

Guía MIPS32: Registros, Instrucciones

2 de junio de 2025

1. Conceptos Fundamentales

1.1. Concepto de Palabra

En MIPS, una **palabra** (word) son 4 bytes (32 bits). Las direcciones de memoria deben estar alineadas a palabras (múltiplos de 4).

2. Registros MIPS

Nombre	Número	¿se mantiene?	Explicación
\$zero	0	Sí (siempre 0)	Siempre contiene el valor 0. No se puede modificar.
\$at	1	No	Registro temporal reservado para el ensamblador (pseudoinstrucciones)
\$v0-\$v1	2-3	No	Valores de retorno de funciones (return values)
\$a0-\$a3	4-7	No	Argumentos para llamadas a funciones (arguments)
\$t0-\$t7	8-15	No	Registros temporales (no preservados entre llamadas)
\$s0-\$s7	16-23	Sí	Registros guardados (preservados entre llamadas)
\$k0-\$k1	26-27	No	Reservados para el núcleo del sistema operativo
\$gp	28	Sí	Puntero global (global pointer)
\$sp	29	Sí	Puntero de pila (stack pointer)
\$fp	30	Sí	Puntero de marco (frame pointer)
\$ra	31	Sí	Dirección de retorno (return address)

- **Número:** Cada registro tiene un identificador numérico (0-31) que se usa en la codificación binaria de las instrucciones. Por ejemplo, \$t0 es el registro 8.

- **¿Se mantiene?:** Indica si el registro debe preservar su valor después de una llamada a función (convención de llamadas):
 - **Sí:** La función llamada debe dejar el registro con el mismo valor que tenía al entrar (usualmente se guarda en la pila).
 - **No:** La función llamada puede modificar el registro sin restricciones.

3. Manejo de Memoria

3.1. Carga y Almacenamiento

```

1 lw $t0, 0($sp)      # Carga palabra de ($sp + 0) en $t0
2 sw $t0, 4($sp)      # Almacena $t0 en ($sp + 4)
3 lb $t0, 0($sp)      # Carga byte (sign-extend) en $t0
4 lbu $t0, 0($sp)     # Carga byte (zero-extend) en $t0
5 sb $t0, 0($sp)      # Almacena byte (los 8 bits bajos de $t0)

```

4. Instrucciones Aritméticas

4.1. Aritmética Básica

```

1 add $t0, $t1, $t2   # $t0 = $t1 + $t2 (suma con signo)
2 addu $t0, $t1, $t2  # $t0 = $t1 + $t2 (suma sin signo)
3 sub $t0, $t1, $t2   # $t0 = $t1 - $t2 (resta con signo)
4 subu $t0, $t1, $t2  # $t0 = $t1 - $t2 (resta sin signo)
5 addi $t0, $t1, 5    # $t0 = $t1 + 5 (suma inmediata)

```

4.2. Multiplicación y División

```

1 mult $t0, $t1        # Hi/Lo = $t0 * $t1 (producto de 32 bits)
2 div $t0, $t1         # Lo = $t0/$t1, Hi = $t0 % $t1 (division con
    signo)
3 mflo $t2             # $t2 = Lo (copia registro bajo)
4 mfhi $t3             # $t3 = Hi (copia registro alto)

```

5. Instrucciones de Control

5.1. Comparaciones

```

1 slt $t0, $t1, $t2    # $t0 = 1 si $t1 < $t2 (con signo), 0 en otro
    caso
2 sltu $t0, $t1, $t2   # $t0 = 1 si $t1 < $t2 (sin signo), 0 en otro
    caso
3 slti $t0, $t1, 10    # $t0 = 1 si $t1 < 10 (inmediato con signo)

```

5.2. Saltos Condicionales

```
1 beq $t0, $t1, etiqueta # Salta si $t0 == $t1
2 bne $t0, $t1, etiqueta # Salta si $t0 != $t1
3 beqz $t0, etiqueta     # Salta si $t0 == 0 (pseudoinstrucci n)
4 bgt $t0, $t1, etiqueta # Salta si $t0 > $t1 (pseudoinstrucci n)
5 blt $t0, $t1, etiqueta # Salta si $t0 < $t1 (pseudoinstrucci n)
6 bge $t0, $t1, etiqueta # Salta si $t0 >= $t1 (pseudoinstrucci n)
7 ble $t0, $t1, etiqueta # Salta si $t0 <= $t1 (pseudoinstrucci n)
```

5.3. Saltos Incondicionales

```
1 j etiqueta           # Salto incondicional
2 jal etiqueta         # Salto y enlace (para llamadas a funci n)
3 jr $ra               # Salto a direcci n en $ra (retorno de
                        funci n)
```

6. Entrada/Salida Bsica

6.1. Lectura y Escritura de Enteros

```
1 # Ejemplo: Leer un entero y mostrarlo
2 .data
3     mensaje: .asciiz "Ingrese un numero: "
4     resultado: .asciiz "El numero ingresado es: "
5
6 .text
7 main:
8     # Mostrar mensaje
9     li $v0, 4           # C digo para imprimir string
10    la $a0, mensaje      # Carga direcci n del mensaje
11    syscall
12
13    # Leer entero
14    li $v0, 5           # C digo para leer entero
15    syscall
16    move $t0, $v0       # Guardar el valor le do en $t0
17
18    # Mostrar resultado
19    li $v0, 4
20    la $a0, resultado
21    syscall
22
23    li $v0, 1           # C digo para imprimir entero
24    move $a0, $t0       # Cargar el valor a imprimir
25    syscall
26
27    # Terminar programa
28    li $v0, 10
29    syscall
```

6.2. Manejo de Caracteres

```
1 .data
2     char: .byte 'A'      # Caracter ASCII
3     buffer: .space 20    # Buffer para cadena
4
5 .text
6 main:
7     # Leer un caracter
8     li $v0, 12            # Codigo para leer caracter
9     syscall
10    sb $v0, char          # Almacenar caracter leido
11
12    # Mostrar caracter
13    li $v0, 11            # Codigo para imprimir caracter
14    lb $a0, char          # Cargar caracter
15    syscall
16
17    # Leer cadena
18    li $v0, 8              # Codigo para leer cadena
19    la $a0, buffer         # Direccion del buffer
20    li $a1, 20             # Longitud maxima
21    syscall
22
23    # Mostrar cadena
24    li $v0, 4              # Codigo para imprimir cadena
25    la $a0, buffer         # Direccion del buffer
26    syscall
```

6.3. Números Decimales (Punto Flotante)

```
1 .data
2     flotante: .float 3.1416
3     doble: .double 2.71828
4     mensaje: .asciiz "El valor es: "
5
6 .text
7 main:
8     # Leer float
9     li $v0, 6              # Codigo para leer float
10    syscall                # Resultado en $f0
11
12    # Mostrar float
13    li $v0, 4              # Codigo para imprimir float
14    la $a0, mensaje        # Direccion del mensaje
15    syscall
16
17    # Leer double
18    li $v0, 7              # Codigo para leer double
19    syscall                # Resultado en $f0-$f1
20
21    # Leer double
22    li $v0, 7              # Codigo para leer double
23    syscall                # Resultado en $f0-$f1
24
```

```
25      # Mostrar double
26      li $v0, 4
27      la $a0, mensaje
28      syscall
29
30      li $v0, 3          # Codigo para imprimir double
31      mov.d $f12, $f0    # Cargar valor a imprimir
32      syscall
```