Operandos MIPS

- **32 Registros:** Son ubicaciones de datos de acceso rápido para la CPU, utilizadas para almacenar datos que serán operandos en operaciones aritméticas. El registro `$zero` siempre contiene el valor 0.
- **30 Palabras de Memoria:** Permiten el acceso a datos directamente desde la memoria, aunque de forma más lenta que los registros. La memoria se utiliza para guardar los datos que se desbordan de los registros.

Lenguaje Ensamblador MIPS

1. Aritmética

- `add`: Suma el contenido de dos registros y guarda el resultado en un tercer registro.
- `subtract`: Resta el contenido de dos registros y guarda el resultado en un tercer registro.
- `addi` (add immediate): Suma el contenido de un registro con una constante (inmediato) y guarda el resultado.

2. Transferencia de Datos

- `load word (lw)`: Carga una palabra (32 bits) desde una dirección de memoria especificada a un registro.
- `store word (sw)`: Almacena una palabra (32 bits) desde un registro a una dirección de memoria especificada.
- `load half (lh)`: Carga media palabra (16 bits) desde una dirección de memoria especificada a un registro.
- `store half (sh)`: Almacena media palabra (16 bits) desde un registro a una dirección de memoria especificada.
- `load byte (lb)`: Carga un byte (8 bits) desde una dirección de memoria especificada a un registro.
- `store byte (sb)`: Almacena un byte (8 bits) desde un registro a una dirección de memoria especificada.
- `load upper immediate (lui)`: Carga una constante de 16 bits en los 16 bits más significativos de un registro, rellenando los 16 bits menos significativos con ceros.

3. Lógica

- `and`: Realiza una operación AND bit a bit entre el contenido de dos registros y guarda el resultado.
- `or`: Realiza una operación OR bit a bit entre el contenido de dos registros y guarda el resultado.
- `nor`: Realiza una operación NOR bit a bit entre el contenido de dos registros y guarda el resultado.
- `and immediate (andi)`: Realiza una operación AND bit a bit entre el contenido de un registro y una constante, guardando el resultado.
- `or immediate (ori)`: Realiza una operación OR bit a bit entre el contenido de un registro y una constante, guardando el resultado.
- `shift left logical (sll)`: Desplaza lógicamente los bits de un registro a la izquierda por un número constante de posiciones.
- `shift right logical (srl)`: Desplaza lógicamente los bits de un registro a la derecha por un número constante de posiciones.

4. Salto Condicional

- `branch on equal (beq)`: Comprueba si el contenido de dos registros es igual. Si lo son, salta a una etiqueta (dirección) especificada.
- `branch on not equal (bne)`: Comprueba si el contenido de dos registros no es igual. Si no lo son, salta a una etiqueta (dirección) especificada.
- `set on less than (slt)`: Comprueba si el contenido de un registro es menor que el de otro. Si es verdadero, el registro destino se establece a 1; de lo contrario, se establece a 0.
- `set on less than immediate (slti)`: Comprueba si el contenido de un registro es menor que una constante. Si es verdadero, el registro destino se establece a 1; de lo contrario, se establece a 0.

5. Salto Incondicional

- `jump (j)`: Salta incondicionalmente a una dirección especificada.
- `jump register (jr)`: Salta incondicionalmente a la dirección contenida en un registro.
- `jump and link (jal)`: Salta incondicionalmente a una dirección especificada y guarda la dirección de retorno (la dirección de la instrucción siguiente a `jal`) en el registro `$ra` (register 31), lo cual es útil para llamadas a subrutinas.