

**INSTITUTO POLITÉCNICO NACIONAL**

**Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas**

|  |
| --- |
|  |
| **Arquitectura de Computadoras** |
|  |
|  |

**Programación avanzada**

19 de octubre de 2017

Alvarado Balbuena Jorge Anselmo

**Práctica B**

19 de octubre de 2017

Alvarado Balbuena Jorge Anselmo

Avalos Vizuett Julio Cesar

Rocha Díaz Brandon

**Objetivos**

* Complementar el análisis teórico visto en clase con herramientas de programación.
* Comprender el análisis y la creación de rutinas en el lenguaje ensamblador, para poder crear timers en nuestros programas.

**Introducción**

Las subrutinas (funciones o procedimientos) son abstracciones que se usan en los lenguajes de alto nivel para simplificar el código y poder reusarlo. A la subrutina se la llama (invoca) desde el programa que se invoca mediante una instrucción particular.

Pero antes de llamar a la subrutina tenemos que haber colocado los datos de entrada a ésta en un lugar accesible (registros reservados o pila). En general al trabajar con subrutinas se divide el trabajo a realizar entre el programa invocante y la subrutina:

* Programa invocante: Poner los parámetros o argumentos de entrada a la subrutina en un lugar donde sean accesibles por ésta. o Registros para pasar parámetros r0 – r3

Ej.: mov r0, r7

* Programa invocante: Transferir el control a la subrutina

Ej.: bl Etiqueta lr=pc+4 y pc=posición de Etiqueta

* Subrutina: Ejecutar la tarea deseada usando los recursos necesarios
* Subrutina: Poner el resultado en un lugar accesible al Programa. En el ARM la subrutina devuelve un único parámetro de salida con el resultado. En nuestro caso, vamos a trabajar siempre con números enteros con lo que el resultado siempre se devolverá únicamente en r0, pero si el resultado ocupase más de 4 bytes se usarían también los otros registros del r1 al r3.

Ej.: mov r0, r7

* Subrutina: Devolver el control al punto inicial, manteniendo el contexto

Ej.: mov pc, lr

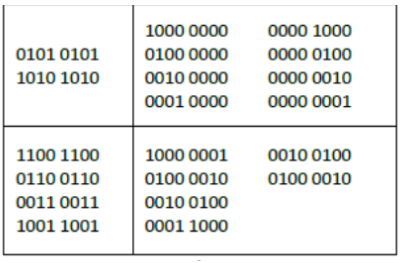
**Desarrollo**

**Ejercicio 1**

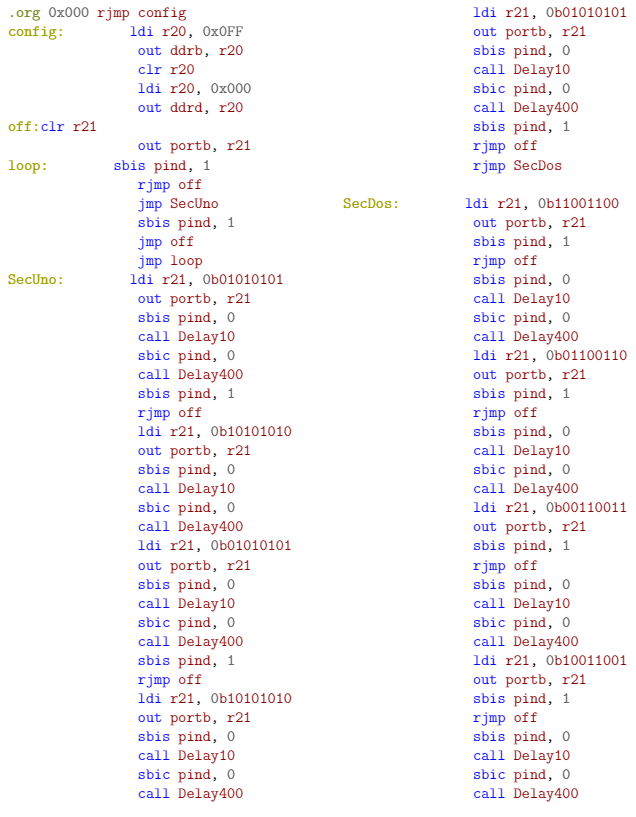
Realizar un programa ensamblador tenga programadas 4 secuencias de 8 leds y pueda tener

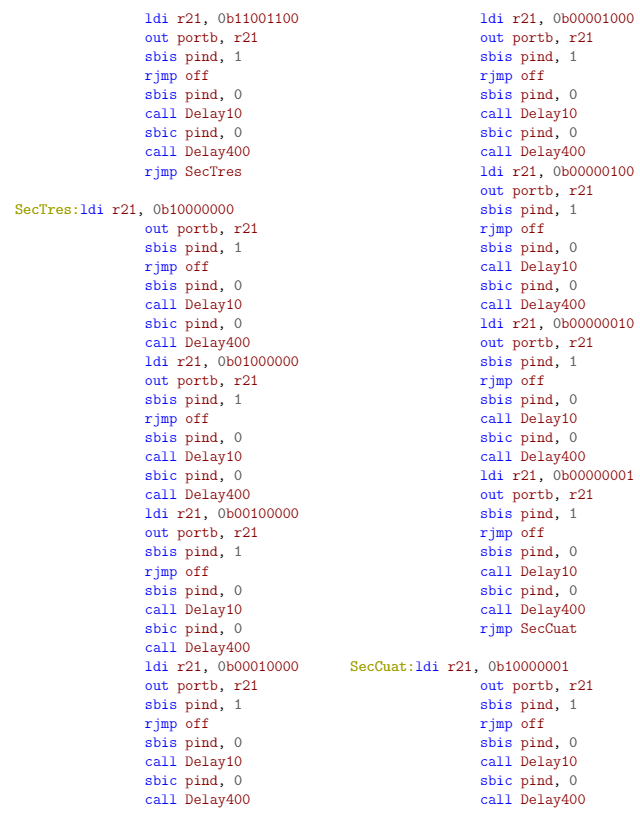
la opción de ponerlo a dos velocidades de cambio distintas (400 ms, y 100 ms), en el diseño

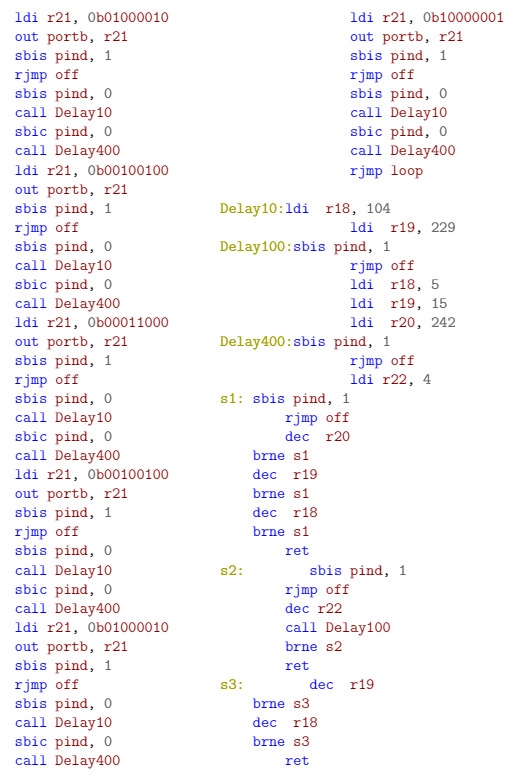
incluir un botón de on/off. Utilizar rutinas de tiempo y rutinas para cada secuencia.



**Código**





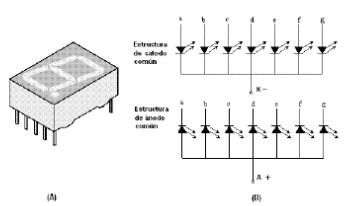


**Ejercicio 2**

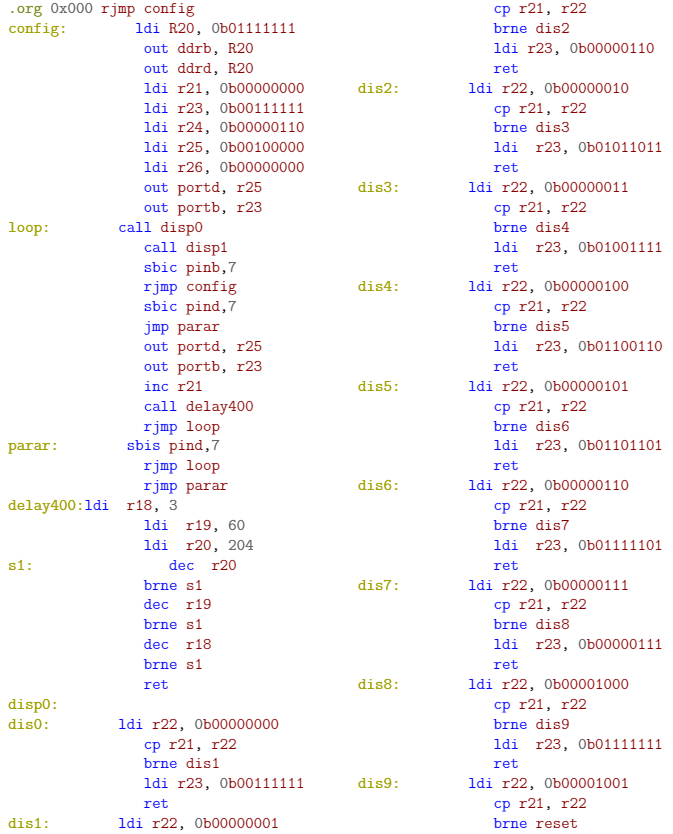
Elaborar un segunde que cuente de 0 a 59 y lo muestre en dos Display de 7 segmentos de

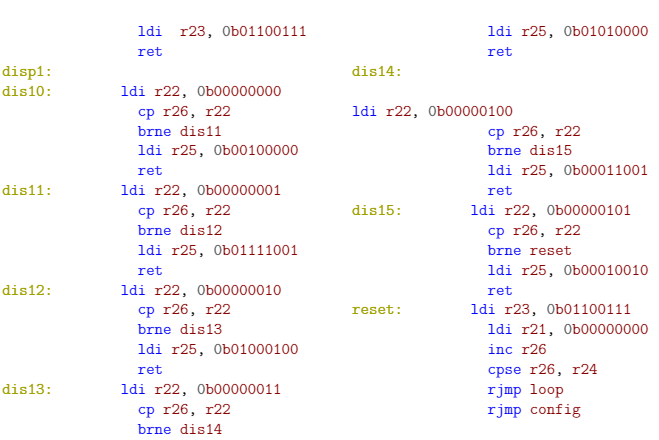
ánodo común el tiempo que va transcurriendo. Habilitar un botón de reset y un botón de

pausa.



**Código**





**Conclusiones**

**Alvarado Balbuena Jorge Anselmo**

Durante el desarrollo de esta práctica practicamos el concepto de rutinas. En el primer ejercicio para prender una serie de leds en secuencias diferentes, así como en velocidades, lo cual sirvió de practica para el segundo ejercicio. En este lo realizado fue un segundero, con lo cual vimos una aplicación más real de lo que es posible hacer con el lenguaje ensamblador.

**Julio Cesar Avalos Vizuett**

Una rutina es una parte fundamental en la programación a través del lenguaje de ensamblador porque nos permite crear retardo, crear un código que se puede usar más de una vez. Un retardo es hacer que un microcontrolador no ejecute nada para hacer que pase el tiempo ya que su tiempo de ejecución es de micro segundos, entonces con los retardos podemos crear tiempo para hacer contadores, relojes, etc.

**Rocha Díaz Brandon**

Las rutinas generadas en esta práctica sin duda me brindaron una visión más cercana a la aplicación en la vida real ya que pueden ser utilizadas para diversas situaciones, llevada a algo conocido sería un reloj, una serie navideña, cronómetro o secuencia de operaciones realizadas en ciertas velocidades y orden optimizando el código para un uso constante del mismo por medio del lenguaje ensamblador.