Ejercicio_2

Parte 1.

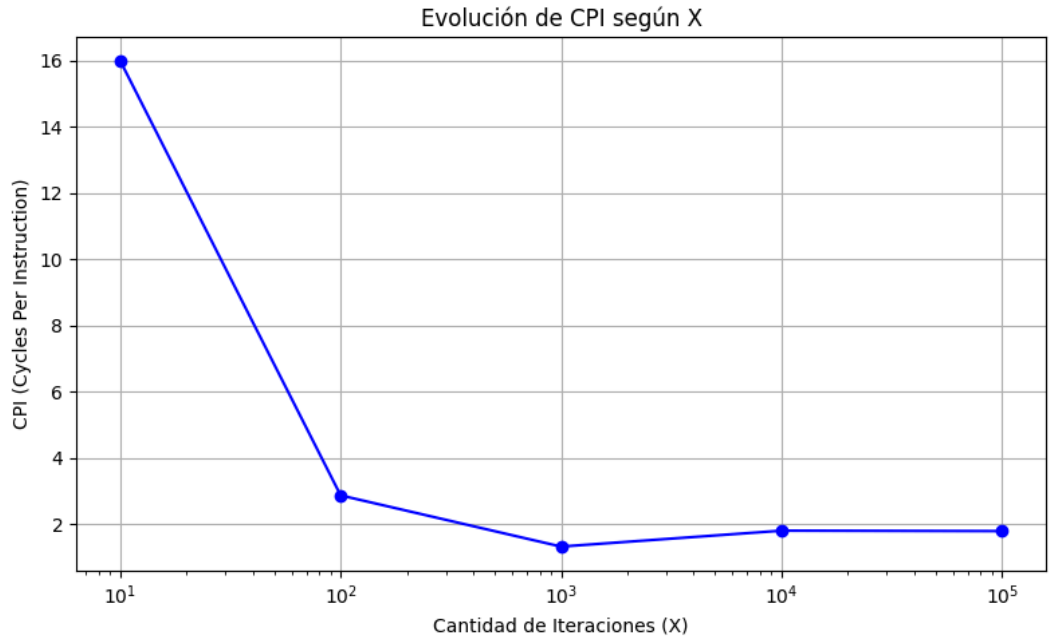
La gráfica de evolución del CPI en función de la cantidad de iteraciones (X) muestra una tendencia clara: el CPI disminuye drásticamente al aumentar X, especialmente en las primeras magnitudes.

Para X = 10, el CPI es extremadamente alto: 16, lo cual indica un gran sobrecosto en ciclos por instrucción cuando se ejecutan pocas iteraciones.
A medida que X aumenta, el CPI se estabiliza entre 1.3 y 1.8, a partir de X = 1000 en adelante.

Esto indica que:

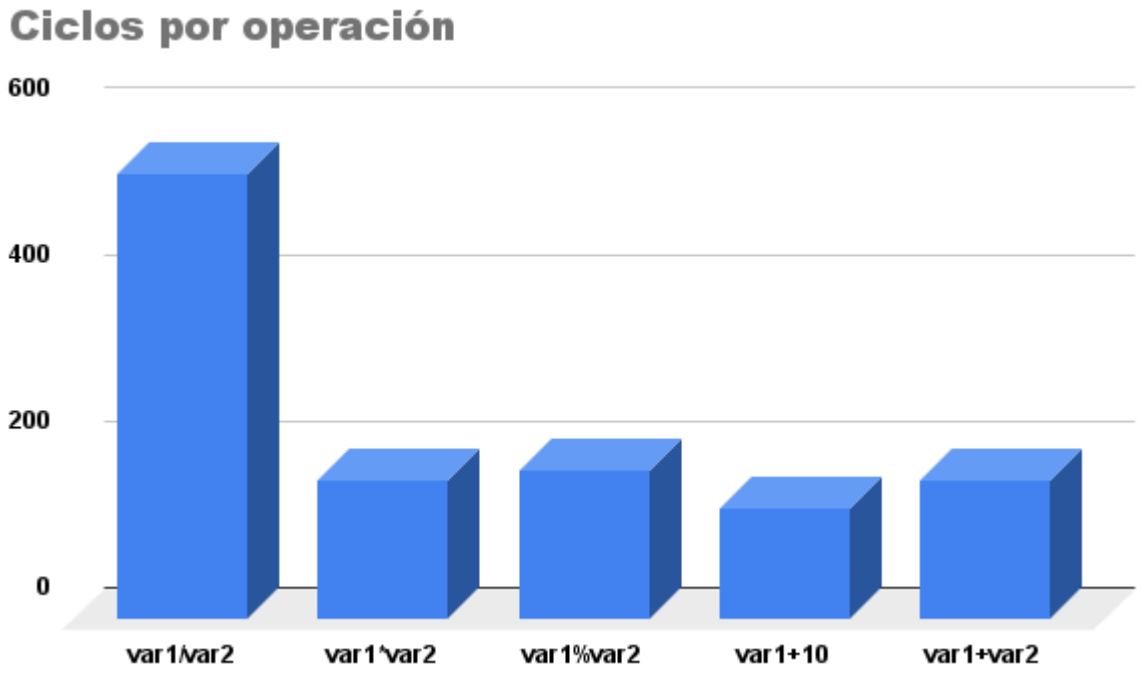
- Para valores pequeños de X, el sistema aún está influenciado por overheads (tiempo de inicio de la medición, latencias iniciales del sistema operativo, planificación de tareas, interrupciones, etc.).
- Con más iteraciones, el peso de estos overheads se diluye, y el CPI refleja mejor el rendimiento real del procesador al ejecutar las instrucciones.

En resumen, la relación entre X y CPI es inversamente proporcional al comienzo, y luego tiende a estabilizarse conforme la ejecución se vuelve más representativa del comportamiento promedio del sistema.



Parte 2.

Para este punto se midió la cantidad total de ciclos de CPU requeridos para ejecutar una única operación repetida X veces (donde X fue fijo y suficientemente grande para obtener una medición confiable). A continuación se presentan los resultados obtenidos:



Operaciones de suma entera (+)

Tanto var1 + var2 como var1 + 10 son operaciones de suma entera, pero la suma con una constante (+10) es más eficiente (133 ciclos) que la suma entre dos variables (167 ciclos). Esto puede deberse a que el compilador y la arquitectura pueden optimizar mejor operaciones con constantes, posiblemente cargando el valor inmediato directamente en el registro.

Operación de módulo (%)

El módulo es más costoso que una suma o multiplicación, con 180 ciclos. Esto es esperable, ya que internamente se realiza como una división y una resta, lo cual implica mayor uso de recursos del ALU (Unidad Aritmética y Lógica).

Multiplicación (*)

Se obtiene un tiempo de 167 ciclos, el mismo que la suma entre dos variables. Esto puede deberse a que, en la arquitectura ESP32, la multiplicación entera está bien optimizada en hardware, y su latencia no es significativamente mayor que una suma. Sin embargo, esto no siempre ocurre en todas las arquitecturas.

División flotante (/)

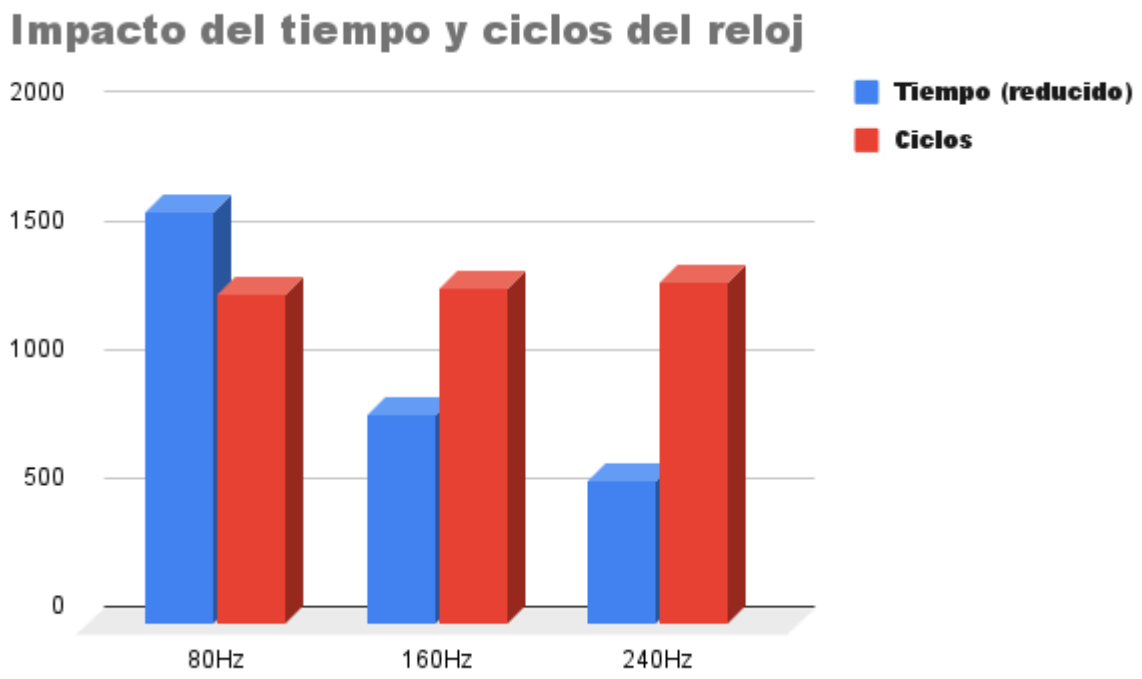
Esta es, con diferencia, la operación más costosa, con 536 ciclos, más de 3 veces mayor que cualquier otra operación medida. Esto tiene sentido, ya que las divisiones flotantes:

- Implican conversión de enteros a flotantes.
- Requieren el uso de la FPU (unidad de punto flotante), que tiene mayor latencia.
- No siempre están completamente implementadas en hardware en todos los microcontroladores.

En conclusión, la cantidad de ciclos varía notablemente según el tipo de operación realizada. Las operaciones más simples como la suma con constante tienen el menor costo, mientras que operaciones complejas como la división en punto flotante tienen un costo mucho mayor. Esto es fundamental para comprender el comportamiento del CPI en microcontroladores, ya que al ejecutar código con muchas divisiones o módulos, el tiempo total y el consumo energético pueden aumentar significativamente.

Parte 3.

Se realizaron pruebas variando la frecuencia del reloj del ESP32 utilizando idf.py menuconfig. Para cada frecuencia se ejecutó el mismo bloque de código, y se midieron tanto los ciclos de CPU utilizados como el tiempo total de ejecución en microsegundos. Los resultados obtenidos fueron los siguientes:



Interpretación de los resultados

El tiempo de ejecución disminuye con mayor frecuencia. A medida que la frecuencia del microcontrolador aumenta de 80 MHz a 240 MHz, el tiempo requerido para ejecutar el mismo código disminuye significativamente.

A 80 MHz se tarda 1602 µs, mientras que a 240 MHz solo 554 µs, es decir, se reduce a aproximadamente un 35% del tiempo original. Esto valida el comportamiento esperado de los sistemas digitales: al incrementar la frecuencia del reloj, se incrementa la cantidad de ciclos por segundo, por lo tanto, el mismo número de instrucciones se ejecuta más rápido.

La cantidad de ciclos se mantiene casi constante (Esto se encuentra reducido en el gráfico para mejor visibilidad de datos). Aunque se observa una leve variación en el número de ciclos (de 127895 a 132853), esta diferencia es pequeña y puede atribuirse a factores externos como interrupciones, planificación de tareas del RTOS o latencias de acceso a memoria. Lo importante es que el número de ciclos no cambia significativamente con la frecuencia, lo cual tiene sentido: los ciclos son una medida relativa al reloj, no al tiempo absoluto.

En conclusión, el tiempo total de ejecución es inversamente proporcional a la frecuencia del reloj. El número de ciclos ejecutados es prácticamente constante, independientemente de la

frecuencia. Esto demuestra que el rendimiento en tiempo puede mejorarse aumentando la frecuencia del procesador, pero que la eficiencia medida en ciclos depende del código y no del clock. También resalta que al trabajar con optimizaciones o perfiles de rendimiento, es crucial distinguir entre tiempo real y ciclos de CPU, ya que uno depende del hardware (frecuencia) y el otro del software (cantidad y tipo de instrucciones ejecutadas).