Example 6.1 TRANSLATING HIGH-LEVEL CODE TO ASSEMBLY LANGUAGE

Translate the following high-level code into assembly language. Assume variables `a`–`c` are held in registers `$s0`–`$s2` and `f`–`j` are in `$s3`–`$s7`.

```
a = b - c;
f = (g + h) - (i + j);
```

Solution: The program uses four assembly language instructions.

```
# MIPS assembly code
# $s0 = a, $s1 = b, $s2 = c, $s3 = f, $s4 = g, $s5 = h
# $s6 = i, $s7 = j
  sub $s0, $s1, $s2  # a = b - c
  add $t0, $s4, $s5  # $t0 = g + h
  add $t1, $s6, $s7  # $t1 = i + j
  sub $s3, $t0, $t1  # f = (g + h) - (i + j)
```

The Register Set

The **MIPS** architecture defines 32 registers. Each register has a name and a number ranging from 0 to 31. Table 6.1 lists the name, number, and use for each register. `$0` always contains the value 0 because this constant is so frequently used in computer programs. We have also discussed the `$s` and `$t` registers. The remaining registers will be described throughout this chapter.

Table 6.1 MIPS register set

Name	Number	Use
`$0`	0	the constant value 0
`$at`	1	assembler temporary
`$v0`–`$v1`	2–3	function return value
`$a0`–`$a3`	4–7	function arguments
`$t0`–`$t7`	8–15	temporary variables
`$s0`–`$s7`	16–23	saved variables
`$t8`–`$t9`	24–25	temporary variables
`$k0`–`$k1`	26–27	operating system (OS) temporaries
`$gp`	28	global pointer
`$sp`	29	stack pointer
`$fp`	30	frame pointer
`$ra`	31	function return address