

Operandos MIPS

Nombre	Ejemplo	Comentarios
32 registros	<code>\$s0-\$s7, \$t0-\$t9, \$zero, \$a0-\$a3, \$v0-\$v1, \$gp, \$fp, \$sp, \$ra, \$at</code>	Localizaciones rápidas para los datos. En MIPS, los datos deben estar en los registros para realizar operaciones aritméticas. El registro MIPS <code>\$zero</code> es siempre igual a 0. El registro <code>\$at</code> está reservado por el ensamblador para manejar constantes grandes.
2 ³⁰ palabras de memoria	<code>Memory[0], Memory[4], . . . , Memory[4294967292]</code>	Accesibles solamente por instrucciones de transferencia de datos. MIPS utiliza direcciones de byte, de modo que las direcciones de palabras consecutivas se diferencian en 4. La memoria guarda las estructuras de datos, las tablas y los registros desbordados (guardados).

Lenguaje ensamblador MIPS

Categoría	Instrucción	Ejemplo	Significado	Comentarios
Aritmética	<code>add</code>	<code>add \$s1,\$s2,\$s3</code>	<code>\$s1 = \$s2 + \$s3</code>	Tres operandos; datos en registros
	<code>subtract</code>	<code>sub \$s1,\$s2,\$s3</code>	<code>\$s1 = \$s2 - \$s3</code>	Tres operandos; datos en registros
	<code>add immediate</code>	<code>addi \$s1,\$s2,100</code>	<code>\$s1 = \$s2 + 100</code>	Usado para sumar constantes
Transferencia de dato	<code>load word</code>	<code>lw \$s1,100(\$s2)</code>	<code>\$s1 = Memory[\$s2 + 100]</code>	Palabra de memoria a registro
	<code>store word</code>	<code>sw \$s1,100(\$s2)</code>	<code>Memory[\$s2 + 100] = \$s1</code>	Palabra de registro a memoria
	<code>load half</code>	<code>lh \$s1,100(\$s2)</code>	<code>\$s1 = Memory[\$s2 + 100]</code>	Media palabra de memoria a registro
	<code>store half</code>	<code>sh \$s1,100(\$s2)</code>	<code>Memory[\$s2 + 100] = \$s1</code>	Media palabra de registro a memoria
	<code>load byte</code>	<code>lb \$s1,100(\$s2)</code>	<code>\$s1 = Memory[\$s2 + 100]</code>	Byte de memoria a registro
	<code>store byte</code>	<code>sb \$s1,100(\$s2)</code>	<code>Memory[\$s2 + 100] = \$s1</code>	Byte de registro a memoria
	<code>load upper immed.</code>	<code>lui \$s1,100</code>	<code>\$s1 = 100 * 2¹⁶</code>	Cargar constante en los 16 bits de mayor peso
Lógica	<code>and</code>	<code>and \$s1,\$s2,\$s3</code>	<code>\$s1 = \$s2 & \$s3</code>	Tres registros operandos; AND bit-a-bit
	<code>or</code>	<code>or \$s1,\$s2,\$s3</code>	<code>\$s1 = \$s2 \$s3</code>	Tres registros operandos; OR bit-a-bit
	<code>nor</code>	<code>nor \$s1,\$s2,\$s3</code>	<code>\$s1 = ~ (\$s2 \$s3)</code>	Tres registros operandos; NOR bit-a-bit
	<code>and immediate</code>	<code>andi \$s1,\$s2,100</code>	<code>\$s1 = \$s2 & 100</code>	AND Bit-a-bit registro con constante
	<code>or immediate</code>	<code>ori \$s1,\$s2,100</code>	<code>\$s1 = \$s2 100</code>	OR Bit-a-bit registro con constante
	<code>shift left logical</code>	<code>sll \$s1,\$s2,10</code>	<code>\$s1 = \$s2 << 10</code>	Desplazamiento a la izquierda por constante
	<code>shift right logical</code>	<code>srl \$s1,\$s2,10</code>	<code>\$s1 = \$s2 >> 10</code>	Desplazamiento a la derecha por constante
Salto condicional	<code>branch on equal</code>	<code>beq \$s1,\$s2,L</code>	<code>if (\$s1 == \$s2) go to L</code> <code>PC + 4 + 100</code>	Comprueba igualdad y bifurca relativo al PC
	<code>branch on not equal</code>	<code>bne \$s1,\$s2,L</code>	<code>if (\$s1 != \$s2) go to L</code> <code>PC + 4 + 100</code>	Comprueba si no igual y bifurca relativo al PC
	<code>set on less than</code>	<code>slt \$s1,\$s2,\$s3</code>	<code>if (\$s2 < \$s3) \$s1 = 1;</code> <code>else \$s1 = 0</code>	Compara si es menor que; usado con <code>beq</code> , <code>bne</code>
	<code>set on less than immediate</code>	<code>slti \$s1,\$s2,100</code>	<code>if (\$s2 < 100) \$s1 = 1;</code> <code>else \$s1 = 0</code>	Compara si es menor que una constante
Salto incondicional	<code>jump</code>	<code>j 2500</code>	<code>go to 10000</code>	Salto a la dirección destino
	<code>jump register</code>	<code>jr \$ra</code>	<code>go to \$ra</code>	Para retorno de procedimiento
	<code>jump and link</code>	<code>jal 2500</code>	<code>\$ra = PC + 4; go to 10000</code>	Para llamada a procedimiento

FIGURA 2.1 Lenguaje ensamblador MIPS tratado en este capítulo. Esta información se encuentra también en la Tarjeta de Datos de Referencia de MIPS, que se incluye con este libro.