Optimización: Reemplazo de multiplicaciones y divisiones por desplazamientos

Lo que estás viendo es un ejemplo clásico de optimización a nivel bajo (bitwise), que consiste en reemplazar multiplicaciones y divisiones por potencias de 2 usando operaciones de desplazamiento (<< y >>). Estas operaciones son mucho más rápidas para el procesador que las multiplicaciones y divisiones tradicionales.

# Código original:

int i, j, a, b=0, m3=8, m5=32;  
for (j=0; j<ITER; j++)  
 for (i=0; i<N; i++) {  
 a = i \* m3;  
 b += a / m5;  
 }

Sabemos que:

* - i \* 8 se puede escribir como i << 3
* - a / 32 se puede escribir como a >> 5

Entonces: (i \* 8) / 32 = i \* (8 / 32) = i \* 1/4 = i / 4 = i >> 2

# Código optimizado:

for (j=0; j<ITER; j++) {  
 for (i=0; i<N; i++) {  
 b += i >> 2;  
 }  
 a = (N - 1) << 3;  
}

## Sobre el valor de a al final

a = (N - 1) << 3 es equivalente a (N - 1) \* 8. En el código original, la variable a se reasignaba en cada iteración del bucle interno, pero al final siempre contenía el mismo valor: (N - 1) \* 8. Como no se usa a dentro del bucle para nada más, se calcula una sola vez al final, optimizando también eso.

# Conclusión

Este tipo de optimización:

* - Reduce el costo computacional, evitando multiplicaciones y divisiones.
* - Aumenta la eficiencia en sistemas embebidos o de bajo nivel.

- Es válida solo cuando los factores son potencias de 2, como en este caso (8 = 2^3, 32 = 2^5, etc.).

¿Querés que te explique cómo hacerlo con números que no son potencias de 2? También se puede, aunque es más complejo y menos exacto.