CENTRO DE CIENCIAS BÁSICAS

DEPARTAMENTO DE SISTEMAS ELECTRÓNICOS

ORGANIZACIÓN COMPUTACIONAL

4° "A"

PRÁCTICA 1

M. en CC. Juan Pedro Cisneros Santoyo

Alumno: Joel Alejandro Espinoza Sánchez

Fecha de Entrega: Aguascalientes, Ags.**, 5** de mayo de 2020

Práctica 1

Objetivo

La manipulación básica del microcontrolador 8051 (en cualquiera de sus variantes). Utilizar el software Keil µVision para realizar el código en lenguaje ensamblador. Realizar el código para el retardo requerido.

Antecedentes

Un microcontrolador es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria realizadas por lenguaje ensamblador. “Está compuesto de varios bloques funcionales que cumplen una tarea dada. Un microcontrolador incluye las tres principales unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada/salida” (Pahrami, 2005).

“Algunos microcontroladores pueden utilizar palabras de cuatro bits y funcionan a velocidad de reloj con frecuencias tan bajas como 4 kHz, con un consumo de baja potencia. Por lo general, es capaz de mantenerse a la espera de un evento como pulsar un botón o de otra interrupción; así, el consumo de energía durante el estado de reposo (reloj de la CPU y los periféricos de la mayoría) es mínimo” (Wackerly, 2008), lo que hace que muchos de ellos sean muy adecuados para aplicaciones con batería de larga duración. Otros microcontroladores pueden servir para roles de rendimiento crítico, donde sea necesario actuar más como un procesador digital de señal (DSP), con velocidades de reloj y consumo de energía más altos.

Cuando es fabricado el microcontrolador, no contiene datos en la memoria ROM. Para que pueda controlar algún proceso es “necesario generar o crear y luego grabar en la EEPROM o equivalente del microcontrolador algún programa, el cual puede ser escrito en lenguaje ensamblador u otro lenguaje para microcontroladores; sin embargo, para que el programa pueda ser grabado en la memoria del microcontrolador, debe ser codificado en sistema numérico hexadecimal que es finalmente el sistema que hace trabajar al microcontrolador cuando este es alimentado con el voltaje adecuado y asociado a dispositivos analógicos y discretos para su funcionamiento” (Wikipedia, 2007).

Pregunta de Investigación

¿Cómo se puede configurar el microcontrolador 8051 con instrucciones que causen un retraso (delay) específico?

Predicción

Puede buscarse una combinación de instrucciones MOV y NOP que pueda causar un número necesario de ciclos máquina para obtener el efecto de retraso requerido.

Materiales

Una computadora con ciertas especificaciones.

El software Keil µVision.
El software Proteus 8.8.

Método (Variables)

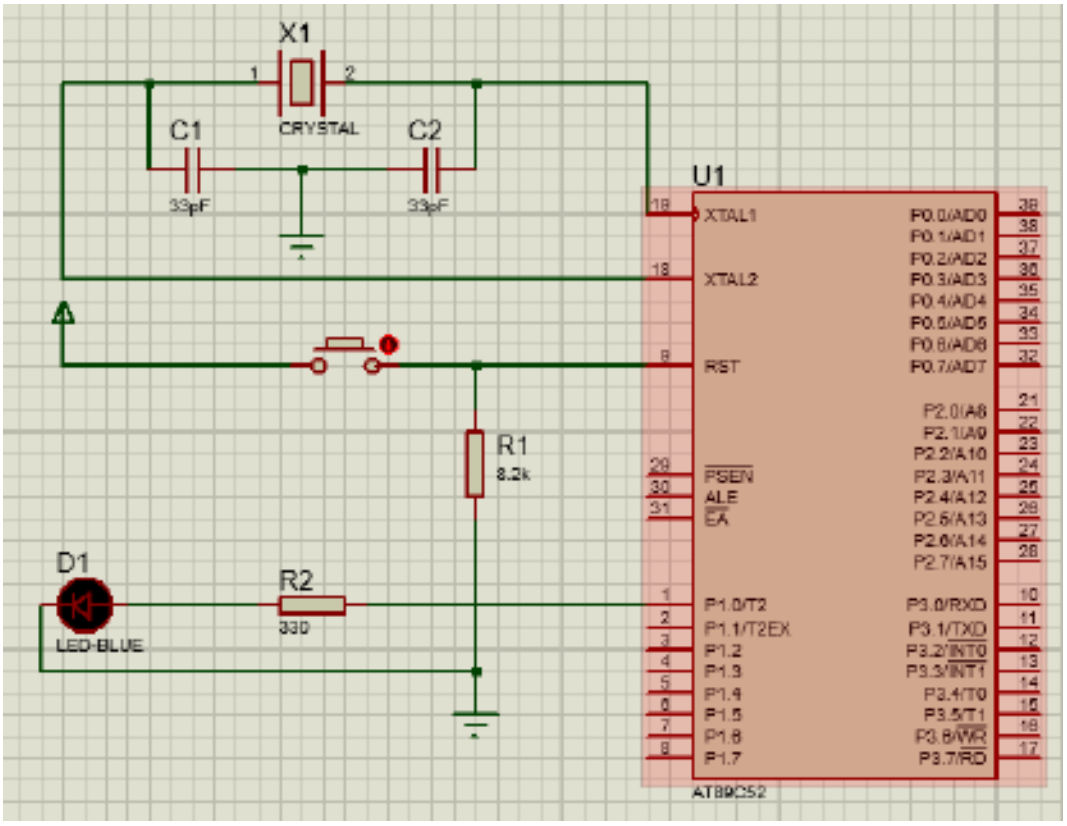
Dependiente: El retraso requerido de la reacción del circuito.
Independiente: Las instrucciones en lenguaje ensamblador del circuito.
Controlada: El circuito elaborado.

Seguridad

No existen riesgos físicos en la elaboración de esta práctica (a excepción de las medidas de precaución con el uso de una computadora).

Procedimiento

1.- Se realizó en el software Proteus 8.8 el siguiente circuito:

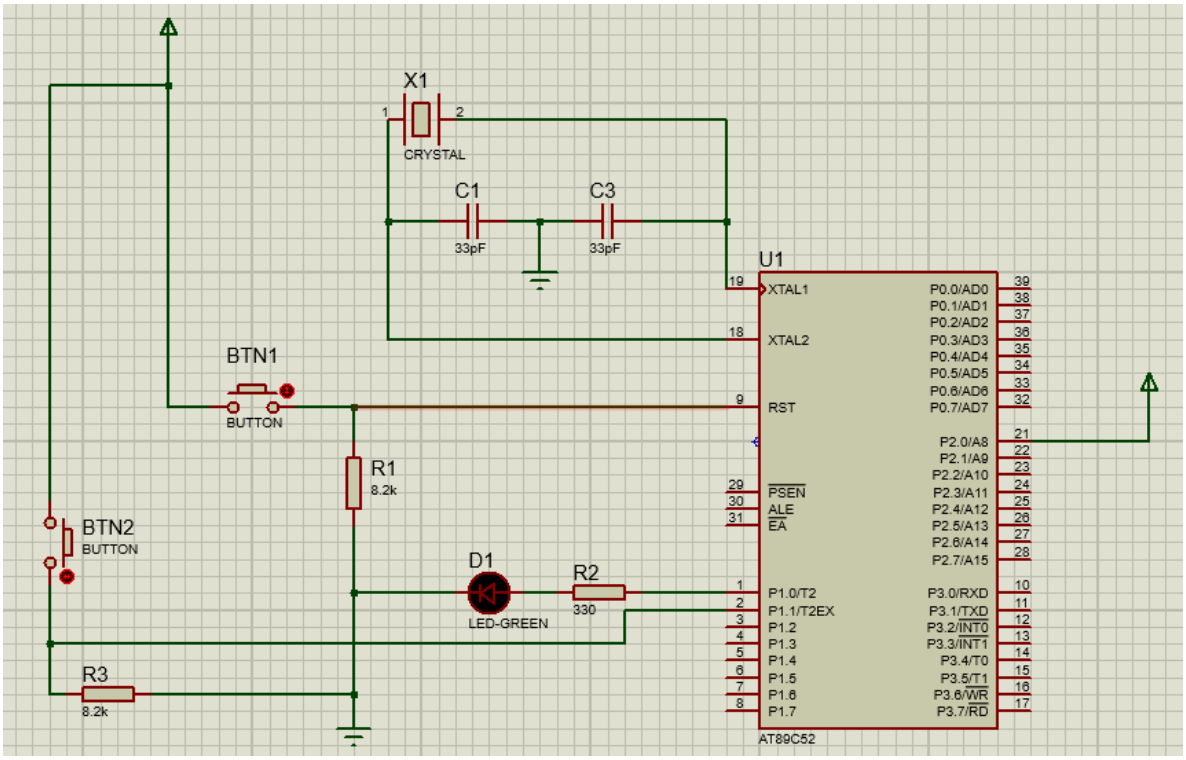


2.- Se escribió un programa en lenguaje ensamblador en Keil µVision para encender y apagar el led del circuito con ciertas modificaciones para que exista un retardo de un segundo como tiempo de encendido y apagado.

3.- Se cargó el programa de lenguaje ensamblador dentro del microcontrolador y se probó el circuito y el programa.

Obtención y Procesamiento de Datos

Primeramente, se realizó el circuito pedido en el software Proteus, quedando de la siguiente manera:



Después se escribió el programa en Keil, el cual en un principio fue el siguiente:

Practica1.asm*

```
1  ORG 0000H
2      inicio:
3      MOV R2,#00H ;Limpiar el puerto
4      MOV P1,R2
5      ciclo:
6      JB P1.1,enciende_led
7      CLR P1.1
8      SJMP ciclo
9      enciende_led:
10     SETB P1.0
11     SJMP enciende_led
12
13 END
```

Y con esta base se conectó con el microcontrolador del programa de Proteus. Seguido a ello se le realizaron modificaciones al programa para conseguir el retraso esperado, el cual era de un segundo. Pues se tenía conocimiento del siguiente programa ensamblador:

```
ORG 0000H
        inicio:
        ACALL delay

        delay: ;Retardo de 1.5 segundos                          ;2 ciclos                    = 2
        MOV R6,#0FAH                                             ;1 ciclo                     = 1
        d1:
        MOV R7,#0F9H                                             ;1 ciclo       1*250         = 250
        NOP                                                      ;1 ciclo       1*250         = 250
        NOP                                                      ;1 ciclo       1*250         = 250
        NOP                                                      ;1 ciclo       1*250         = 250
        NOP                                                      ;1 ciclo       1*250         = 250
        NOP                                                      ;1 ciclo       1*250         = 250
        d2:
        NOP                                                      ;1 ciclo       1*250         = 250
        NOP                                                      ;1 ciclo       1*250         = 250
        NOP                                                      ;1 ciclo       1*250         = 250
        NOP                                                      ;1 ciclo       1*250         = 250
        NOP                                                      ;1 ciclo       1*250         = 250
        NOP                                                      ;1 ciclo       1*250         = 250
        DJNZ R7,d2 ;Decrement Jump if Not Zero                   ;2 ciclos      2*249*250     = 124500
        DJNZ R6,d1                                               ;2 ciclos      2*250         = 500
        RET                                                      ;2 ciclos                    = 2
                                                                 ;total                       = 500,005

END
```

Con el cual se conocía un programa en el que se podía realizar un retraso de 500,005 milisegundos. Fue entonces que tomando como base el primer código mostrado en Keil, se agregó el código anterior con el segmento de código que contenía el apartado “`delay`”. Podemos observar en la imagen anterior que este segmento crea un retraso de acción de 500,005 milisegundos, muy cercano a medio segundo, por lo tanto se agregaron dos segmentos “`delay`” a los segmentos de “`enciende_led`” y “`apaga_led`” que se harían para el encendido del led durante un segundo y el apagado del mismo otro segundo respectivamente (véase anexo 1).

Para la elaboración del segmento “`enciende_led`” se usaron las instrucciones:

```
SETB P1.0
ACALL delay
ACALL delay
SJMP apaga_led
```

Pues en orden de ejecución, primero se activaba el led, posteriormente se llamaba dos veces al segmento que nos producía medio segundo de retraso (consiguiendo un segundo total) y saltaba finalmente a la sección de apagado del led.

Para la elaboración del segmento “`apaga_led`” se usaron las instrucciones:

```
CLR P1.1
CLR P1.0
ACALL delay
ACALL delay
SJMP enciende_led
```

Igualmente, en un análisis siguiendo el orden de las instrucciones, primero se limpian los puertos 1.1 y 1.0 por seguridad, después se procede a activar el segundo

total de retraso buscado y vuelve a saltar el programa al apartado para encender el led. El ciclo, como se observa se repite continuamente sin efectos de parar.

Conclusiones

Creo que es muy importante recalcar que se está trabajando con elementos lógicos, por lo que aunque conocemos poco del lenguaje ensamblador, podemos crear algunos programas con lo que sabemos. Claro que desarrollando la práctica sé que el código puede tener algunos errores, o mejor aclarar, instrucciones que pueden ser obviadas, mejorar el código, hacerlo menor e incluso haber buscado nuevas instrucciones para optimizar éste.

Sin embargo, por lo pronto al ser una práctica con un código ensamblador pequeño, podían usarse programas de actividades pasadas para recrear el encendido del led y el retraso, consiguiendo juntar ambos propósitos en uno. Esto nos lleva a que claramente lo hecho está bajo una simulación, pues si ejecutamos el programa del anexo 1 en el software Proteus, conseguimos esta animación, sin embargo, a reserva de probarlo con el microcontrolador (pues las condiciones mundiales no lo permiten) se debe mantener con este modelo de animación por software.

Referencias

- Anónimo. (2007). Microcontrolador. Mayo 2, 2020, de Wikipedia Sitio web: https://es.wikipedia.org/wiki/Microcontrolador
- Anónimo. (2013). Microcontrolador. Mayo 3, 2020, de EcuRed Sitio web: https://www.ecured.cu/Microcontrolador
- Pahrami, B. (2005). Arquitectura de Computadoras. México: McGraw Hill.
- Wackerly, J. (2008). Diseño Digital. Principios y prácticas. México: Pearson.

Anexos

Anexo 1: Código del programa en ensamblador

```
ORG 0000H
      inicio:
      MOV R2,#00H
      MOV P1,R2
      ciclo:
      JB P1.1,enciende_led
      CLR P1.1
      SJMP ciclo

      enciende_led:
      SETB P1.0
      ACALL delay
      ACALL delay
      SJMP apaga_led

      apaga_led:
      CLR P1.1
      CLR P1.0
      ACALL delay
      ACALL delay
      SJMP enciende_led

      delay:
      MOV R6,#0FAH
      d1:
      MOV R7,#0F9H
      NOP
      NOP
      NOP
      NOP
      NOP
      d2:
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      DJNZ R7,d2
      DJNZ R6,d1
      RET
END
```