



# INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y  
TECNOLOGÍAS AVANZADAS

## “Práctica 0: Introducción al entorno AVR Studio”

Asignatura: Microprocesadores, Microcontroladores e Interfaz

Docente: Trejo Salazar David Benjamín

Equipo 2:

- Aguilar Zarazúa Luis Fernando
- Martínez Pérez Nestor



Grupo: 2MM6  
Fecha de entrega: 10/04/2016

# ÍNDICE

1.-INTRODUCCIÓN .....	1
2.- OBJETIVOS.....	1
2.1.- GENERAL .....	1
2.2.- PARTICULARES.....	1
3.- MARCO TEÓRICO .....	1
3.1.- ATMEL STUDIO.....	1
3.2.- ATMEGA328P .....	1
3.3.- OPERACIONES ARITMÉTICO LÓGICAS.....	2
3.3.1.- AND .....	2
3.3.2.- OR.....	2
3.3.3.- SUMA.....	3
3.3.4.- RESTA .....	3
4.- MATERIAL .....	4
5.- DESARROLLO EXPERIMENTAL .....	4
5.1.- CONTROL DE ENCENDIDO Y APAGADO DE LEDS .....	4
5.1.1.- DIAGRAMA DE FLUJO .....	4
5.1.2.- DIAGRAMA ELÉCTRICO .....	5
5.1.3.- PROCEDIMIENTO.....	5
5.2.- OPERACIONES ARITMÉTICAS LÓGICAS Y BANDERAS H, S Y Z .....	8
5.2.1.- DIAGRAMA DE FLUJO .....	8
5.1.2.- DIAGRAMA ELÉCTRICO .....	10
5.2.3.- PROCEDIMIENTO.....	10
6.- RESULTADOS .....	12
6.1.- CONTROL DE ENCENDIDO Y APAGADO DE LEDS .....	12
6.2.- OPERACIONES ARITMÉTICAS LÓGICAS Y BANDERAS H, S Y Z .....	13
7.- CONCLUSIONES.....	16
- Zarazúa Aguilar Luis Fernando .....	16
- Martínez Pérez Nestor .....	16
8.- BIBLIOGRAFÍA.....	17

## ÍNDICE DE FIGURAS

<b>Figura 1.-</b> Tabla de verdad de la compuerta AND.....	2
<b>Figura 2.-</b> Tabla de verdad de la compuerta OR. ....	2
<b>Figura 3.-</b> Tabla de verdad del semirestador.....	3
<b>Figura 4.-</b> Circuito a realizar en el laboratorio. ....	5
<b>Figura 5.-</b> Circuito realizado en el laboratorio. ....	6
<b>Figura 6.-</b> Código escrito en Atmel Studio.....	7
<b>Figura 7.-</b> Localización de la opción construir. ....	7
<b>Figura 8.-</b> Selección del archivo .hex para simularlo en proteus.....	8
<b>Figura 9.-</b> Segmento de código del segundo ejercicio. ....	11
<b>Figura 10.-</b> Simulación en proteus. ....	12
<b>Figura 11.-</b> Prueba 1 del circuito.....	13
<b>Figura 12.-</b> Prueba 2 del circuito.....	13
<b>Figura 13.-</b> Operación resta 1-1 simulada.....	14
<b>Figura 14.-</b> Operación resta 0-1 simulada.....	14
<b>Figura 15.-</b> Operación resta 1-1 sobre el circuito.....	15
<b>Figura 16.-</b> Operación resta 0-1 sobre el circuito.....	15
<b>Figura 17.-</b> Operación or entre 0 y 0 sobre el circuito.....	15
<b>Figura 18.-</b> Operación or entre F y F sobre el circuito.....	16

## **1.-INTRODUCCIÓN**

En esta práctica se realizaron las primeras aproximaciones al software Atmel Studio en donde se programaron instrucciones en lenguaje ensamblador. Las instrucciones programadas permitían realizar el control del encendido y apagado de leds por medio de DIP switches y también se programaron las operaciones como la suma, la resta, y también operaciones lógicas como la and y la or además de utilizar las banderas H, S y Z que nos ofrece el microcontrolador para mostrar cuales se habían encendido y cuales no.

## **2.- OBJETIVOS**

### **2.1.- GENERAL**

1. Que el alumno conozca la herramienta de desarrollo AVR Studio para proyectos en el microcontrolador de ATMEL ATmega328P, y aprender a simular dichos programas en la herramienta de simulación con que cuenta el AVR Studio así como programar el ATmega328P.

### **2.2.- PARTICULARES**

1. Programar correctamente las instrucciones.
2. Armar correctamente los circuitos eléctricos.

## **3.- MARCO TEÓRICO**

### **3.1.- ATMEL STUDIO**

Atmel Studio 7 es la plataforma de desarrollo integrado (IDP) para desarrollar y depurar aplicaciones de SMART Atmel® Atmel AVR® microcontrolador (MCU) ARM basado-y. Studio 7 es compatible con todos los AVR de Atmel y MCUs de SMART. El Atmel Studio IDP le da un ambiente transparente y fácil de usar para escribir, generar y depurar sus aplicaciones escritas en C / C ++ o código ensamblador. Además, se conecta sin problemas a los depuradores Atmel y kits de desarrollo.

Además, Atmel Studio 7 incluye Gallery, una tienda de aplicaciones en línea que le permite ampliar su entorno de desarrollo con plug-ins desarrollados por Atmel, así como por la herramienta de terceros y proveedores de software embebidos. Atmel Studio 7 puede también importar sus bocetos Arduino como proyectos de C ++.

### **3.2.- ATMEGA328P**

El Atmega328P AVR 8-bit es un Circuito integrado de alto rendimiento que está basado un microcontrolador RISC, combinando 32 KB ISPflash una memoria con la capacidad de leer-mientras-escribe, 1 KB de memoria EEPROM, 2 KB de SRAM, 23 líneas de E/S de propósito general, 32 registros de proceso general, tres temporizadores flexibles/contadores con modo de comparación, interrupciones internas y externas, programador de modo USART, una interface serial orientada a byte de 2 cables, SPI puerto serial, 6-canales 10-bit Conversor A/D (8-canales en TQFP y QFN/MLF packages), "watchdog timer" programable con oscilador interno, y cinco modos de ahorro de energía seleccionables por software. El dispositivo opera entre 1.8 y 5.5 voltios. Por medio de la ejecución de poderosas instrucciones en un solo ciclo de reloj, el dispositivo alcanza una respuesta de 1 MIPS, balanceando consumo de energía y velocidad de proceso.

### 3.3.- OPERACIONES ARITMÉTICO LÓGICAS

#### 3.3.1.- AND

La puerta AND es una de las puertas básicas con la que se construyen todas las funciones lógicas. Una puerta AND puede tener dos o más entradas y una única salida y realiza la operación que se conoce como multiplicación lógica.

La puerta AND genera una salida a nivel ALTO sólo cuando todas las entradas están a nivel ALTO. Cuando cualquiera o todas las entradas están a nivel BAJO, la salida se pone a nivel BAJO.

La multiplicación booleana es lo mismo que la función AND.

En la **figura 1** se muestra la tabla de verdad de una compuerta AND.

Entradas		Salida
<i>A</i>	<i>B</i>	<i>X</i>
0	0	0
0	1	0
1	0	0
1	1	1
1 = ALTO, 0 = BAJO		

**Figura 1.-** Tabla de verdad de la compuerta AND.

#### 3.3.2.- OR

Una puerta OR puede tener dos o más entradas y realiza la operación que se conoce como suma lógica.

Una puerta OR genera un nivel ALTO a la salida cuando cualquiera de sus entradas está a nivel ALTO. La salida se pone a nivel BAJO sólo cuando todas las entradas están a nivel BAJO.

La suma booleana es lo mismo que la función OR.

En la **figura 2** se muestra la tabla de verdad de la puerta OR.

Entradas		Salida
<i>A</i>	<i>B</i>	<i>X</i>
0	0	0
0	1	1
1	0	1
1	1	1
1 = ALTO, 0 = BAJO		

**Figura 2.-** Tabla de verdad de la compuerta OR.

### 3.3.3.- SUMA

El proceso a seguir en la realización de una suma binaria es muy similar al empleado habitualmente en la aritmética decimal. Sean A y B dos informaciones binarias de n bits.

$$A = A_{n-1} A_{n-2} \dots A_0 \quad B = B_{n-1} B_{n-2} \dots B_0$$

La suma de  $A + B$  será un proceso de n sumas parciales, comenzando por los bits de menor peso. El resultado de esta operación  $A_0 + B_0$  será un bit de suma  $S_0$  y un bit de acarreo  $C_0$  que se propagará como elemento integrante de la suma, en los siguientes dos bits, en orden ascendente de peso  $A_1 + B_1$ . A partir de este punto el resultado de todas las operaciones incorporarán un bit de suma y un bit de acarreo que se propagará como elemento integrante de la suma en los dos bits siguientes y así sucesivamente. El resultado será:

$$S = C_{n-1} S_{n-1} S_{n-2} \dots S_0$$

### 3.3.4.- RESTA

A continuación se expresa la tabla de verdad del semirestador como función  $R = A - B$ .

Entradas		Salidas	
B	A	C	R
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

**Figura 3.-** Tabla de verdad del semirestador.

A partir de la **figura 3** se obtienen las funciones lógicas R y C, que se muestran a continuación:

$$R = \bar{B}A + B\bar{A} = B \oplus A$$

$$C = \bar{B}A$$

Este sumador no se emplea en la práctica pues es preferible realizar una codificación de números con signo y emplear sumadores basados en el principio:

$$A - B = A + (-B)$$

Para la conversión del operador B a un valor con signo se puede emplear la codificación en complemento a uno o complemento a dos, siendo esta última la más utilizada en la práctica.

#### 4.- MATERIAL

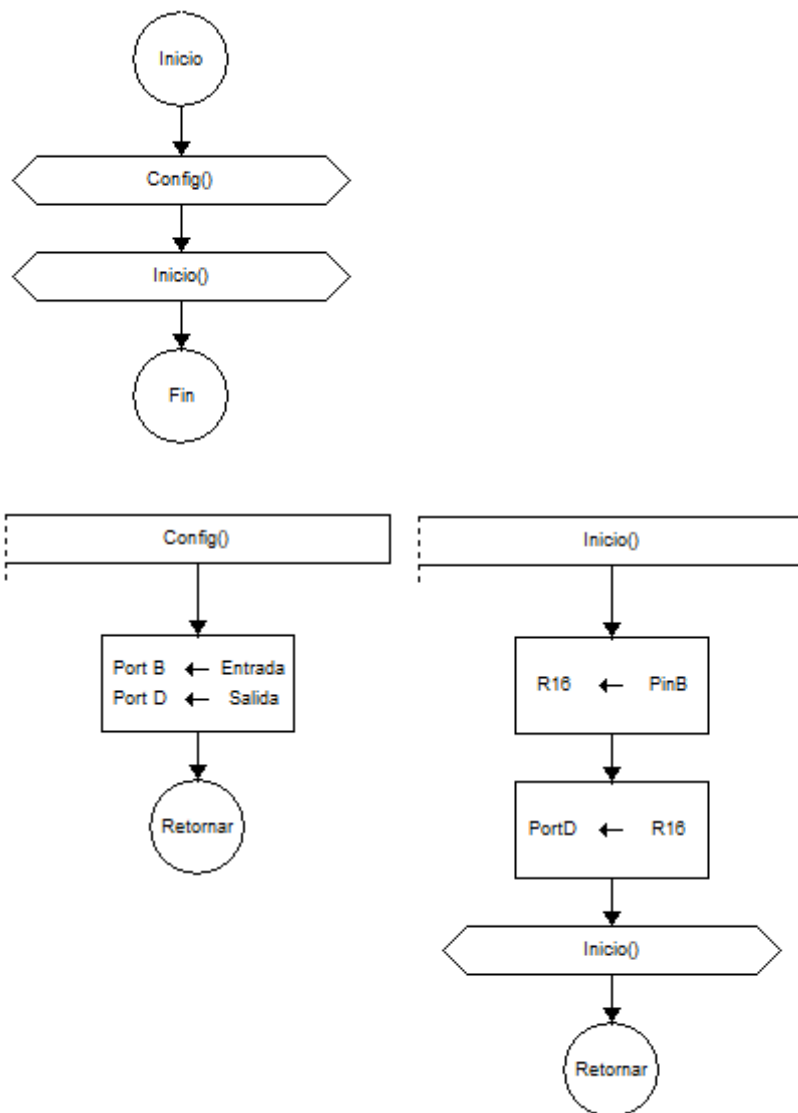
1. Fuente de alimentación.
2. ATmega328P.
3. Circuito eléctrico.
4. Programador.
5. PC.

#### 5.- DESARROLLO EXPERIMENTAL

##### 5.1.- CONTROL DE ENCENDIDO Y APAGADO DE LEDS

##### 5.1.1.- DIAGRAMA DE FLUJO

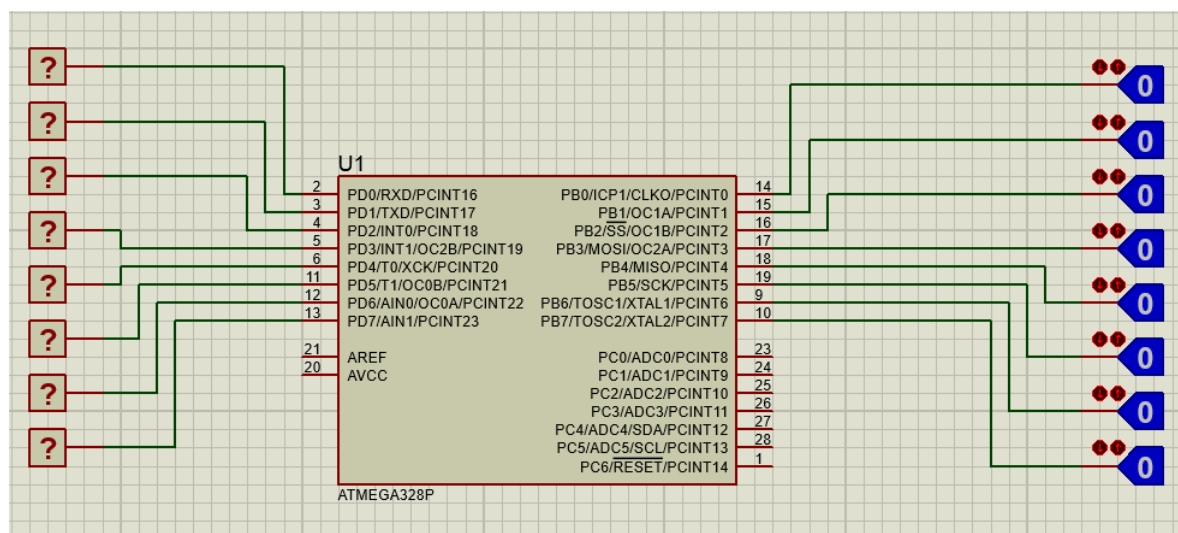
A continuación se presenta el diagrama de flujo correspondiente al primer ejercicio de la práctica cero.



Como se observa en el diagrama de flujo se tienen solo dos subprogramas uno de los cuales se encarga de la configuración de los puertos de entrada y salida y el otro subprograma es el encargado de asignar los valores del puerto de entrada a un registro y de asignar el valor del registro a los puertos de salida para así poder realizar el control de los leds, el subprograma con el nombre de inicio se queda encerrado en un ciclo por lo que se estará ejecutando indefinidamente con lo que garantizamos que las salidas responderán a cada una de las entradas que se proporcionen en cualquier momento.

### 5.1.2.- DIAGRAMA ELÉCTRICO

A continuación se presenta el diagrama del circuito a realizar en el laboratorio.



**Figura 4.-** Circuito a realizar en el laboratorio.

Como se observa en la **figura 4** se utilizaron en la simulación probadores y estados lógicos para simplificar el circuito simulado pero en la práctica se conectaron leds y DIP switches para observar el comportamiento de esta primera parte de la práctica.

### 5.1.3.- PROCEDIMIENTO

1. Armar el circuito en un protoboard.
2. Abrir el AVR Studio, inicializar un proyecto y escribir el siguiente código, grabar el código con la extensión .ASM.



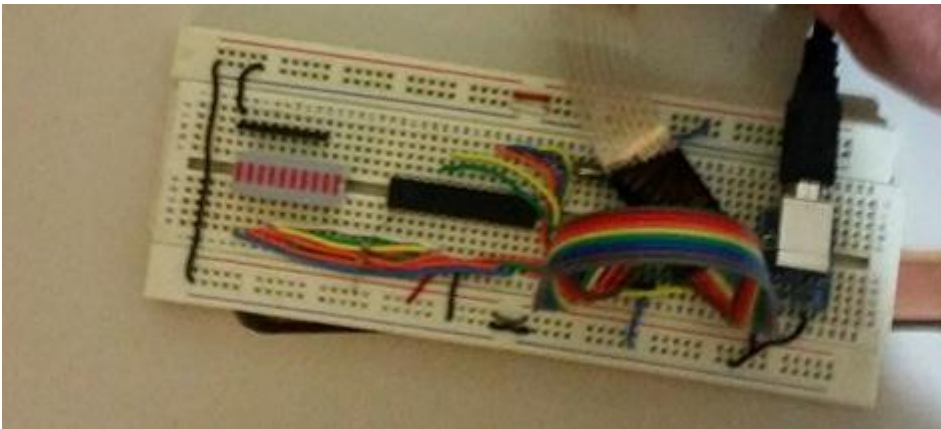
```

.include "m328pdef.inc"
.def temp=R16;
.org 0x0000 jmp config;
.org 0x3C00 jmp config;
config:
    LDI temp,0x00;
    out DDRB,temp;
    LDI temp,0xFF;
    out DDRD,temp;
inicio:
    in temp,PINB;
    out PORTD,temp;
    jmp inicio;

```

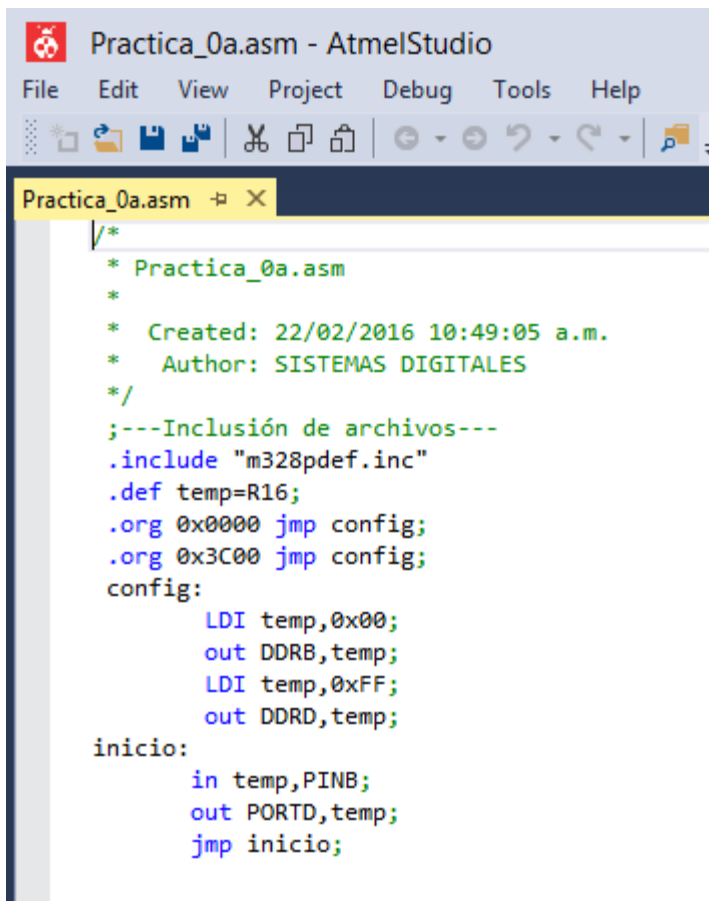
3. Compilar el programa y simularlo.
4. Grabar el programa en el ATmega328P.
5. Simular el circuito con la ayuda del programa PROTEUS.

Para iniciar con la práctica, primero se armó el circuito que a continuación se muestra:



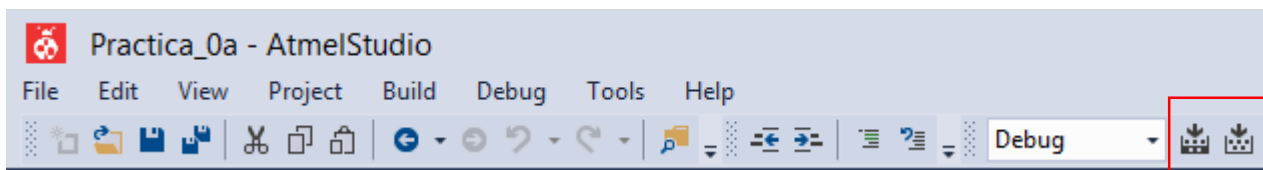
**Figura 5.-** Circuito realizado en el laboratorio.

Una vez armado el circuito de la **figura 5** se procedió a crear un nuevo proyecto en el software Atmel Studio en el cual se escribió el código sugerido en la práctica.



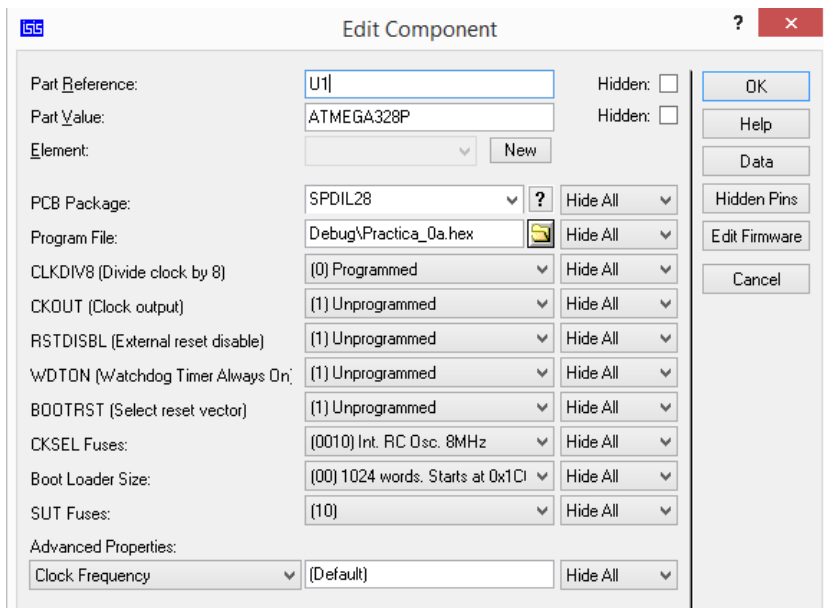
**Figura 6.-** Código escrito en Atmel Studio.

Una vez escrito el código en Atmel Studio se seleccionó la opción de construir para generar el archivo .hex.



**Figura 7.-** Localización de la opción construir.

Después de generar el archivo .hex, se procedió a cargar este archivo al circuito simulado en proteus para lo cual basta con dar doble clic sobre el microcontrolador simulado y se abrirá una ventana en la cual seleccionaremos el archivo que deseamos montar, en este caso es el correspondiente al de la práctica 0a tal como se muestra en la **figura 8**.



**Figura 8.-** Selección del archivo .hex para simularlo en proteus.

Los resultados obtenidos en la simulación se presentan en la sección de resultados de esta práctica en el apartado que corresponde al primer ejercicio de la práctica.

Después de corroborar que la programación era la correcta se procedió a cargar el archivo .hex al microcontrolador por lo que se utilizó, para este fin, el software extreme Burner y el programador usbasp teniendo cuidado de conectar correctamente cada una de las terminales del microcontrolador.

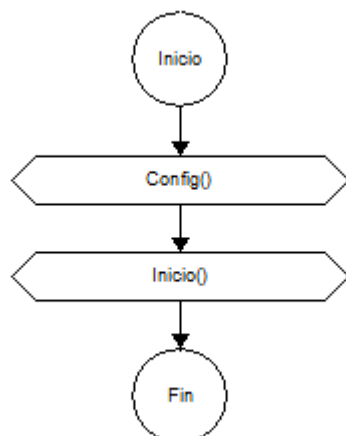
En este paso se colocó el microcontrolador de nuevo en el circuito armado para probar su correcto funcionamiento alimentando el circuito con una fuente de 5 V y obteniendo resultados positivos en el funcionamiento del circuito.

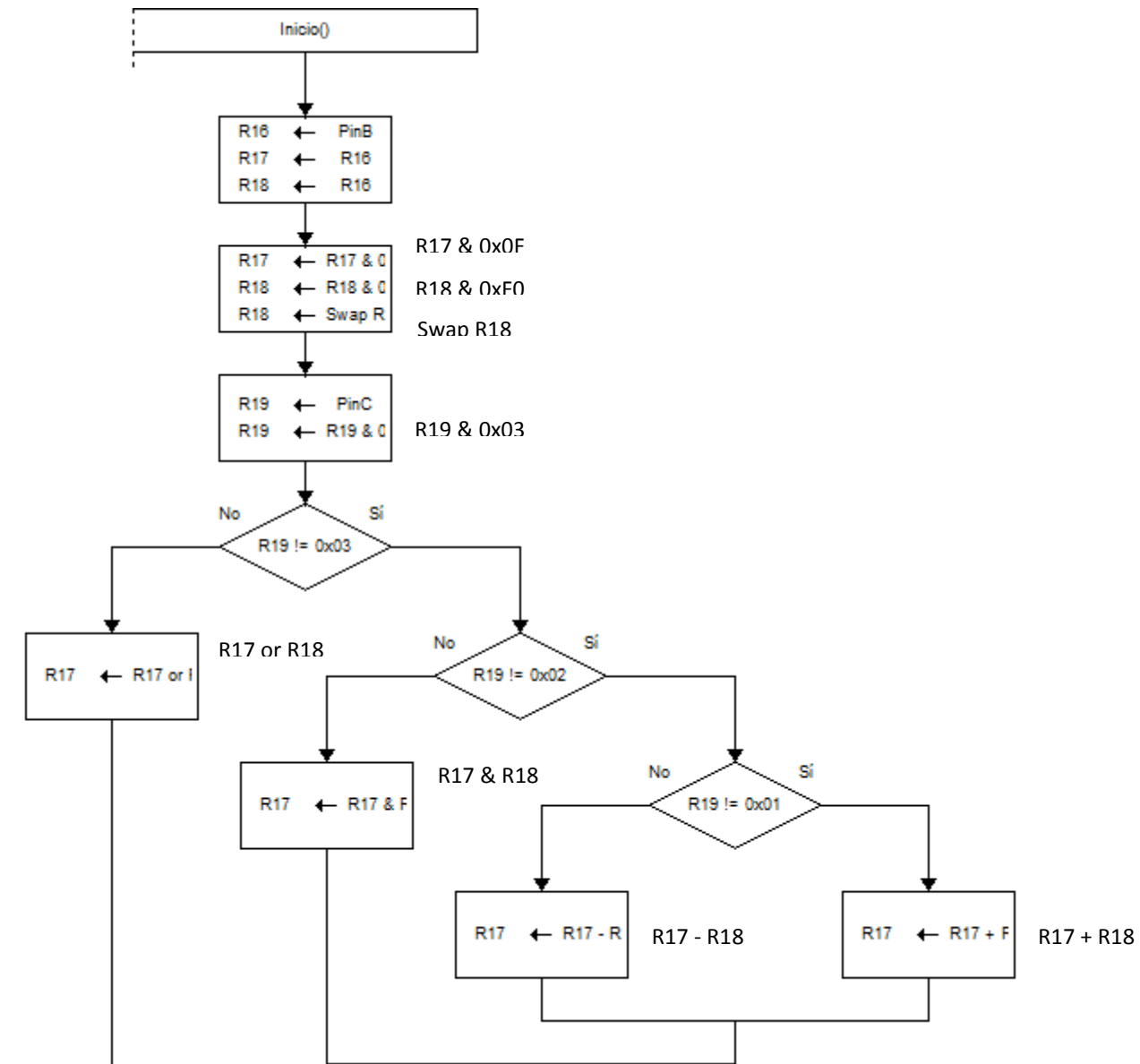
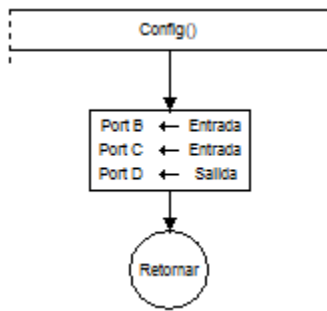
Los resultados obtenidos en este primer ejercicio de la práctica están documentados en la sección de resultados en su apartado correspondiente para el primer ejercicio.

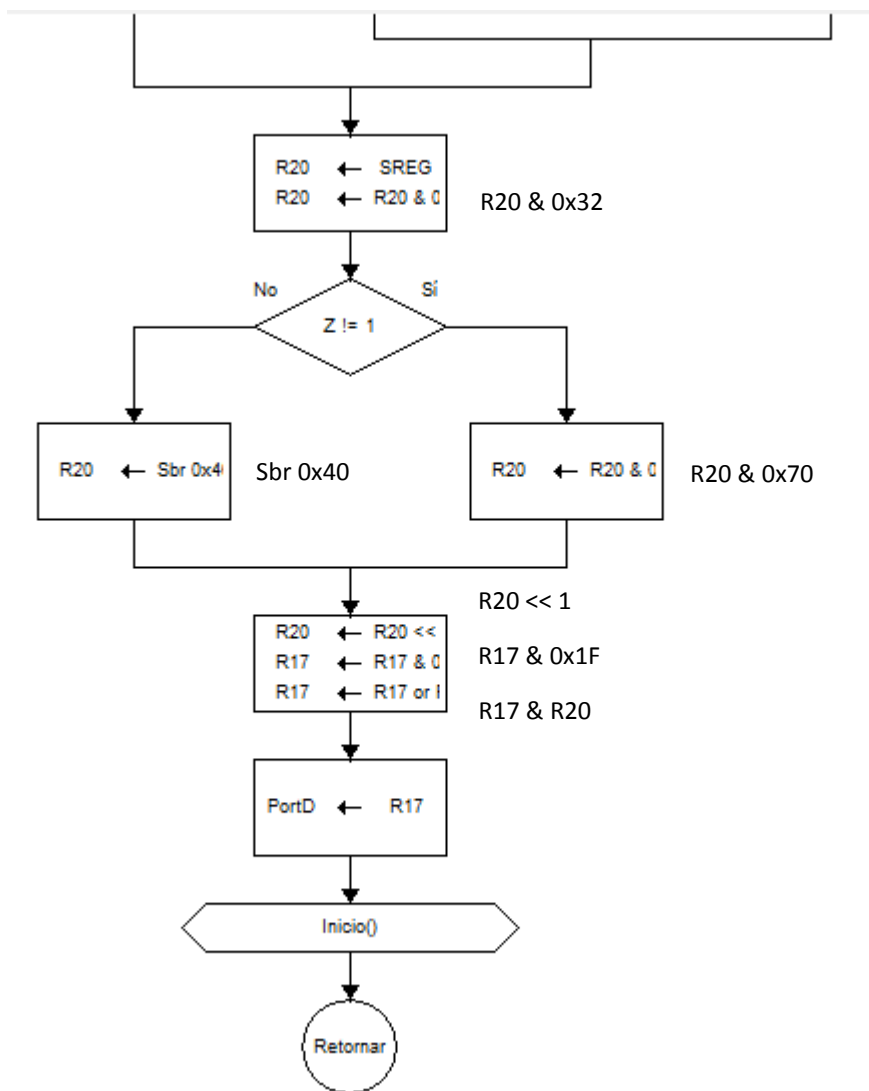
## **5.2.- OPERACIONES ARITMÉTICAS LÓGICAS Y BANDERAS H, S Y Z**

### **5.2.1.- DIAGRAMA DE FLUJO**

A continuación se presenta el diagrama de flujo correspondiente al segundo ejercicio de la práctica cero.







El diagrama de flujo consta del programa principal en el cual se llaman dos subprogramas que se encargan de realizar todas las operaciones necesarias para poder observar los correspondientes resultados en los indicadores leds conectados.

Esas fueron todas las rutinas y subrutinas correspondientes a este ejercicio de la práctica 0 pero debido al software utilizado, algunos datos no quedaron visibles por lo que se colocaron cuadros de texto que aclararan los valores y operaciones que no estuvieran visibles.

### 5.1.2.- DIAGRAMA ELÉCTRICO

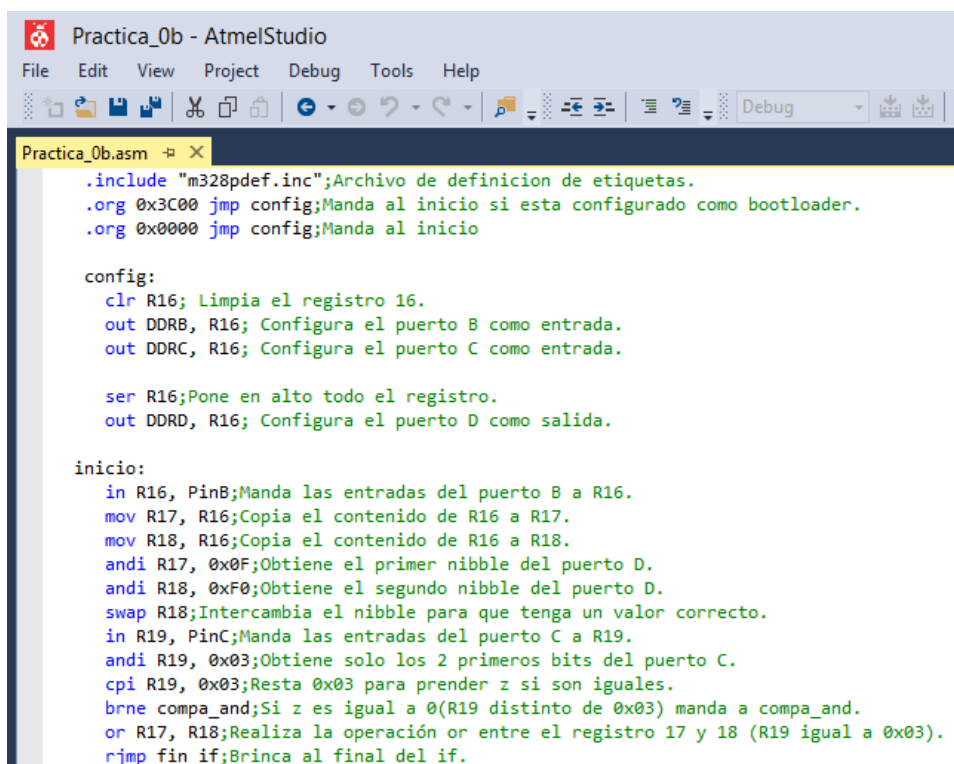
Es necesario mencionar que para el segundo ejercicio de la práctica se utilizó el mismo circuito utilizado en el ejercicio 1 por lo que el referente del circuito eléctrico para este segundo ejercicio es el mostrado en la **figura 4**.

### 5.2.3.- PROCEDIMIENTO

1. Realizar un programa donde comprobemos las operaciones aritméticas lógicas básicas (suma, resta, and y or) entre dos nibbles (configurar un puerto D como entrada) y mostrar la salida con ayuda de leds en el puerto B, también mostrar el estado de las banderas H,S y Z.
2. En este segundo ejercicio repetir los pasos 2 al 5 del ejercicio 1.

Para este segundo ejercicio se utilizó el mismo circuito eléctrico que se utilizó en el ejercicio uno por lo tanto el circuito eléctrico referente se encuentra en la **figura 5**. Cabe aclarar también que los procedimientos para generar los archivos y simular los circuitos son exactamente los mismos que en el ejercicio 1 con la salvedad de que el nombre que se le dio a estos nuevos archivos es practica\_0b por lo que se omitirá la explicación de todos los pasos que pueden ser consultados en el procedimiento del ejercicio 1.

A continuación se muestra un segmento del código que se escribió en el software Atmel Studio



```

Practica_0b - AtmelStudio
File Edit View Project Debug Tools Help
Practica_0b.asm
.include "m328pdef.inc"; Archivo de definicion de etiquetas.
.org 0x3C00 jmp config; Manda al inicio si esta configurado como bootloader.
.org 0x0000 jmp config; Manda al inicio

config:
    clr R16; Limpia el registro 16.
    out DDRB, R16; Configura el puerto B como entrada.
    out DDRC, R16; Configura el puerto C como entrada.

    ser R16; Pone en alto todo el registro.
    out DDRD, R16; Configura el puerto D como salida.

inicio:
    in R16, PinB; Manda las entradas del puerto B a R16.
    mov R17, R16; Copia el contenido de R16 a R17.
    mov R18, R16; Copia el contenido de R16 a R18.
    andi R17, 0x0F; Obtiene el primer nibble del puerto D.
    andi R18, 0xF0; Obtiene el segundo nibble del puerto D.
    swap R18; Intercambia el nibble para que tenga un valor correcto.
    in R19, PinC; Manda las entradas del puerto C a R19.
    andi R19, 0x03; Obtiene solo los 2 primeros bits del puerto C.
    cpi R19, 0x03; Resta 0x03 para prender z si son iguales.
    brne compa_and; Si z es igual a 0 (R19 distinto de 0x03) manda a compa_and.
    or R17, R18; Realiza la operación or entre el registro 17 y 18 (R19 igual a 0x03).
    rjmp fin_if; Brinca al final del if.
  
```

**Figura 9.-** Segmento de código del segundo ejercicio.

Para este ejercicio se ocuparon más instrucciones porque ahora había que realizar enmascaramientos y comparaciones para saber que operación se había seleccionado también para saber que bandera se había activado y para seleccionar solo las banderas que nos interesaba.

Con el fin de poder seleccionar la operación que se desea realizar, se agregaron dos DIP switches con los que se puede tener las 4 combinaciones que definan a cada operación obedeciendo a las siguientes reglas propuestas por nosotros:

Suma:	00
Resta:	01

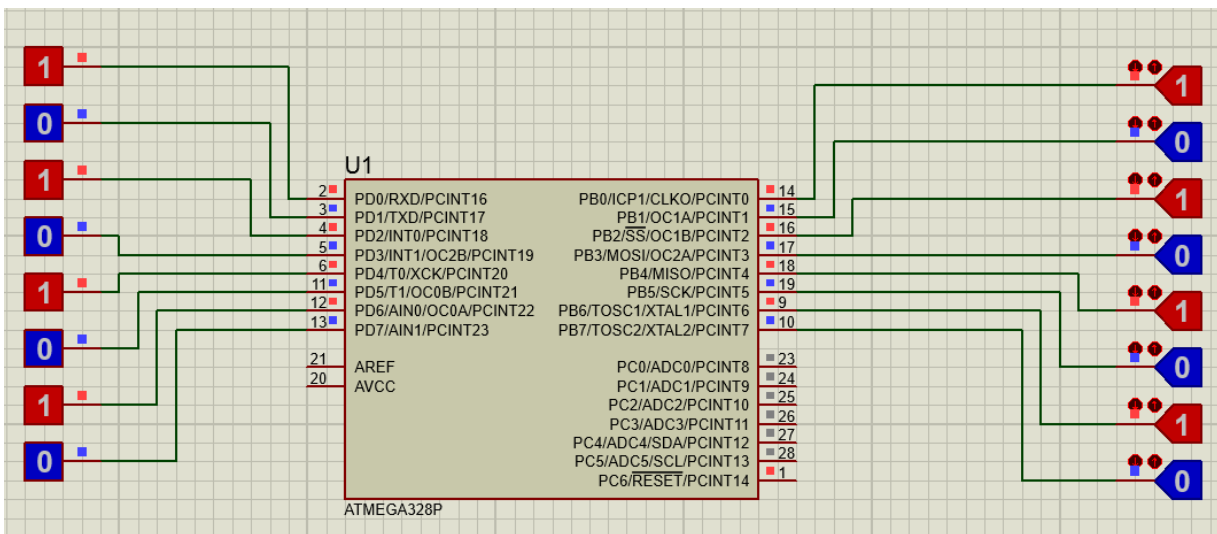
Multiplicación: 10  
División: 11

Los resultados obtenidos tanto en este ejercicio como en el primero se encuentran en sus respectivos apartados en la sección de resultados.

## 6.- RESULTADOS

### 6.1.- CONTROL DE ENCENDIDO Y APAGADO DE LEDS

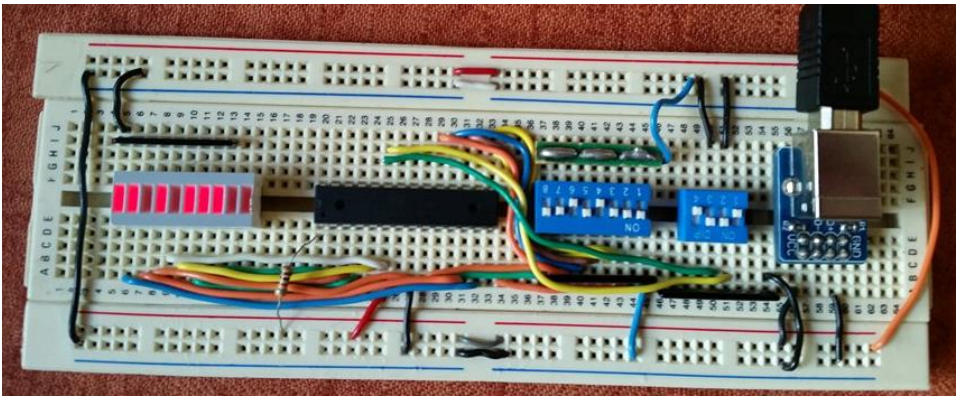
Para comprobar que la programación era la correcta, se simuló el comportamiento que tendría el microcontrolador en proteus y tal como se ve a continuación la programación fue correcta.



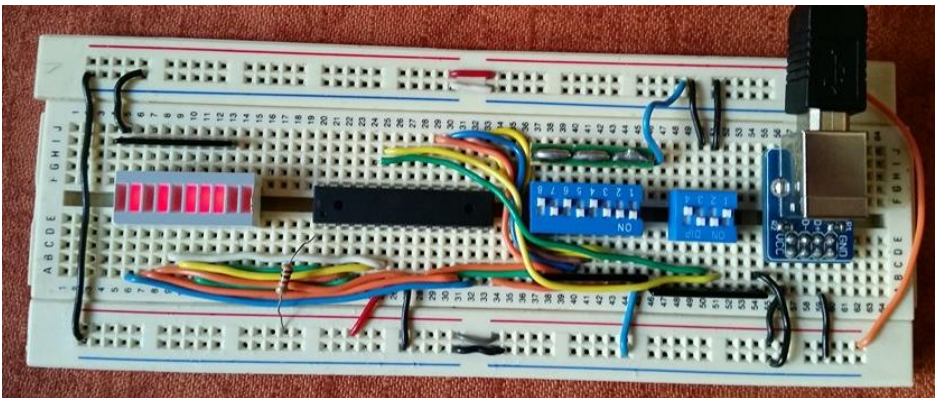
**Figura 10.-** Simulación en proteus.

Como se ve en la **figura 10** el circuito funcionó correctamente pues cada uno de los indicadores lógicos funciona de acuerdo a las entradas lógicas proporcionadas.

En las **figuras 11 y 12** se puede observar el correcto funcionamiento del circuito ya que dependiendo de las entradas proporcionadas por los DIP switches, se encenderán sus respectivos leds indicadores.



**Figura 11.-** Prueba 1 del circuito.



**Figura 12.-** Prueba 2 del circuito.

Tanto en la **figura 11** y como en la **figura 12** los leds funcionales son 8 contando de izquierda a derecha y los DIP switches que los controlan son los del primer bloque de izquierda a derecha.

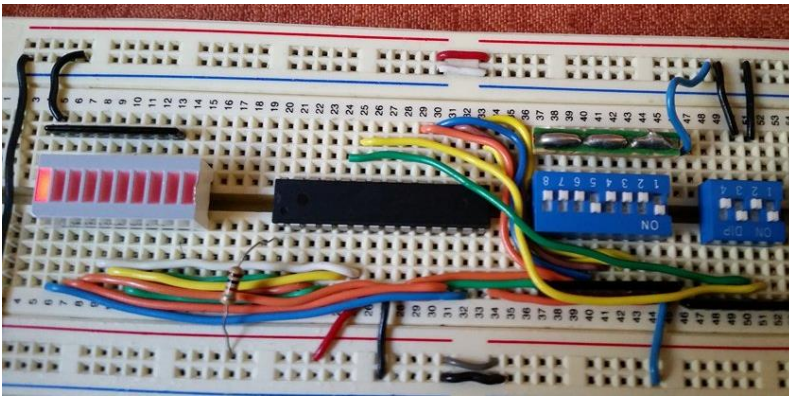
## 6.2.- OPERACIONES ARITMÉTICAS LÓGICAS Y BANDERAS H, S Y Z

Para comprobar que la programación era la correcta, se simuló el comportamiento que tendría el microcontrolador en proteus y tal como se ve a continuación la programación fue correcta.

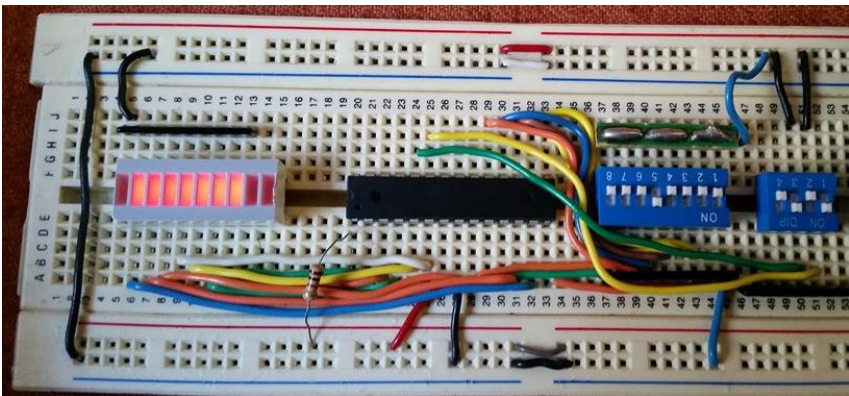
En las **figuras 13** y **14** los estados lógicos están colocados en orden descendente, divididos en 2 nibbles comenzando por el bit menos significativo. Después de los 8 bits con los cuales se llevaran a cabo las operaciones se colocaron dos estados lógicos más para seleccionar la operación que se desee:



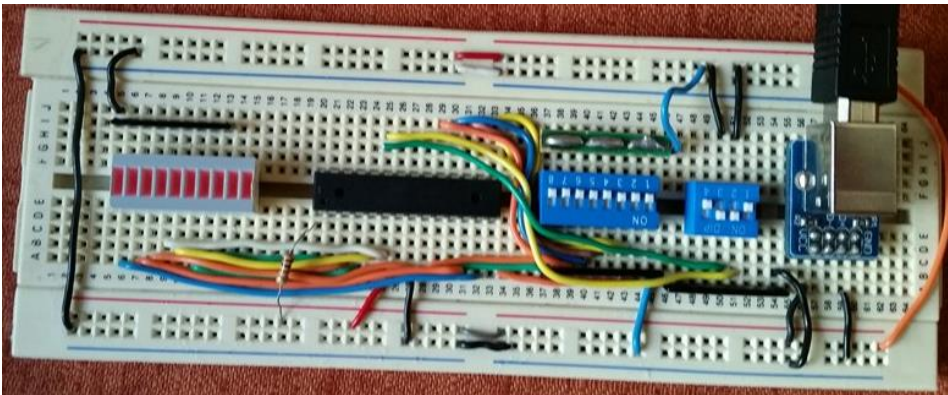




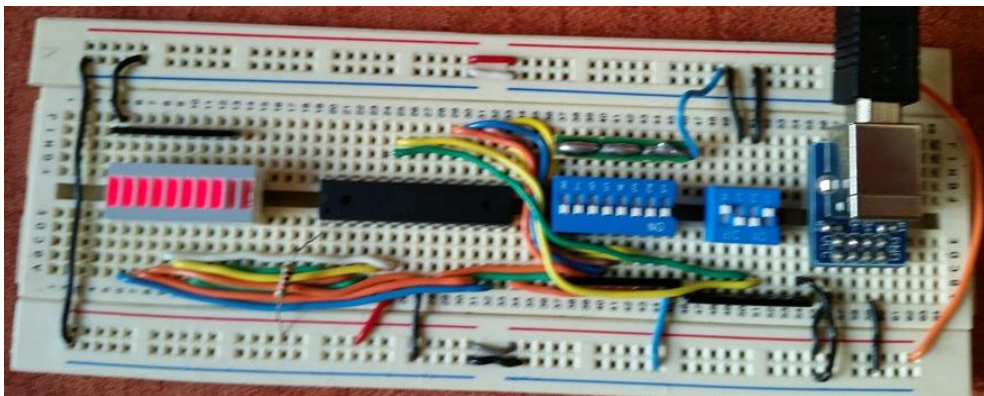
**Figura 15.-** Operación resta 1-1 sobre el circuito.



**Figura 16.-** Operación resta 0-1 sobre el circuito.



**Figura 17.-** Operación or entre 0 y 0 sobre el circuito.



**Figura 18.-** Operación or entre F y F sobre el circuito.

En las **figuras** de la **15** a la **18** se puede observar cómo funciona el circuito armado y como el microcontrolador envía las señales de salida correctas en cada operación que realiza así como también se observa el estado de las banderas que posee el microcontrolador.

## 7.- CONCLUSIONES

### - Zarazúa Aguilar Luis Fernando

El microcontrolador ATMEGA 328P es un circuito integrado que nos ofrece demasiadas ventajas a la hora de usarlo, ya que entre otras cosas es un circuito reprogramable que posee varios registros que son de utilidad a la hora de hacer operaciones aritméticas o lógicas que nos ayudan en un proceso, como lo es el registro de banderas en cual la bandera que más destaca es la bandera Z, ya que se usa prácticamente para cualquier comparación.

Por otra parte el lenguaje que se usa en el microcontrolador nos da la opción de ver su estructura inicial y con qué hardware interno cuenta, dándonos así un mejor panorama al nivel que puede llegar y hasta donde se puede usar, siendo así un elemento muy útil para la ingeniería ya que con un bajo costo podemos realizar un procesamiento básico medio de distintas señales que quisiéramos medir.

### - Martínez Pérez Nestor

Con la realización de esta práctica pudimos introducirnos al ambiente de programación en lenguaje ensamblador, el cual se encuentra en uno de los niveles más bajos de programación, para poder programar el microcontrolador y poder utilizar sus diferentes puertos como entrada y salida de datos y lograr que realizara diferentes tareas que en este caso fueron el control del encendido y apagado de los leds y la realización de operaciones aritméticas lógicas básicas como suma, resta, and y or.

Se introdujeron algunos conceptos tales como el complemento a dos que fue necesario para poder realizar la resta de los números, la activación de diferentes banderas y también se aprendió a acceder a dichas banderas que nos ofrece el microcontrolador para poder ocuparlas en futuras aplicaciones.

Se pudo comprobar el correcto funcionamiento del programa sobre un software de simulación de circuitos que en este caso fue proteus para poder asegurarnos de que la programación que realizamos era la correcta de acuerdo al comportamiento que se esperaba obtener sobre el circuito ya armado.

## **8.- BIBLIOGRAFÍA**

- [1].- Fundamentos de sistemas digitales. Thomas L. Floyd. Pearson Prentice Hall, Novena edición.
- [2].- Electrónica digital introducción a la lógica digital. Santiago Acha. Alfaomega.