La aparente contradicción en los tiempos se debe a dos factores clave:

Asignación redundante en el código optimizado: La línea a = (N-1) << 3; fuera del bucle interno añade una operación innecesaria que no existe en la versión original, incrementando ligeramente el tiempo de ejecución. Además, aunque a no se usa posteriormente, el compilador no elimina esta operación al usar -O0.

Optimizaciones implícitas en la versión original: Aunque compilado con -O0, el código original podría beneficiarse de ciertas optimizaciones mínimas del compilador. Por ejemplo, el cálculo de a = i * 8 seguido de a / 32 podría simplificarse automáticamente a i / 4, reduciendo operaciones, mientras que la versión optimizada introduce una asignación extra.

Soluciones para mejorar la versión sin optimizar:
Eliminar la variable a en el código original: Reemplazar directamente b += (i * 8) / 32 por b += i >> 2 sin cálculos intermedios.

Usar tipos unsigned para desplazamientos: Evitar problemas de comportamiento indefinido con enteros con signo:

c

```
unsigned i, j, a, b = 0;
for (j = 0; j < ITER; j++)
    for (i = 0; i < N; i++)
        b += i >> 2;
```

Simplificar aún más el código: Eliminar cualquier operación redundante que el compilador no optimiza en -O0.

¿Por qué la versión optimizada es más lenta?

Operación adicional: La línea a = (N-1) << 3; añade una operación por iteración del bucle externo, lo que incrementa el tiempo.

Falta de optimizaciones: Con -O0, el compilador no elimina código muerto ni reordena instrucciones, haciendo que la versión optimizada manualmente incluya pasos innecesarios.

Conclusión: La versión optimizada contiene una operación redundante que penaliza su rendimiento. Eliminando a = (N-1) << 3; y usando tipos unsigned, la versión optimizada superaría a la original. Para la versión sin optimizar, aplicar desplazamientos directamente y eliminar variables temporales mejora su eficiencia incluso con -O0.

Respuesta final:

La versión optimizada es más lenta debido a una operación redundante (a = (N-1) << 3;). Para mejorar la versión sin optimizar, reemplaza las multiplicaciones/divisiones por desplazamientos y usa tipos unsigned:

c

```
unsigned i, j, b = 0;
for (j = 0; j < ITER; j++)
    for (i = 0; i < N; i++)
        b += i >> 2;
```

Esto elimina cálculos innecesarios y aprovecha mejor las operaciones de bits.