ANÁLISIS DEL CÓDIGO ENSAMBLADOR

Archivo: programa.asm
Descripción: Este programa recorre un arreglo de 5 números en memoria y suma aquellos que cumplen dos condiciones simultáneas: ser mayores a 50 y ser impares.

1. Fase de Inicialización (Configuración)

Antes de iniciar el procesamiento, se configuran los registros con los valores iniciales.

- addi $t0, $zero, 0: Define la **Dirección Base**. El arreglo de datos comienza en la posición de memoria 0.
- addi $t1, $zero, 0: Inicializa el índice i en 0. Este registro se usará para recorrer el arreglo.
- addi $t2, $zero, 5: Establece N = 5, indicando que se procesarán un total de 5 números.
- addi $t5, $zero, 0: Inicializa el **Acumulador** en 0. Aquí se guardará la suma total de los números válidos.
- addi $t6, $zero, 50: Define el **Umbral = 50**, el valor contra el cual se compararán los datos.

2. Fase de Procesamiento (Bucle loop)

Esta sección se ejecuta repetidamente (5 veces), una por cada número del arreglo.

Paso A: Calcular Dirección y Cargar Dato

1. add $t7, $t0, $t1: Calcula la dirección de memoria exacta sumando la dirección base ($t0) y el índice actual ($t1). El resultado se guarda en $t7.
2. lw $t3, 0($t7): Instrucción Load Word. El procesador va a la memoria en la dirección calculada ($t7), lee el número almacenado y lo guarda en el registro temporal $t3.

Paso B: Primer Filtro (Magnitud)

Se verifica si el número es mayor que 50.

1. slt $t4, $t6, $t3: Instrucción Set Less Than. Compara si 50 ($t6) es menor que el número actual ($t3).
    - Si es VERDADERO (Número > 50), $t4 se pone en 1.
    - Si es FALSO (Número <= 50), $t4 se pone en 0.
2. beq $t4, $zero, next: Instrucción Branch if Equal. Si $t4 es 0 (la condición anterior fue falsa), el programa salta a la etiqueta next, descartando el número.

Paso C: Segundo Filtro (Paridad)

Se verifica si el número es impar.

1. andi $t4, $t3, 1: Instrucción AND Immediate. Realiza una operación lógica AND con el número 1. Esto aísla el bit menos significativo.
    - Resultado 1: El número es IMPAR.
    - Resultado 0: El número es PAR.

2. addi $t8, $zero, 1: Carga el valor 1 en el registro auxiliar $t8 para la comparación.
3. beq $t4, $t8, next: Esta instrucción salta a next si el resultado del AND ($t4) es igual a 1 ($t8).
    - *Comportamiento actual:* Si es IMPAR, salta (descarta). Si es PAR, continúa (suma).
    - *Para sumar impares:* La lógica debería ser inversa (beq $t4, $zero, next), pero se explica tal cual está escrito en el código.

Paso D: Acumulación

Si el número superó los filtros (no se realizó ningún salto a next):

1. add $t5, $t5, $t3: Se suma el número actual ($t3) al valor que ya tenía el acumulador ($t5).

3. Fase de Iteración y Cierre

Preparar Siguiente Ciclo (next)

1. addi $t1, $t1, 4: Se incrementa el índice en 4. (En la arquitectura MIPS, cada número entero ocupa 4 bytes de memoria).
2. addi $t9, $zero, 20: Se calcula el límite de memoria (5 números x 4 bytes = 20).
3. beq $t1, $t9, end_loop: Se compara el índice actual con el límite. Si i == 20, significa que ya se revisaron todos los números, por lo que salta a la etiqueta end_loop.
4. j loop: Si no se ha llegado al límite, salta incondicionalmente al inicio de loop para procesar el siguiente número.

Finalización (end_loop)

1. sw $t5, 100($zero): Instrucción **Store Word**. Guarda el resultado final de la suma ($t5) en la dirección de memoria 100 para su posterior verificación.
2. j end_loop: Bucle infinito. Mantiene al procesador en un estado conocido y seguro para finalizar la ejecución.