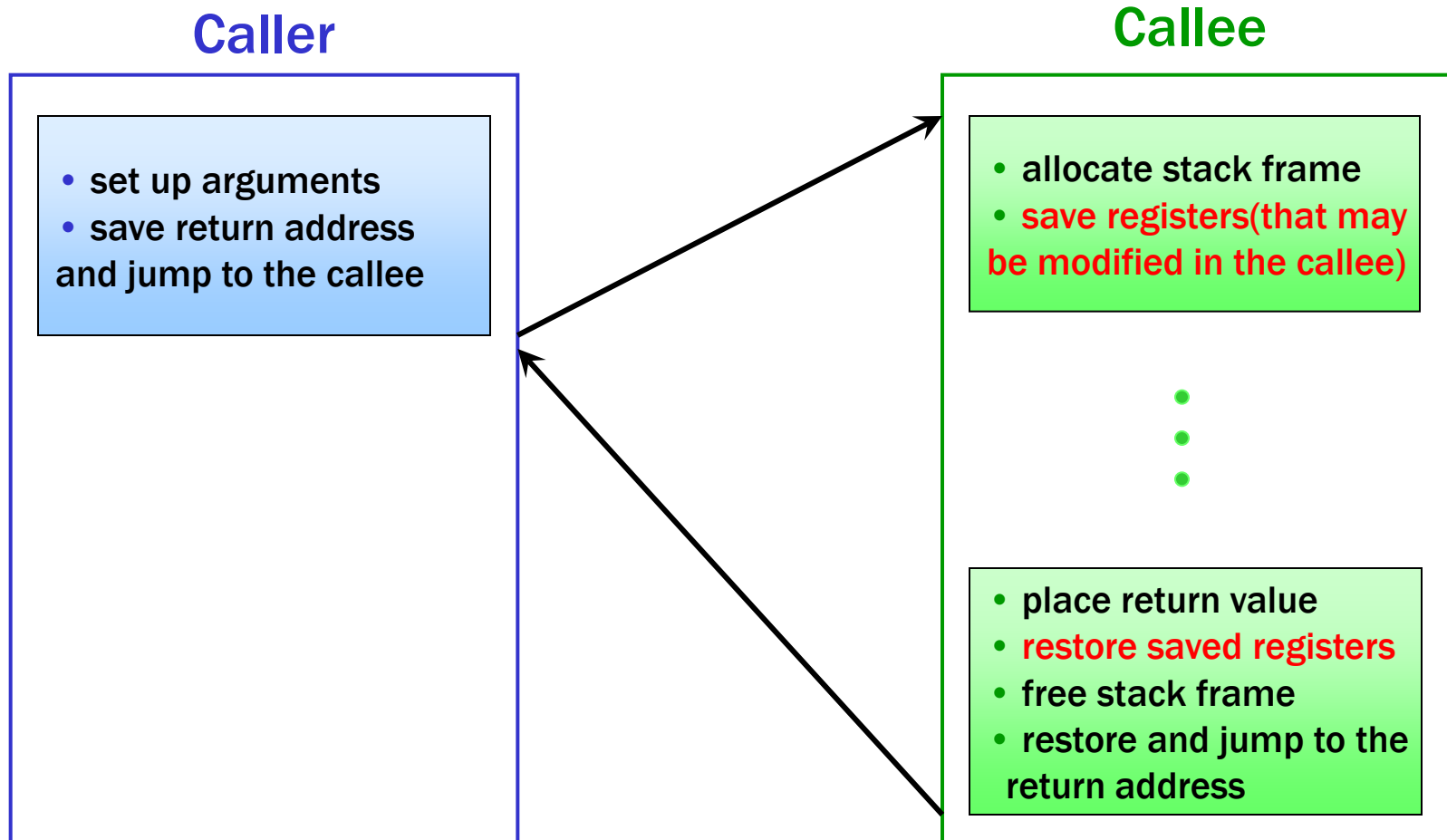
# MIPS Register Naming

0	\$zero constant 0	16	<b>\$s0 permanent</b> (For variables in a high-level language program)
1	\$at reserved for assembler	...	
2	\$v0 return values	23	\$s7
3	\$v1	24	<b>\$t8 temporary</b>
4	<b>\$a0 arguments</b>	25	<b>\$t9</b>
5	\$a1	26	\$k0 OS kernel (reserved)
6	\$a2	27	\$k1
7	\$a3	28	\$gp global pointer
8	<b>\$t0 temporary</b>	29	\$sp stack pointer
...		30	\$fp frame pointer
15	<b>\$t7</b>	31	\$ra return address

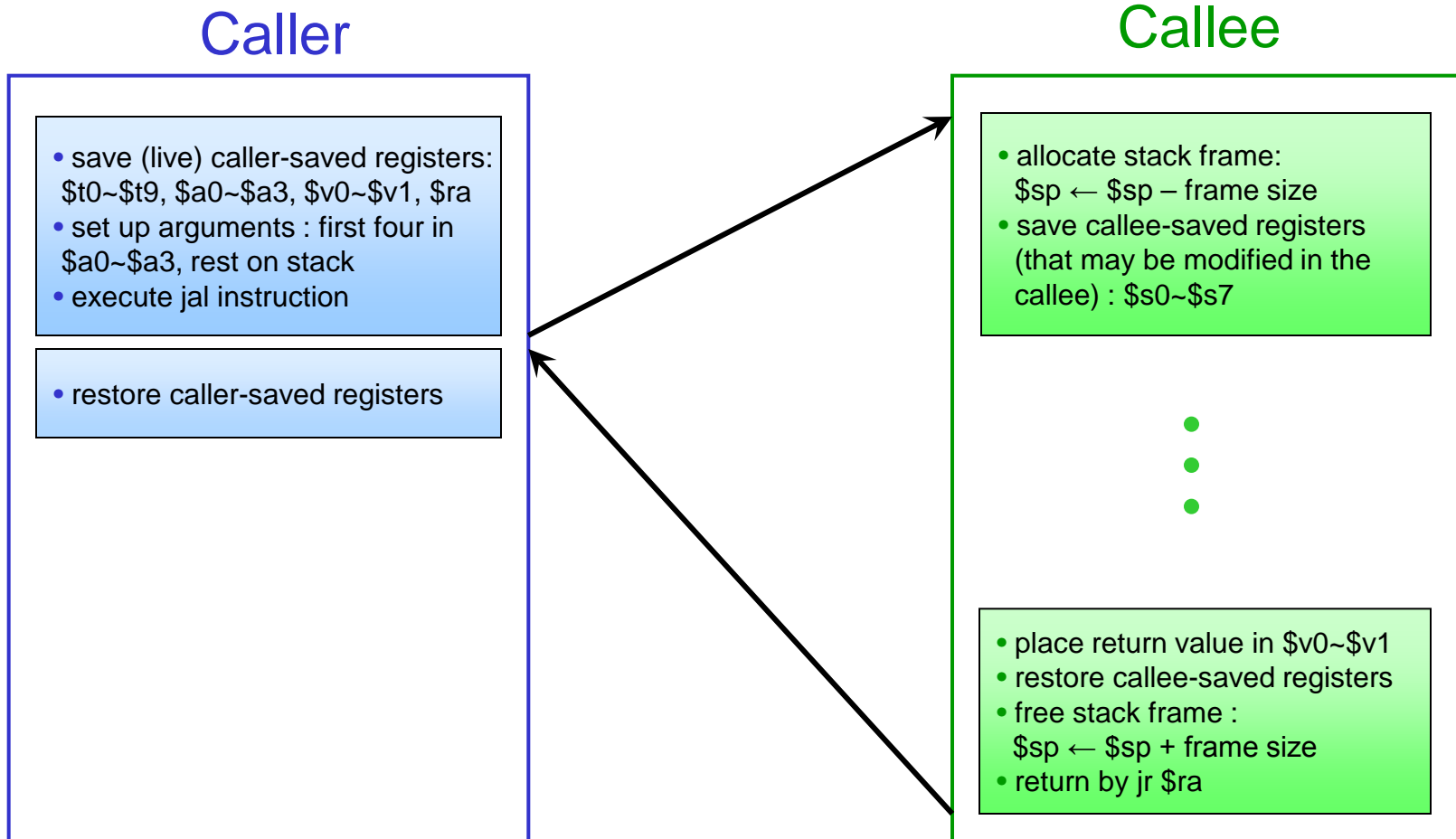
# Calling Convention (Caller Save)



# Calling Convention (Callee Save)



# MIPS Calling Convention (Hybrid)



# MIPS Register Convention

0	\$zero constant 0	16	<b>\$s0 permanent (callee saves)</b> (For variables in a high-level language program)
1	\$at reserved for assembler	...	
2	<b>\$v0 function return value</b>	23	<b>\$s7</b>
3	<b>\$v1 (caller saves)</b>	24	<b>\$t8 temporary (caller saves)</b>
4	<b>\$a0 arguments (caller saves)</b>	25	<b>\$t9</b>
5	\$a1	26	<b>\$k0 OS kernel (reserved)</b>
6	\$a2	27	<b>\$k1</b>
7	\$a3	28	\$gp global pointer (caller saves)
8	<b>\$t0 temporary (caller saves)</b>	29	\$sp stack pointer (preserved)
...		30	\$fp frame pointer (\$s8 for cc) (callee saves)
15	<b>\$t7</b>	31	<b>\$ra return address(jal) (caller saves)</b>

# 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

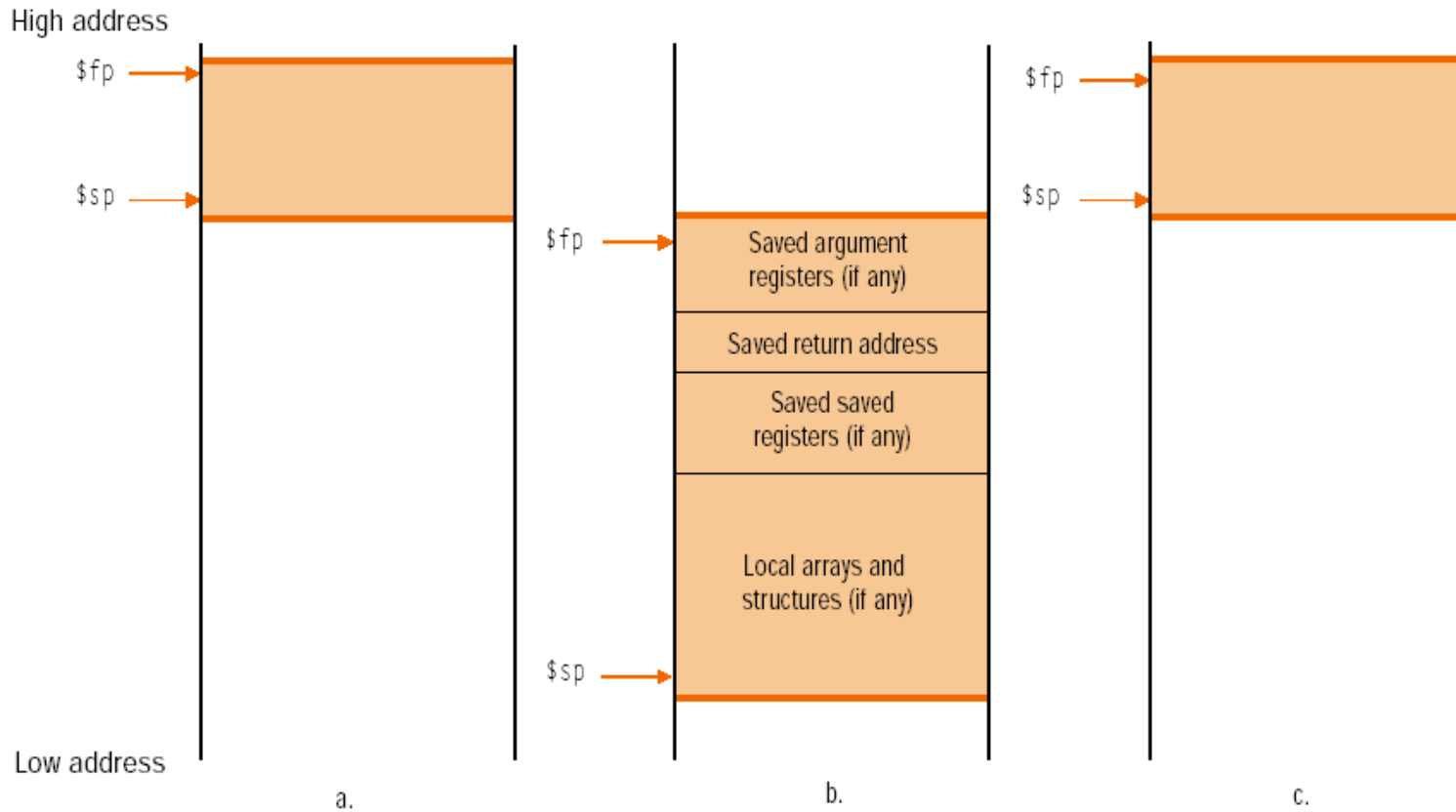


FIGURE 3.12 Illustration of the stack allocation (a) before, (b) during, and (c) after the procedure call



# 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

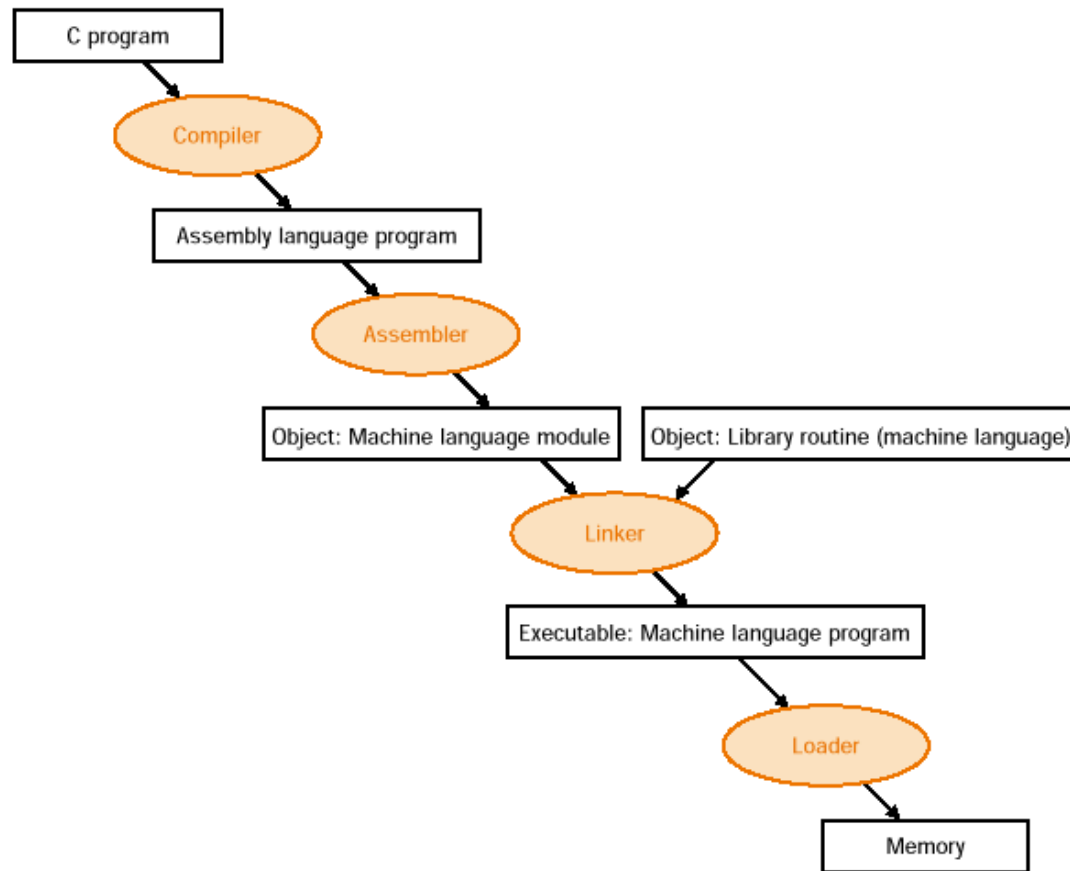
```
int fact ( int n )
{
  if ( n < 1 ) return (1);
  else
    return ( n * fact ( n - 1) );
}
```



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
```

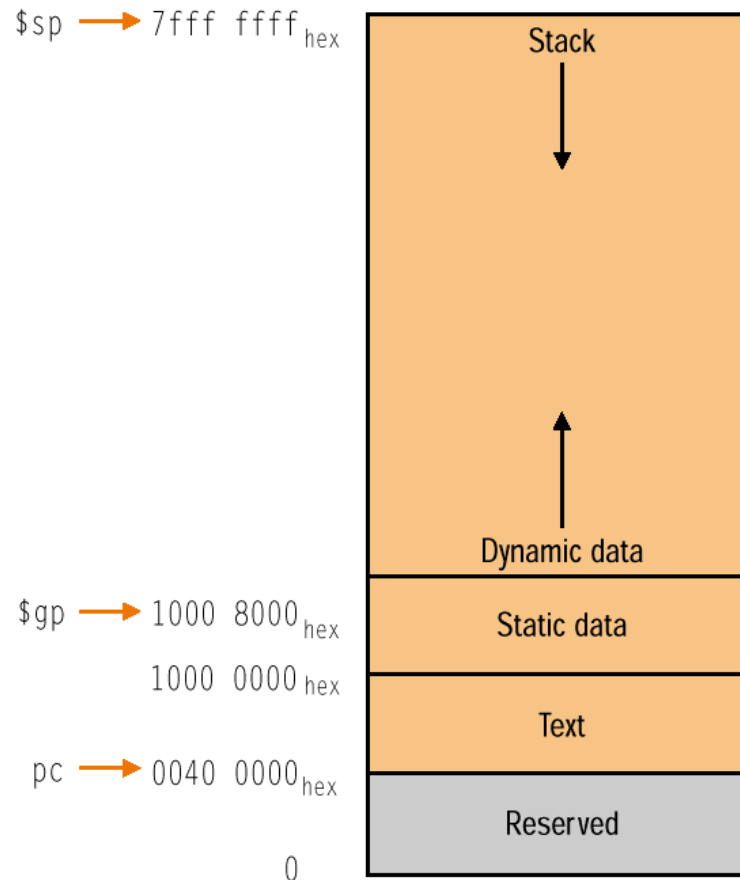
# Compiling and Starting a Program



# Object File Example

Object file header				Object file header			
	Name	Procedure A			Name	Procedure B	
	Text size	100 <sub>hex</sub>			Text size	200 <sub>hex</sub>	
	Data size	20 <sub>hex</sub>			Data size	30 <sub>hex</sub>	
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 <sub>hex</sub>
	Data size	50 <sub>hex</sub>
Text segment	Address	Instruction
	0040 0000 <sub>hex</sub>	lw \$a0, 8000 <sub>hex</sub> (\$gp)
	0040 0004 <sub>hex</sub>	jal 40 0100 <sub>hex</sub>
	...	...
	0040 0100 <sub>hex</sub>	sw \$a1, 8020 <sub>hex</sub> (\$gp)
	0000 0104 <sub>hex</sub>	jal 40 0000 <sub>hex</sub>
	...	...
Data segment	Address	
	1000 0000 <sub>hex</sub>	(X)
	...	...
	1000 0020 <sub>hex</sub>	(Y)
	...	...