¿Vale la pena usar desplazamientos de bits en lugar de multiplicaciones/divisiones?

La pregunta central del post era:

¿Por qué usar desplazamientos de bits y sumas/restas en lugar de multiplicaciones o divisiones? ¿No lo optimiza ya el compilador?

📌 Puntos clave extraídos del debate:

Depende del microcontrolador (MCU):

Algunos MCUs como el MSP430 no tienen instrucciones de multiplicación o división por hardware.

En esos casos, la multiplicación/división se implementa en software y puede tardar muchos ciclos.

Compiladores modernos:

Pueden optimizar x \* 4 a x << 2 automáticamente.

Pero solo cuando los operandos son constantes. Si uno es variable (x \* y), eso ya no es tan sencillo.

Desempeño y consumo energético:

Los shifts y sumas son de muy bajo consumo y casi siempre de 1 ciclo.

Las multiplicaciones, aunque a veces rápidas, son más costosas en energía y pueden no ser de un solo ciclo.

Portabilidad y comportamiento definido:

C tiene reglas un tanto ambiguas sobre los desplazamientos con números negativos.

Al dejar que el compilador lo haga, se asegura un comportamiento válido y definido.

Optimización manual en DSPs:

En aplicaciones como filtros (FIR, IIR), donde hay multiplicaciones fijas, es útil preprocesar los coeficientes para expresarlos como combinaciones de shifts + sumas/restas (como hizo uno de los usuarios con un script en Python).

Rendimiento medible solo con pruebas:

Lo mejor es comparar las versiones compiladas y revisar el ensamblador generado.

Una operación puede ocupar una sola instrucción pero muchos ciclos.

🧪 Conclusión técnica:

Si estás usando un procesador como el MSP430, sí tiene sentido usar shifts/sumas.

Si estás en un entorno moderno con FPU o multiplicación por hardware rápida, puede que no valga la pena optimizarlo manualmente.

Siempre es buena práctica medir si el cambio realmente mejora el rendimiento.