Instituto Politécnico Nacional

Escuela superior de Cómputo

Unidad de aprendizaje: Introducción a los microcontroladores

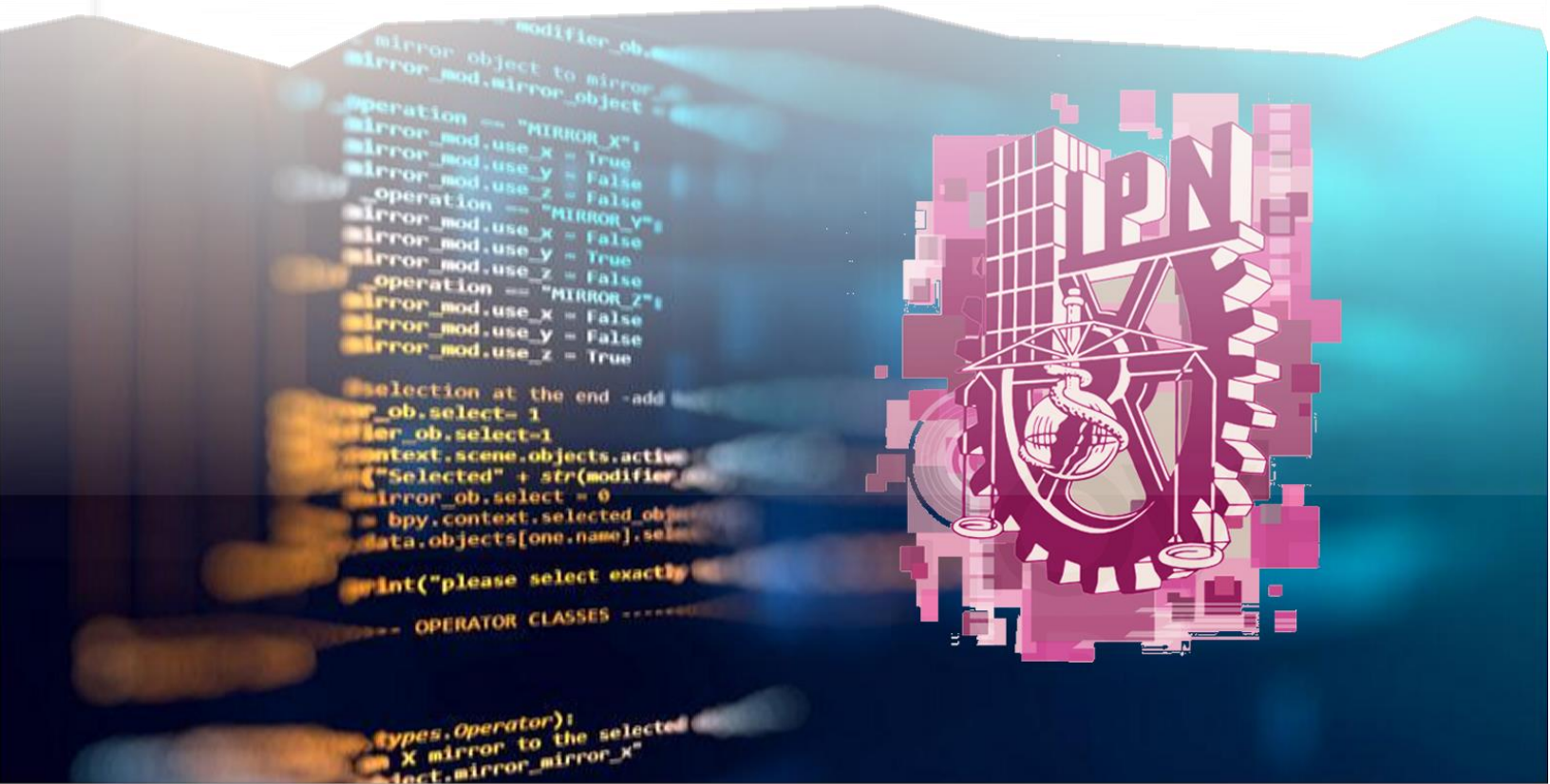
“Reportes prácticas 1, 2, 3 y 4”

Profesor: Pérez Pérez José Juan

Grupo: 3CM8

Alumnos:

Alarcón Zamudio Rafael Sebastián Arturo
De la Rosa Medina Christian Iván
López Romero Joel
Martínez San Román Aarón Hazel



Índice

Práctica 1.

Objetivo.

En esta práctica los alumnos capturaran y simularan los siguientes programas con la finalidad de familiarizarse con el entorno de AVR Studio, además de comprender la lógica que emplea el microcontrolador ATMega8535 por medio del lenguaje ensamblador.

Introducción.

AVR Studio es un entorno de desarrollo Integrado (IDE) de software desarrollado por Atmel. Proporcionando una plataforma de desarrollo única para las familias de microcontroladores AVR de 8 bits.

Práctica.

Capturar y simular los siguientes programas.

Material y equipo.

- Equipo de cómputo con AVR Studio.
- Fuente de voltaje
- 16 leds
- Microcontrolador ATMega8535
- 1 dip switch

Desarrollo.

Diagrama.

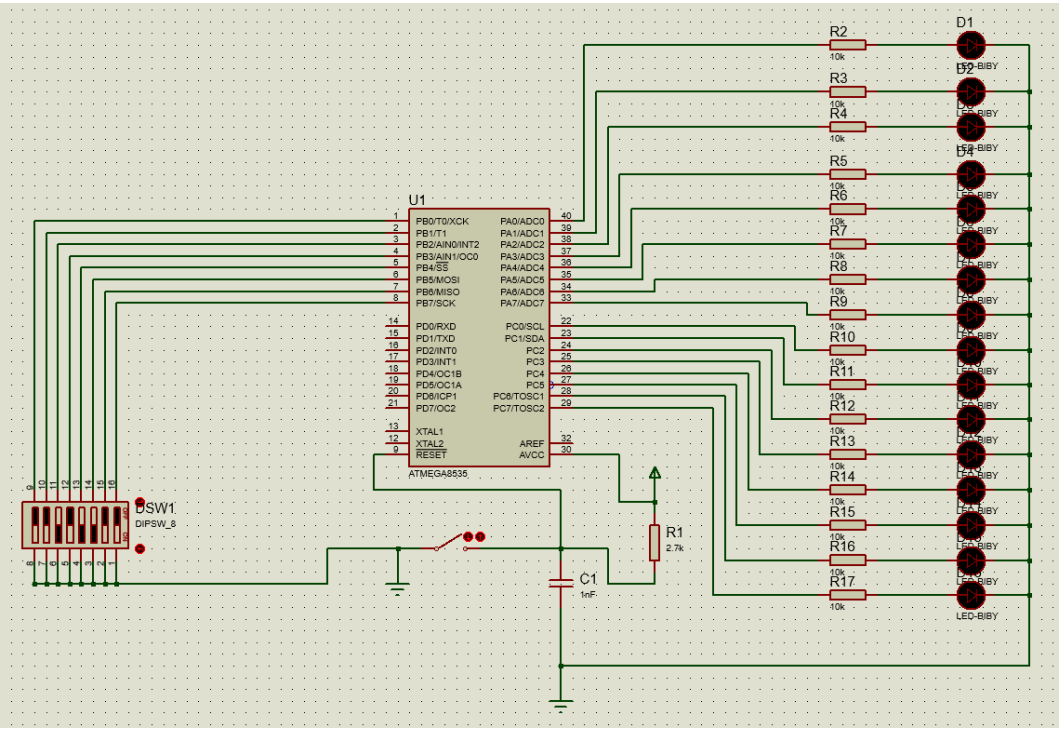


Diagrama del circuito.

Código.

1.

```
Ldi R16,20;  Carga la constante 20 en el registro 16
Ldi R17, $20;Carga la constante 0010 0000 en el registro 17
      Loop1:
      clr R19;  Indica la etiqueta loop1, limpia lo que haya dentro del registro 19
        mov R19, R17;  Mueve el valor del registro 17   al 19
        add R16, R19;  Suma el valor del registro 16 y 19
        inc R19;  Incrementa el valor del registro 19 en 1
            rjmp loop1;  Salta de regreso a la etiqueta loop1
```

2.

```
.include "m8535def.inc";  Incluye la libreria del ATMega8535
.def R16 = aux;  Define la variable aux que va apuntar al valor del registro 16
.def R17 = dato1;  Define la variable dato1 que va apuntar al valor del registro 17
.def R18 = dato2; Define la variable dato2 que va apuntar al valor del registro 18
Ldi aux, low(RAMEND); Cargamos el byte bajo del dato que se encuentra en la SRAM en la variable aux
out SPL, aux; El valor de aux se manda a la parte baja del Stack Pointer
Ldi aux, high(RAMEND); Cargamos el byte alto del dato que se encuentra en la SRAM en la variable aux
out SPH, aux; El valor de aux se manda a la parte alta del Stack Pointer
ser aux; Rellena el registro apuntado con la variable aux con 0xFF
out DDRA, aux; Se manda la variable aux al puerto A del registro de datos
out DDRC, aux; Se manda la variable aux al puerto C del registro de datos
out PORTB, aux; Se manda la variable aux al puerto de salida B
      alla:
            in dato1, PINB; Indica la etiqueta alla, guarda en la variable dato1 lo que entre por el PIN B de
                            entrada
            mov dato2, dato1; Mueve el valor del dato1 al dato2
            neg dato2; Saca el complemento a 2 de la variable dato2
            out PORTA, dato2; Manda al puerto de salida A el dato que se encuentra en la variable dato2
            out PORTC, dato1; Manda al puerto de salida C el dato que se encuentra en la variable dato1
      rjmp alla; Salta a la etiqueta alla
```

Conclusiones.

Alarcón Zamudio Rafael Sebastián Arturo

Está práctica fue muy didáctica ya que pudimos interactuar con la interfaz que nos ofrece el entorno de AVR estudio y practicar algunos comandos sencillos.

De la Rosa Medina Christian Iván

Tras el correcto desarrollo y análisis de la práctica en cuestión pudimos comprender mejor como navegar dentro del estudió AVR para después poder ejecutar los programas que se pidan.

López Romero Joel

Después de la ejecución de la práctica adquirimos conocimiento práctico relacionado a la programación en el estudio AVR la cual será de gran ayuda para prácticas posteriores.

Martínez San Román Aarón Hazel

Gracias a la exitosa realización de la práctica pudimos reafirmar conocimiento práctico como el ensamble de algunos componentes necesarios para el cableado así como la programación de los mismos.

Práctica 2.

Objetivo.

En esta práctica los alumnos desarrollaran un programa, el cual deberá leer el dato en los puertos y de acuerdo a su signo, este será enviado al puerto D o C, respectivamente. Además de configurar los pull-ups.

Introducción.

Cuando algún pin o los pines de AVR se han configurado como entradas digitales, mediante el registro PORTx se activa o desactiva unas resistencias Pull Up internas al microcontrolador AVR, cuando el bit del registro PORTx correspondiente a algún pin que es utilizado como entrada digital, se pone a 1 se activará la resistencia pull up correspondiente al pin, cuando se configura como 0 la resistencia pull up de ese pin estará desactivada.

Material y equipo.

- Equipo de cómputo con AVR Studio.
- Fuente de voltaje
- 30 leds
- Microcontrolador ATMega8535
- 2 dip switch
- Push Buttom

Práctica.

En esta práctica se debe desarrollar un programa para el microcontrolador ATMega8535.

1) Leer el dato del puerto **B**, si es positivo mandarlo al puerto **D**, si es negativo mandarlo al puerto **C**.
2) Configurar la mitad (pa7 – pa4) del puerto **A** como entrada y la otra mitad (pa3 – pa0) como salida, habilitando los pull-ups de los bits configurados como entrada, leer el dato presente en los 4 bits de entrada, complementarlos a 1 y escribirlos en los 4 bits configurados como salida, ciclar el programa de forma que se repita indefinidamente.

Desarrollo.

Diagrama.

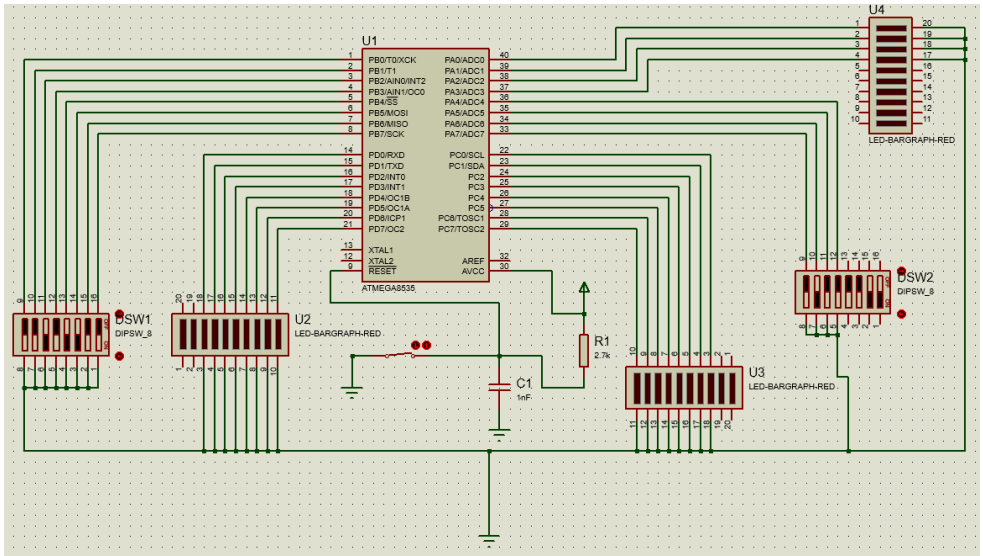


Diagrama del circuito

Código.

```
.include "m8535def.inc"; Incluye la libreria del ATMega8535
ser r16; Rellena el registro 16 con 0xFF
out DDRC,r16; Se manda el valor del registro 16 al puerto C del registro de datos
out DDRD,r16; Se manda el valor del registro 16 al puerto D del registro de datos
out portb,r16; Se manda el valor del registro 16 al puerto de salida B
ldi r16, $0f; Carga en el registro 16 el valor 0000 1111
out ddra,r16; Se manda el valor del registro 16 al puerto A del registro de datos
swap r16; Intercambia los nibbles del registro 16 para que quede 1111 0000
out porta,r16; Se manda el valor del registro 16 al puerto de salida A
      loop: Se define la etiqueta loop
            in r16,pinb; Carga en el registro 16 el dato mandado a través del pin B
            tst r16; Prueba si el registro es menor o igual que 0
            brpl pos; Aumenta el Contador del Programa en 1 y salta a la etiqueta pos si es positivo
            out portc,r16; Si no lo es, manda el valor del registro 16 al puerto de salida C
      rjmp otro; Salta a la etiqueta otro
      pos: Define la etiqueta pos
            out portd,r16; Se manda el valor del registro 16 al puerto de salida D
      otro: Define la etiqueta otro
            in r16,pina; Carga el valor del registro 16 al pin a
            com r16; Realiza el complemento a 1 del registro 16
            swap r16; Cambia los nibbles del registro 16
            out porta,r16; Se manda el valor del registro 16 al puerto de salida A
      rjmp loop; Regresa a la etiqueta loop
```

Conclusiones.

Alarcón Zamudio Rafael Sebastián Arturo

En esta práctica tuvimos que hacer uso de sentencias iterativas para poder cumplir con lo requerido, así como complementarlo con lo visto en la práctica anterior

De la Rosa Medina Christian Iván

Tras el correcto desarrollo y análisis de la práctica en cuestión pudimos comprender mejor como utilizar nuevas sentencias de código del estudió AVR como por ejemplo (Loop) para realizar ciclos con la finalidad de poder ejecutar los programas que se pidan.

López Romero Joel

Después de la ejecución de la práctica adquirimos conocimiento práctico relacionado a la programación en el estudio AVR, en este caso diferenciar números positivos y hacer uso de ciclos lo cual será de gran ayuda para prácticas posteriores.

Martínez San Román Aarón Hazel

Gracias a la exitosa realización de la práctica pudimos reafirmar conocimiento práctico como el ensamble de algunos componentes necesarios para el cableado así como la programación de los microcontroladores para que pudiera diferenciar entre los números que se le asignaban.

Práctica 3.

Objetivo.

En esta práctica, los alumnos desarrollaran e implementaran un programa en ensamblador, el cual consta en un decodificador de binario a decimal en displays de 7 segmentos.

Introducción.

El microcontrolador ATMega8535 permite desarrollar un decodificador mediante sus entradas en binario dando como salida 1s y 0s para formar la decodificación en decimal en displays de 7 segmentos.

Material y equipo.

- Equipo de cómputo con AVR Studio.
- Fuente de voltaje
- 2 displays
- Microcontrolador ATMega8535
- 1 dip switch
- Push Buttom

Práctica.

Desarrollar un decodificador Binario - Decimal en leguaje ensamblador e implementarlo en el circuito con 2 displays de 7 segmentos.

Desarrollo.

Diagrama.

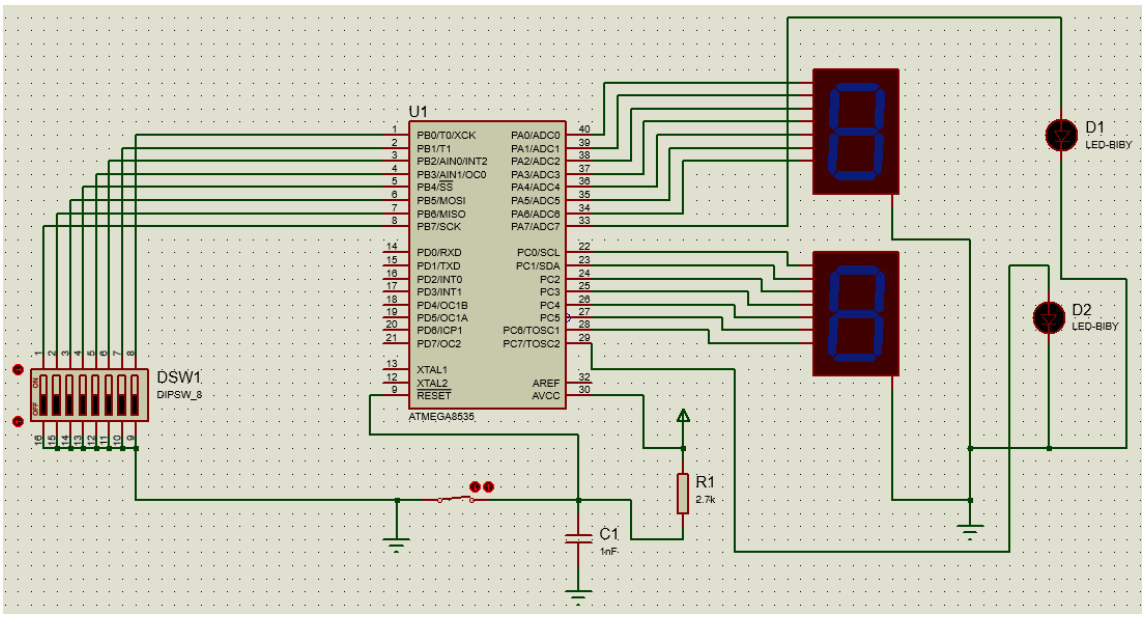


Diagrama del circuito

Código.

```
.include"m8535def.inc";  Incluye la librería del ATMega8535
.def dato1=r16; Define la variable dato1 que va apuntar al valor del registro 16
.def dato2=r17; Define la variable dato2 que va apuntar al valor del registro 17
ser dato1; Rellena al registro que apunta dato1 con 0xFF
out ddra,dato1;  Se manda el valor de dato1 al puerto A del registro de datos
out ddrc,dato1; Se manda el valor de dato1 al puerto C del registro de datos
out portb,dato1; Se manda el valor de dato1 al puerto B del registro de datos
ldi r20,$3f;  Carga el valor 0011 1111 al registro 2
ldi r21,6; Carga el valor 6 al registro 21
ldi r22,$5b;  Carga el valor 0101 1011 al registro 22
ldi r23,$4f;  Carga el valor 0100 1111 al registro 23
ldi r24,$66;  Carga el valor 0110 0110 al registro 24
ldi r25,$6d;  Carga el valor 0110 1101 al registro 25
ldi r26,$7d;  Carga el valor 0111 1101 al registro 26
ldi r27,$27;  Carga el valor 0010 0111 al registro 27
ldi r28,$7f;  Carga el valor 0111 1111 al registro 28
ldi r29,$6f;  Carga el valor 0110 1111 al registro 29
clr zh; Limpia la parte alta del registro 31
      otro:    Define la etiqueta otro
            ldi zl,20;  Carga el valor 20 a la parte baja del registro 31
            in dato1,pinb;  Carga en la variable dato1 lo que se mande por el pin B
            mov dato2,dato1;  Mueve el valor del dato1 al dato2
            andi dato1,$0f;  Realiza una operación and del dato1 con el valor 0000 1111
            cpi dato1,$0f;  Compara si es igual el dato1 con el valor 0000 1111
            cpi dato1,$0A; Compara si es igual el dato1 con el valor 0000 1010
            brsh punto1; Salta a la etiqueta punto1 si el bit en SREG está definido con 1
            add zl,dato1;  Suma dato1 con la parte baja del registro 31
            ld dato1,z;  Carga el valor del registro 31 en el registro apuntado por la variable dato1
            out porta,dato1;  Manda al puerto de salida A lo almacenado en dato1
      otro2:  Define la etiqueta otro2
            swap dato2 ;  Cambia los nibbles de la variable dato2
            andi dato2,$0f ;  Realiza una operación and entre el dato2 con la variable 0000 1111
            cpi dato2,$0A ;  Compara si es igual el dato2 con el valor 0000 1010
            brsh punto2;  Salta a la etiqueta punto2 si el bit en SREG está definido con 1
            ldi zl,20;  Carga en la parte baja del registro 31 el valor 20
            add zl,dato2;  Suma dato2 con la parte baja del registro 31
            ld dato2,z;  Carga el valor del registro 31 en la variable dato2
            out portc,dato2;  Manda al puerto de salida C lo almacenado en dato2
      rjmp otro;  Salta a la etiqueta otro
      punto1:  Define la etiqueta punto1
            ldi dato1,$80;  Carga en el dato1 el valor de 1000 0000
            out porta,dato1;  Manda al puerto de salida A lo almacenado en dato1
      rjmp otro2;  Salta de regreso a la etiqueta otro2
      punto2:  Define la etiqueta otro2
            ldi dato2,$80;  Carga en el dato2 el valor 1000 0000
            out portc,dato2;  Manda al puerto de salida C lo almacenado en dato2
      rjmp otro;  Salta de regreso a la etiqueta otro
```

Conclusiones.

Alarcón Zamudio Rafael Sebastián Arturo

La práctica 3 nos permitió implementar dos tipos diferentes de operaciones, cada una con su grado de complejidad, pero que, finalmente, gracias a los elementos proporcionados por el lenguaje ensamblador pudimos cubrir para así, cumplir con las operaciones solicitadas en esta práctica teniendo así un mayor conocimiento en la implementación de soluciones a través de un microcontrolador.

De la Rosa Medina Christian Iván

La realización de esta práctica fue de gran ayuda para conocer aún más el funcionamiento del microcontrolador, así como el lenguaje ensamblador, permitiendo implementar ciclos gracias a las instrucciones rjmp. Al utilizar un mayor número de complementos electrónicos tuvimos que disponer de una manera más eficiente los puertos disponibles en el microcontrolador.

López Romero Joel

La práctica aquí presente dividida en tres partes permitió realizar contadores y conversores de unidades a través de la lógica implementada en el lenguaje ensamblador para el microcontrolador. También, al incluir displays de 7 segmentos pudimos implementar programas que utilizan de manera eficiente las salidas y las entradas al microcontrolador.

Martínez San Román Aarón Hazel

Ésta práctica nos permitió implementar un mayor número de funciones y sentencias del lenguaje ensamblador para el microcontrolador, en esta ocasión, al realizar la práctica en 3 partes pudimos comparar las funciones utilizadas y así tener un mayor conocimiento para la implementación de posteriores prácticas eligiendo los métodos que sean más convenientes según los requerimientos de cada práctica.

Práctica 4.

Objetivo.

Los alumnos desarrollan un contador ascendente y descendente en ensamblador mediante el uso de subrutinas y un decodificar para ser mostrados en displays.

Introducción.

Para el microcontrolador ATMega8535, delay nos permite crear subrutinas de tiempo por milisegundos. En esta práctica se usara para crear subrutinas de tiempo de 250 milisegundos en un contador, además de deco para poder mostrar tal conteo en displays.

Material y equipo.

- Equipo de cómputo con AVR Studio.
- Fuente de voltaje
- 30 leds
- Microcontrolador ATMega8535
- 2 dip switch
- Push Buttom

Práctica.

Para el circuito dado, este deberá empezar una cuenta de la forma siguiente:

- Si el PB7=0 (ON), la cuenta deberá ser ascendente (0-9) cíclica.
- Si PB=1 (OFF), la cuenta deberá ser descendente (9-0) cíclica.

En ambos casos se deberá mostrar la cuenta en binario en el puerto **C**, el retardo entre cuenta deberá ser de 250 milisegundos, para esto se debe usar una “Subrutina” de tiempo (delay) de duración de 250 milisegundos, además se debe usar una subrutina de decodificación (deco) para tal efecto.

Desarrollo.

Diagrama.

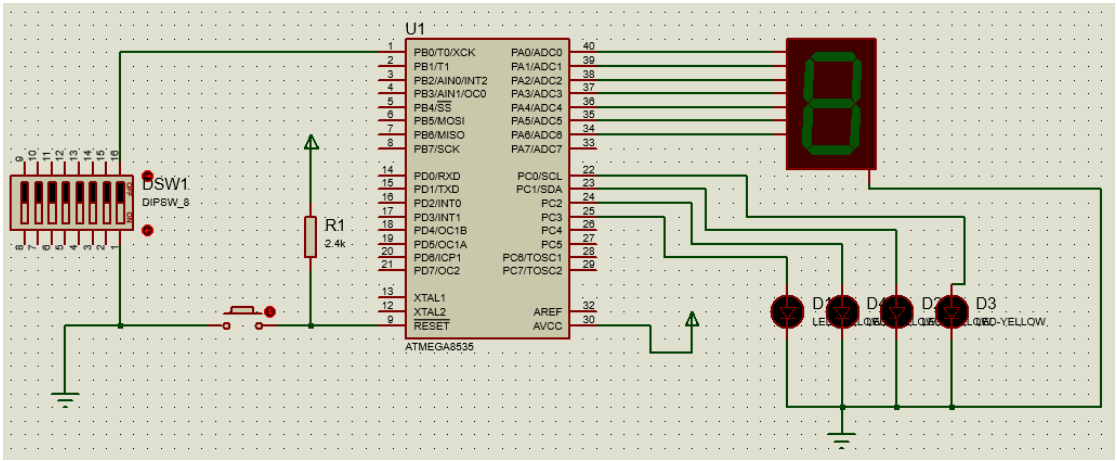


Diagrama del circuito

Código.

```
.include"m8535def.inc";  Incluye la libreria del ATMega8535
.def dato = R16;  Define la variable dato que va a apuntar al registro 16
.def aux = R17;  Define la variable aux que va a apuntar al registro 17
clr aux;  Llena con 0000 0000 la variable aux (r16)
out sph,aux; Manda a la parte alta del Stack Pointer el valor de la variable aux (r16)
ldi aux,$69;  Carga en la variable aux el valor de 0110 1001
out spl,aux;  Manda a la parte baja del Stack Pointer el valor en aux (SP=$0069)
ldi aux,$6f;  Carga en la variable aux el valor de 0110 1111     (Muestra el #9 en el display)
push aux;  Mete el valor de la variable aux en la Pila (STACK = 0110 1111)
ldi aux,$7f;  Carga en la variable aux el valor de 0111 1111     (Muestra el #8 en el display)
push aux;  Mete el valor de la variable aux en la Pila (STACK = 0111 1111)
ldi aux,$27;  Carga en la variable aux el valor de 0010 0111     (Muestra el #7 en el display)
push aux;  Mete el valor de la variable aux en la Pila (STACK = 0010 0111)
ldi aux,$7d;  Carga en la variable aux el valor de 0111 1100     (Muestra el #6 en el display)
push aux;  Mete el valor de la variable aux en la Pila (STACK = 0111 1100)
ldi aux,$6d;  Carga en la variable aux el valor de 0110 1100     (Muestra el #5 en el display)
push aux;  Mete el valor de la variable aux en la Pila (STACK = 0110 1100)
ldi aux,$66;  Carga en la variable aux el valor de 0110 0110     (Muestra el #4 en el display)
push aux;  Mete el valor de la variable aux en la Pila (STACK = 0110 0110)
ldi aux,$4f;  Carga en la variable aux el valor de 0100 1111     (Muestra el #3 en el display)
push aux;  Mete el valor de la variable aux en la Pila (STACK = 0100 1111)
ldi aux,$5b;  Carga en la variable aux el valor de 0101 1011     (Muestra el #2 en el display)
push aux;  Mete el valor de la variable aux en la Pila (STACK = 0101 1011)
ldi aux,$06;  Carga en la variable aux el valor de 0000 0110     (Muestra el #1 en el display)
push aux;  Mete el valor de la variable aux en la Pila (STACK = 0000 0110)
ldi aux,$3f;  Carga en la variable aux el valor de 0011 1111     (Muestra el #0 en el display)
push aux; Mete el valor de la variable aux en la Pila (STACK = 0000 1111)
ldi aux,low(RAMEND);  Carga el valor de la parte baja de la SRAM en la variable aux
out spl,aux;  Manda a la parte baja Stack Pointer el valor de la variable aux
ldi aux,high(RAMEND);  Carga el valor de la parte alta de la SRAM en la variable aux
out sph,aux;  Manda a la parte alta Stack Pointer el valor de la variable aux
ser aux;  Llena aux con 0xFF
out DDRA,aux;  Se manda el valor de aux al puerto A del registro de datos
out PORTB,aux;  Se manda el valor de aux al puerto de salida B
out PORTC,aux;  Se manda el valor de aux al puerto de salida C
      uno: Se define la etiqueta uno
            clr dato ;  Se llena la variable dato con 0x00
      dos:   Se define la etiqueta dos
            rcall deco ;  Se manda a llamar a la subrutina deco
            out porta, aux ;  Se manda el valor de aux al puerto de salida A
            rcall delay ;  Se manda a llamar a la subrutina delay
            in aux,pinb ;  Carga el valor que se manda en el pin B a la variable aux
            andi aux,$80 ;  Realiza una operacion and con la variable aux y el valor 1000 0000
            brmi tres ;  Si el resultado es negativo ve a la etiqueta tres
            inc dato ;  Si no, Incrementa el valor de dato en 1
            cpi dato,10 ;  Compara si es igual el dato1 con el valor 10
            brne dos;  Si el valor no es igual a 10 salta a la etiqueta dos
      rjmp uno;  Si no, salta a la etiqueta uno
      tres: Define a la etiqueta tres
            dec dato ;  Decrementa el valor de dato en 1
            brmi cuatro ;  Si el resultado es negativo salta a la etiqueta cuatro
            rjmp dos ;  Si no, salta a la etiqueta dos
      cuatro:    Define a la etiqueta cuatro
            ldi dato,9 ;  Carga en dato el valor 9
```

```
        rjmp dos ;  Salta a la etiqueta dos
deco:   Define a la etiqueta deco
        ldi zl,$60 ;  Carga en la parte alta del registro 31 el valor de 0110 0000
        add zl,dato ;  Suma la parte alta del registro 31 con el valor de dato
        ld aux,z ;  Carga en la variable aux el valor de z
        ret ;  Decrementa el Stack Pointer en 1 (Sale de la subrutina)
delay:  Define la etiqueta delay
        ldi R18, $A7 ;  Carga en el registro 18 el valor de 1010 0111
WGLOOP0: ldi R19, $02 ;  Define la etiqueta WGLOOP0, Carga en el registro 19 el valor de 0000 0010
WGLOOP1: ldi R20, $F8 ;  Define la etiqueta WGLOOP1, Carga en el registro 20 el valor de 1111 1000
WGLOOP2: dec R20 ;  Define la etiqueta WGLOOP0, Decrementa el valor del registro 20 en 1
brne WGLOOP2 ;  Si el resultado de la operación no es igual salta a la etiqueta WGLOOP2
dec R19 ;  Si no, decrementa en 1 al Registro 19
brne WGLOOP1;  Si el resultado de la operación no es igual salta a la etiqueta WGLOOP1
dec R18 ;  Si no, decrementa en 1 al Registro 18
brne WGLOOP0;  Si el resultado de la operación no es igual salta a la etiqueta WGLOOP0
nop ;  No hace nada
ret ;  Decrementa el Stack Pointer en 1 (Sale de la subrutina)
```

Conclusiones.

Alarcón Zamudio Rafael Sebastián Arturo

Esta práctica nos trajo varios problemas ya que al principio no podíamos implementar las entradas y salidas necesarias para el funcionamiento correcto del circuito, así como la condición de contador circular ya que nos mostraba basura al llegar al "tope" del contador. Sin embargo, gracias al uso de las diversas funciones ofrecidas por el microcontrolador pudimos completar esta práctica.

De la Rosa Medina Christian Iván

Al realizar esta práctica notamos el avance en el grado de complejidad de las prácticas debido a que se nos complicó la implementación de entradas externas al microcontrolador para así permitir el avance en el conteo sobre el display. Una parte importante fue la "recursividad" ya que también se nos complicó.

López Romero Joel

La realización de esta práctica nos permitió implementar un contador similar a uno implementado en el primer parcial. Sólo que en esta ocasión es necesaria una entrada al microcontrolador para realizar cada paso. Esta práctica fue un poco más complicada debido a las entradas y salidas que se debieron implementar.

Martínez San Román Aarón Hazel

Esta práctica permitió aplicar ciclos para implementar el contador circular. Un detalle importante de la práctica fue su funcionamiento de manera "manual" es decir, con entradas al microcontrolador se realizaba la operación de avanzar. Fue un poco difícil su implementación, pero finalmente se pudo realizar.