**TRABAJO PRÁCTICO GRUPAL**

**ARQUITECTURA DE COMPUTADORAS**

**2023**


**INTEGRANTES:**

Lautaro Lombardi DNI 43989019

Lola Torres DNI 44507859

Santiago Saitz DNI 43518748

Marco Pappalardo DNI 45131689


---

**Consigna**:

Desarrollar el código para los siguientes ejercicios y probarlos en la herramienta Proteus con los tres circuitos provistos por el docente (existe un circuito correspondiente para cada Ejercicio). La complejidad de los ejercicios aumenta progresivamente, pero están diseñados de manera que se pueda reutilizar el código de los puntos anteriores.

Ej. Para hacer el Ejercicio 1-b, se puede tomar lo hecho para el Ejercicio 1-a y solamente modificar unas líneas de código. Por este mismo motivo, solo se va a pedir que se entreguen los últimos puntos de cada ejercicio (1-e, 2-d, 3-c). Es recomendable que igual los hagan todos.

**Forma de Entrega:**

• La entrega puede hacerse de manera grupal. Si es así, en el espacio habilitado en el campus se debe hacer una sola entrega por grupo. • Debe entregarse el código con una captura de su simulación en Proteus en un .zip/.rar junto con un .pdf que explique brevemente la resolución de cada ejercicio.

• El documento pdf no debe tener código, para eso se los pido aparte. La idea es que cuenten como lo resolvieron, que registros tuvieron que usar, cálculos para valores de los registros, etc.

• El archivo comprimido debe tener los apellidos de todos los integrantes del grupo.

# Resolución

## Ejercicio 1-E:

**CÓMO LO RESOLVIMOS:**

Primero configuramos los pines desde el **RB0** hasta el **RB3** como salidas, limpiamos **PORTB**. Creamos una subrutina que se llama "**MAIN**" esta subrutina lo que hace es:

-Setear desde **PORTB,0** hasta **PORTB,3** con **un bit**, lo cual hace que se enciendan los leds conectados a esos pines

-Con una subrutina llamada **"delay_500mili"** generamos un delay de **500 milisegundos** entre cada seteo de los pines

-cuando los pines desde **RB0** hasta **RB3** están seteados en **1** (están los leds prendidos), con la instrucción **BCF** ponemos los bits desde RB3 hasta **RB0** en **0** con un intervalo de **500 milisegundos** entre ellos.

-La subrutina **"MAIN" e**s un loop infinito por lo que la secuencia se va a ejecutar indefinidamente

**REGISTROS UTILIZADOS:**

**-STATUS**

**-TRISB**

**-PORTB**

-También usamos **0x20, 0x21** y **0x22** como contadores para ayudarnos a generar los delay.

**CÁLCULOS PARA VALORES DE REGISTROS:**

Para generar **1 MILISEGUNDO**: hicimos un loop que consta de **3 instrucciones (NOP,DECFSZ Y GOTO)** entre las cuales suman **4 MICROSEGUNDOS**, con el uso de un contador en **250,** un loop con el **GOTO** y la instrucción **DECFSZ** hicimos **250** loops de **4 MICROSEGUNDOS** o **1000 MICROSEGUNDOS.**

Con la misma lógica que generamos **1 milisegundo**, hicimos un loop que contiene un **CALL** el cual **"llama" 250** veces a la subrutina para generar **1 MILISEGUNDO**, así generando un delay de **250 MILISEGUNDOS.**

Con las subrutinas anteriores creadas y siguiendo la misma lógica, generamos **500 MILISEGUNDOS** llamando **2** veces a "**delay de 250 MILISEGUNDOS**"

---

## Ejercicio 2-E:

**CÓMO LO RESOLVIMOS:**

Primero habilitamos las interrupciones por **RB0**, luego configuramos el pin **RB0** como entrada y luego **RB1** Y como salidas, limpiamos **PORTB**.

Después de eso movemos el literal **1** a la variable **UNIDAD** la cual usamos como contador. Cada vez que se ejecute una interrupción en **RB0** (se presione el botón) el programa se va a dirigir al vector de interrupciones. Cada vez que se ejecute una interrupción movemos lo que está en **UNIDAD** a **W** y se llama a una subrutina **LUCES** la cual, a través del contador de programa, realizará la secuencia correspondiente

**REGISTROS UTILIZADOS:**

**STATUS**

**TRISB**

**PORTB**

**INTCON**

**OPTION_REG**

**CONT1: 0X20, CONT2: 0X21, UNIDAD: 0X30**

**CÁLCULOS PARA VALORES DE REGISTROS:**

**CONT1** y **CONT2** tienen el literal **.250** para generar el delay de **1ms**

**UNIDAD** tiene el literal **.1** al principio para poder manipular el contador de programa

---

## Ejercicio 3-C:

CÓMO LO RESOLVIMOS:

-En la subrutina **INICIO** configuramos las interrupciones por **Timer0** y **RB0,** configuramos los **pull-ups** para así usar el pulsador, configuramos el **bit T0CS** en **0** para así usarlo como contador de eventos internos, configuramos psa en **0** para usar preescaler y seteamos los bits **PS0-PS1-PS2** en **1** para usar el **preescaler** en **256** y configuramos **RB0** como **entrada** y los demás **RB** como **salida**.

-Definimos una variable llamada **UNIDAD** que se va a utilizar como iterador del display.

-Definimos una variable llamada **CONT** que se usará como un contador.

-Definimos una variable llamada **CONT_TMR0** que se usa para calcular **1** segundo de demora con el **Timer0.**

-Creamos la tabla de display, que retorna los valores necesarios para mostrar en un display los números desde el **0 al 9** y las letras desde la **A hasta la F**. La tabla de display también tiene código para reiniciar el contador y la variable de iteración a su valor original.

-En la rutina de interrupción **copiamos su código** para generar 1 segundo de delay con **Timer0** porque no pudimos hacerlo solos.

-La forma en la que hacemos que muestren toda la tabla de display y después la muestre en orden descendente:

-Creamos una subrutina **MOSTRAR_DISPLAY** que mueve a **W** el valor del iterador y se lo pasa a la tabla de display y display retorna el valor para pasarlo a **PORTB.**

-La forma en la que planteamos la parte de mostrar la secuencia al revés está de hecha de manera que un contador al cual se le va a restar **1** cada vez que se muestre un valor en el display se mantenga en **0** hasta que muestre los **16** valores, cuando **DECFSZ** se saltea la primera instrucción return al contador se le pasa el valor **1** para que cuando salga de la interrupción y vuelva a entrar siga salteando la instrucción de retorno y ejecute lo que está debajo.

-Cuando entra a la interrupción al iterador siempre se le suma **1**, así que para pasarle al display los valores del **16** hasta el **0** le restamos **2** al iterador.

-Cuando el iterador llegue a **0** (osea que ya se mostró la secuencia al revés) se ejecuta un **GOTO** en el display llamado **REINICIAR** que reinicia el iterador y el contador a sus valores originales, para así mostrar la tabla de display en el orden ascendente otra vez.

-Cuando se toca el botón, el bit **T0CS** pasa a ser contador de eventos externos y así deteniendo la secuencia en el display.

-Cuando se toca otra vez el **botón,** el bit **T0CS** vuelve a su valor original y continúa contando eventos internos.

-El código para cambiar el valor de **T0CS** también **se lo copiamos a usted.**

**REGISTROS UTILIZADOS:**

**-INTCON**

**-STATUS**

**-OPTION_REG**

**-TRISB**

**-PORTB**

**-TMR0**

**-0X30=iterador**

**-0X31=contador TIMER0**

**-0X27=contador**

**-PCL**

**CÁLCULOS PARA VALORES DE REGISTROS:**

**UNIDAD:** Vale **.7** porque antes de el retorno para mostrar un **0** en el display hay código.

**CONT:** Vale **.17** porque son los valores del **0** hasta la **F + 1** para la instrucción **DECFSZ.**