For example, the following instructions:

```
li   $t0, 42
move $t1, $t0
```

will move the contents of register **$t0**, 42 in this example, into the **$t1** register.

The *mfhi*, *mflo*, *mtho*, and *mtlo* instructions are required only when performing 64-bit integer multiply and divide operations.

The floating-point section will include examples for moving data between integer and floating-point registers.

5.4 Integer Arithmetic Operations

The arithmetic operations include addition, subtraction, multiplication, division, remainder (remainder after division), logical AND, and logical OR. The general format for these basic instructions is as follows:

Instruction		Description
add	**Rdest, Rsrc, Src**	Signed addition Rdest = Rsrc + Src or Imm
addu	**Rdest, Rsrc, Src**	Unsigned addition Rdest = Rsrc + Src or Imm
sub	**Rdest, Rsrc, Src**	Signed subtraction Rdest = Rsrc – Src or Imm
subu	**Rdest, Rsrc, Src**	Unsigned subtraction Rdest = Rsrc – Src or Imm
mul	**Rdest, Rsrc, Src**	Signed multiply with no overflow Rdest = Rsrc * Src or Imm
mulu	**Rdest, Rsrc, Src**	Unsigned multiply with no overflow Rdest = Rsrc * Src or Imm
mulo	**Rdest, Rsrc, Src**	Signed multiply with overflow Rdest = Rsrc * Src or Imm

mulou Rdest, Rsrc, Src	Unsigned multiply with overflow Rdest = Rsrc * Src or Imm
mult Rsrc1, Rsrc2	Signed 64-bit multiply **$hi**/**$lo** = Rsrc1 * Rsrc2
multu Rsrc1, Rsrc2	Unsigned 64-bit multiply **$hi**/**$lo** = Rsrc1 * Rsrc2
div Rdest, Rsrc, Src	Signed divide Rdest = Rsrc / Src or Imm
divu Rdest, Rsrc, Src	Unsigned divide Rdest = Rsrc / Src or Imm
div Rsrc1, RSrc2	Signed divide with remainder **$lo** = Rsrc1 / RSrc2 **$hi** = Rsrc1 % RSrc2
divu Rsrc1, RSrc2	Unsigned divide with remainder **$lo** = Rsrc1 / RSrc2 **$hi** = Rsrc1 % RSrc2
rem Rdest, Rsrc, Src	Signed remainder Rdest = Rsrc % Src or Imm
remu Rdest, Rsrc, Src	Unsigned remainder Rdest = Rsrc % Src or Imm
abs Rdest, Rsrc	Absolute value Rdest = | Rsrc |
neg Rdest, Rsrc	Signed negation Rdest = - Rsrc

These instructions operate on 32-bit registers (even if byte or halfword values are placed in the registers).

Assuming the following data declarations:

```
wnum1:      .word       651
wnum2:      .word       42
wans1:      .word       0
wans2:      .word       0
wans3:      .word       0
```

Instructions and PseudoInstructions

The following is an abbreviated list of MIPS instructions and SPIM pseudoinstructions. This list is not complete. Notably missing are all Floating Point and coprocessor instructions.

• - Indicates an actual MIPS instruction. Others are SPIM pseudoinstructions.

	Instruction		Function
•	add	Rd, Rs, Rt	Rd = Rs + Rt (signed)
•	addu	Rd, Rs, Rt	Rd = Rs + Rt (unsigned)
•	addi	Rd, Rs, Imm	Rd = Rs + Imm (signed)
•	sub	Rd, Rs, Rt	Rd = Rs - Rt (signed)
•	subu	Rd, Rs, Rt	Rd = Rs - Rt (unsigned)
•	div	Rs, Rt	lo = Rs/Rt, hi = Rs **mod** Rt (integer division, signed)
•	divu	Rs, Rt	lo = Rs/Rt, hi = Rs **mod** Rt (integer division, unsigned)
	div	Rd, Rs, Rt	Rd = Rs/Rt (integer division, signed)
	divu	Rd, Rs, Rt	Rd = Rs/Rt (integer division, unsigned)
	rem	Rd, Rs, Rt	Rd = Rs **mod** Rt (signed)
	remu	Rd, Rs, Rt	Rd = Rs **mod** Rt (unsigned)
	mul	Rd, Rs, Rt	Rd = Rs * Rt (signed)
•	mult	Rs, Rt	hi, lo = Rs * Rt (signed, hi = high 32 bits, lo = low 32 bits)
•	multu	Rd, Rs	hi, lo = Rs * Rt (unsigned, hi = high 32 bits, lo = low 32 bits)