Laboratorio 1 | Entorno de desarrollo y subrutinas

Objetivos

- Dar los primeros pasos con el uso de la placa.
- Desarrollar subrutinas que acceden a puertos de entrada y salida (E/S).
- Validar subrutinas básicas para prácticas siguientes.
- Comprender los ciclos de máquina involucrados en la ejecución de una subrutina.

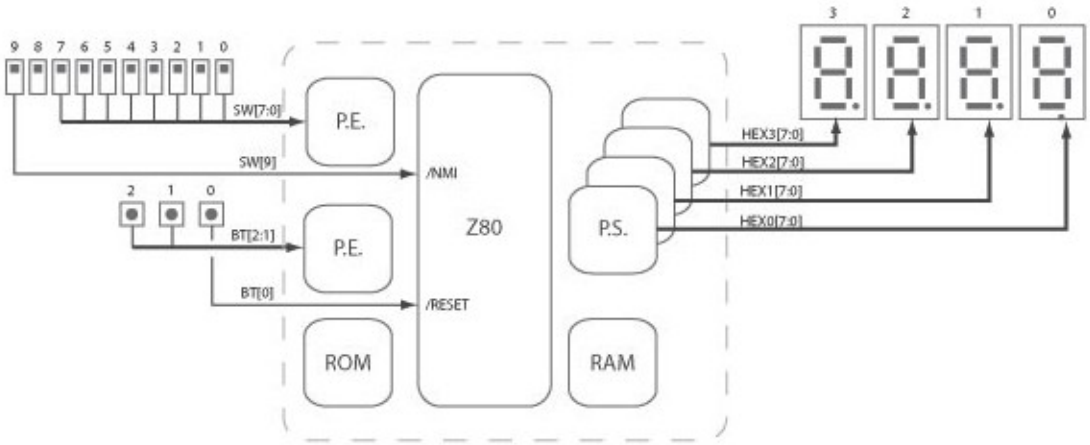
Descripción General

Requerimiento previo a la práctica: Para un correcto aprovechamiento de la práctica se debe realizar previamente el paso: *"2.2 Trabajando con un sistema real cargado en la placa DE0"* de la *"Guía para las herramientas del Laboratorio".*

Al finalizar la presente práctica se contará con un programa que despliega en los display 7 segmentos de la placa una cuenta de segundos y centésimas de segundo. La cuenta no se decrementará automáticamente, sino manualmente manipulando los pulsadores.

Este programa estará estructurado en subrutinas que hacen uso unas de otras, lo que permitirá partir el problema en partes más sencillas de resolver y contar con subrutinas auxiliares que serán útiles en siguientes prácticas.

El siguiente esquema muestra el hardware disponible en la placa. Si bien el sistema cuenta con más puertos y periféricos, solo se ilustran las partes necesarias para llevar adelante esta práctica. El sistema incluye puertos de entrada para los switches SW[9..0] y los pulsadores BUTTON[2..0] y puertos de salida para manejar los leds LEDG[7..0] y los displays de 7 segmentos HEX3, ...HEX0. En el anexo 3 de la *Guía para las herramientas del Laboratorio* se indican en detalle los recursos disponibles y su mapa de direcciones.



IMPORTANTE 1:

Es responsabilidad del programa de usuario inicializar el registro SP para que el stack esté contenido en la memoria disponible para el usuario (0xB000 a 0xFFFF). Se sugiere usar el área más alta de memoria ya que el stack "crece hacia abajo".

IMPORTANTE 2

Cuando no dispongan de la placa igual pueden probar las subrutinas utilizando el ambiente QEMU+GDB al igual que en la Tarea 1.

En ese caso, cuando se mencione un puerto de entrada o salida (switches, botones y displays 7 segmentos) estos serán emulados mediante los archivos in.txt y output.log. Notar que los cambios en los puertos de salida pueden observarse tanto en el archivo output.log como en la ventana del emulador QEMU, y el valor leído desde un puerto de entrada por una instrucción IN es tomado por el simulador desde el archivo in.txt.

Para probar las siguientes subrutinas se sugiere establecer un breakpoint que permita correr el loop principal de a una iteración por vez. De esta forma es posible modificar el archivo in.txt entre iteraciones sucesivas y así emular los cambios en el puerto de entrada.

Organización de los archivos

Se sugiere escribir todas las subrutinas en un único archivo subrutinas.s

Para verificar el buen funcionamiento de cada subrutina se escribirá un programa de prueba separado que la invoque (p. ej. prueba_hexa7seg). Para incluir el código de la subrutina en el programa de prueba se utilizará la directiva include, de la siguiente forma:

.include "subrutinas.s"

Esta directiva hace que antes de invocar al ensamblador se inserte en ese lugar del programa de prueba el contenido completo del archivo incluido (subrutinas.s en el ejemplo).

Subrutinas

hexa7seg

Descripción: Subrutina que convierte un número de cuatro bits a la representación en un display de 7 segmentos del dígito hexadecimal correspondiente (0-9, A-F).

Parámetros: recibe en los cuatro bits menos significativos del acumulador (registro A) el número a convertir y devuelve en el acumulador el número convertido a 7 segmentos con el punto decimal apagado. Preserva todos los registros salvo A y F.

Ejemplo: recibe A= 0x01 y devuelve A= 11111001B

Prueba: Para la prueba de la subrutina se sugiere realizar un programa que implemente un bucle infinito donde se leen los switches[3..0] hacia el acumulador, se invoca la subrutina hexa7seg y se escribe el resultado de la conversión en el dígito menos significativo (HEX0) de los display 7 segmentos.

Notas:

- Esta subrutina se entrega implementada. Se debe analizar el código y entender su funcionamiento.

- La correspondencia entre bits del byte y los segmentos del display se encuentran en el anexo 3 de la guía.
- Se utiliza una tabla para la conversión hexa a 7 segmentos. A continuación, se da el código de la subrutina y el contenido de la tabla para los primeros valores. Se supone que el comienzo de la tabla en memoria está alineado al comienzo de una página de 256 posiciones en memoria, o dicho de otra forma que el byte bajo de la dirección tab_h7s vale 0. Esto último es para facilitar la implementación de la subrutina en cuestión.

```
tab_h7s:

;       _gfedcba
   db   11000000B  ; 0
   db   11111001B  ; 1
   db   10100100B  ; 2
   db   10110000B  ; 3
   db   10011001B  ; 4
   db   ........   ; 5
   db   ........   ; 6
   db   ........   ; 7
   db   ........   ; 8
   db   ........   ; 9
   db   ........   ; A
   db   ........   ; b
   db   ........   ; c
   db   ........   ; d
   db   ........   ; E
   db   ........   ; F

hexa7seg:
      push hl
      ld hl, tab_h7s
      and 0x0f
      ld l, a
      ld a, (hl)
      pop hl
      ret
```

pbcda7seg

Descripción: Subrutina que convierte un BCD empaquetado de 2 dígitos a 7 segmentos 2 dígitos.

Parámetros: recibe en A los dos dígitos en BCD empaquetado y devuelve en el par de registros BC los 2 dígitos en 7 segmentos con el punto apagado (en B el más significativo y en C el menos significativo). Preserva todos los registros salvo A, F, B y C.

Ejemplo: recibe A= 0x01 y devuelve B=11000000B y C= 11111001B

Prueba: Para la prueba de la subrutina se sugiere realizar un programa que implemente un bucle infinito donde se leen 2 dígitos BCD empaquetado de los switches[7..0] hacia el acumulador, se invoca la subrutina pbcda7seg y se escribe el resultado de la conversión en los 2 dígitos menos significativos de los display 7 segmentos HEX1 y HEX0.

Nota: Observar que los dígitos BCD coinciden con los 10 primeros dígitos hexadecimales, por lo que extrayendo los dígitos del BCD empaquetado puede utilizarse la rutina hexa7seg para la conversión de cada dígito.

binapbcd

Descripción: Subrutina que convierte a BCD empaquetado de 2 dígitos un número menor o igual a 99 codificado en binario.

Parámetros: recibe en A el número en binario y devuelve en A los dos dígitos en BCD empaquetado. Preserva todos los registros salvo A y F.

Ejemplo: recibe A= 0x0F= 00001111B y devuelve A=00010101B

Prueba: Para la prueba de la subrutina se sugiere realizar un programa que implemente un bucle infinito donde se lee un número binario de los switches y se utilizan las subrutinas anteriores para mostrarlo en los 2 dígitos menos significativos del display 7 segmentos (HEX1 y HEX0).

Nota: para la representación BCD empaquetado del binario, es necesario conocer el valor de las decenas y unidades del número. Una forma de determinar estos valores es calcular el cociente y el resto de la división entera por 10. Para ello se puede contar cuántas veces se debe restar 10 hasta que el resultado sea negativo.

despreloj

Descripción: Subrutina que modifica el estado de los display 7 segmentos de acuerdo al contenido de variables en memoria.

Parámetros: recibe en IX la dirección *base* de comienzo de la siguiente estructura de datos en memoria:

lugar	**nombre**	**descripción**
base +0	seg	segundos codificados en binario
base +1	cen	centésimas de segundo codificados en binario
base +2	flagpd	bandera que puede tomar los valores 00H y FFH

La subrutina despliega el contenido de las variables preservando registros:

- dígitos 7 segmentos HEX3, HEX2: contenido de seg
- dígitos 7 segmentos HEX1, HEX0: contenido de cen
- los puntos decimales de los display 7 segmentos: según el valor de flagpd (00H encendidos, FFH apagados).

Prueba: Para la prueba de la subrutina se sugiere realizar un programa que inicialice las variables y luego invoque en un bucle infinito la subrutina despreloj. Los valores de memoria se modificarán utilizando el debugger para probar los diferentes valores posibles.

decreloj

Descripción: Subrutina que cada vez que se invoca decrementa en forma anidada centésimas de segundo y segundos. Cada vez que es invocada decrementa 10 centésimas de segundo. Cada vez que se decremente un segundo se debe complementar una bandera. El decremento debe saturar en 0.

Parámetros: recibe en IX la dirección *base* en memoria:

lugar	nombre	descripción
base+0	seg	segundos codificados en binario
base+1	cen	centésimas de segundo codificadas en binario
base+2	flagpd	bandera que puede tomar los valores 0x00 y 0xFF

Decrementa las variables preservando todos registros.

Prueba: Se debe implementar un programa principal con el siguiente pseudocódigo. Para el caso que se esté utilizando el ambiente QEMU+GDB, Notar que el avance de a una iteración por vez puede controlarse cambiando el valor leído en los pulsadores mediante el archivo in.txt

```
   Inicializo sistema
Loop:
   Espero pulsador 2
   Invoco decreloj()
   Invoco despreloj()
   Espero pulsador 1
   Vuelvo a Loop
```

Ensamblado, ciclos y tiempo de ejecución

Ensamblado

Ensamblar el siguiente código escribiendo la tabla de símbolos y completando los espacios en blanco de la tabla (ignorar los casilleros tachados con “XXXXXXX”):

```
CTE16bit equ 0x4242
.org   0x0A00   ;relativo   a   comienzo   del   programa   en   0xB000
espero:
       push AF
       push DE
       ld DE, CTE16bit
loop:  dec DE
       ld A,D
       or E
       jr NZ, loop
       pop DE
       pop AF
       ret
```

Contador de posiciones	**Instrucción**	**Código de máquina**	**Nro de Bytes**	**Ciclos M**	**Ciclos T**
0xBA00	push AF	1111 0101	1	M1, WR^{MEM}, WR^{MEM}	11T
0xBA01	push DE	1101 0101	1	M1, WR^{MEM}, WR^{MEM}	
0xBA02	ld DE, CTE16bit				
0xBA05	dec DE	XXXXXXXXXXX		XXXXXXXXXXX	
	ld A,D	XXXXXXXXXXX		XXXXXXXXXXX	
	or E	XXXXXXXXXXX		XXXXXXXXXXX	
	jr NZ, loop				
	pop DE				
	pop AF	XXXXXXXXXXX		XXXXXXXXXXX	
	Ret	XXXXXXXXXXX		XXXXXXXXXXX	

Símbolo	**Valor**

Ciclos

Para cada una de las instrucciones ensambladas, llenar la tabla indicando para cada ciclo los valores que se observan en el bus de datos y de direcciones.

Al comenzar a ejecutar la subrutina, el valor de los registros es: SP=0xFFF0, AF=0x1234, DE=0xABCD.

Instrucción: push DE (ejemplo)

CICLO	M1	WR^{MEM}	WR^{MEM}		
BUS DIRECCIONES	0xBA01	0xFFED	0xFFEC		
BUS DATOS	11010101	0xAB	0xCD		

Instrucción: ld DE, CTE16bit

CICLO					
BUS DIRECCIONES					
BUS DATOS					

Instrucción: jr NZ, loop

CICLO					
BUS DIRECCIONES					
BUS DATOS					

Instrucción: pop AF

CICLO					
BUS DIRECCIONES					
BUS DATOS					

Tiempo de ejecución

Determinar un nuevo valor para la constante `CTE16bit` de forma que la duración de la subrutina desde que se ejecuta la primera instrucción hasta que se ejecuta *ret* sea lo más próximo posible a 10 ms (milisegundos) . Tener en cuenta que la frecuencia del reloj es 50MHz.

Informe

La entrega consiste de:

- informe.pdf conteniendo:
  - pseudocódigo o diagrama de flujo de todas las subrutinas y programas de prueba
  - la solución del ejercicio planteado de ensamblado, ciclos y tiempo de ejecución
- el código de todas las subrutinas (subrutinas.s)
- programas de prueba para cada una de las subrutinas (prueba_hexa7seg.s, prueba_pbcda7seg.s, prueba_binapbcd.s, prueba_despreloj.s y prueba_decreloj.s) que incluyen el archivo anterior mediante la directiva .include

Se entregará por dos vías antes del **lunes 4 de abril a las 12:30hs**
- versión papel en secretaría del IIE
- archivo formato .zip con todos los ítems mediante la tarea correspondiente en EVA

Cada estudiante **deberá llevar registro de las horas dedicadas a la práctica**. Se les solicitará que ingresen esa información a través de la página del curso.

La defensa se hará en forma presencial en el laboratorio de software del IIE en el horario y día asignado durante la inscripción.