

**Universidad autónoma de baja california**

**Ingeniero en computación**

**microcontroladores**

**practica 4**

**Maestro: Jose Isabel Garcia Rocha**

**Erik Garcia Chávez 01275863**

**30 de septiembre del 2024**

**Teoría:**

**Programación en lenguaje C en microcontroladores**

en lenguaje de programación C es el lenguaje por excelencia cuando se trata de programar estos circuitos integrados, por la alta capacidad que tiene C en el manejo en memoria, manejo de registros, pines de I/O. así como a la facilidad de comunicarse con el ensamblador.

**Convención de llamadas a funciones en C en gcc para AVR (avr-gcc)**

En el entorno de desarrollo con avr-gcc para microcontroladores AVR, la convención de llamadas a funciones se refiere a cómo se pasan los argumentos a las funciones y cómo se retorna el valor. Aquí hay algunas convenciones comunes:

Convención de llamada por defecto: En C, la convención de llamada por defecto es cdecl, donde los argumentos se pasan de derecha a izquierda y el llamador limpia la pila después de la llamada.

Convención de llamada stdcall: Similar a cdecl, pero el llamador no limpia la pila; la función llamada se encarga de ello. Esta convención es común en las API de Windows.

Convención de llamada fastcall: Los primeros argumentos se pasan en registros, lo que puede hacer la llamada más rápida, pero solo es útil para un número limitado de argumentos.

Convención de llamada naked: No se generan secuencias de entrada/salida por el compilador, lo que permite al programador manejar todo el código de la función en ensamblador

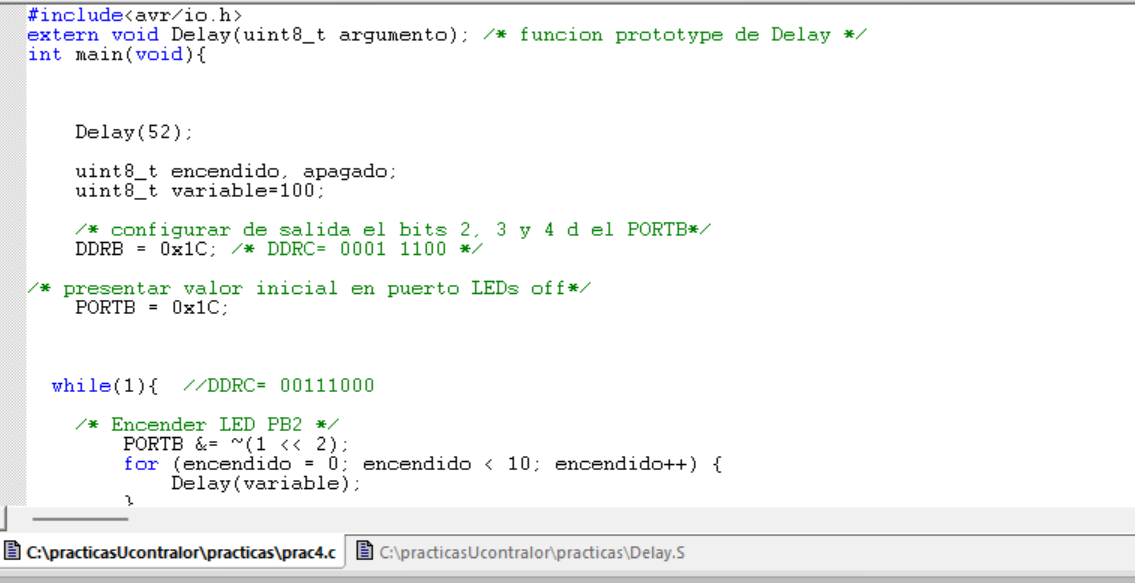
**Instrucciones:**

Haciendo uso de los pasos descritos en la Practica 1 crear un nuevo proyecto, pero esta vez para hacer uso del lenguaje C con base en el Listado 1 (Prac4.c) y del lenguaje ensamblador con el Listado 2 (Delay.S). El programa tiene como fin el mostrar el llamado a funciones escritas en lenguaje ensamblador desde código escrito lenguaje C. ensamblador.

**Desarrollo:**

En primera instancia contamos con la declaración de la función, la cual es llamada de C a ensamblador, la función en ASM se encargara de hacer el retardo de 0-255 ms, con la excepción que no se podrá hacer de manera efectiva para 0ms, porque para 0 ms se necesitan hacer 0 ticks del procesador, lo cual no puede realizarse de manera efectiva, por el solo hecho de ir de C a ASM consta de 6 ticks, más aun los Branch asi como el ret, al final podría ser alrededor de 20 ticks de procesador, el solo procesar y hacer una excepción sobre que no se puede hacer un delay de 0ms. Para 1ms no es tan exacto, pero para todos los demás se pretendió ser lo más exacto posible.

La función recibe un argumento, el cual es el tiempo que se quiere procesar, el cual será representado en mS., la función es de tipo void ya que no necesito que proese nada, tan solo que queme tiempo de procesador.

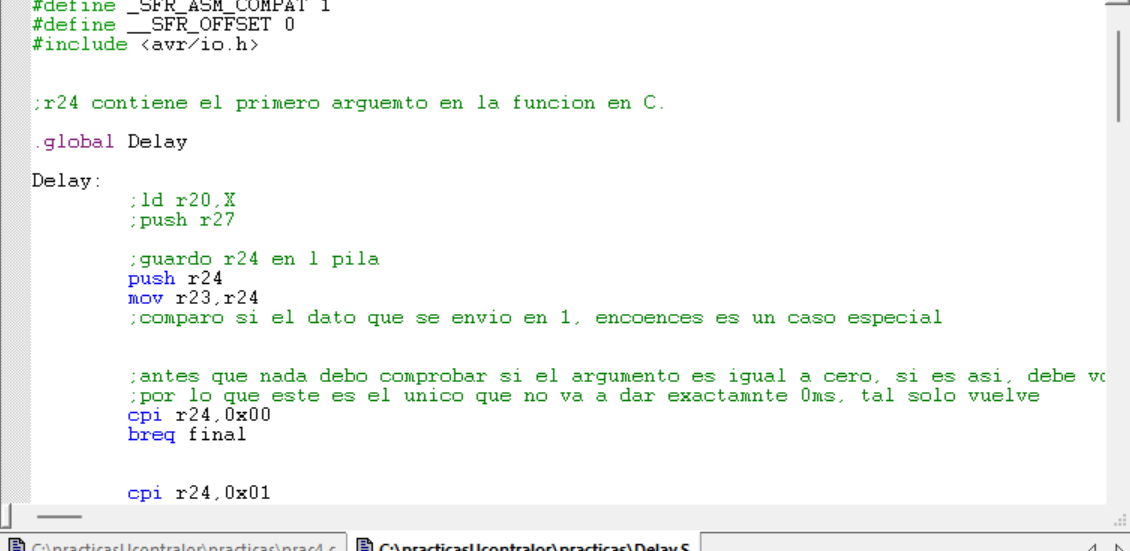


**Delay.S**

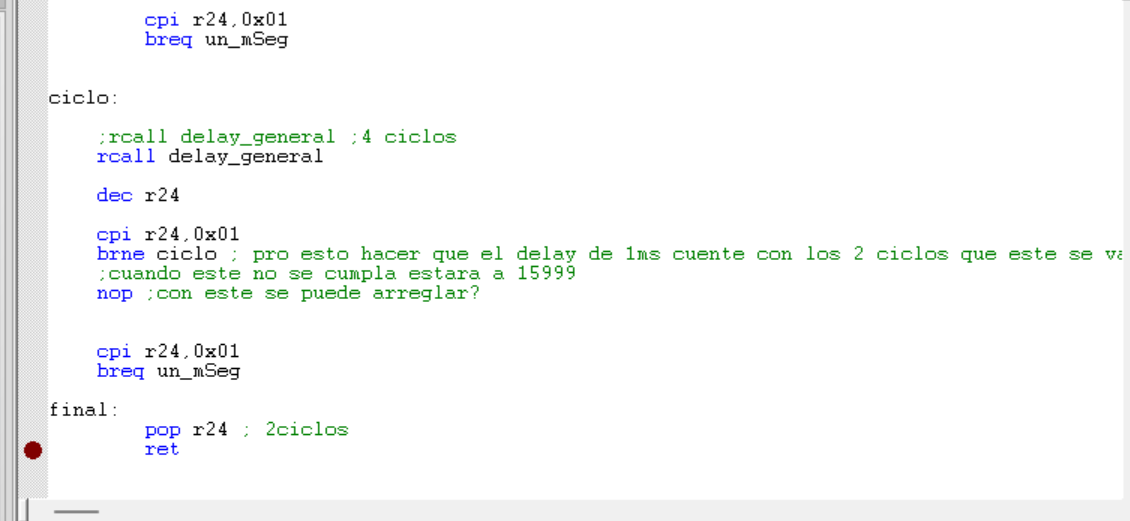
Establecemos la etiqueta con la cual pude ser llamada por cualquier archivo. Siempre y cunado se enlace el objeto con este archivo claramento, pero eso AVR lo hace por nosotros.

R24 es el registro que trae nuestro parámetro a ASM, por lo que lo guardo, lo realice así porque si no hacia eso me dama errores.

Hacemos la primera comparación que es verificar antes que realizar cualquier operación lógica del delay es saber si el numero no es 0, si es cero, entonces este tiene que volver a C, porque es un no valido, en el sentido que no se puede realizar un delay de 0 mS con 0 ticks. Por lo que regresamos a C. tan solo, por lo que si no es 0 entonces el programa puede seguir.

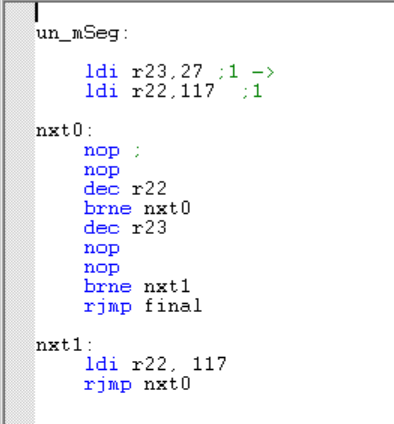


Se tiene una segunda condicional, en la cual si el parámetro que se recibió es 1, entonces se manda a llamar a una función especial, la cual calcula 16000 ciclos de reloj, pero ese toma en cuenta todos los ticks se se realizan desde que se manda a llamar de C hasta el regreso, el RET, pero si R24 no es 1 entonces es cualquier otro numero por lo que se calcula los ciclos exactos a excepción del último, como se puede ver la etiqueta, lo que hace es un ciclo que manda a llamar a la etiqueta “***delay\_general***”, el delay\_general calcula mS a mS, hace 16000 ciclos exactos regresa a esta etiqueta “ciclo”, hace la comparación si R24 no es 1 entonces se vuelve a hacer la llamada al “***delay\_general***” cuando ya se está en la última iteración se llama a la etiqueta que tiene el delay que dura igual 16000 ciclos pero toma en cuenta en su cálculo todos los ciclos que se toma para decisiones, así como el CALL y el RET de C a ASM y devuelta.



**Ciclos un “un\_mSeg”**:

Se utiliza 2 ciclos, para hacer el retardo de 16000 ciclos, pero esta toma en cuenta, todos los ticks se hicieron hasta antes de llegar a la subrutina, las comparaciones, el tick de ir de C a AMS, etc. por lo que puede ir de 11 a 14 tikcs, previos, 11 ticks en el caso de que se mande 1 mS como retardo, por lo que no seguirá a la otra subrutina que calcula los 16000 tikcs exactos,



La fórmula calculada a partir del siguiente ciclo para calcular los ticks que necesito, necesito 16000 tikcs de reloj para 1mS

Formula=1 +1 + 1XY + 1XY+ 1XY + 2X(Y-1) + 1X+ 1X + 1X+ 2(X-1) +4 + 1X + 4X

Que despejándola nos queda

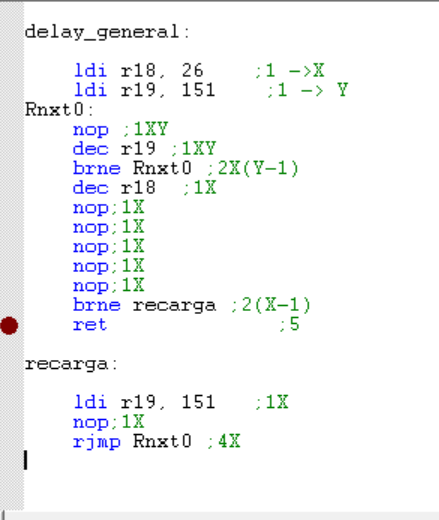
Ticks = 8X + 5XY + 4

Sustituyendo los valores con X = 27 y Y=117, nos queda

Tikcs = 8(27) + 5(27+117) +4 = 16015

Nos acercamos mucho

Se tiene a “***delay\_general”*** el cual calcula los 16000 ciclos exactos para todos los mS antes de llegar al último, el delay es muy similar a cunado es para el ultimo mS, solo que en este sus valores cambian, y se agregan más nops, dado que se requiere llegar a más ciclos.



La ecuación para calcular la formula quedo de la siguiente manera:

Tikcs delay general=

1+1+1XY+1XY+2X(Y-1) +1X+1X+1X+1X+1X+1X+2(X-1) +5+1X+1X+4X

Donde si la despejamos nos dejaría con

Tikcs delay general= 12X + 4XY + 5, donde si sustituimos los variables con los valores de X y Y que asignamos a r18 y r19 respectivamente, tenemos:

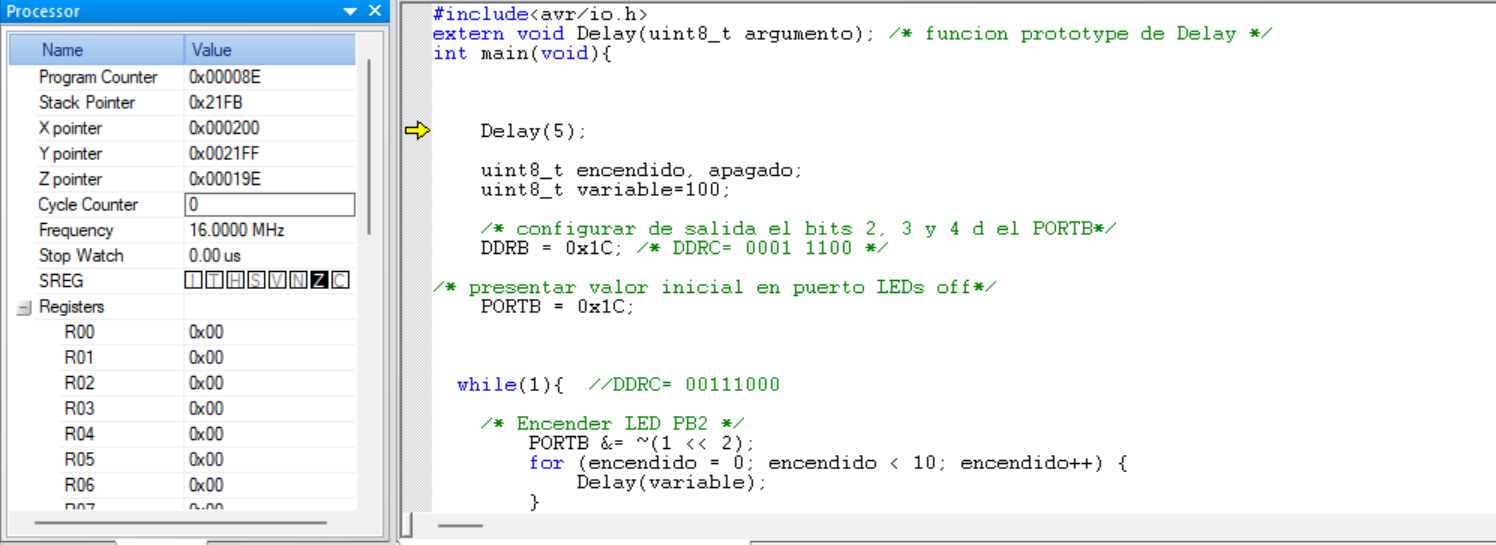
Ticks delay general = 12(26) + 4(26\*151) +5 = 16021

Igual nos quedamos bastante cercanos al valor que se estima.

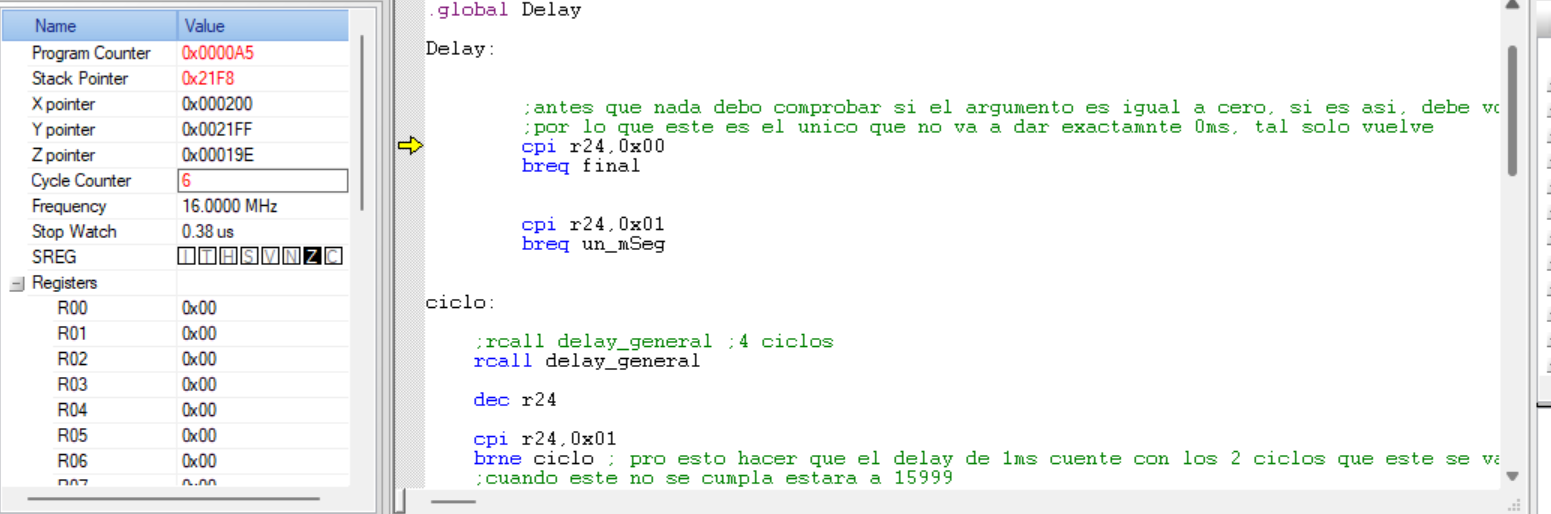
**Ejecución del programa:**

Si queremos un delay para 5mS los ticks totales tendrían que ser de 80000 ticks de ciclo de reloj, desde va de C a ASM hasta que vuelve de ASM a C.

Primero se inicia el programa y limpiamos los ciclos que se ejecutaron para iniciar el programa:

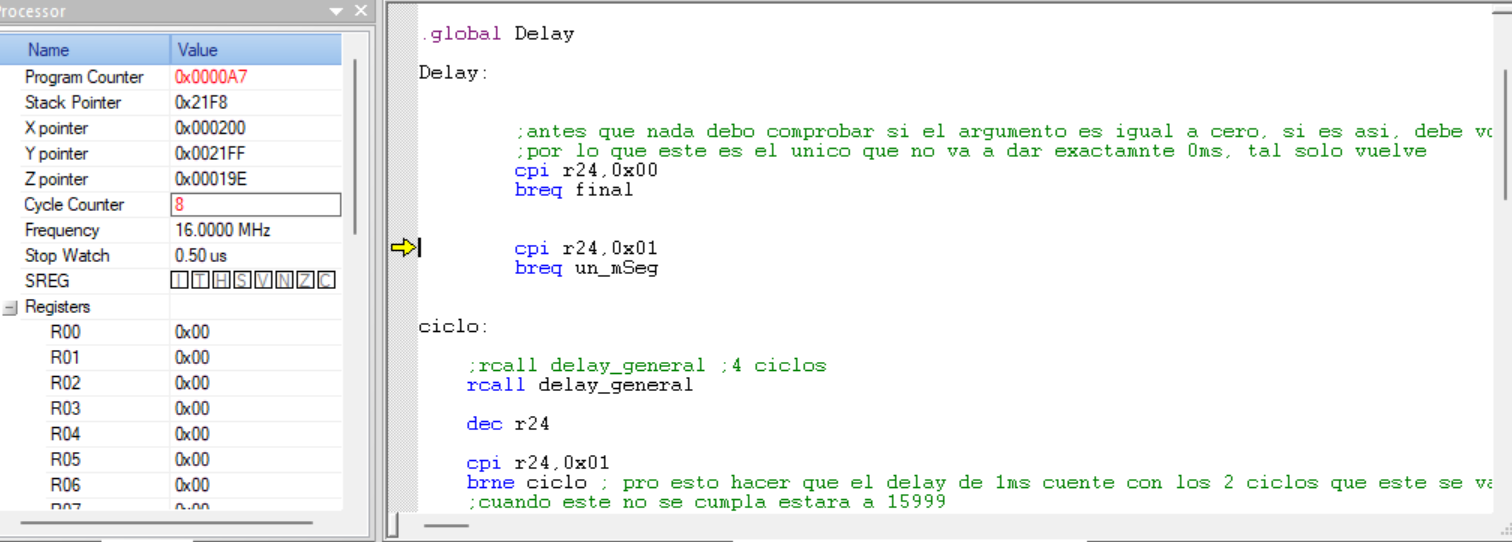


Se llama a la función que esta en ASM desde C lo que consume **6 tikcs.**

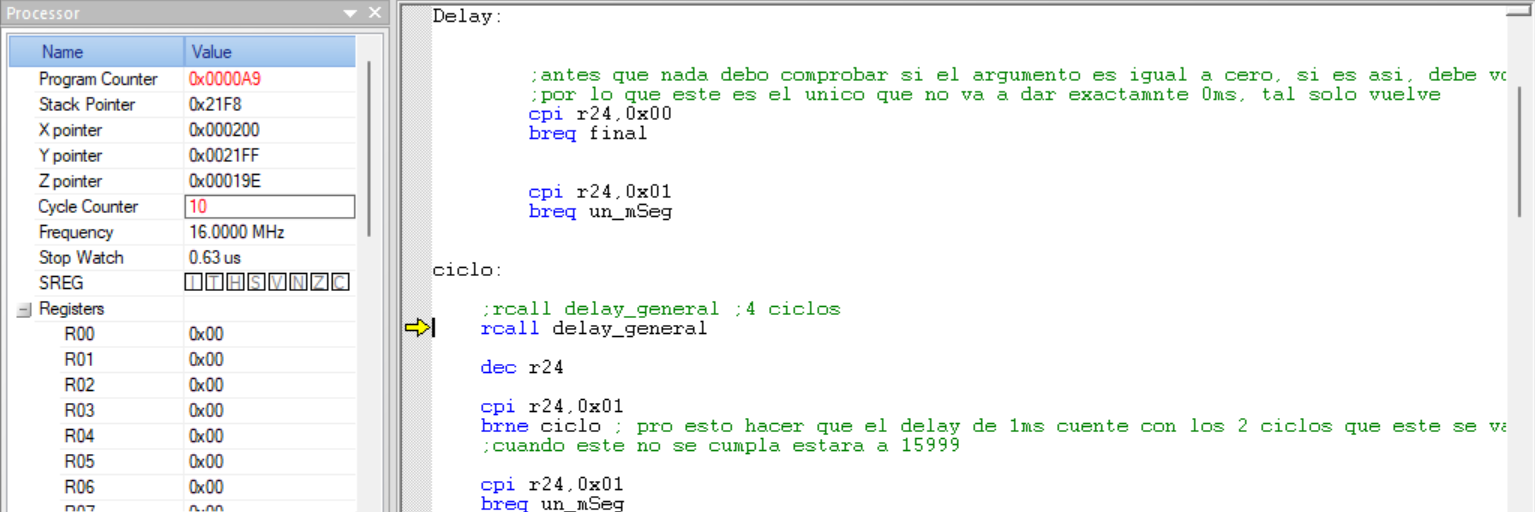


Ahora se tiene que comprobar si el dato que se mandó como parámetro es diferente a 0. Cunado esto suceda, el programa ira inmediatamente a ret a regresar a C, es la única vez cunado no se va a poder cumplir la condición de 0 mS de retardo, es imposible 0 ticks de ciclos de reloj. Tan solo el ir consumió 6 ticks

En este caso no es, por lo que aquí habrán **2 ticks** más, 1 de la comparación y 1 del Branch, dado que no se cumplió por lo que es 1.

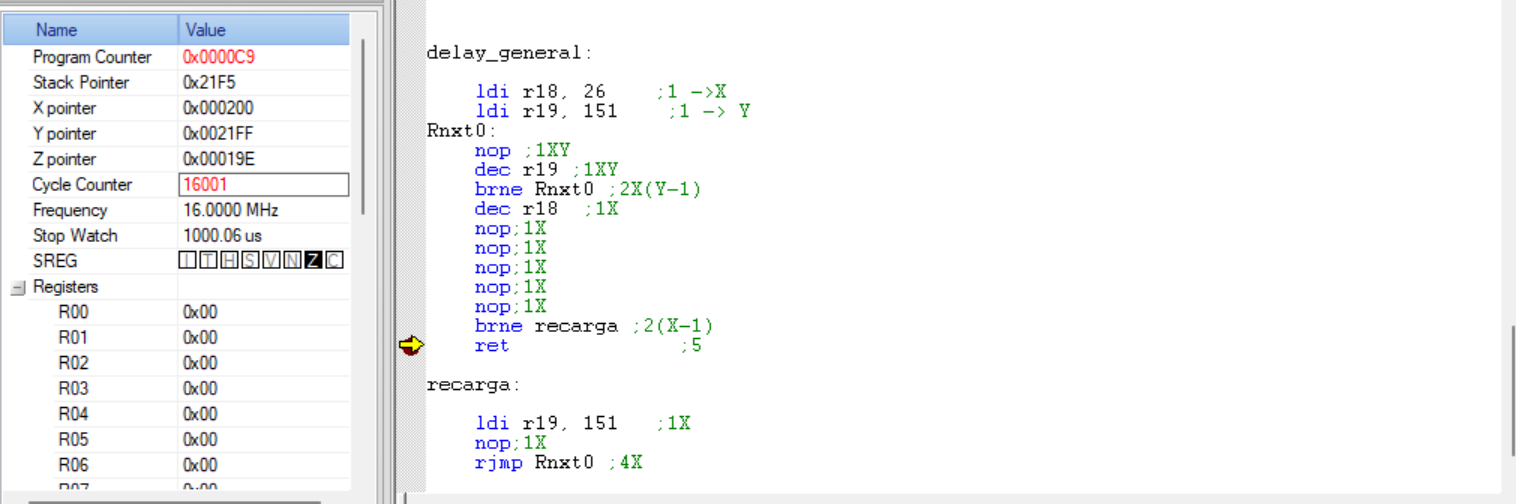


Ahora se compara si lo que se mando es 1mS, en caso de serlo llama a la subrutina. En nuestro caos no por lo que serán otros **2 Ticks** de reloj.

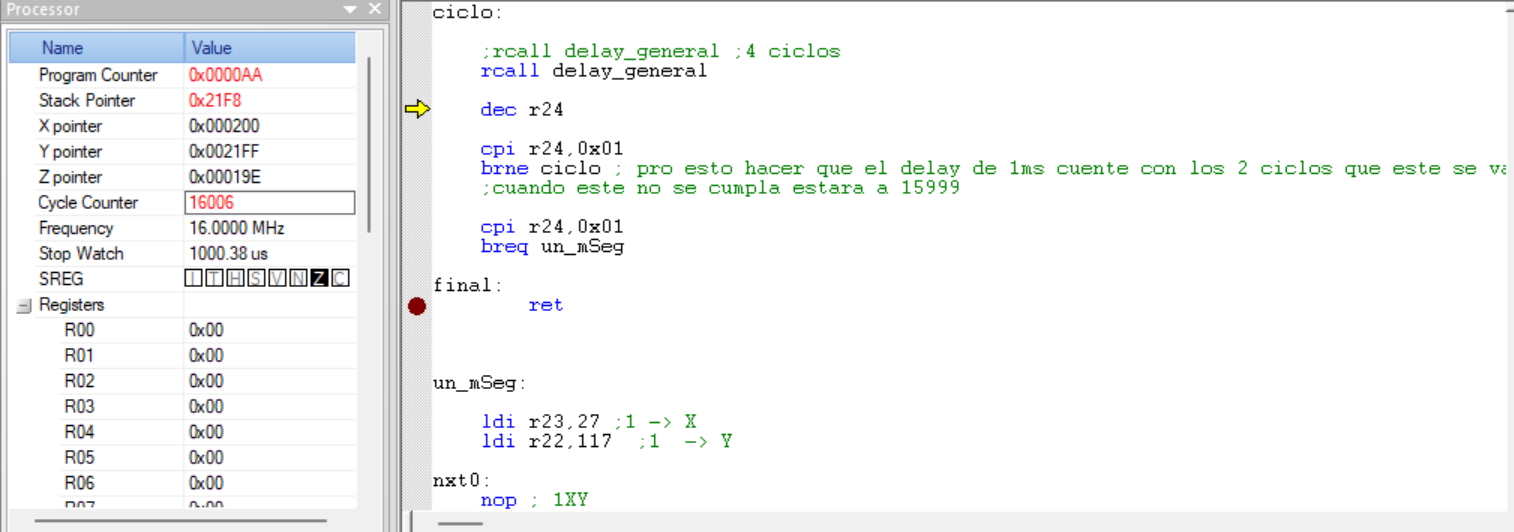


En este punto tenemos 10 ticks de reloj, por lo que siempre se se vuelva a llamar a ***delay\_general*** serán 14 tikcs más, con rcall con 4 ticks más.

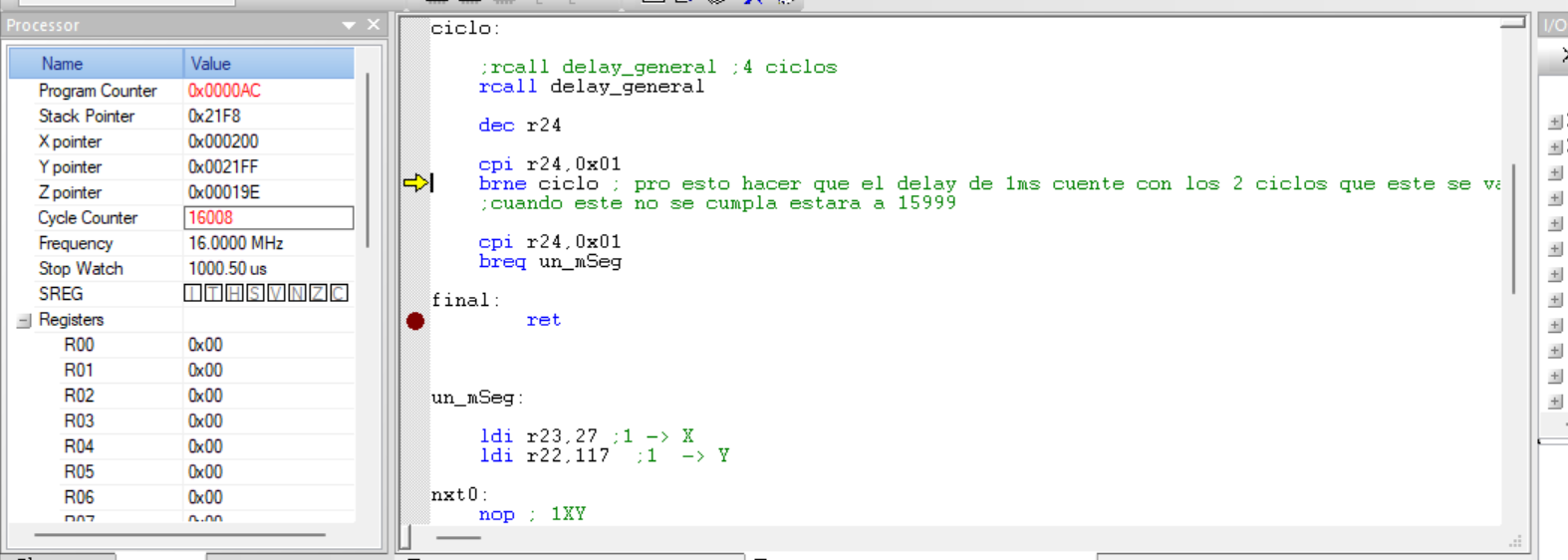
Por lo que se calculan 16000 ciclos, junto con ret, y las otras operaciones siempre y cunado r24 no sea igual 1.



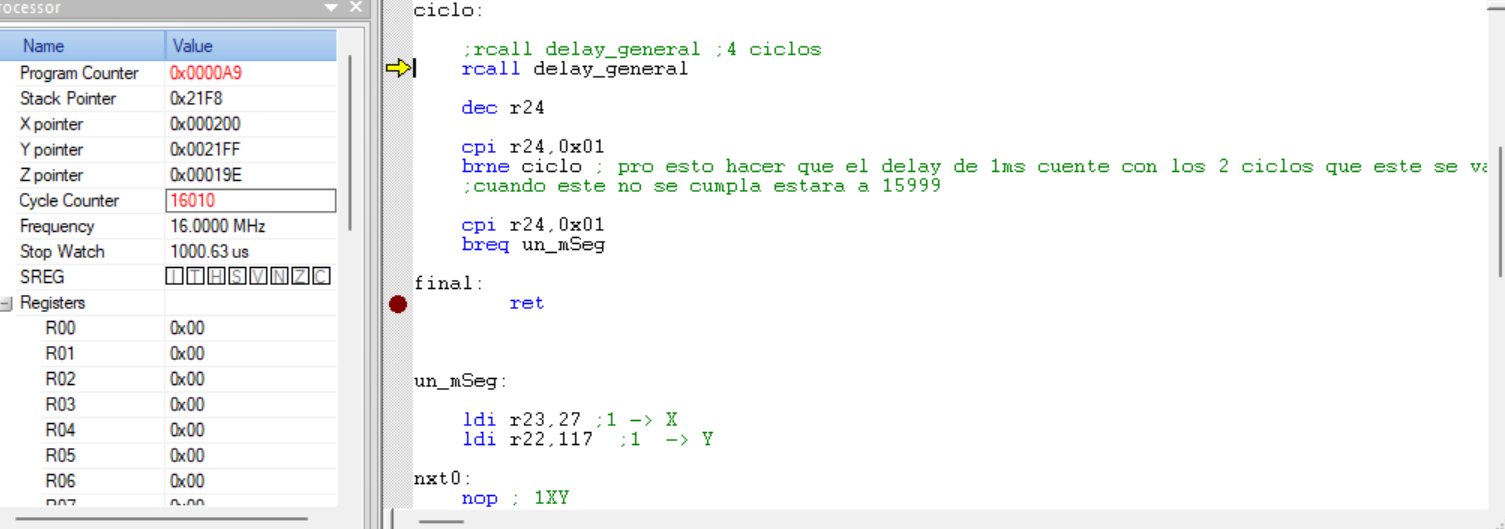
Decrementamos R24



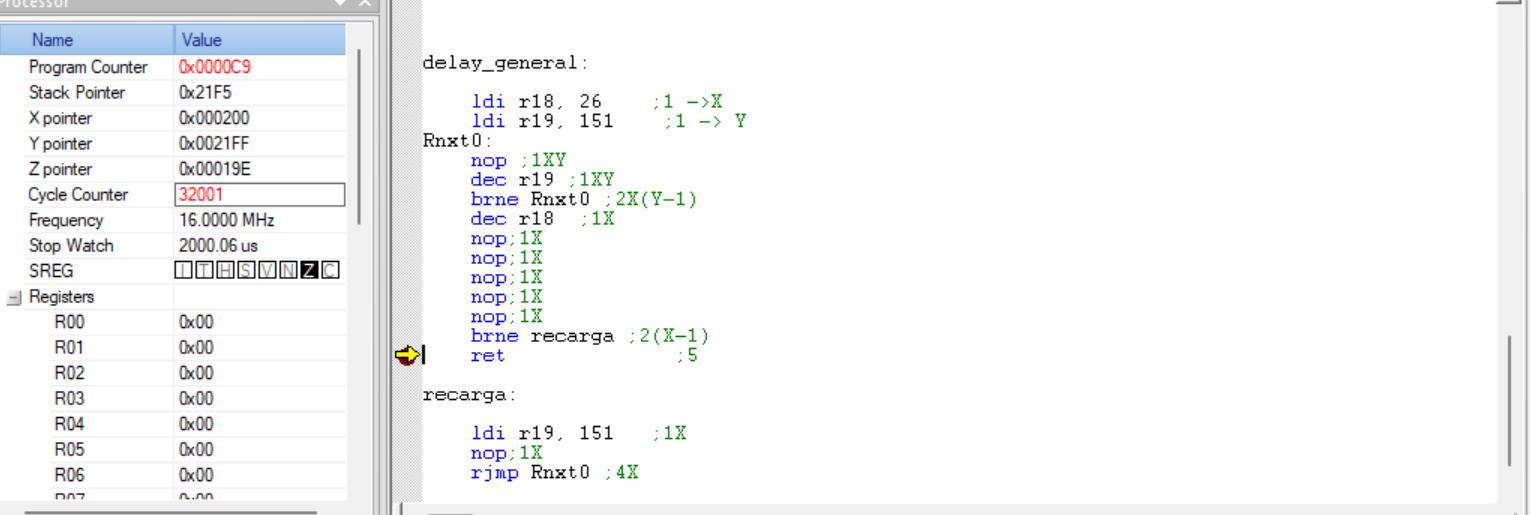
En caso de no ser 1 vuelve a *ciclo*



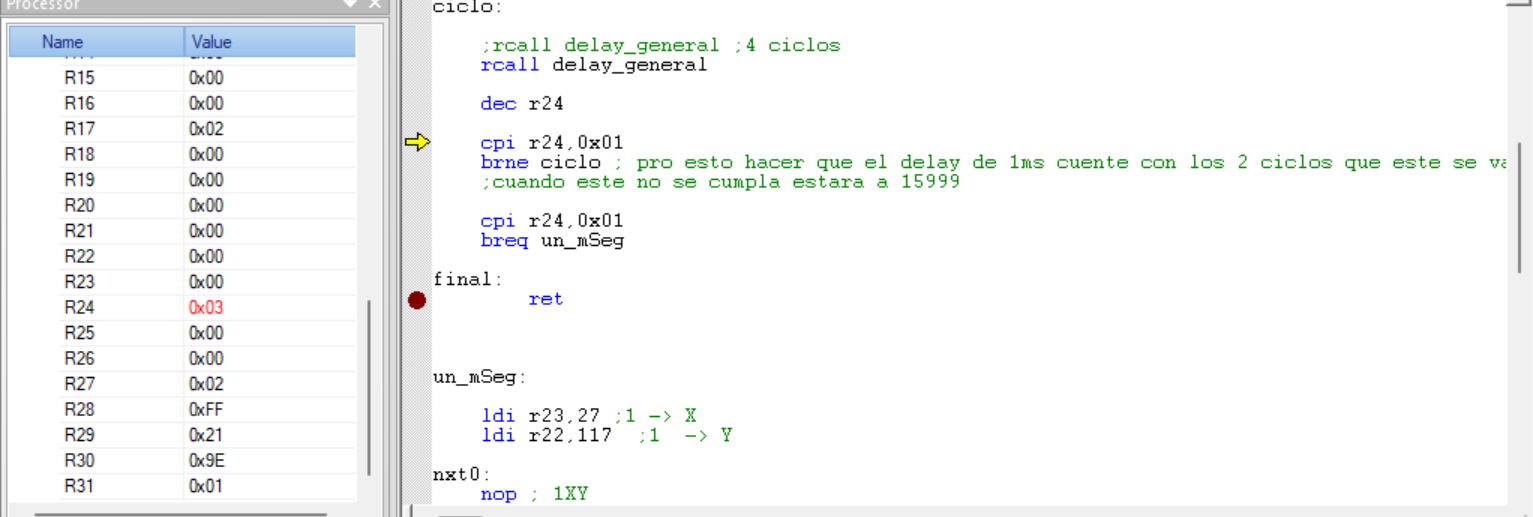
La operación es la misma hasta llegar cunado R24 = 1;



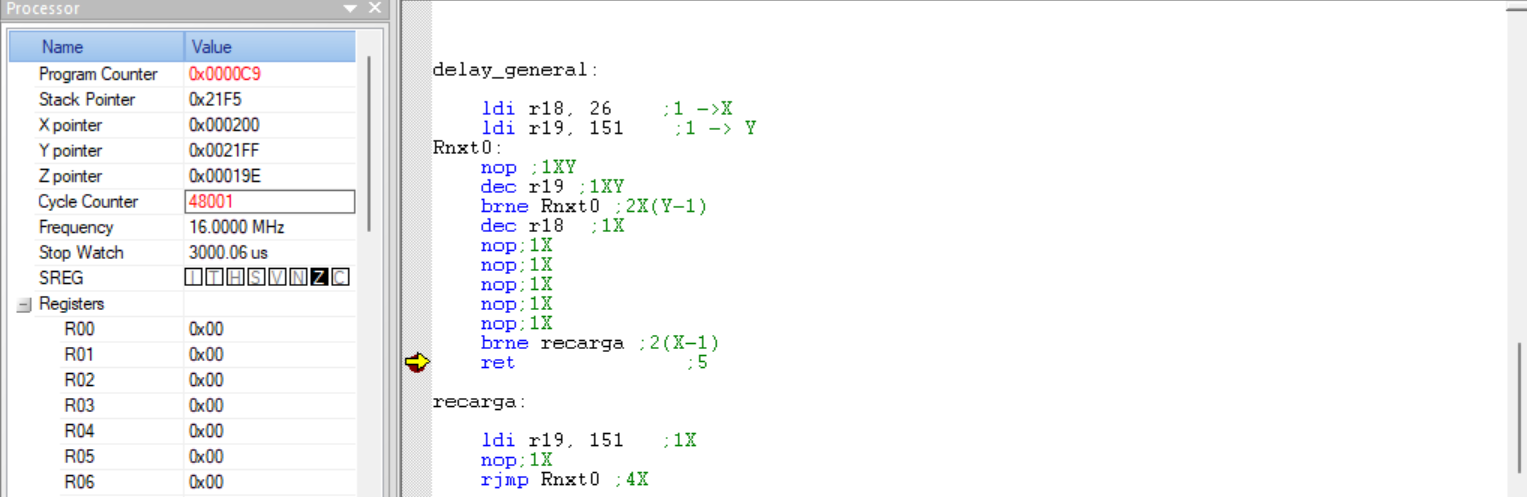
2mS= 32000 ticks



R24-=1;

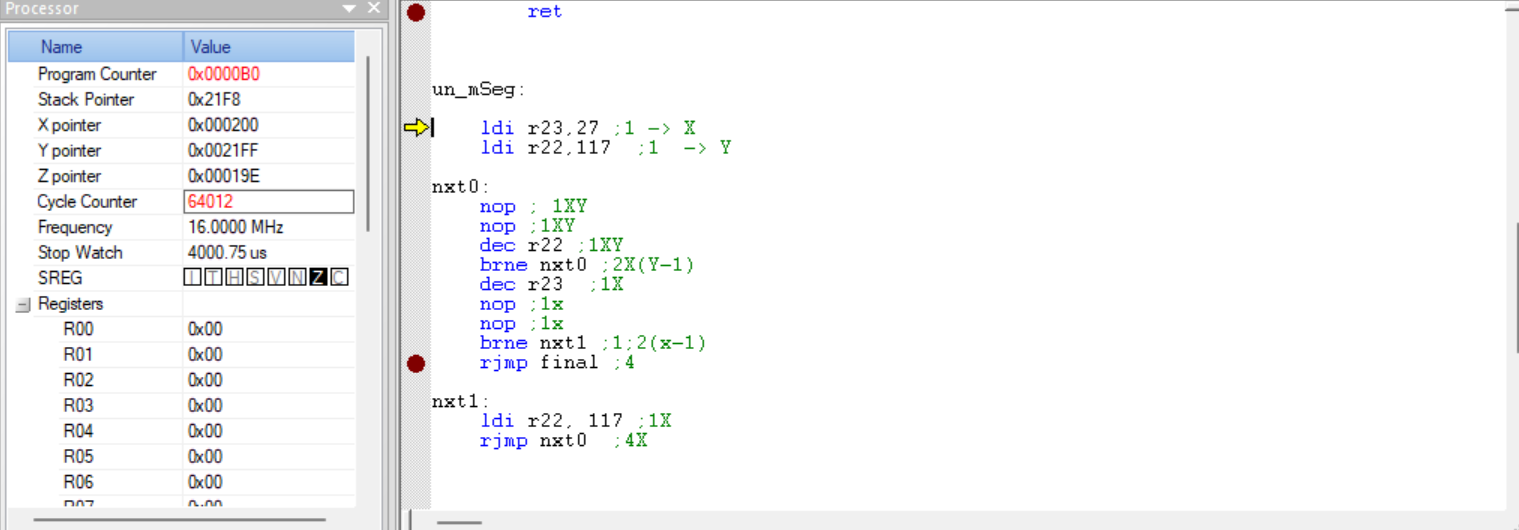


R24!=1?

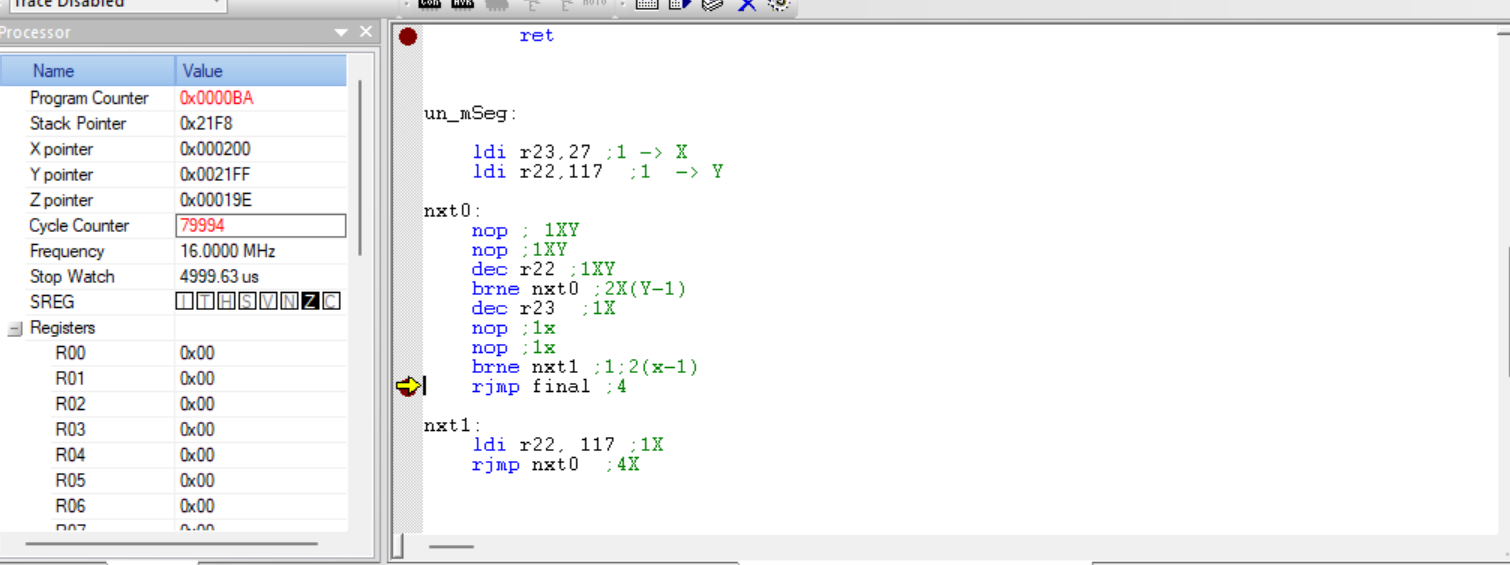


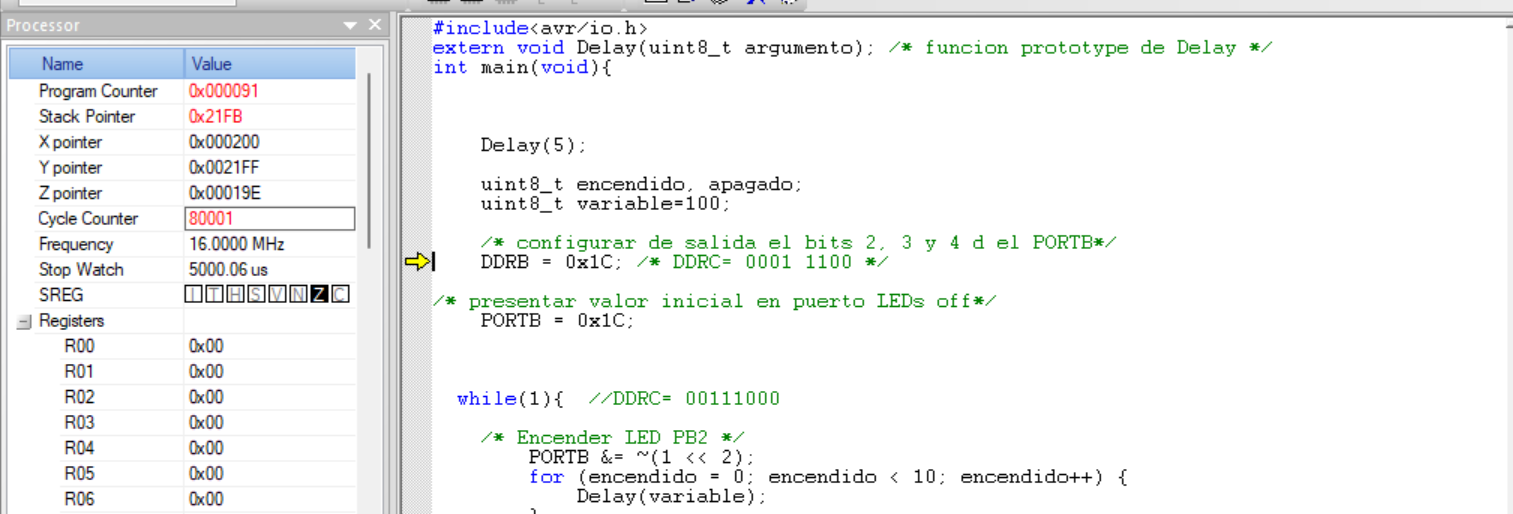
Así ira hasta que r24=1;

En este punto R24 en efecto es igual a 1. Por lo que ira el branch es aceptado e ira a *un\_mSeg*



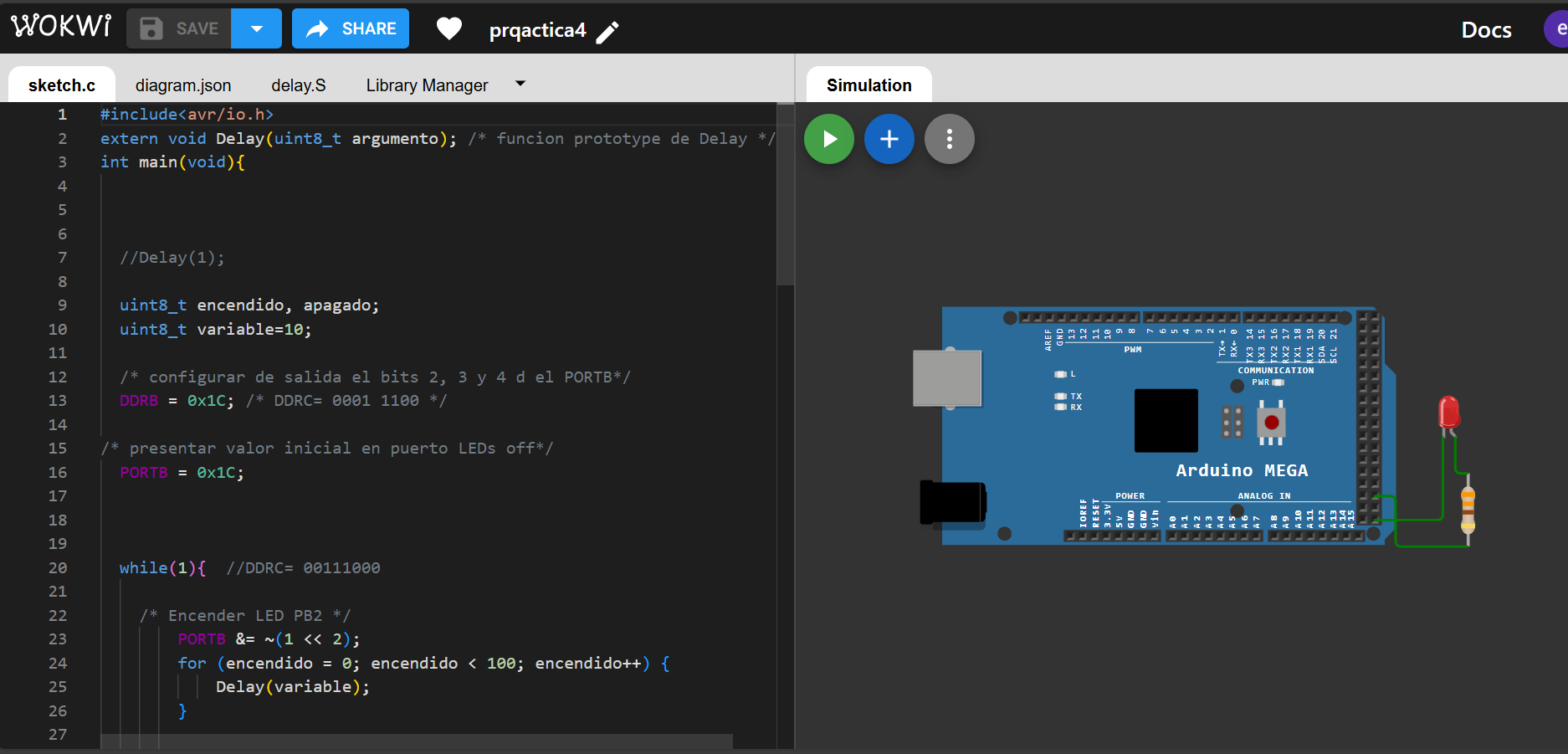
Son los ultimo 16000 ciclos, pero esta toma en cuenta todos los ticks se hicieron antes de estar llamando a las distintas etiquetas.





Programa cargado a Arduino Atmega 2560:

En este caso esta en un simulador:



Debería de prender y apagar entre 1 segundo, asi infiniticamente ya que se enceuntran dentro de una función while.

Video: https://www.youtube.com/watch?v=9czb5wVnDQw