**INSTITUTO POLITÉCNICO NACIONAL**


## Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas

# Arquitectura de Computadoras

**PRÁCTICA B**

19 de octubre de 2017

Alvarado Balbuena Jorge Anselmo
Avalos Vizuett Julio Cesar
Rocha Díaz Brandon

## Objetivos

- Complementar el análisis teórico visto en clase con herramientas de programación.
- Comprender el análisis y la creación de rutinas en el lenguaje ensamblador, para poder crear timers en nuestros programas.

## Introducción

Las subrutinas (funciones o procedimientos) son abstracciones que se usan en los lenguajes de alto nivel para simplificar el código y poder reusarlo. A la subrutina se la llama (invoca) desde el programa que se invoca mediante una instrucción particular.

Pero antes de llamar a la subrutina tenemos que haber colocado los datos de entrada a ésta en un lugar accesible (registros reservados o pila). En general al trabajar con subrutinas se divide el trabajo a realizar entre el programa invocante y la subrutina:

- Programa invocante: Poner los parámetros o argumentos de entrada a la subrutina en un lugar donde sean accesibles por ésta. o Registros para pasar parámetros r0 – r3
  Ej.: mov r0, r7
- Programa invocante: Transferir el control a la subrutina
  Ej.: bl Etiqueta lr=pc+4 y pc=posición de Etiqueta
- Subrutina: Ejecutar la tarea deseada usando los recursos necesarios
- Subrutina: Poner el resultado en un lugar accesible al Programa. En el ARM la subrutina devuelve un único parámetro de salida con el resultado. En nuestro caso, vamos a trabajar siempre con números enteros con lo que el resultado siempre se devolverá únicamente en r0, pero si el resultado ocupase más de 4 bytes se usarían también los otros registros del r1 al r3.
  Ej.: mov r0, r7
- Subrutina: Devolver el control al punto inicial, manteniendo el contexto
  Ej.: mov pc, lr

## Desarrollo

## Ejercicio 1

Realizar un programa ensamblador tenga programadas 4 secuencias de 8 leds y pueda tener la opción de ponerlo a dos velocidades de cambio distintas (400 ms, y 100 ms), en el diseño incluir un botón de on/off. Utilizar rutinas de tiempo y rutinas para cada secuencia.

| | | |
|---|---|---|
| 0101 0101<br>1010 1010 | 1000 0000<br>0100 0000<br>0010 0000<br>0001 0000 | 0000 1000<br>0000 0100<br>0000 0010<br>0000 0001 |
| 1100 1100<br>0110 0110<br>0011 0011<br>1001 1001 | 1000 0001<br>0100 0010<br>0010 0100<br>0001 1000 | 0010 0100<br>0100 0010 |

## Código

```
.org 0x000 rjmp config
config:        ldi r20, 0x0FF
               out ddrb, r20
               clr r20
               ldi r20, 0x000
               out ddrd, r20
off:clr r21
               out portb, r21
loop:        sbis pind, 1
               rjmp off
               jmp SecUno
               sbis pind, 1
               jmp off
               jmp loop
SecUno:       ldi r21, 0b01010101
               out portb, r21
               sbis pind, 0
               call Delay10
               sbic pind, 0
               call Delay400
               sbis pind, 1
               rjmp off
               ldi r21, 0b10101010
               out portb, r21
               sbis pind, 0
               call Delay10
               sbic pind, 0
               call Delay400
               ldi r21, 0b01010101
               out portb, r21
               sbis pind, 0
               call Delay10
               sbic pind, 0
               call Delay400
               sbis pind, 1
               rjmp off
               ldi r21, 0b10101010
               out portb, r21
               sbis pind, 0
               call Delay10
               sbic pind, 0
               call Delay400
               ldi r21, 0b01010101
               out portb, r21
               sbis pind, 1
               call Delay10
               sbic pind, 0
               call Delay400
               sbis pind, 1
               rjmp off
               rjmp SecDos

SecDos:       ldi r21, 0b11001100
               out portb, r21
               sbis pind, 1
               rjmp off
               sbis pind, 0
               call Delay10
               sbic pind, 0
               call Delay400
               ldi r21, 0b01100110
               out portb, r21
               sbis pind, 1
               rjmp off
               sbis pind, 0
               call Delay10
               sbic pind, 0
               call Delay400
               ldi r21, 0b00110011
               out portb, r21
               sbis pind, 1
               rjmp off
               sbis pind, 0
               call Delay10
               sbic pind, 0
               call Delay400
               ldi r21, 0b10011001
               out portb, r21
               sbis pind, 1
               rjmp off
               sbis pind, 0
               call Delay10
               sbic pind, 0
               call Delay400
```

```
                ldi r21, 0b11001100
                out portb, r21
                sbis pind, 1
                rjmp off
                sbis pind, 0
                call Delay10
                sbic pind, 0
                call Delay400
                rjmp SecTres

SecTres:ldi r21, 0b10000000
                out portb, r21
                sbis pind, 1
                rjmp off
                sbis pind, 0
                call Delay10
                sbic pind, 0
                call Delay400
                ldi r21, 0b01000000
                out portb, r21
                sbis pind, 1
                rjmp off
                sbis pind, 0
                call Delay10
                sbic pind, 0
                call Delay400
                ldi r21, 0b00100000
                out portb, r21
                sbis pind, 1
                rjmp off
                sbis pind, 0
                call Delay10
                sbic pind, 0
                call Delay400
                ldi r21, 0b00010000
                out portb, r21
                sbis pind, 1
                rjmp off
                sbis pind, 0
                call Delay10
                sbic pind, 0
                call Delay400
                ldi r21, 0b00001000
                out portb, r21
                sbis pind, 1
                rjmp off
                sbis pind, 0
                call Delay10
                sbic pind, 0
                call Delay400
                ldi r21, 0b00000100
                out portb, r21
                sbis pind, 1
                rjmp off
                sbis pind, 0
                call Delay10
                sbic pind, 0
                call Delay400
                ldi r21, 0b00000010
                out portb, r21
                sbis pind, 1
                rjmp off
                sbis pind, 0
                call Delay10
                sbic pind, 0
                call Delay400
                ldi r21, 0b00000001
                out portb, r21
                sbis pind, 1
                rjmp off
                sbis pind, 0
                call Delay10
                sbic pind, 0
                call Delay400
                rjmp SecCuat

SecCuat:ldi r21, 0b10000001
                out portb, r21
                sbis pind, 1
                rjmp off
                sbis pind, 0
                call Delay10
                sbic pind, 0
                call Delay400
```

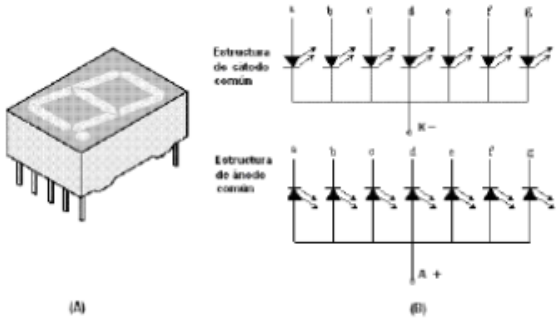
```
ldi r21, 0b01000010
out portb, r21
sbis pind, 1
rjmp off
sbis pind, 0
call Delay10
sbic pind, 0
call Delay400
ldi r21, 0b00100100
out portb, r21
sbis pind, 1
rjmp off
sbis pind, 0
call Delay10
sbic pind, 0
call Delay400
ldi r21, 0b00011000
out portb, r21
sbis pind, 1
rjmp off
sbis pind, 0
call Delay10
sbic pind, 0
call Delay400
ldi r21, 0b00100100
out portb, r21
sbis pind, 1
rjmp off
sbis pind, 0
call Delay10
sbic pind, 0
call Delay400
ldi r21, 0b01000010
out portb, r21
sbis pind, 1
rjmp off
sbis pind, 0
call Delay10
sbic pind, 0
call Delay400
```

```
    ldi r21, 0b10000001
    out portb, r21
    sbis pind, 1
    rjmp off
    sbis pind, 0
    call Delay10
    sbic pind, 0
    call Delay400
    rjmp loop

Delay10:ldi  r18, 104
        ldi  r19, 229
Delay100:sbis pind, 1
        rjmp off
        ldi  r18, 5
        ldi  r19, 15
        ldi  r20, 242
Delay400:sbis pind, 1
        rjmp off
        ldi r22, 4
s1: sbis pind, 1
        rjmp off
        dec  r20
    brne s1
    dec  r19
    brne s1
    dec  r18
    brne s1
        ret
s2:       sbis pind, 1
        rjmp off
        dec r22
        call Delay100
        brne s2
        ret
s3:       dec  r19
    brne s3
    dec  r18
    brne s3
        ret
```

## Ejercicio 2

Elaborar un segunde que cuente de 0 a 59 y lo muestre en dos Display de 7 segmentos de ánodo común el tiempo que va transcurriendo. Habilitar un botón de reset y un botón de pausa.



## Código

```
.org 0x000 rjmp config
config:        ldi R20, 0b01111111
               out ddrb, R20
               out ddrd, R20
               ldi r21, 0b00000000
               ldi r23, 0b00111111
               ldi r24, 0b00000110
               ldi r25, 0b00100000
               ldi r26, 0b00000000
               out portd, r25
               out portb, r23
loop:        call disp0
               call disp1
               sbic pinb,7
               rjmp config
               sbic pind,7
               jmp parar
               out portd, r25
               out portb, r23
               inc r21
               call delay400
               rjmp loop
parar:        sbis pind,7
               rjmp loop
               rjmp parar
delay400:ldi  r18, 3
               ldi  r19, 60
               ldi  r20, 204
s1:               dec  r20
               brne s1
               dec  r19
               brne s1
               dec  r18
               brne s1
               ret
disp0:
dis0:        ldi r22, 0b00000000
               cp r21, r22
               brne dis1
               ldi r23, 0b00111111
               ret
dis1:        ldi r22, 0b00000001
               cp r21, r22
               brne dis2
               ldi r23, 0b00000110
               ret
dis2:        ldi r22, 0b00000010
               cp r21, r22
               brne dis3
               ldi  r23, 0b01011011
               ret
dis3:        ldi r22, 0b00000011
               cp r21, r22
               brne dis4
               ldi  r23, 0b01001111
               ret
dis4:        ldi r22, 0b00000100
               cp r21, r22
               brne dis5
               ldi  r23, 0b01100110
               ret
dis5:        ldi r22, 0b00000101
               cp r21, r22
               brne dis6
               ldi  r23, 0b01101101
               ret
dis6:        ldi r22, 0b00000110
               cp r21, r22
               brne dis7
               ldi  r23, 0b01111101
               ret
dis7:        ldi r22, 0b00000111
               cp r21, r22
               brne dis8
               ldi  r23, 0b00000111
               ret
dis8:        ldi r22, 0b00001000
               cp r21, r22
               brne dis9
               ldi  r23, 0b01111111
               ret
dis9:        ldi r22, 0b00001001
               cp r21, r22
               brne reset
```

```
                ldi  r23, 0b01100111
                ret
disp1:
dis10:          ldi r22, 0b00000000
                cp r26, r22
                brne dis11
                ldi r25, 0b00100000
                ret
dis11:          ldi r22, 0b00000001
                cp r26, r22
                brne dis12
                ldi r25, 0b01111001
                ret
dis12:          ldi r22, 0b00000010
                cp r26, r22
                brne dis13
                ldi r25, 0b01000100
                ret
dis13:          ldi r22, 0b00000011
                cp r26, r22
                brne dis14
                ldi r25, 0b01010000
                ret
dis14:
ldi r22, 0b00000100
                cp r26, r22
                brne dis15
                ldi r25, 0b00011001
                ret
dis15:          ldi r22, 0b00000101
                cp r26, r22
                brne reset
                ldi r25, 0b00010010
                ret
reset:          ldi r23, 0b01100111
                ldi r21, 0b00000000
                inc r26
                cpse r26, r24
                rjmp loop
                rjmp config
```

## Conclusiones

**Alvarado Balbuena Jorge Anselmo**

Durante el desarrollo de esta práctica practicamos el concepto de rutinas. En el primer ejercicio para prender una serie de leds en secuencias diferentes, así como en velocidades, lo cual sirvió de practica para el segundo ejercicio. En este lo realizado fue un segundero, con lo cual vimos una aplicación más real de lo que es posible hacer con el lenguaje ensamblador.

**Julio Cesar Avalos Vizuett**

Una rutina es una parte fundamental en la programación a través del lenguaje de ensamblador porque nos permite crear retardo, crear un código que se puede usar más de una vez. Un retardo es hacer que un microcontrolador no ejecute nada para hacer que pase el tiempo ya que su tiempo de ejecución es de micro segundos, entonces con los retardos podemos crear tiempo para hacer contadores, relojes, etc.

**Rocha Díaz Brandon**

Las rutinas generadas en esta práctica sin duda me brindaron una visión más cercana a la aplicación en la vida real ya que pueden ser utilizadas para diversas situaciones, llevada a algo conocido sería un reloj, una serie navideña, cronómetro o secuencia de operaciones realizadas en ciertas velocidades y orden optimizando el código para un uso constante del mismo por medio del lenguaje ensamblador.