Instituto Politécnico Nacional

Escuela Superior de Cómputo

Introducción a los microcontroladores

Grupo: 3CM8

Reporte de Prácticas

Primer parcial

Alumnos:

Barriga Vargas Martín Eduardo

Mejia Matehuala Alfredo Daniel

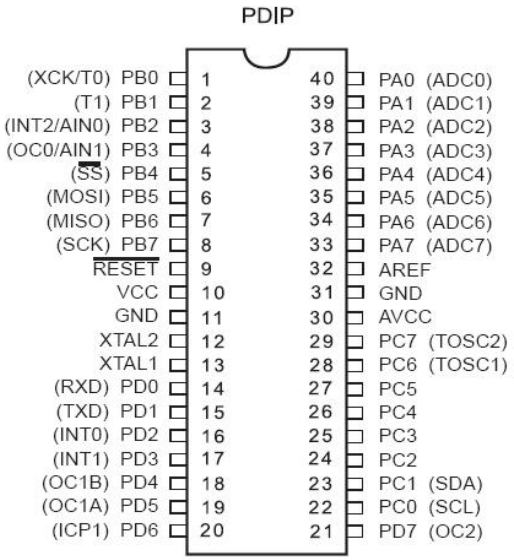
Ramirez Vives José Manuel

Índice

Práctica 1

Introducción:

Para realizar esta práctica, se deberán de conocer los fundamentos de ensamblador, así como de nuestro microcontrolador, que será el ATMega8535. Se realizará una suma de los valores insertados en los puertos d y b, y la veremos reflejada como salida en el puerto a, así como los valores del sreg en el puerto c.



Material:

- Microcontrolador ATMega8535
- 16 leds
- 1 protoboard
- 2 dipswitch
- 1 push button
- 1 capacitor de 1 microfaradio a 16 volts.
- 16 resistencias 100 ohms
- Fuente de voltaje 5v

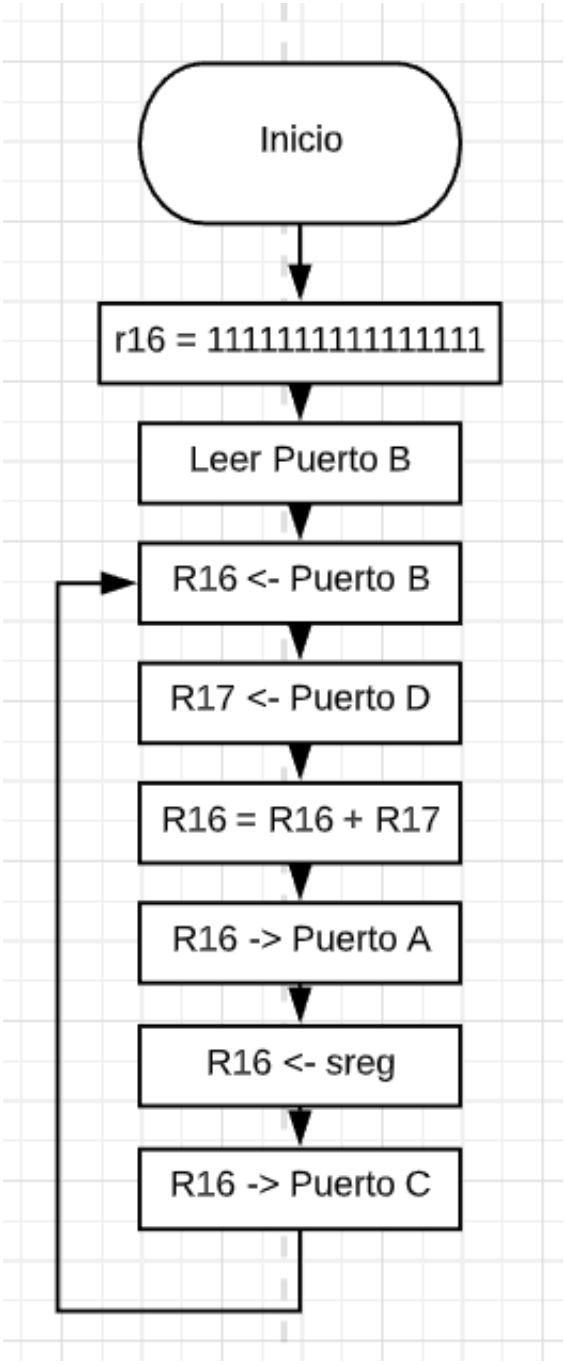
- Jumpers

Desarrollo:

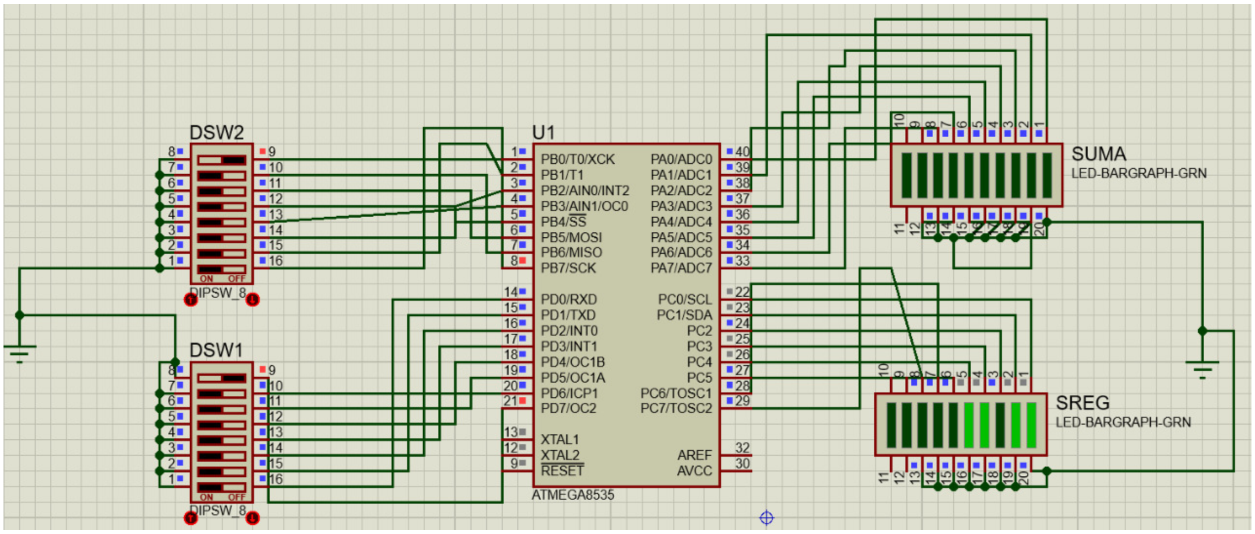
Código:

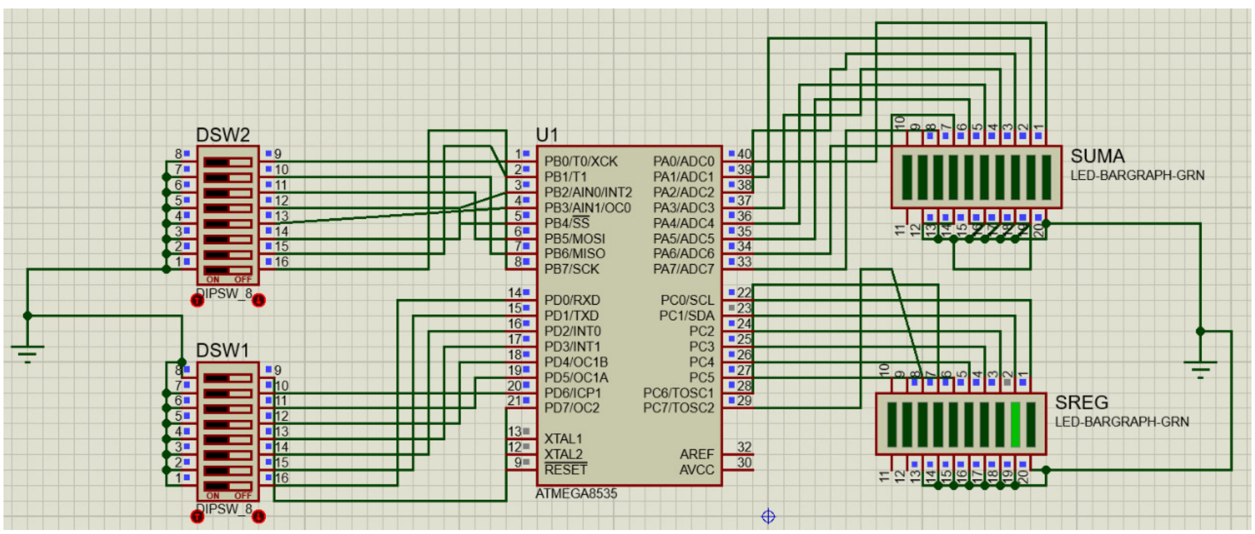
```
.include"m8535def.inc"
ser r16 //Llenamos el registro 16 con 1's
out ddra, r16 //Asignamos el puerto a como salida
out ddrc, r16 //Asignamos el puerto c como salida
out portb, r16 //Asignamos el puerto b como entrada
out portd, r16 //Asignamos el puerto d como entrada
uno:in r16, pinb // Guardamos en el registro 16, lo leído del puerto b
    in r17, pind // Guardamos en el registro 17, lo leído del puerto d
    add r16, r17 //Sumamos lo contenido en el registro 16 y 17
                 // y sobreescribimos el registro 16 con el resultado
    out porta, r16 //Mandamos al puerto de salida A, lo que tenga r16
    in r16, sreg //Sobreescribimos el registro 16 con el valor de sreg
    out portc, r16 //Mandamos al puerto de salida C, el valor de r16
    rjmp uno //Saltamos al target uno, creando un ciclo infinito

```

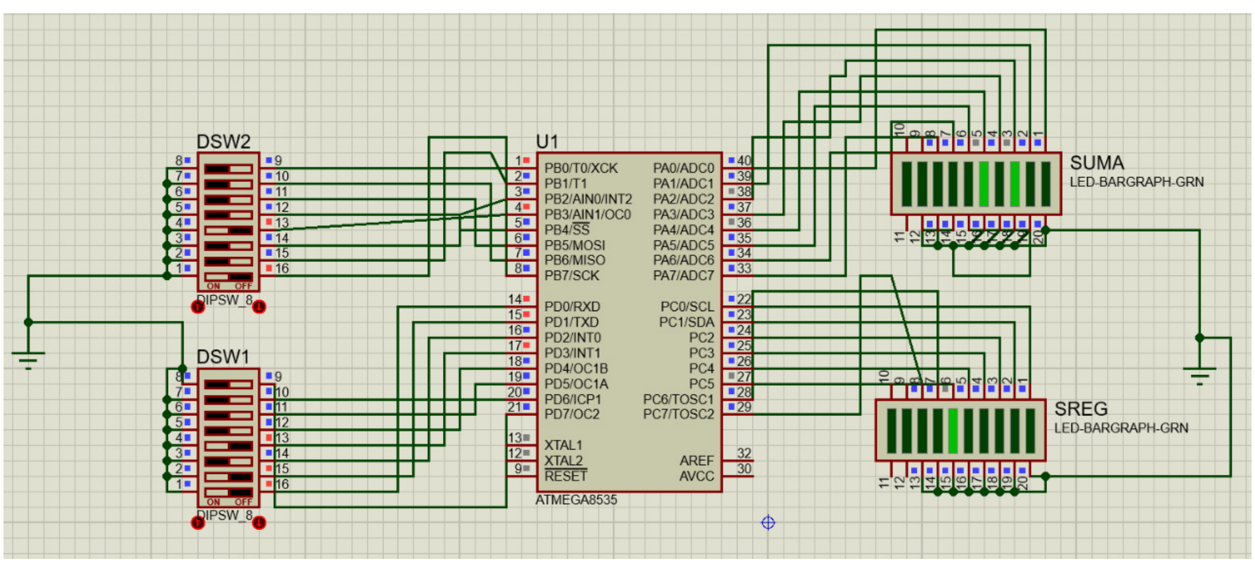
Diagrama de flujo:



Simulación:


DSW2
DIPSW_8
DSW1
DIPSW_8
U1
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
XTAL1
XTAL2
RESET
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PC0/SCL
PC1/SDA
PC2
PC3
PC4
PC5
PC6/TOSC1
PC7/TOSC2
AREF
AVCC
ATMEGA8535
SUMA
LED-BARGRAPH-GRN
SREG
LED-BARGRAPH-GRN


DSW2
DIPSW_8
DSW1
DIPSW_8
U1
ATMEGA8535
SUMA
LED-BARGRAPH-GRN
SREG
LED-BARGRAPH-GRN


DSW2
DIPSW_8
DSW1
DIPSW_8
U1
ATMEGA8535
SUMA
LED-BARGRAPH-GRN
SREG
LED-BARGRAPH-GRN

Conclusiones:

Barriga Vargas Martín Eduardo: Al ser la primer práctica, fue un poco difícil acostumbrarnos a la parte de usar registros para almacenar la información, así como la forma en la que se realizaba la suma y la entrada y salida de información por los puertos del microcontrolador.

Mejía Matehuala Alfredo: Ésta práctica conllevó el aprendizaje mas significativo respecto al uso del microprocesador y el programador. Fue el primer encuentro con este tipo de procesador y el usarlo fue realmente útil. El armado no fue complicado ni el diseño del programa en sí, pero fue muy útil en general.

Ramirez Vives José Manuel: Con esta primera práctica aprendimos el uso básico del microcontrolador con un programa sencillo. Vimos como hacer uso de los puertos de entrada y salida, las funciones básicas para utilizar en el programa del microcontrolador, as como de las variables que ya están previamente definidas como lo es el SREG.

Bibliografía

https://www.unioviedo.es/ate/alberto/TEMA3-Ensamblador.pdf
https://staffweb.cms.gre.ac.uk/~sp02/assembly/lecture1.html

Práctica 2

Introducción:

En esta práctica se introducirá por el puerto b un número en hexadecimal, esperando la salida del número correspondiente en ascii por el puerto a. Y viceversa, introducir un número ascii por el puerto d y obtener como salida en el puerto c.
El sistema hexadecimal tiene como base el número 16; se utiliza a menudo para una representación condensada del número binario mediante cadenas de 4 bits. Utiliza cifras del 0 al 9 y letras de la A a la F.
El ascii es un código numérico que representa los caracteres, usando una escala decimal del 0 al 127.

Material:

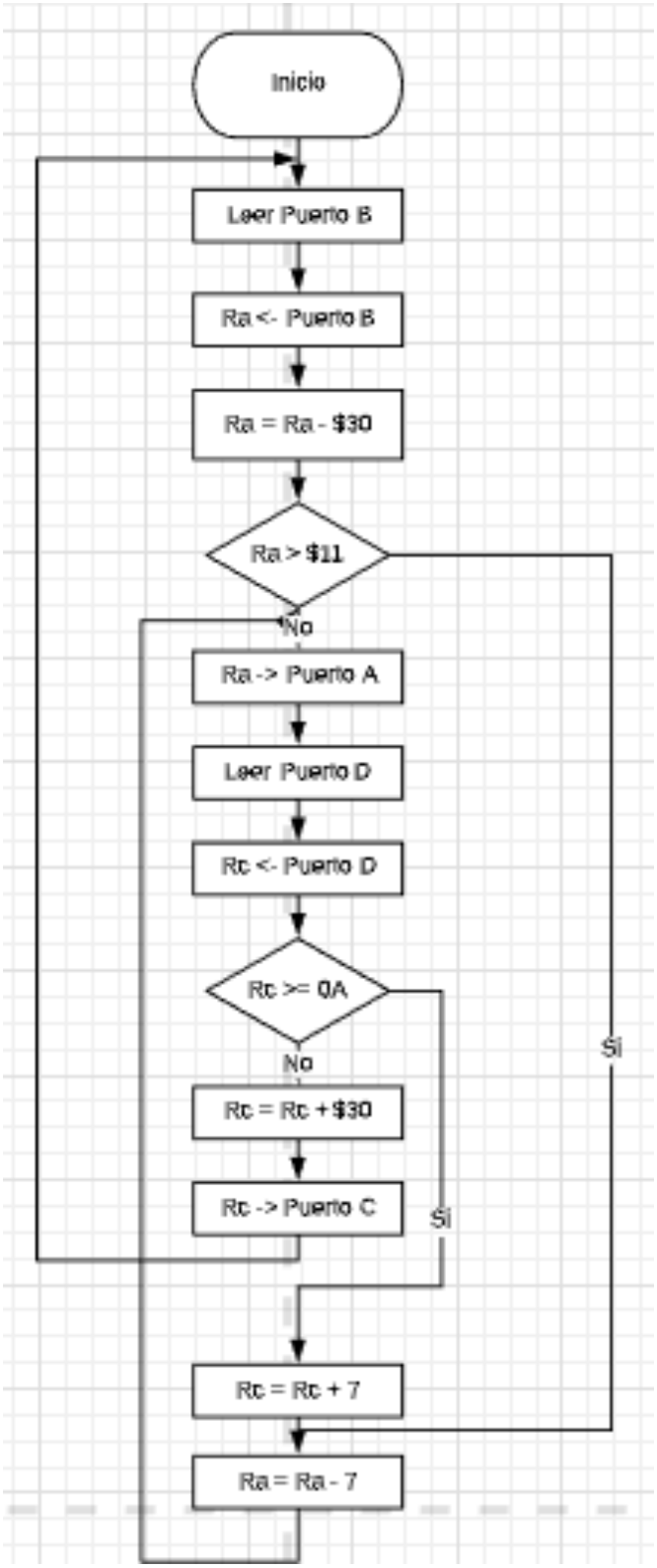
- Microcontrolador ATMega8535
- 16 leds
- 1 protoboard
- 2 dipswitch
- 1 push button
- 1 capacitor de 1 microfaradio a 16 volts.
- 16 resistencias 100 ohms
- Fuente de voltaje 5v
- Jumpers

Desarrollo:

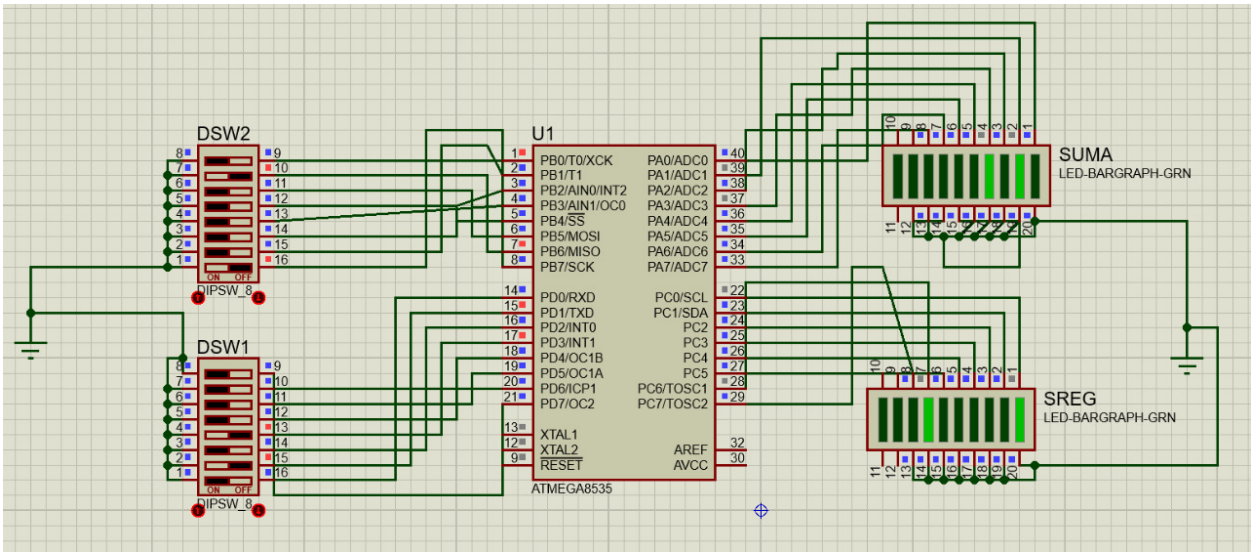
Código:

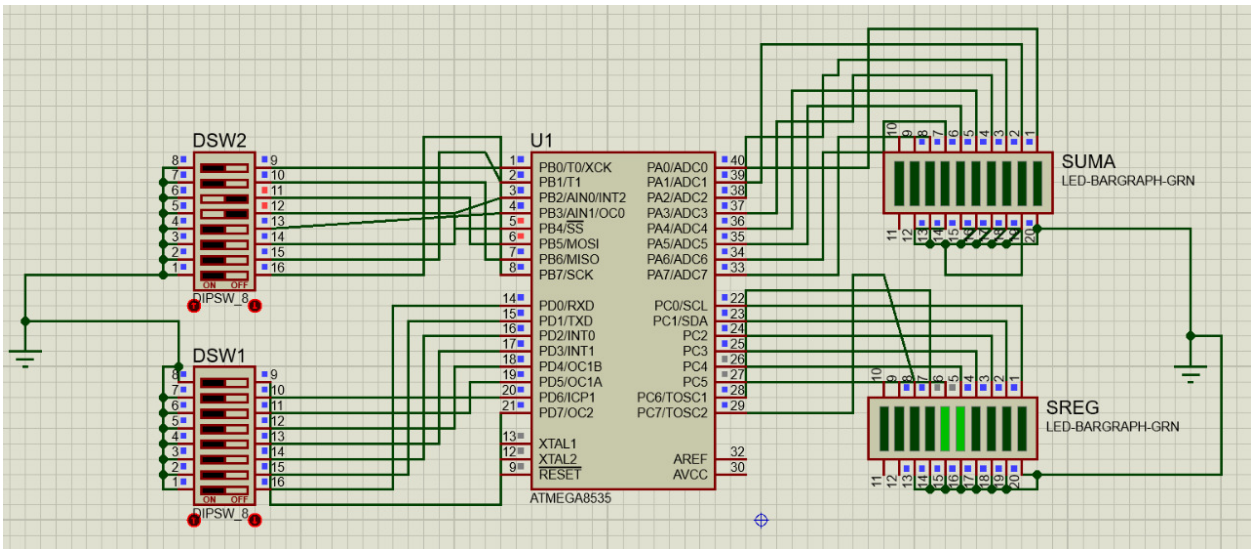
```
.include "m8535def.inc"
.def ra = r16 //Vinculamos ra con r16
.def aux = r17 //Vinculamos aux con r17
.def rc = r18 //Vinculamos rc con r18
.def rtreinta = r19 //Vinculamos rtreinta con r19
.def rsiete = r20 //Vinculamos rsiete con r20

ser aux //Llenamos aux con 1's
out ddra, aux //Asignamos el puerto a como salida
out ddrc, aux //Asignamos el puerto c como salida
out portb, aux //Asignamos el puerto b como entrada
out portd, aux //Asignamos el puerto d como entrada
ldi rtreinta, $30 //Metemos el valor $30 dentro del reg.
ldi rsiete, $07 //Metemos el valor $07 dentro del reg.

otro:
    in ra, pinb //leemos el valor del puerto b y lo metemos en ra
    subi ra, $30  //le restamos a ra el valor $30
    cpi ra, $11 //preguntamos si el valor restante es mayor a $11,
                //si es así, se trata de una letra y pasa a brge letra
    brge letra //Salta a la etiqueta letra

    sal1:
        out porta, ra //Mandamos al puerto a el valor en ra
        in rc, pind //Leemos del puerto d y lo almacenamos en rc
        cpi rc, $0A //Preguntamos si rc es mayor o igual a 0A
        brge hex//En caso de haber sido mayor, lo mandamos al tag hex
        sal2:
            add rc, rtreinta //finalmente le aumentamos $30
            out portc, rc //mandamos al puerto c , el valor en rc
            rjmp otro // saltamos al tag otro, para crear un ciclo inf.
        hex:
            add rc, rsiete //Le sumamos 7, pues es la dif. entre $41 y $39
            rjmp sal2
    letra:
        subi ra, $07 //Se resta 7 a ra, pues si nos encontramos dentro de
                     //esta etiqueta es porque ra ahora vale por lo menos
                     //11, y la diferencia entre 11 y 0A, que es a lo que
                     //queremos igualarlo, es de 7.
        rjmp sal1 //saltamos al tag sal1

```

Diagrama de flujo:



Simulación:


DSW2
DSW1
DIPSW_8
U1
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
XTAL1
XTAL2
RESET
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PC0/SCL
PC1/SDA
PC2
PC3
PC4
PC5
PC6/TOSC1
PC7/TOSC2
AREF
AVCC
ATMEGA8535
SUMA
LED-BARGRAPH-GRN
SREG
LED-BARGRAPH-GRN


DSW2
DSW1
DIPSW_8
U1
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
XTAL1
XTAL2
RESET
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PC0/SCL
PC1/SDA
PC2
PC3
PC4
PC5
PC6/TOSC1
PC7/TOSC2
AREF
AVCC
ATMEGA8535
SUMA
LED-BARGRAPH-GRN
SREG
LED-BARGRAPH-GRN

Conclusiones:

Barriga Vargas Martín Eduardo: Al inicio parecía muy complejo el resolver dos problemas a la vez, pero se nos ocurrió que podríamos resolver primero el problema de hexadecimal a ascii, y justo antes de volver a ciclar el programa, hacer el código para pasar de ascii a hexadecimal. Una vez que se entendía cuanta era la magnitud de diferencia entre hexadecimal y ascii, se volvía muy fácil resolver el problema.

Mejía Matehuala Alfredo: En la segunda práctica no tuvimos mayores complicaciones. EL armado se complicó un poco pero con el uso de buenas prácticas se logró. Identificamos que algunos DIP switch meten algo de ruido en las señales, así que lo probamos varias veces para que no diera problemas.

Ramirez Vives José Manuel: Con esta práctica hicimos uso de otras funciones ms específicas del microcontrolador que implican operaciones con las entradas y saltos condicionados según los resultados de estas operaciones. Aunque al inicio nos cost un poco de trabajo comprender como funcionaban los saltos e interrupciones, pero finalmente conseguimos generar el código adecuado.

Bibliografía

https://www.unioviedo.es/ate/alberto/TEMA3-Ensamblador.pdf
https://staffweb.cms.gre.ac.uk/~sp02/assembly/lecture1.html
https://es.wikipedia.org/wiki/Sistema_hexadecimal

Práctica 3

Introducción:

Para esta práctica se construirá un programa, en el que se pueda introducir un número en hexadecimal, y que se muestre a través de un display un: 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, dependiendo del número introducido.
El display de 7 segmentos es un dispositivo electrónico que se utiliza para representar visualmente números y algunos caracteres. Este display es muy popular debido a su gran efectividad y simplicidad al momento de utilizarlo.
El sistema hexadecimal tiene como base el número 16; se utiliza a menudo para una representación condensada del número binario mediante cadenas de 4 bits. Utiliza cifras del 0 al 9 y letras de la A a la F.

Material:

- Microcontrolador ATMega8535
- 1 protoboard
- 1 dipswitch
- 1 display
- 1 push button
- 1 capacitor de 1 microfaradio a 16 volts.
- Fuente de voltaje 5v
- Jumpers

Desarrollo:

Código:

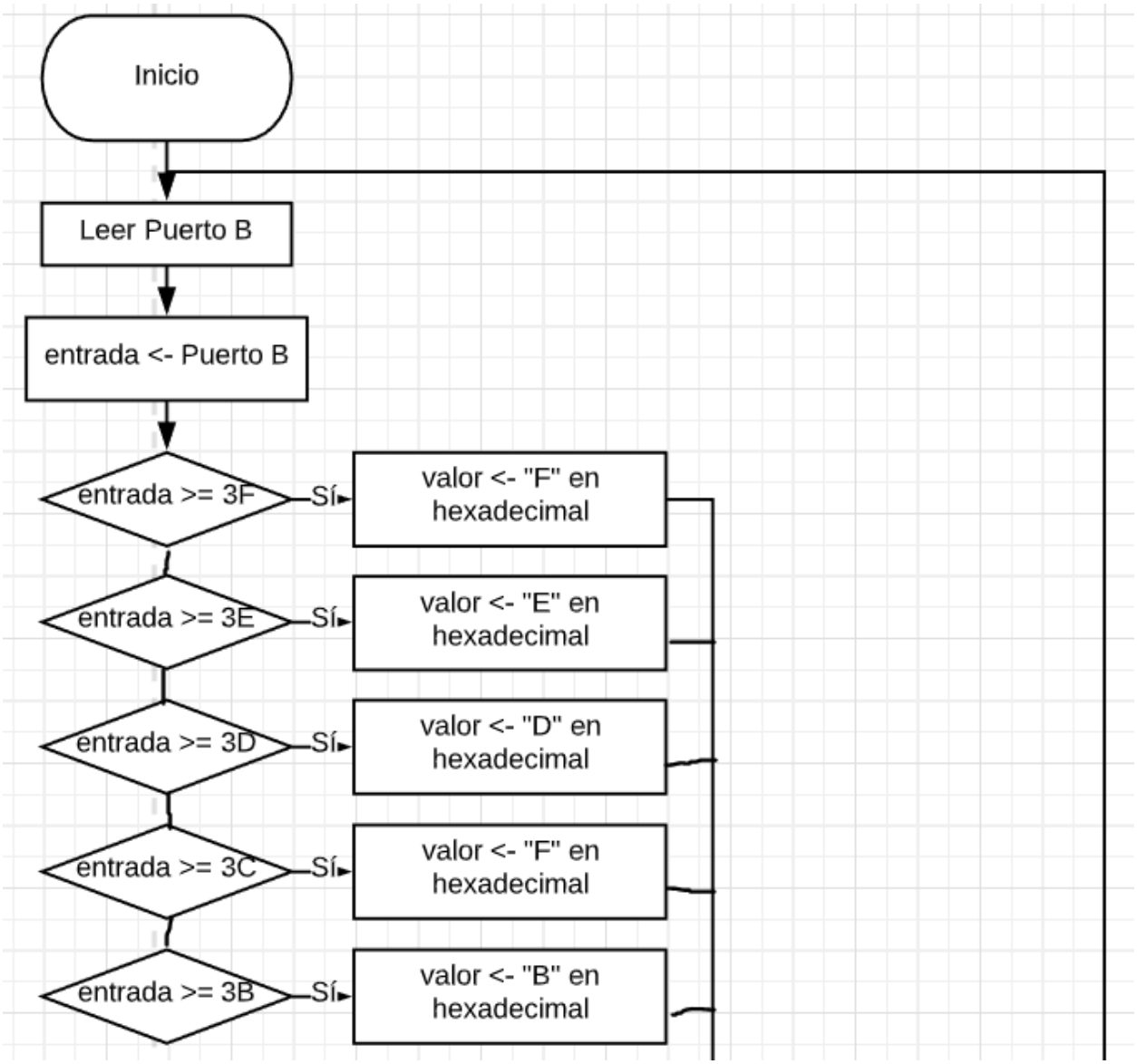
```
.include "m8535def.inc"

.def cont = r17 //vinculamos cont con r17
.def valor = r18 //vinculamos valor con r18
.def entrada = r19 //vinculamos entrada con r19
.def aux = r20 //vinculamos aux con r20

ser aux //llenamos aux con 1's
out ddra, aux //definimos ddra como puerto de salida
out portb, aux//definimos port b como puerto de entrada

inic:
    in entrada, pinb //guardamos en entrada el valor del puerto b

    cpi entrada, $46 //preguntamos si el valor es mayor o igual a 3F
    brge targetF //si es el caso, lo mandamos al tag F

    cpi entrada, $45 //preguntamos si el valor es mayor o igual a 3E
    brge targetE //si es el caso, lo mandamos al tag E

    cpi entrada, $44 //preguntamos si el valor es mayor o igual a 3D
    brge targetD //si es el caso, lo mandamos al tag D

    cpi entrada, $43 //preguntamos si el valor es mayor o igual a 3C
    brge targetC //si es el caso, lo mandamos al tag C

    cpi entrada, $42 //preguntamos si el valor es mayor o igual a 3B
    brge targetB //si es el caso, lo mandamos al tag B

    cpi entrada, $41 //preguntamos si el valor es mayor o igual a 3A
    brge targetA //si es el caso, lo mandamos al tag A

    cpi entrada, $39 //preguntamos si el valor es mayor o igual a 39
    brge targetNueve //si es el caso, lo mandamos al tag nueve

    cpi entrada, $38 //preguntamos si el valor es mayor o igual a 38
    brge targetOcho //si es el caso, lo mandamos al tag ocho

    cpi entrada, $37 //preguntamos si el valor es mayor o igual a 37
    brge targetSiete //si es el caso, lo mandamos al tag siete

    cpi entrada, $36 //preguntamos si el valor es mayor o igual a 36
    brge targetSeis //si es el caso, lo mandamos al tag seis

```

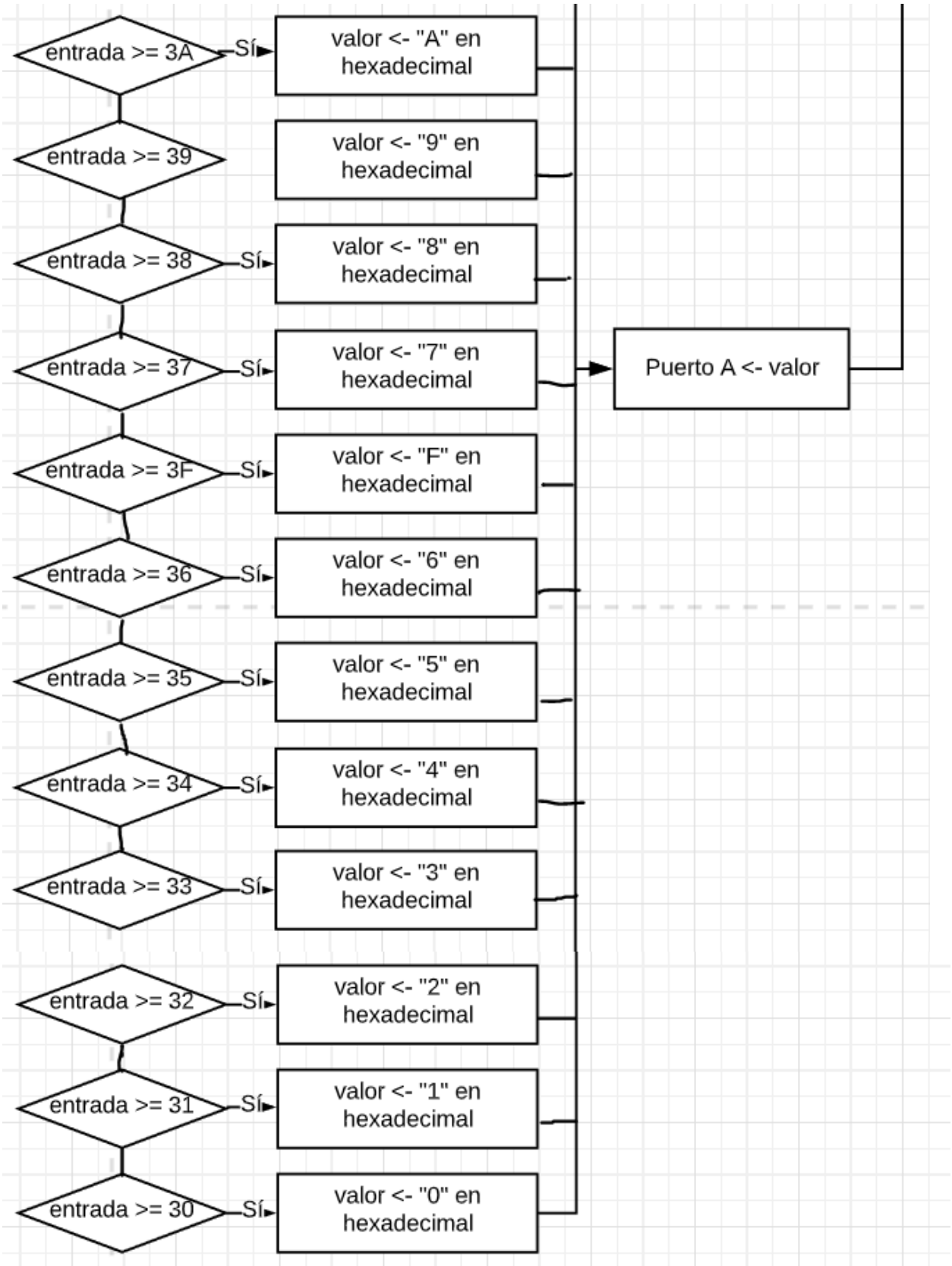
```
    cpi entrada, $35 //preguntamos si el valor es mayor o igual a 35
    brge targetCinco //si es el caso, lo mandamos al tag cinco

    cpi entrada, $34 //preguntamos si el valor es mayor o igual a 34
    brge targetCuatro //si es el caso, lo mandamos al tag cuatro

    cpi entrada, $33 //preguntamos si el valor es mayor o igual a 33
    brge targetTres //si es el caso, lo mandamos al tag tres

    cpi entrada, $32 //preguntamos si el valor es mayor o igual a 32
    brge targetDos //si es el caso, lo mandamos al tag dos

    cpi entrada, $31 //preguntamos si el valor es mayor o igual a 31
    brge targetUno //si es el caso, lo mandamos al tag uno

    cpi entrada, $30 //preguntamos si el valor es mayor o igual a 30
    brge targetCero //si es el caso, lo mandamos al tag cero

    rjmp inic // saltamos al tag inic


targetA:
    ldi valor,$77  //le asignamos la representacion de la A en display en hex.
    out porta, valor //Mandamos al puerto A, lo almacenado en valor
    rjmp inic //Saltamos al tag inic para hacer un loop infinito

targetB:
    ldi valor,$7C   //le asignamos la representacion de la B en display en hex.
    out porta, valor //Mandamos al puerto A, lo almacenado en valor
    rjmp inic //Saltamos al tag inic para hacer un loop infinito

targetC:
    ldi valor,$39  //le asignamos la representacion de la C en display en hex.
    out porta, valor //Mandamos al puerto A, lo almacenado en valor
    rjmp inic //Saltamos al tag inic para hacer un loop infinito

targetD:
    ldi valor,$5E  //le asignamos la representacion de la D en display en hex.
    out porta, valor //Mandamos al puerto A, lo almacenado en valor
    rjmp inic //Saltamos al tag inic para hacer un loop infinito

targetE:
    ldi valor,$79  //le asignamos la representacion de la E en display en hex.
    out porta, valor //Mandamos al puerto A, lo almacenado en valor
    rjmp inic //Saltamos al tag inic para hacer un loop infinito
```

```
targetF:
    ldi valor,$71  //le asignamos la representacion de la F en display en hex.
    out porta, valor //Mandamos al puerto A, lo almacenado en valor
    rjmp inic //Saltamos al tag inic para hacer un loop infinito


targetNueve:
    ldi valor,$6f  //le asignamos la representacion deL 9 en display en hex.
    out porta, valor //Mandamos al puerto A, lo almacenado en valor
    rjmp inic //Saltamos al tag inic para hacer un loop infinito

targetOcho:
    ldi valor,$7f  //le asignamos la representacion del 8 en display en hex.
    out porta, valor //Mandamos al puerto A, lo almacenado en valor
    rjmp inic //Saltamos al tag inic para hacer un loop infinito

targetSiete:
    ldi valor,$27  //le asignamos la representacion del 7 en display en hex.
    out porta, valor //Mandamos al puerto A, lo almacenado en valor
    rjmp inic //Saltamos al tag inic para hacer un loop infinito

targetSeis:
    ldi valor,$7d  //le asignamos la representacion del 6 en display en hex.
    out porta, valor //Mandamos al puerto A, lo almacenado en valor
    rjmp inic //Saltamos al tag inic para hacer un loop infinito

targetCinco:
    ldi valor,$6d  //le asignamos la representacion del 5 en display en hex.
    out porta, valor //Mandamos al puerto A, lo almacenado en valor
    rjmp inic //Saltamos al tag inic para hacer un loop infinito

targetCuatro:
    ldi valor,$66  //le asignamos la representacion del 4 en display en hex.
    out porta, valor //Mandamos al puerto A, lo almacenado en valor
    rjmp inic //Saltamos al tag inic para hacer un loop infinito

targetTres:
    ldi valor,$4f  //le asignamos la representacion del 3 en display en hex.
    out porta, valor //Mandamos al puerto A, lo almacenado en valor
    rjmp inic //Saltamos al tag inic para hacer un loop infinito

targetDos:
    ldi valor,$5b  //le asignamos la representacion del 2 en display en hex.
    out porta, valor //Mandamos al puerto A, lo almacenado en valor
    rjmp inic //Saltamos al tag inic para hacer un loop infinito
```

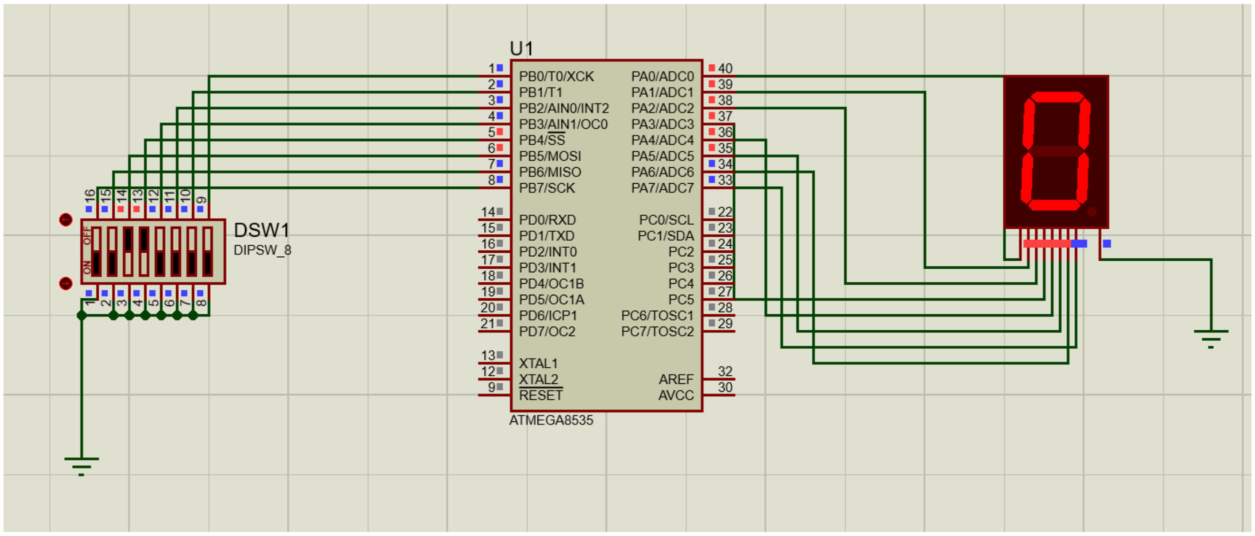
```
targetUno:
    ldi valor,$06  //le asignamos la representacion del 1 en display en hex.
    out porta, valor //Mandamos al puerto A, lo almacenado en valor
    rjmp inic //Saltamos al tag inic para hacer un loop infinito

targetCero:
    ldi valor,$3f  //le asignamos la representacion del 0 en display en hex.
    out porta, valor //Mandamos al puerto A, lo almacenado en valor
    rjmp inic //Saltamos al tag inic para hacer un loop infinito
```

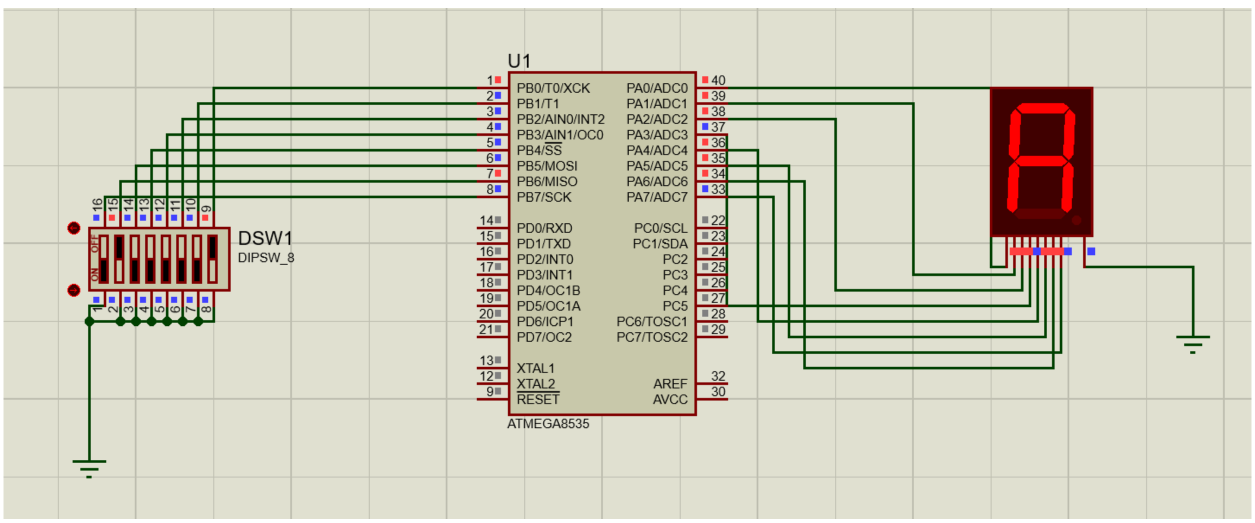
Diagrama de flujo:

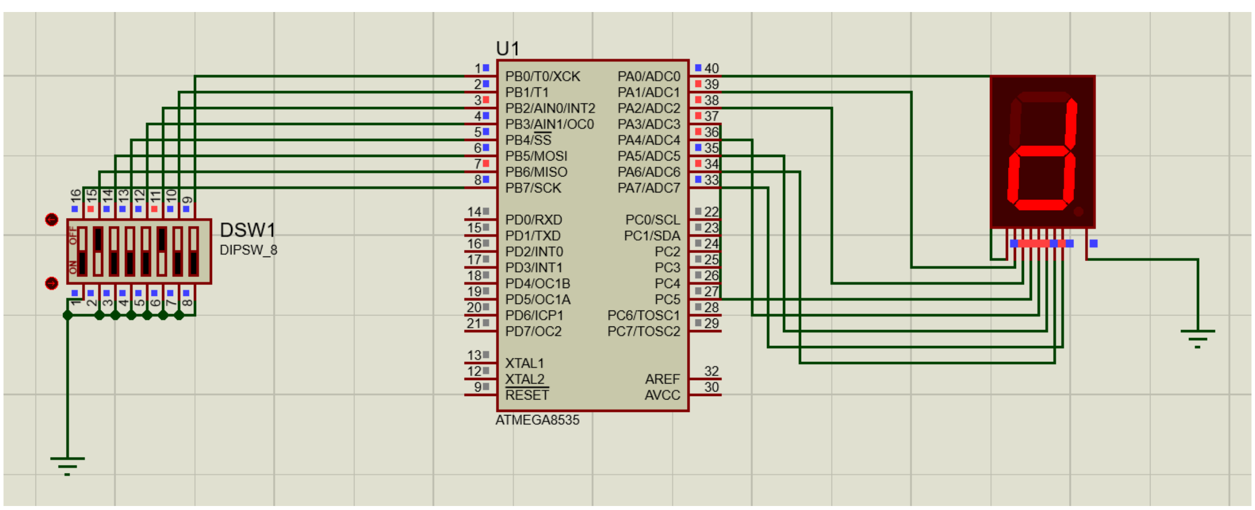


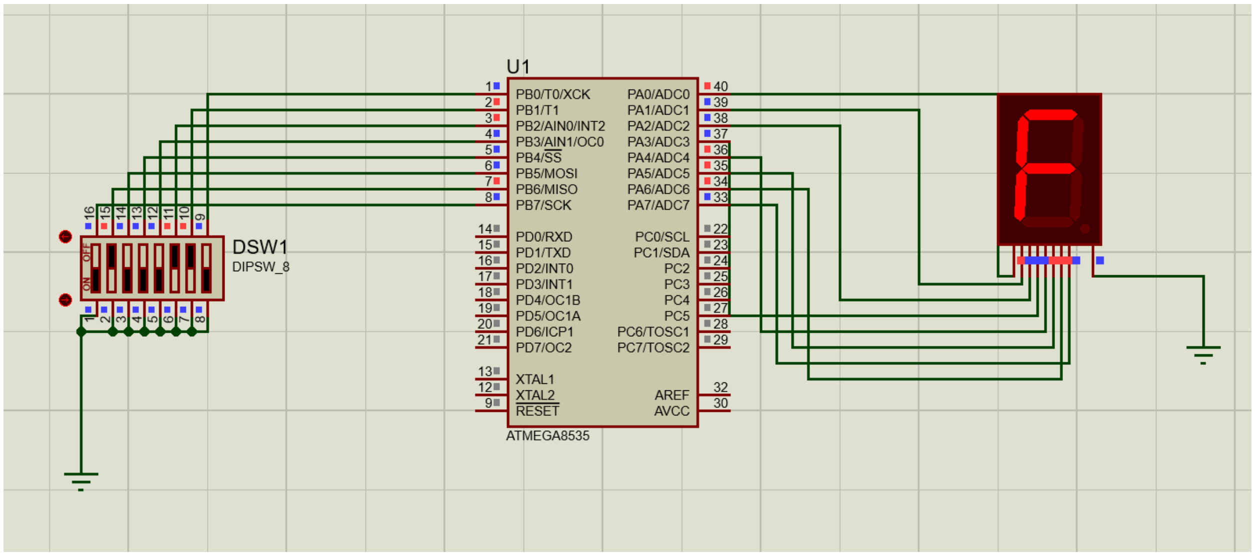

entrada >= 3A
Sí
valor <- "A" en hexadecimal
entrada >= 39
valor <- "9" en hexadecimal
entrada >= 38
Sí
valor <- "8" en hexadecimal
entrada >= 37
Sí
valor <- "7" en hexadecimal
Puerto A <- valor
entrada >= 3F
Sí
valor <- "F" en hexadecimal
entrada >= 36
Sí
valor <- "6" en hexadecimal
entrada >= 35
Sí
valor <- "5" en hexadecimal
entrada >= 34
Sí
valor <- "4" en hexadecimal
entrada >= 33
Sí
valor <- "3" en hexadecimal
entrada >= 32
Sí
valor <- "2" en hexadecimal
entrada >= 31
Sí
valor <- "1" en hexadecimal
entrada >= 30
Sí
valor <- "0" en hexadecimal

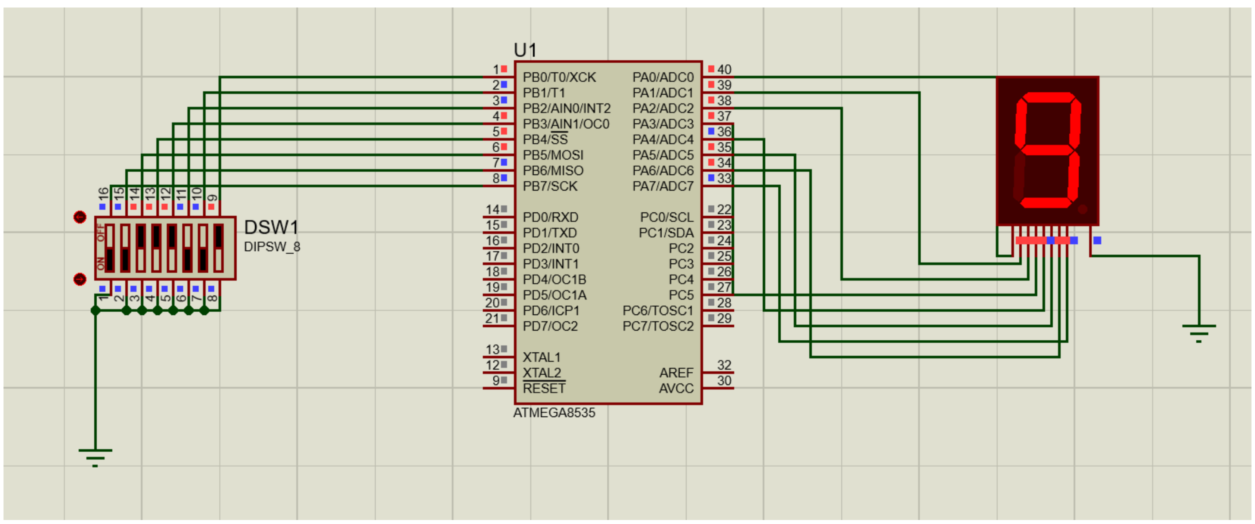
Simulación:

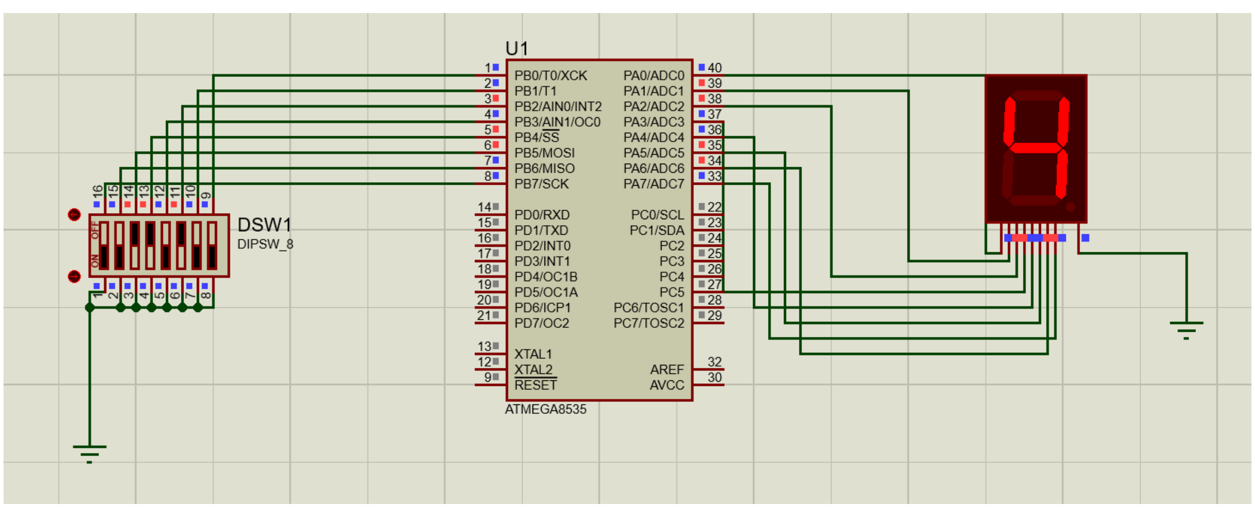

U1
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
XTAL1
XTAL2
RESET
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PC0/SCL
PC1/SDA
PC2
PC3
PC4
PC5
PC6/TOSC1
PC7/TOSC2
AREF
AVCC
ATMEGA8535
DSW1
DIPSW_8


U1
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
XTAL1
XTAL2
RESET
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PC0/SCL
PC1/SDA
PC2
PC3
PC4
PC5
PC6/TOSC1
PC7/TOSC2
AREF
AVCC
ATMEGA8535
DSW1
DIPSW_8


U1
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
XTAL1
XTAL2
RESET
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PC0/SCL
PC1/SDA
PC2
PC3
PC4
PC5
PC6/TOSC1
PC7/TOSC2
AREF
AVCC
ATMEGA8535
DSW1
DIPSW_8


U1
DSW1
DIPSW_8
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
XTAL1
XTAL2
RESET
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PC0/SCL
PC1/SDA
PC2
PC3
PC4
PC5
PC6/TOSC1
PC7/TOSC2
AREF
AVCC
ATMEGA8535


U1
DSW1
DIPSW_8
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
XTAL1
XTAL2
RESET
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PC0/SCL
PC1/SDA
PC2
PC3
PC4
PC5
PC6/TOSC1
PC7/TOSC2
AREF
AVCC
ATMEGA8535


U1
DSW1
DIPSW_8
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
XTAL1
XTAL2
RESET
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PC0/SCL
PC1/SDA
PC2
PC3
PC4
PC5
PC6/TOSC1
PC7/TOSC2
AREF
AVCC
ATMEGA8535

Conclusiones:

Barriga Vargas Martín Eduardo: Algo que no conocíamos pero imprescindible era la parte de representar de forma hexadecimal nuestro número en el display, pues la parte de saber a qué número equivale el que entra por el puerto ya lo hemos hecho en la práctica 2. Por lo cual una vez que supimos eso y lo combinamos con la 2, nos dio como resultado esta práctica de forma exitosa.

Mejía Matehuala Alfredo: Esta fue la práctica que más trabajo costó. El código implementado al inicio no funcionaba y experimentamos problemas de mal funcionamiento en varias etapas del sistema. Al final, después de análisis y varias pruebas, concluimos que los microprocesadores no estaban funcionando correctamente. En suma, fue arduo el trabajo, pero dio resultados gracias a la cooperación de todos. Además de todo fue interesante lo ingenioso que se requiere ser para la programación a bajo nivel.

Ramirez Vives José Manuel: Esta práctica realmente fue un poco más sencilla que las anteriores pues ya estamos un poco más acostumbrados a la manera de trabajar este lenguaje de programación y las funciones que podemos utilizar en el microcontrolador. La única complicación fue al momento de generar las salidas adecuadas para formar el número o letra en el display.

Bibliografía

https://www.ingmecafenix.com/electronica/display-de-7-segmentos/
https://www.unioviedo.es/ate/alberto/TEMA3-Ensamblador.pdf
https://staffweb.cms.gre.ac.uk/~sp02/assembly/lecture1.html
https://es.wikipedia.org/wiki/Sistema_hexadecimal

Práctica 4

Introducción:

Se realizará un programa, en el que se mostrará como salida en un display, el valor de un contador que va del 0 al 9, un vez en el nueve regresaremos al 0. Se utilizará un delay de medio segundo entre cada cambio de valor.
Un contador es una variable cuyo valor se incrementa o decrementa en un valor fijo (en cada iteración de un bucle). Suele utilizarse para contar el número de veces que itera un bucle.
El display de 7 segmentos es un dispositivo electrónico que se utiliza para representar visualmente números y algunos caracteres. Este display es muy popular debido a su gran efectividad y simplicidad al momento de utilizarlo.

Material:

- Microcontrolador ATMega8535
- 1 protoboard
- 1 display
- 1 push button
- 1 capacitor de 1 microfaradio a 16 volts.
- Fuente de voltaje 5v
- Jumpers

Desarrollo:

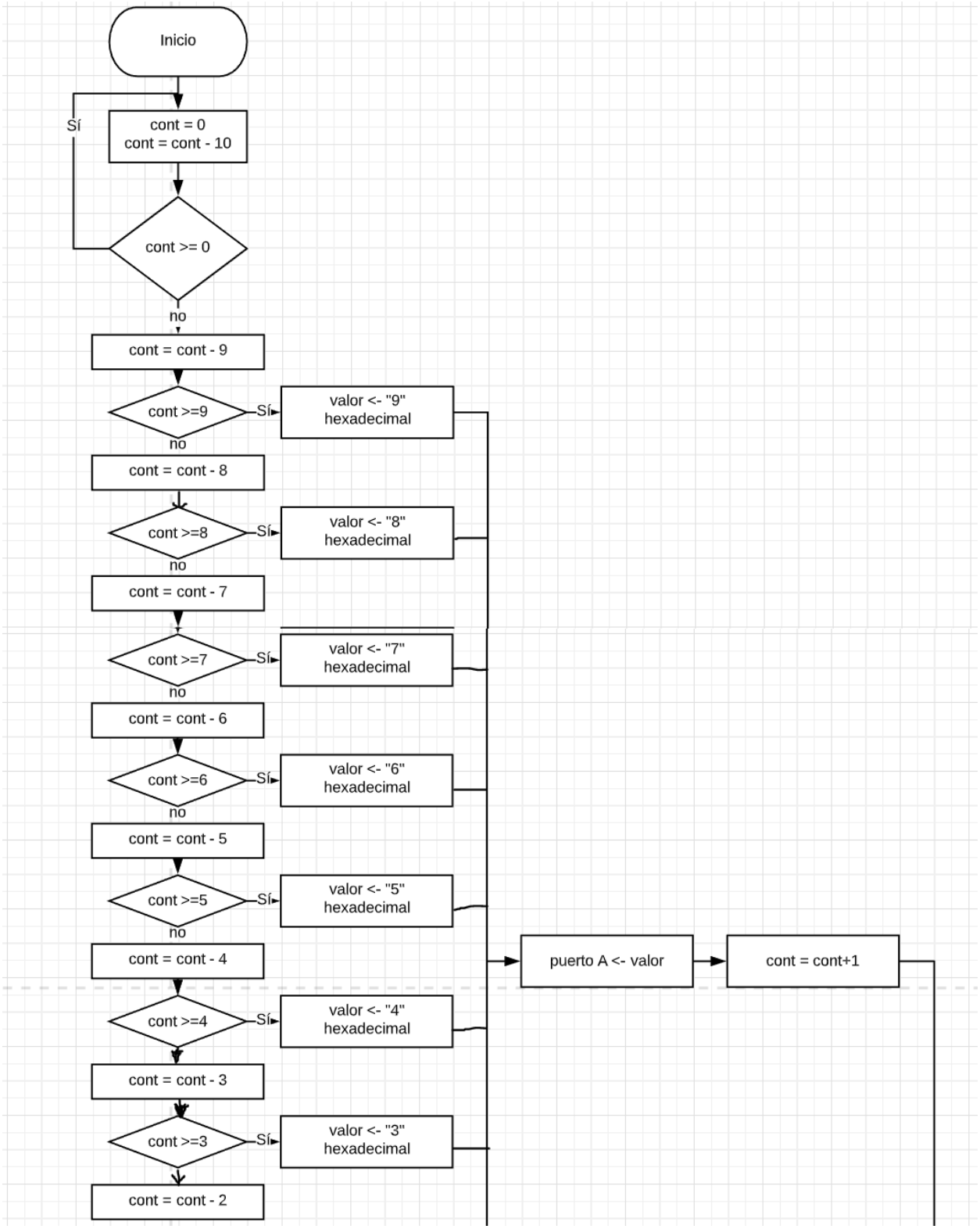
Código:

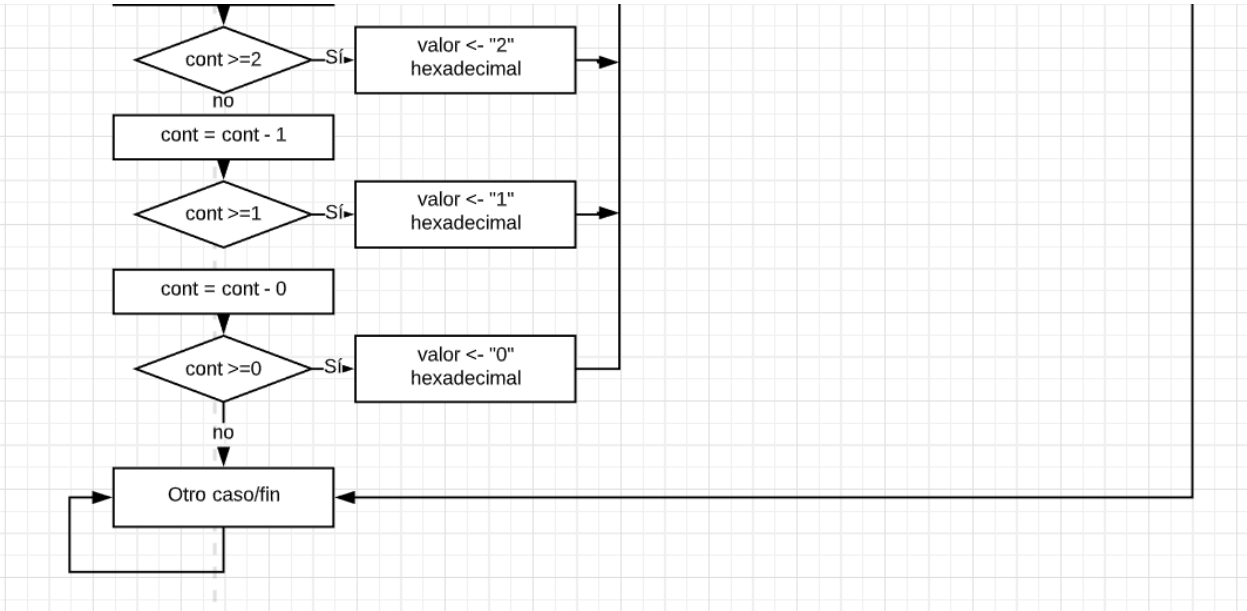
```
.include "m8535def.inc"
.def valor = r20 //vinculamos valor con r20
.def aux = r21 //vinculamos aux con r21
.def cont = r22 //vinculamos cont con r22

ser aux //le asignamos 1's a aux
out ddra, aux //asingamos el puerto A como salida

inic:
    ldi cont,0 // Metemos el valor 0 dentro de cont

otro:

    cpi cont, 10 //hacemos la resta de cont y 10
    brge inic //si el resultado fue mayor o igual a 0, vamos a inic

    rcall delay //mandamos a llamar el delay. lo ponemos aquí porque no
                //nos interesa un delay para cuando el valor es 10

    cpi cont, 9  //hacemos la resta de cont y 9
    brge targetNueve //si el resultado fue mayor o igual a 9, vamos a targetNueve

    cpi cont, 8 //hacemos la resta de cont y 8
    brge targetOcho //si el resultado fue mayor o igual a 8, vamos a targetOcho

    cpi cont, 7 //hacemos la resta de cont y 7
    brge targetSiete //si el resultado fue mayor o igual a 7, vamos a targetSiete

    cpi cont, 6 //hacemos la resta de cont y 6
    brge targetSeis //si el resultado fue mayor o igual a 6,  vamos a targetSeis

    cpi cont, 5 //hacemos la resta de cont y 5
    brge targetCinco //si el resultado fue mayor o igual a 5, vamos a targetCinco

    cpi cont, 4 //hacemos la resta de cont y 4
    brge targetCuatro //si el resultado fue mayor o igual a 4, vamos a targetCuatro

    cpi cont, 3 //hacemos la resta de cont y 3
    brge targetTres //si el resultado fue mayor o igual a 3, vamos a targetTres

    cpi cont, 2 //hacemos la resta de cont y 2
    brge targetDos //si el resultado fue mayor o igual a 2, vamos a targetDos

    cpi cont, 1 //hacemos la resta de cont y 1
    brge targetUno //si el resultado fue mayor o igual a 1, vamos a targetUno
```

```
    cpi cont, 0 //hacemos la resta de cont y 0
    brge targetCero //si el resultado fue mayor o igual a 0, vamos a targetCero

    rjmp otro //saltamos a este mismo tag

targetNueve:
    ldi valor,$6f //le asignamos a valor la representacion de un nueve en display, en hex
    out porta, valor //mandamos al puerto A lo que haya en valor
    inc cont //incrementamos en uno el contador
    rjmp otro //saltamos al tag otro

targetOcho:
    ldi valor,$7f //le asignamos a valor la representacion de un ocho en display, en hex
    out porta, valor //mandamos al puerto A lo que haya en valor
    inc cont //incrementamos en uno el contador
    rjmp otro //saltamos al tag otro

targetSiete:
    ldi valor,$27 //le asignamos a valor la representacion de un siete en display, en hex
    out porta, valor //mandamos al puerto A lo que haya en valor
    inc cont //incrementamos en uno el contador
    rjmp otro //saltamos al tag otro

targetSeis:
    ldi valor,$7d //le asignamos a valor la representacion de un seis en display, en hex
    out porta, valor //mandamos al puerto A lo que haya en valor
    inc cont //incrementamos en uno el contador
    rjmp otro //saltamos al tag otro

targetCinco:
    ldi valor,$6d //le asignamos a valor la representacion de un cinco en display, en hex
    out porta, valor //mandamos al puerto A lo que haya en valor
    inc cont //incrementamos en uno el contador
    rjmp otro //saltamos al tag otro

targetCuatro:
    ldi valor,$66 //le asignamos a valor la representacion de un cuatro en display, en hex
    out porta, valor //mandamos al puerto A lo que haya en valor
    inc cont //incrementamos en uno el contador
    rjmp otro //saltamos al tag otro

targetTres:
    ldi valor,$4f //le asignamos a valor la representacion de un tres en display, en hex
    out porta, valor //mandamos al puerto A lo que haya en valor
    inc cont //incrementamos en uno el contador
```

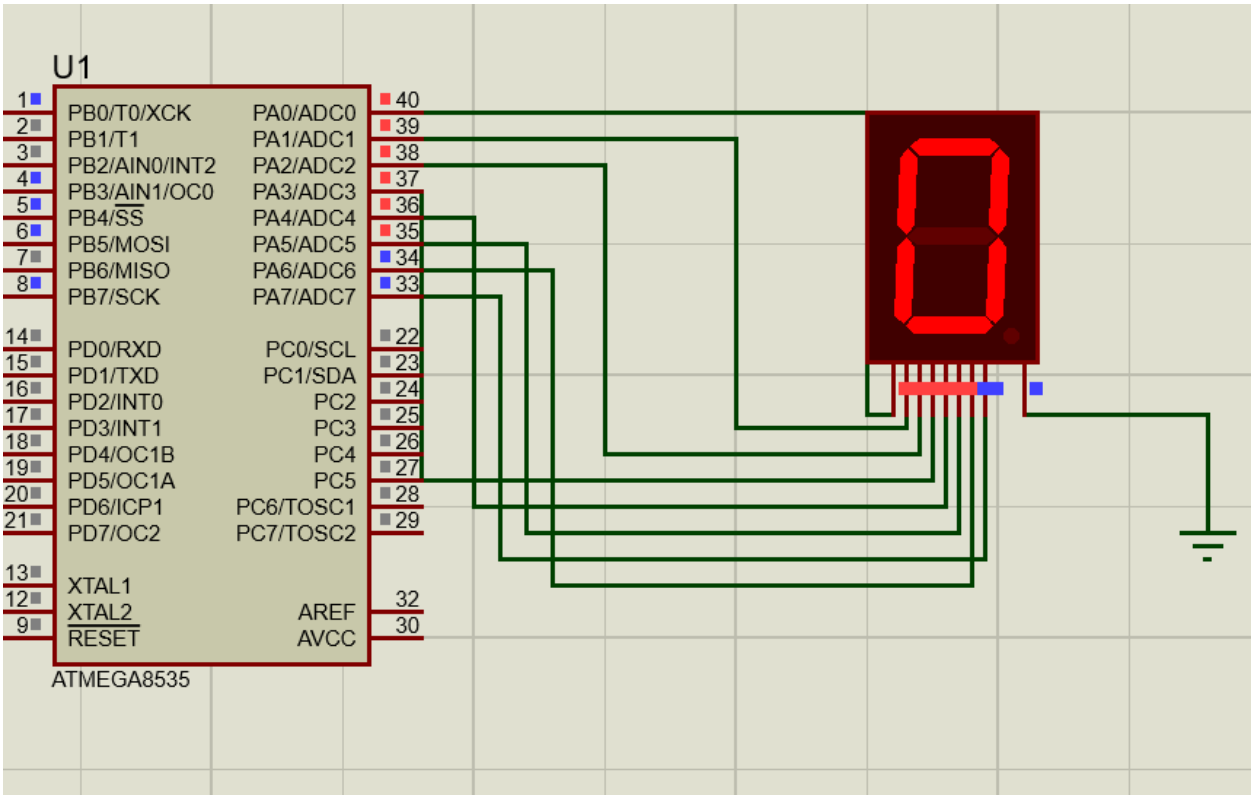
```
106  targetCero:
107      ldi valor,$3f //le asignamos a valor la representacion de un cero en display, en hex
108      out porta, valor //mandamos al puerto A lo que haya en valor
109      inc cont //incrementamos en uno el contador
110      rjmp otro //saltamos al tag otro
111
112
113  delay:
114      push r17
115      push r18
116      push r19
117      ldi r17, $09
118
119  WGLOOP0: ldi r18, $bc
120  WGLOOP1: ldi r19, $c4
121  WGLOOP2: dec r19
122
123      brne WGLOOP2
124      dec r18
125      brne WGLOOP1
126      dec r17
127      brne WGLOOP0
128      nop
129      pop r19
130      pop r18
131      pop r17
132      ret
133
```

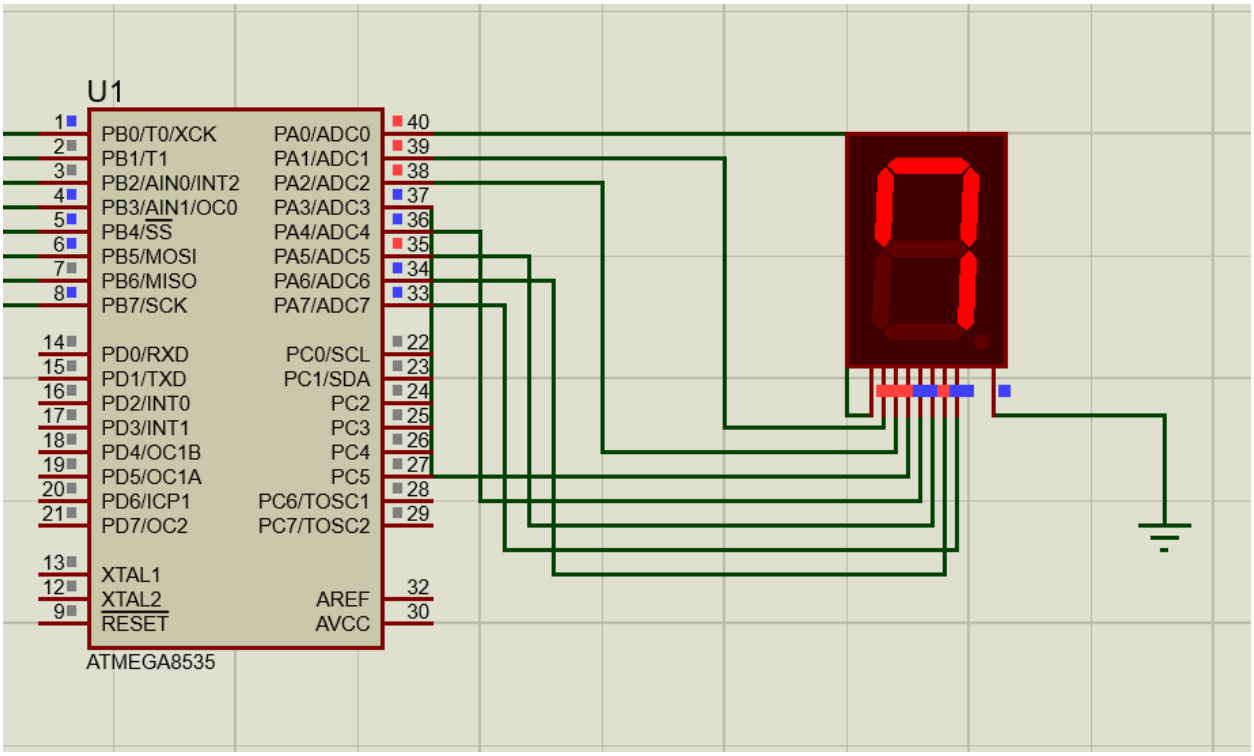
Diagrama de flujo:

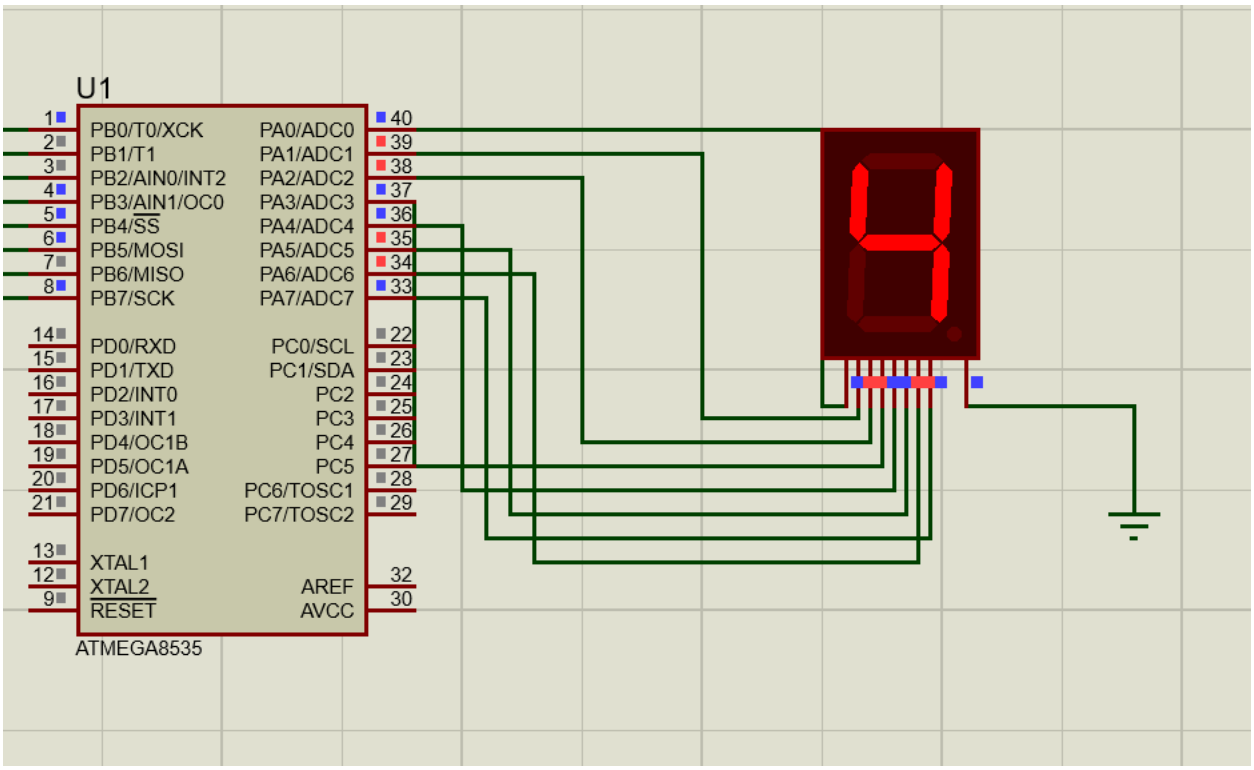


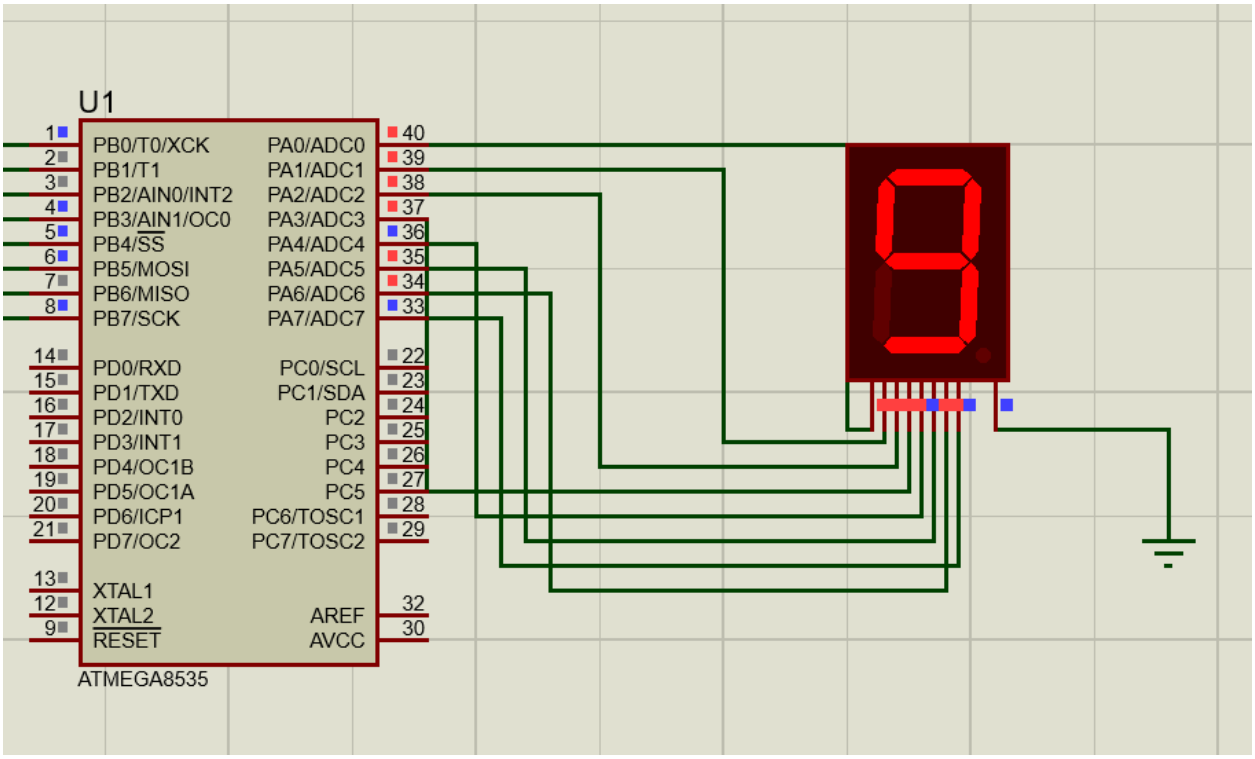

cont >=2
Sí
valor <- "2"
hexadecimal
no
cont = cont - 1
cont >=1
Sí
valor <- "1"
hexadecimal
cont = cont - 0
cont >=0
Sí
valor <- "0"
hexadecimal
no
Otro caso/fin

Simulación:


U1
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
XTAL1
XTAL2
RESET
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PC0/SCL
PC1/SDA
PC2
PC3
PC4
PC5
PC6/TOSC1
PC7/TOSC2
AREF
AVCC
ATMEGA8535


U1
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
XTAL1
XTAL2
RESET
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PC0/SCL
PC1/SDA
PC2
PC3
PC4
PC5
PC6/TOSC1
PC7/TOSC2
AREF
AVCC
ATMEGA8535


U1
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
XTAL1
XTAL2
RESET
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PC0/SCL
PC1/SDA
PC2
PC3
PC4
PC5
PC6/TOSC1
PC7/TOSC2
AREF
AVCC
ATMEGA8535



Conclusiones:

Barriga Vargas Martín Eduardo: La sección más crítica del programa fue el delay, debido a que hacer un contador que aumentara de uno en uno por ciclo y el comparador era una tarea más sencilla, puesto que sólo consistía en mandar al tag correspondiente a imprimir, dependiendo del valor que tuviera el contador.

Mejía Matehuala Alfredo: Desde mi parecer, hasta ahorita ésta ha sido la práctica que mas me ha llamado la atención con su funcionamiento. El código es algo complejo comparado con el resto, pero es fácil de entender y reproducir. Ver encender los displays con información que se introdujo en un “idioma” completamente diferente y ver salir algo que

es tan fácil entender para nosotros fue gratificante. Crear el diagrama de flujo fue interesante ya que el programa es muy poco lineal como suelen ser los códigos anteriores.

Ramirez Vives José Manuel: Con esta práctica pudimos ver una manera un poco rudimentaria para generar un delay para poder observar mejor los resultados de las operaciones o sencillamente algo que se quiere mostrar, como en este caso. El delay utilizado fue el generado con la calculadora sugerida por el profesor que realiza muchas operaciones en ciclos para ocupar el tiempo deseado.

Bibliografía:

http://www.carlospes.com/minidiccionario/variable_contador.php
https://www.ingmecafenix.com/electronica/display-de-7-segmentos/
https://www.unioviedo.es/ate/alberto/TEMA3-Ensamblador.pdf
https://staffweb.cms.gre.ac.uk/~sp02/assembly/lecture1.html
https://es.wikipedia.org/wiki/Sistema_hexadecimal