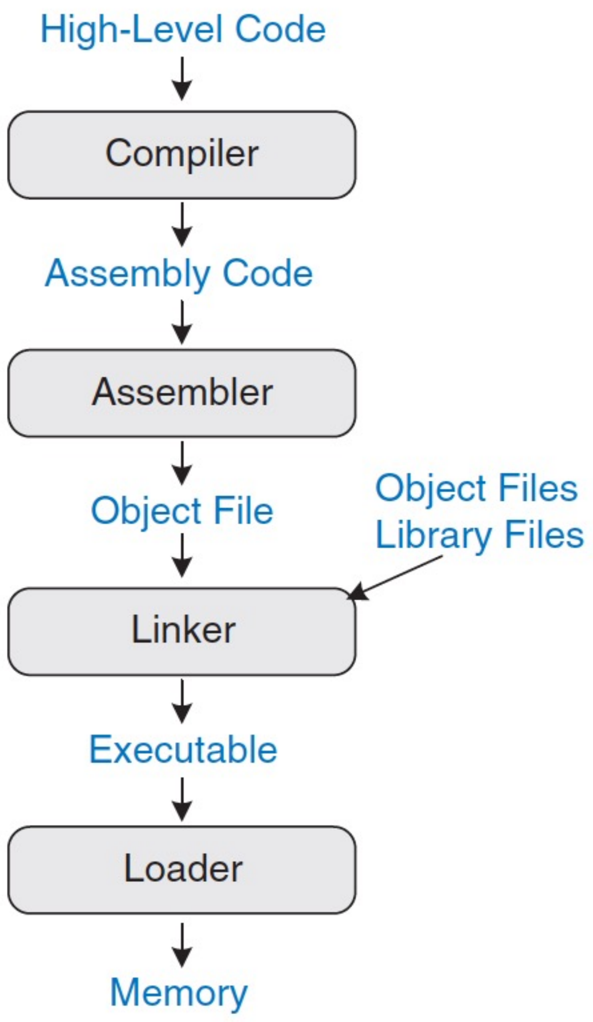
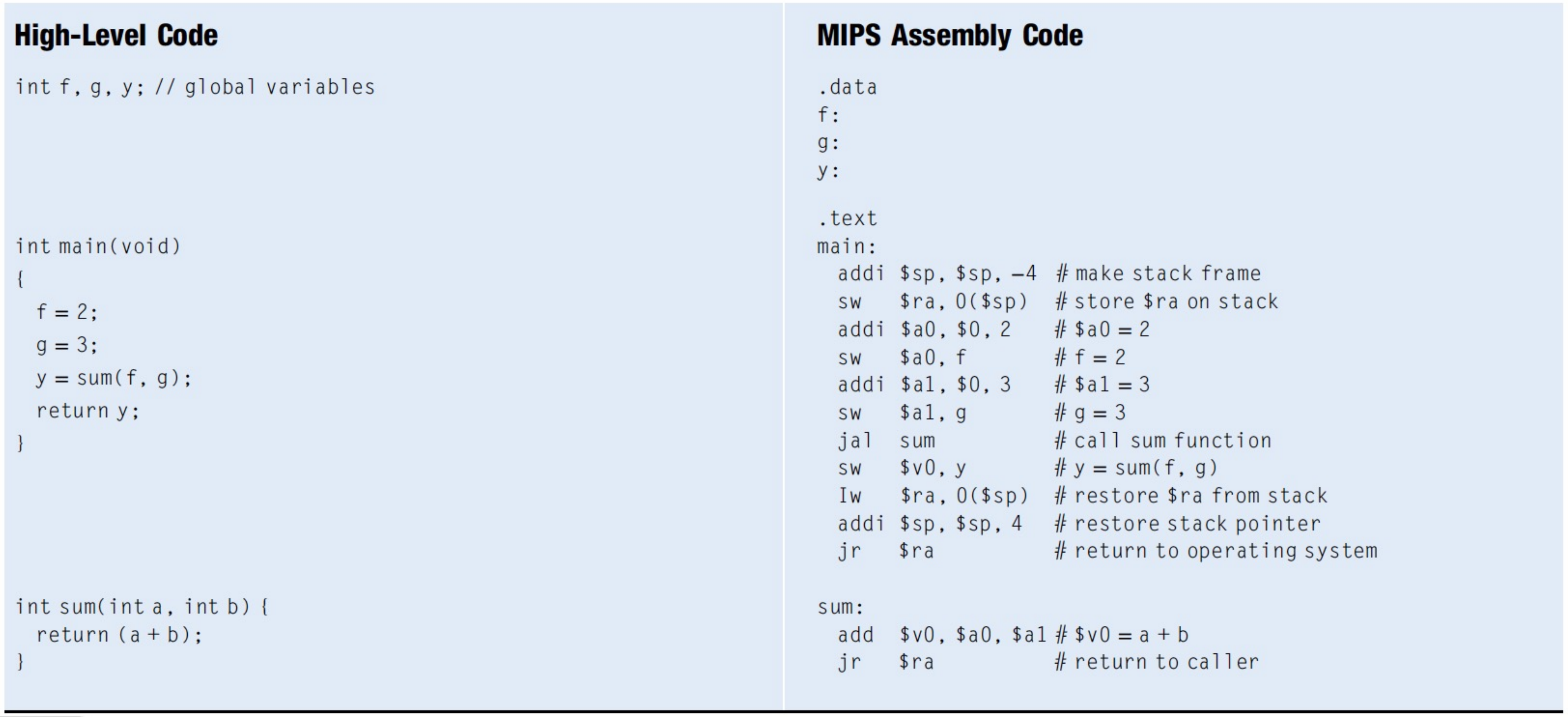
Loading a Program

Translating and Starting a Program



Compile

Translates high-level code into assembly language.



High-Level Code

```
int f, g, y; // global variables




int main(void)
{
  f = 2;
  g = 3;
  y = sum(f, g);
  return y;
}




int sum(int a, int b) {
  return (a + b);
}
```

MIPS Assembly Code

```
.data
f:
g:
y:

.text
main:
  addi $sp, $sp, -4  # make stack frame
  sw   $ra, 0($sp)   # store $ra on stack
  addi $a0, $0, 2    # $a0 = 2
  sw   $a0, f        # f = 2
  addi $a1, $0, 3    # $a1 = 3
  sw   $a1, g        # g = 3
  jal  sum           # call sum function
  sw   $v0, y        # y = sum(f, g)
  lw   $ra, 0($sp)   # restore $ra from stack
  addi $sp, $sp, 4   # restore stack pointer
  jr   $ra           # return to operating system

sum:
  add  $v0, $a0, $a1 # $v0 = a + b
  jr   $ra           # return to caller
```

Assemble

Two passes:

1. Assign instruction addresses and find symbols
    1. Labels and global variable names
2. Fill in the symbol addresses once they are known.

Machine language code and symbol table are stored in an object file.

```
0x00400000 main: addi $sp, $sp, -4
0x00400004       sw   $ra, 0($sp)
0x00400008       addi $a0, $0, 2
0x0040000C       sw   $a0, f
0x00400010       addi $a1, $0, 3
0x00400014       sw   $a1, g
0x00400018       jal  sum
0x0040001C       sw   $v0, y
0x00400020       lw   $ra, 0($sp)
0x00400024       addi $sp, $sp, 4
0x00400028       jr   $ra
0x0040002C sum:  add  $v0, $a0, $a1
0x00400030       jr   $ra
```

Symbol	Address
f	0x10000000
g	0x10000004
y	0x10000008
main	0x00400000
sum	0x0040002C

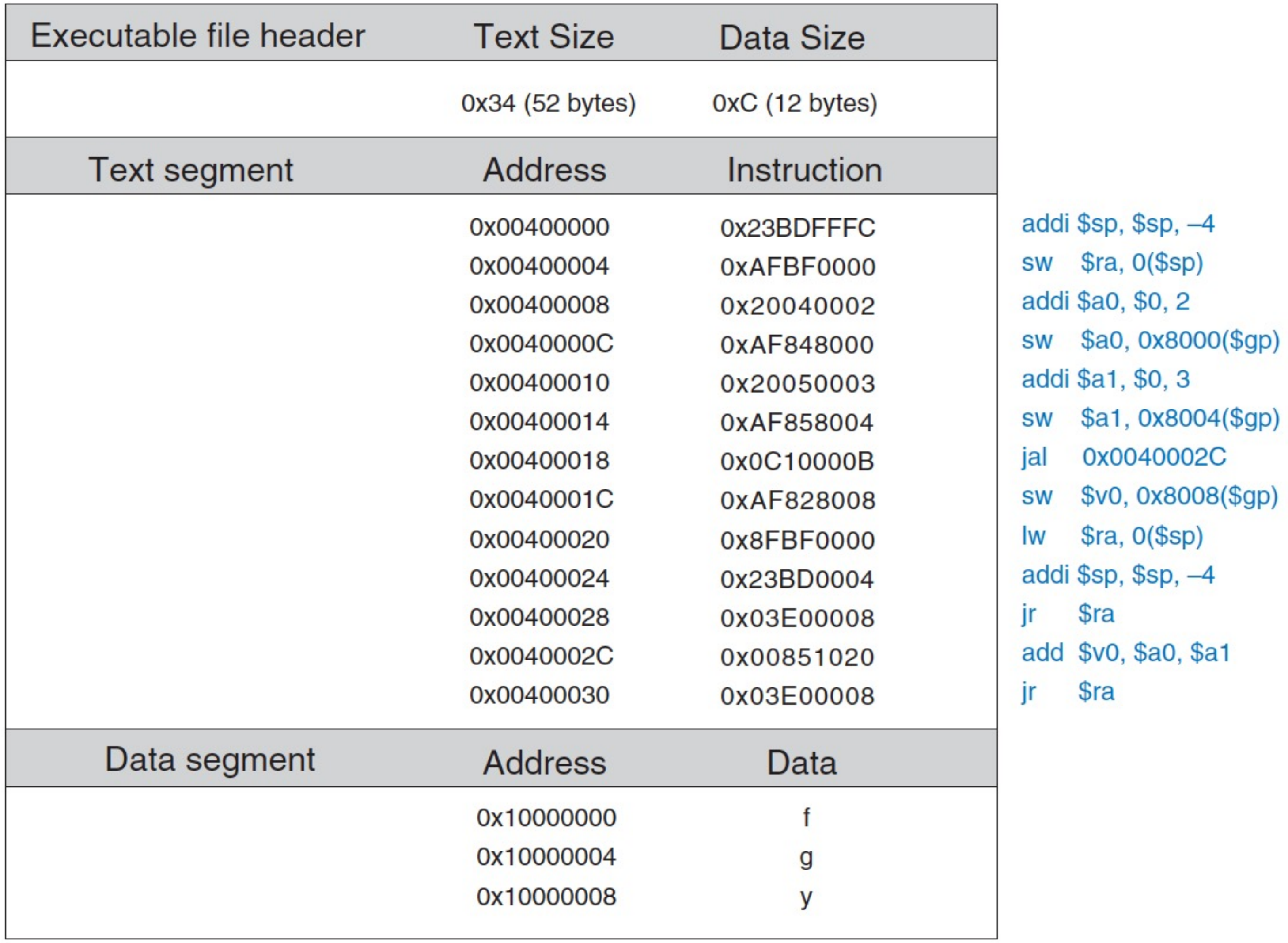
Linking

Multiple files are often used for a single program.

- A change to one file might result in total recompilation.

Linker builds an executable from these compiled files.

- Handles remapping the global variables and instruction addresses.

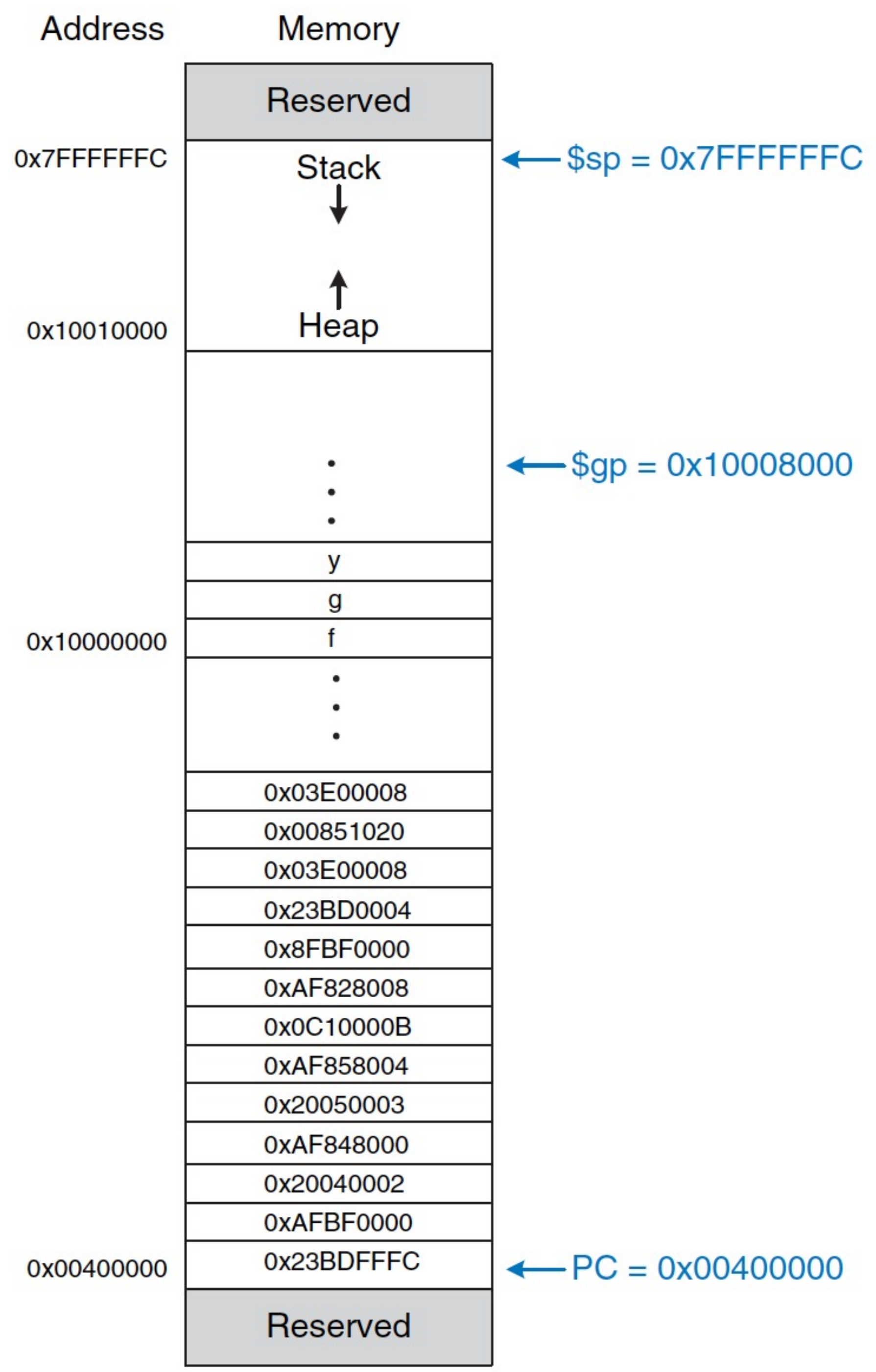


Executable file header	Text Size	Data Size	
	0x34 (52 bytes)	0xC (12 bytes)	
Text segment	**Address**	**Instruction**	
	0x00400000	0x23BDFFFC	addi $sp, $sp, –4
	0x00400004	0xAFBF0000	sw $ra, 0($sp)
	0x00400008	0x20040002	addi $a0, $0, 2
	0x0040000C	0xAF848000	sw $a0, 0x8000($gp)
	0x00400010	0x20050003	addi $a1, $0, 3
	0x00400014	0xAF858004	sw $a1, 0x8004($gp)
	0x00400018	0x0C10000B	jal 0x0040002C
	0x0040001C	0xAF828008	sw $v0, 0x8008($gp)
	0x00400020	0x8FBF0000	lw $ra, 0($sp)
	0x00400024	0x23BD0004	addi $sp, $sp, –4
	0x00400028	0x03E00008	jr $ra
	0x0040002C	0x00851020	add $v0, $a0, $a1
	0x00400030	0x03E00008	jr $ra
Data segment	**Address**	**Data**	
	0x10000000	f	
	0x10000004	g	
	0x10000008	y	

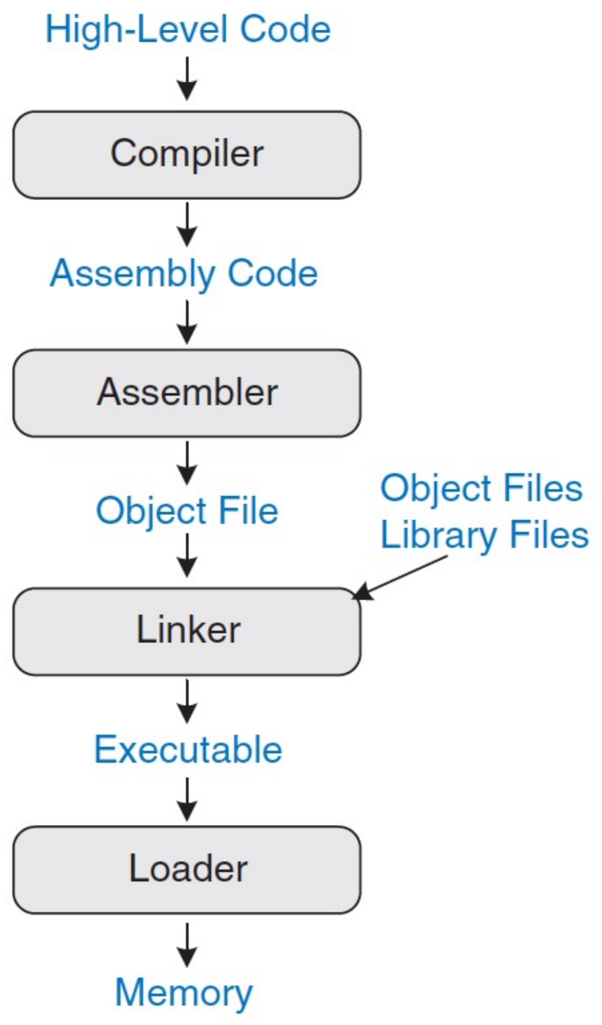
Loading

Read a text segment of the executable into memory.

Executable file header	Text Size	Data Size	
	0x34 (52 bytes)	0xC (12 bytes)	
Text segment	**Address**	**Instruction**	
	0x00400000	0x23BDFFFC	addi $sp, $sp, –4
	0x00400004	0xAFBF0000	sw $ra, 0($sp)
	0x00400008	0x20040002	addi $a0, $0, 2
	0x0040000C	0xAF848000	sw $a0, 0x8000($gp)
	0x00400010	0x20050003	addi $a1, $0, 3
	0x00400014	0xAF858004	sw $a1, 0x8004($gp)
	0x00400018	0x0C10000B	jal 0x0040002C
	0x0040001C	0xAF828008	sw $v0, 0x8008($gp)
	0x00400020	0x8FBF0000	lw $ra, 0($sp)
	0x00400024	0x23BD0004	addi $sp, $sp, –4
	0x00400028	0x03E00008	jr $ra
	0x0040002C	0x00851020	add $v0, $a0, $a1
	0x00400030	0x03E00008	jr $ra
Data segment	**Address**	**Data**	
	0x10000000	f	
	0x10000004	g	
	0x10000008	y	



Translating and Starting a Program


High-Level Code
Compiler
Assembly Code
Assembler
Object File
Object Files
Library Files
Linker
Executable
Loader
Memory