Mips

Samuel Primera CI: 31.129.684

4 de junio de 2025

1. Operandos Mips

1.1. 32 Registros

Localizaciones rápidas para los datos. En MIPS, los datos deben estar en los registros para realizar operaciones aritméticas. El registro MIPS $zero es siempre igual a O. El registro $ at está reservado por el ensamblador para manejar constantes grandes. Ejemplo:

$s0-$s7. $t0-$t9.
$zero. $a0-$a3.
$vO-$v1. $gp. $fp. $sp.
$ra. $at

1.2. 2^{30} Palabras de memoria

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). Ejemplo:

Memory[0],
Memory[4], ... ,
Memory[4294967292]

2. Lenguaje Ensamblador Mips

Categoría	Instrucción	Ejemplo	Significado	Comentarios
Aritmética	add	`add $s1,$s2,$s3`	$s1 = $s2 + $s3	Tres operandos; datos en registros
	subtract	`sub $s1,$s2,$s3`	$s1 = $s2 – $s3	Tres operandos; datos en registros
	add immediate	`addi $s1,$s2,100`	$s1 = $s2 + 100	Usado para sumar constantes
Transferencia de dato	load word	`lw $s1,100($s2)`	$s1 = Memory[$s2 + 100]	Palabra de memoria a registro
	store word	`sw $s1,100($s2)`	Memory[$s2 + 100] = $s1	Palabra de registro a memória
	load half	`lh $s1,100($s2)`	$s1 = Memory[$s2 + 100]	Media palabra de memoria a registro
	store half	`sh $s1,100($s2)`	Memory[$s2 + 100] = $s1	Media palabra de registro a memoria
	load byte	`lb $s1,100($s2)`	$s1 = Memory[$s2 + 100]	Byte de memoria a registro
	store byte	`sb $s1,100($s2)`	Memory[$s2 + 100] = $s1	Byte de registro a memoria
	load upper immed.	`lui $s1,100`	$s1 = 100 * 2^{16}	Cargar constante en los 16 bits de mayor peso
Lógica	and	`and $s1,$s2,$s3`	$s1 = $s2 & $s3	Tres registros operandos; AND bit-a-bit
	or	`or $s1,$s2,$s3`	$s1 = $s2 | $s3	Tres registros operandos; OR bit-a-bit
	nor	`nor $s1,$s2,$s3`	$s1 = ~ ($s2 |$s3)	Tres registros operandos; NOR bit-a-bit
	and immediate	`andi $s1,$s2,100`	$s1 = $s2 & 100	AND Bit-a-bit registro con constante
	or immediate	`ori $s1,$s2,100`	$s1 = $s2 | 100	OR Bit-a-bit registro con constante
	shift left logical	`sll $s1,$s2,10`	$s1 = $s2 << 10	Desplazamiento a la izquierda por constante
	shift right logical	`srl $s1,$s2,10`	$s1 = $s2 >> 10	Desplazamiento a la derecha por constante
Salto condicional	branch on equal	`beq $s1,$s2,L`	if ($s1 == $s2) go to L PC + 4 + 100	Comprueba igualdad y bifurca relativo al PC
	branch on not equal	`bne $s1,$s2,L`	if ($s1 != $s2) go to L PC + 4 + 100	Comprueba si no igual y bifurca relativo al PC
	set on less than	`slt $s1,$s2,$s3`	if ($s2 < $s3) $s1 = 1; else $s1 = 0	Compara si es menor que; usado con beq, bne
	set on less than immediate	`slt $s1,$s2,100`	if ($s2 < 100) $s1 = 1; else $s1 = 0	Compara si es menor que una constante
Salto incondicional	jump	`j 2500`	go to 10000	Salto a la dirección destino
	jump register	`jr $ra`	go to $ra	Para retorno de procedimiento
	jump and link	`jal 2500`	$ra = PC + 4; go to 10000	Para llamada a procedimiento