**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**(ESPOL)**



**FACULTAD DE INGENIERIA EN**

**ELECTRICIDAD Y COMPUTACIÓN**

**(FIEC)**

**LABORATORIO DE**

**MICROCONTROLADORES**

**Práctica 1:**

Encendido de LEDS

**Profesor:**

Ing. Carlos Valdivieso

**Integrantes:**

Burbano Moreira William Paúl

Mendoza Narea Jorge Luis

**Paralelo:**

8

**Grupo:**

#3

**Fecha de presentación:**

01/06/2011

**2011 – I TÉRMINO**

1. **ENUNCIADO DEL PROYECTO**

Para el **Encendido de LEDS** se utilizó un programa sencillo que permite encender LEDS en el Puerto B mediante retardos de tiempo siguiendo las instrucciones.

Realizamos el siguiente procedimiento:

* Creamos un proyecto nuevo con el MPLAB.
* Ejecutamos paso a paso.
* Observamos varios registros.
* Medimos el tiempo de retardo a diferentes frecuencias de reloj.
* Simulamos en el software PROTEUS.

1. **DIAGRAMA DE BLOQUES.**

**ENTRADA**

**MICROCONTROLADOR**

**SALIDAS**

**PIC16F887**

1. **Diagrama de flujo funcional del programa principal (nivel alto).**



1. **Diagrama de flujo de cada una de las subrutinas empleadas.**
2. **Descripción del algoritmo o estrategia utilizado.**

Comenzamos el programa con el vector de reset, luego seteamos los puertos, seleccionando primero el banco contenido en TRISB, luego configurando el puerto B como salida, con ANSEL y ANSELH configuramos los puertos como entradas digitales, luego seleccionamos el puerto B como salida, encero nuevamente los puertos, para luego entrar a un lazo que me permite probar si el bit del puerto A es cero, si es verdad esto me voy al nuevo valor donde se moverá 0x55 al registro W para después ese mismo valor se pasa al puerto B. Cuando el bit del puerto A no es cero el NUEVO VALOR se moverá a 0xAA al registro W, para pasar el valor al puerto B.

1. **Listado del programa fuente en lenguaje ensamblador con comentarios en las líneas de código que considere fundamentales.**

;PROGRAMA A1

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; LECTURA DE SWITCH EN PORTA Y ENCENDIDO DE LEDS EN PORTB

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; NOMBRE: p1\_led.asm

; FECHA: 26/05/2010

; VERSION: 1.00

; PROGRAMADOR: Carlos Valdivieso

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; DESCRIPCION:

; Se coloca un interruptor en RA0. Cuando RA0=1 en el Puerto B

; deberá leerse el valor hexadecimal AA y cuando RA0=0 deberá

; cambiar al valor hexadecimal 55.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

LIST p=16F887 ; Tipo de microcontrolador

INCLUDE P16F887.INC ; Define los SFRs y bits del ; P16F887

\_\_CONFIG \_CONFIG1, \_CP\_OFF&\_WDT\_OFF&\_XT\_OSC

; Ingresa parámetros de

; Configuración

errorlevel -302 ; Deshabilita mensajes de

; Advertencia por cambio

; Bancos

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; INICIO DEL PROGRAMA

ORG 0x00 ; Comienzo del programa (Vector de Reset)

; SETEO DE PUERTOS

BANKSEL TRISB ; selecciona el banco conteniendo TRISB

CLRF TRISB ; puerto B configurado como salida

BANKSEL ANSEL

CLRF ANSEL ; configura puertos con entradas digitales

CLRF ANSELH ; configura puertos con entradas digitales

BANKSEL PORTB ; selecciona el puerto B como salida

CLRF PORTB

CLRF PORTA

; DESARROLLO DEL PROGRAMA

LOOP

BTFSS PORTA,0 ; prueba del bit 0 del puerto A

GOTO NUEVO\_VALOR

MOVLW B'10101010' ; mueve 0xAA al registro W

MOVWF PORTB ; pasa el valor al puerto B

GOTO LOOP

NUEVO\_VALOR

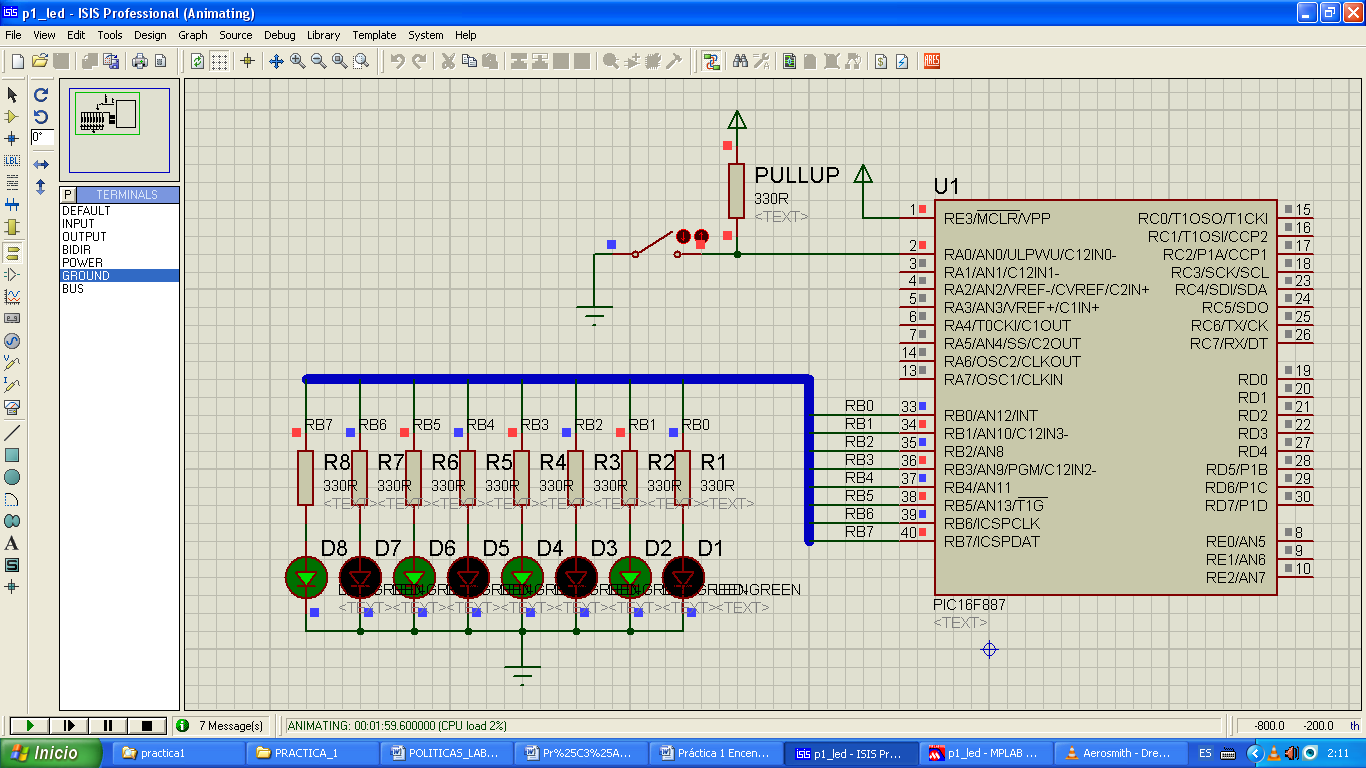
MOVLW B'01010101' ; mueve 0x55 al registro W

MOVWF PORTB ; pasa el valor al puerto B

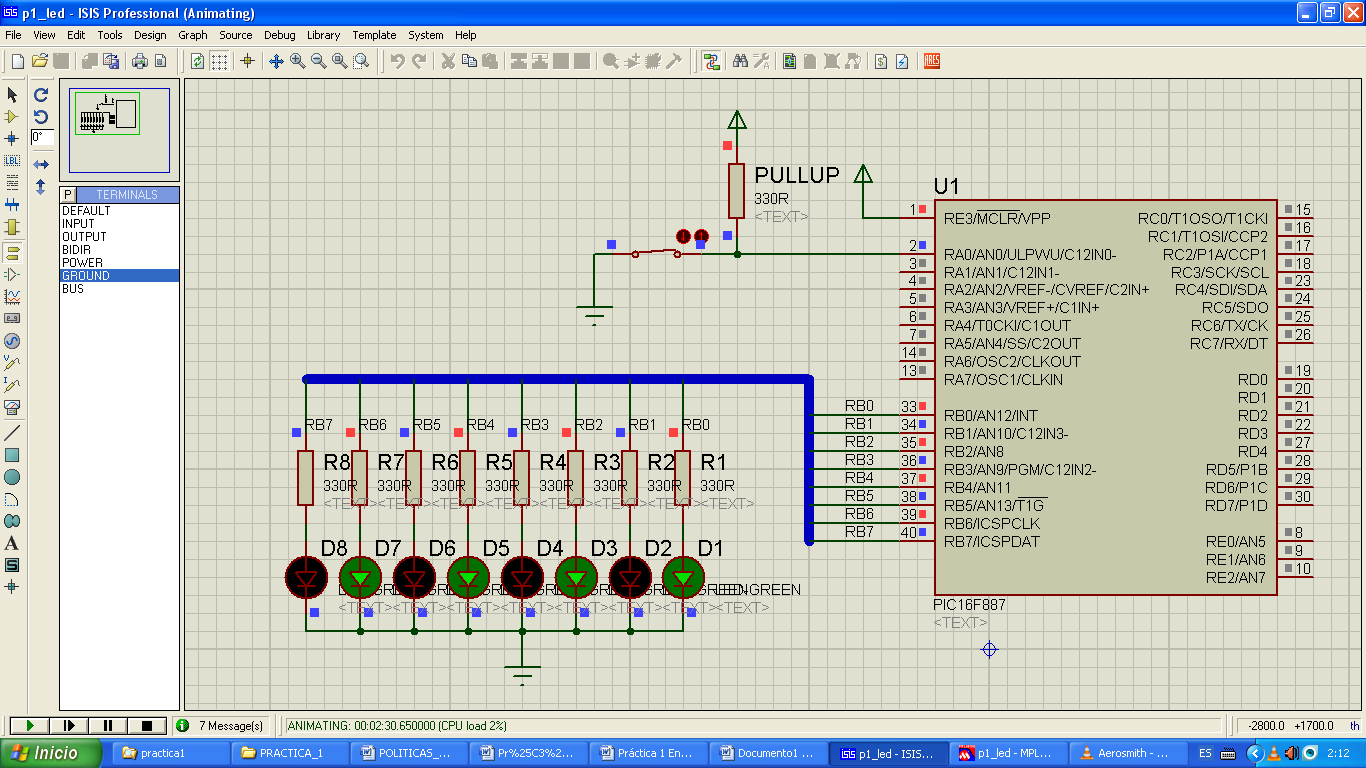
GOTO LOOP

END ; fin del programa

1. **Copia impresa del circuito armado en PROTEUS para la simulación en el momento de su ejecución (Por ejemplo con LEDS encendidos).**

**Gráfica #1.** *Ejecución con Switch Abierto.*

En ésta gráfica se puede apreciar que se enciende los led D8, D6, D4, D2 que posee una entrada en alto debido a que el switch se abrió, generando según el lenguaje ensamblador una salida “10101010”.

**Gráfica #2.** *Ejecución con Switch Cerrado.*

En ésta gráfica se puede apreciar que se enciende los led D7,D5,D3,D1, ya q el switch se cerró generando a la vez una entrada en bajo, q según el lenguaje ensamblador su salida debe estar dada por “01010101”

1. **CONCLUSIONES.**

* En el programa MPLAB pude observar que, me permite editar el archivo fuente en lenguaje ensamblador de nuestro proyecto, además de ensamblarlo y simularlo en pantalla, puede ejecutarlo posteriormente en modo paso a paso y ver como evolucionaría de forma real tanto sus registros internos, la memoria RAM y/o EEPROM de usuario como la memoria de programa, según se fueran ejecutando las instrucciones.
* Tener en cuenta cuando ya se valla polarizar el PIC, los registros TRISA, B, C, D por defecto quedan cargados con 1 y el reset en 1.
* MPLAB me permite visualizar como se va ejecutando el programa instrucción por instrucción y ver las direcciones en memoria todo esto gracias a **disassembly listing.**

1. **RECOMENDACIONES.**

* Para que no salgan esos molestosos mensajes de advertencia en el compilador del programa en el código fuente antes del inicio del programa se debe escribir esta línea de código:

**errorlevel -302 ;**Esta instrucción me deshabilita los mensajes de advertencia.

* A medida que hacen las prácticas en diferentes proyectos, guardar los archivos de manera ordenada en una carpeta que contenga el nombre del proyecto, dentro de la misma que deberá también contener el archivo de la simulación en Proteus.

**ANEXO DE RESPUESTAS (PRÁCTICA 1)**

**INTEGRANTES:**

* Burbano Moreira William Paúl
* Mendoza Narea Jorge Luis

**PARALELO:** 8

**GRUPO #3**

***Indique la función que desempeñan cada uno de los siguientes íconos:***

* **Run.-** Al activarse esta opción el programa se ejecuta a toda velocidad en "*modo tiempo real*", aunque no hay que olvidar que se trata de una simulación por ordenador. La ejecución comienza en la dirección actual del contador de programa (PC), que se muestra en la barra de estado y en la ventana Program Memory.
* **Halt.-**Detiene la ejecución del programa, el contador de programa se detiene y se actualiza la información.
* **Animate.-** Esta opción es similar a la función de RUN pero en este caso el programa se simula de forma automática desde la dirección que indique el PC en el instante en que se active. Equivale a pulsar continuamente Step Into. Se detiene con Halt.
* **Step Into.-** Esta acción ejecuta la instrucción cuya dirección de memoria coincida con el valor al que apunta el PC antes de activarla. Si algún registro se modifica se resalta en color rojo. Como puede apreciarse cuando se utiliza esta acción, podemos comprobar paso a paso como se ejecuta el programa, y de esta forma, ver si hay fallos y ver dónde está el error para depurarlo. También es especialmente interesante cuando se comienza a estudiar el ensamblador y queremos comprobar el repertorio de instrucciones del mismo. En definitiva, ejecuta la instrucción actual y se detiene, actualizándose la información. Si se realiza un **Step Into** sobre una instrucción **SLEEP**, como en la realidad, el dispositivo pasa a modo sleep, siendo necesario despertarlo para continuar, por ejemplo mediante un reset.
* **Step Over.-** Ejecuta la instrucción actual y se detiene, actualizándose la información. Aparentemente funciona como **Step Into** pero si la instrucción a ejecutar es **CALL**, ejecuta la subrutina llamada y se detiene en la instrucción siguiente a la llamada. Muy útil para ejecutar subrutinas ya comprobadas y seguir con la depuración del programa.
* **Step Out.-** Si la localización actual está dentro de una subrutina, la termina y retorna al programa principal.
* **Reset.-** Esta opción inicializa el sistema. El Contador de Programa (PC), que es la dirección de memoria donde se encuentra la primera instrucción que ejecuta el microcontrolador cuando se realiza un reset en el sistema, se pone a 0. A esta dirección de memoria se le denomina vector de reset.

