PRÁCTICA 6

Carlos Raso Alonso

Introducción a los ordenadores

Objetivos

En esta práctica se busca que nos familiaricemos con el simulador i8085, más en concreto con las subrutinas, la gestión de la memoria y nos iniciamos en el uso de dispositivos de entrada y salida

Parte 1

Pregunta 1: ¿Cuál es el modo de direccionamiento de la instrucción LXI?

LXI tiene dos tipos de almacenamiento: almacenamiento inmediato en sus dos últimos bits ya que en ellos se encuentra la información explícita que debe ser cargada. Almacenamiento directo en la parte de la instrucción que indica en qué par de registros se va a almacenar el dato contenido en los dos últimos bits de la instrucción. Esto es almacenamiento directo ya que la instrucción contiene el indicador del par de registros en el que se cargará el inmediato.

Pregunta 2: ¿Qué instrucción guarda en la pila el PC?

Call pone el contenido del par de registros pc en la pila por defecto y luego fija el par de registros al valor indicado en la instrucción para realizar el salto de flujo de programa. Con la instrucción push pc también se consigue poner el contenido del pc en la pila

Pregunta 3 y 4: ¿Qué espacio ocupa la subrutina suma? ¿Cuántos ciclos tarda en ejecutarse?

```
PUSH PSW        ; 1 byte, 12 ciclos
LDAX D          ; 1 byte, 7 ciclos
ADD M           ; 1 byte, 7 ciclos
STAX D          ; 1 byte, 7 ciclos
INX H           ; 1 byte, 6 ciclos
INX D           ; 1 byte, 6 ciclos
POP PSW         ; 1 byte, 10 ciclos
RET             ; 1 byte, 10 ciclos
```

He comentado el código incluyendo la memoria que ocupa cada instrucción y también los ciclos que tarda en ejecutarse, por tanto, para hallar la memoria total que ocupa toda la subrutina, así como los ciclos que tarda en ejecutarse, basta con sumar los datos correspondientes de cada instrucción. Así pues la subrutina tarda 65 ciclos en ejecutarse y ocupa 8 bytes de memoria

Tarea 1: Dibujar el mapa de memoria de datos: direcciones y contenido. Indicar las instrucciones que la modifican y cómo lo hacen. Incluir también la memoria del programa.

Queda indicado en el esquema de la tarea 2

Tarea 2: Indicar qué instrucciones modifican la pila

Directivas y etiquetas	Dirección	Contenido (número o código)	Repercusión en la memoria o en la pila
.data Mat 1:	0	1	
	1	2	
Mat 2:	2	3	
	3	4	
Mat 3:	4	0	
	5	0	
		...	
.data Pila:	20	0	
		...	
.org	600	LXI H, pila	Ninguna
	603	SPHL	Ninguna, solo inicializa el SP
	604	MVI B, num	Ninguna
	606	LXI D, mat 1	Ninguna
	609	LXI H, mat2	Ninguna
loop:	60C	CALL suma	En la pila se añade el valor de PC
	60F	DCR B	Ninguna
	610	JNZ loop	Ninguna
	613	NOP	Ninguna
	614	HLT	Ninguna
Suma:	615	PUSH PSW	En la pila se añade el valor de AF
	616	LDAX D	Ninguna
	617	ADD M	Ninguna
	618	STAX D	En la posición de memoria apuntada por DE se inscribe el valor que alberga el acumulador (A)
	619	INX H	Ninguna
	61A	INX D	Ninguna
	61B	POP PSW	El último valor introducido en la pila (en este caso el antiguo valor de AF) se inscribe en el registro AF y se borra de la pila
	61C	RET	El último valor introducido en la pila (en este caso el valor de PC antes de entrar a la subrutina) se inscribe en el registro PC y se borra de la pila

Código con las explicaciones de lo que hace:

```
.define
num 02h
.data 00h
mat1: db 1,2
mat2: db 3,4
mat3: db 0,0
.data 20h
pila:
.org 600h           ; en esta primera parte de código se asigna al SP el valor
                    ; correspondiente para que apunte a la etiqueta pila
                    ; y se cargan mat1 y mat2 en los pares de registros DE y HL y
num en el B
LXI H, pila         ; HL <= pila
SPHL                ; SP <= HL
MVI B, num          ; B <= [num]
LXI D, mat1         ; DE <= mat1
LXI H, mat2         ; HL <= mat2
loop:               ; el loop llama a la rutina suma y luego resta 1 al registro
                    ; B. Si después de restar 1, B es distinto de 0,
                    ; entonces vuelve a empezar el loop. Cuando se sale del loop
                    ; se para el procesador.A
CALL suma
DCR B               ; B <= B - 1
JNZ loop            ; si A != 0 entonces saltar a loop
NOP
HLT                 ; detener el procesador
suma:               ; Esta subrutina deja la palabra de estado de programa intacta
                    ; y momentáneamente alberga el resultado de la suma del
                    ; contenido de la dirección almacenada en HL y en DE en A,
                    ; luego guarda esta suma almacenada en A en la posición de
                    ; memoria apuntada por los registros DE. Finalmente les suma 1
                    ; a los pares de registros HL y DE (este cambio altera los
                    ; registros también fuera de la subrutina).
PUSH PSW            ; push AF
LDAX D              ; A <= [DE]
ADD M               ; A <= A + [HL]
STAX D              ; [DE] <= A
INX H               ; HL <= HL + 1
INX D               ; DE <= DE + 1
POP PSW             ; pop AF
```

RET

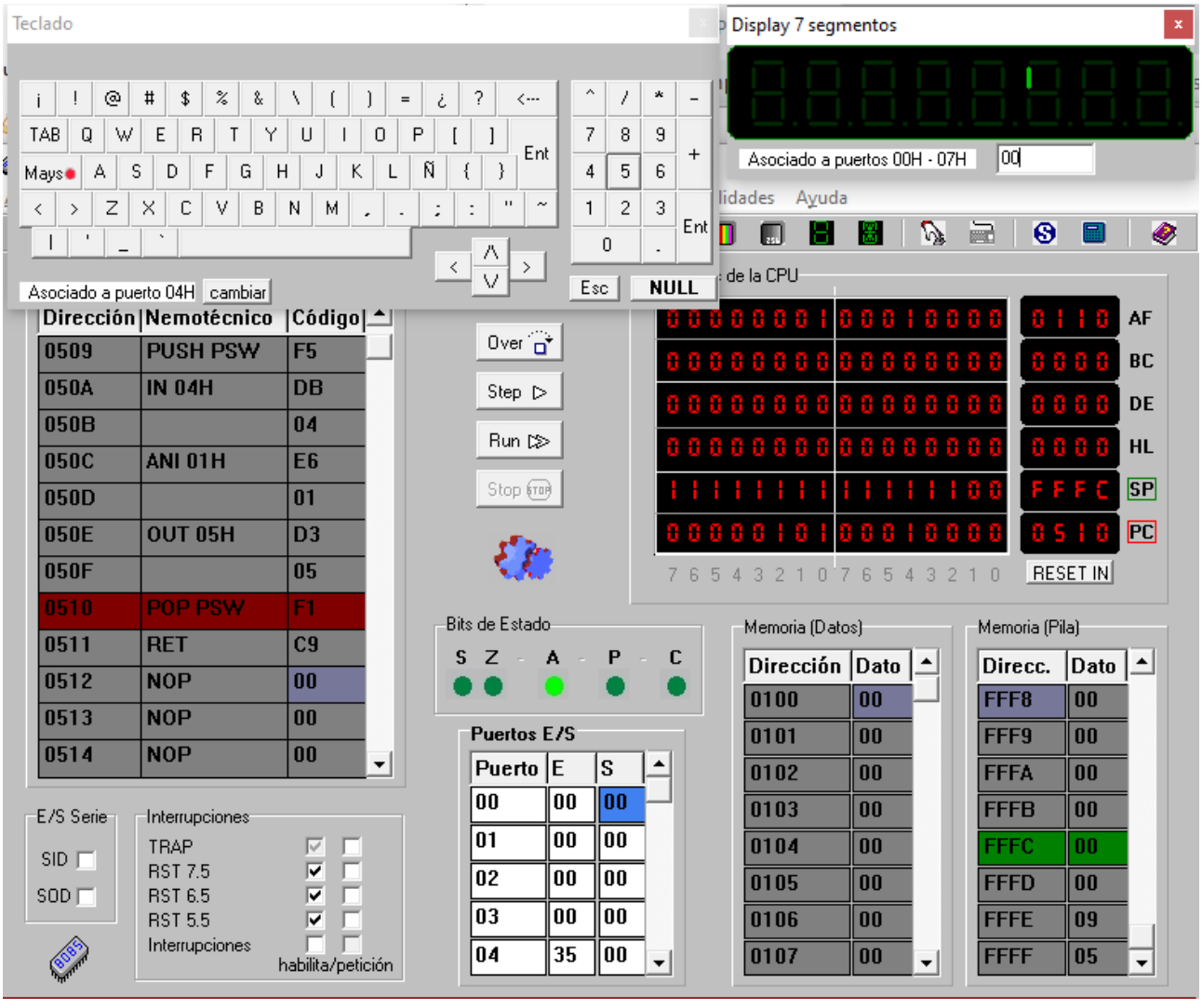
Parte 2

Tarea 3: ¿Qué hace la subrutina puertos? Introduce datos con los interruptores y el teclado y observa el comportamiento del programa en un puerto de salida.

La subrutina puertos empieza guardando en la pila la palabra de estado de programa (PSW o AF) en la pila. Posteriormente guarda en el acumulador el valor que hay en el puerto de entrada 04h. Posteriormente hace la operación A and 01h y la almacena en A. Esto es una máscara que transforma el contenido de A de tal modo que se transforma en 1h si el valor anterior de A es impar y se transforma en 0 si el valor anterior de A es par. Posteriormente, se inscribe en el puerto de salida 05h el valor de A y se recupera el primer valor de la pila en los registros AF. En definitiva lo que hace puertos es que comprueba si la entrada del puerto 04h es impar o impar. Si es par por el puerto de salida 05h sale 0 y si es par, por el puerto de salida 05h sale 01h.

Al hacer pruebas con el programa se comporta de la manera esperada, por ejemplo si muestro el display de 7 segmentos y tomo como periférico de entrada el teclado, si la tecla introducida por teclado tiene codificación en el número que se muestra como entrada par, no se enciende ningún led y si su codificación es impar, se enciende un único led, el asociado al 01h del puerto de salida 05h

Imagen en la que vemos cómo se enciende uno de los leds del display cuando estamos presionando la tecla 5 del teclado con codificación impar (35h):



Código explicado del programa:

```
.data 100h
pila:
.org 24h            ; cada vez que se produce una interrupción se salta a este punto
                    ; del programa y a su vez se salta a la etiqueta ports, cuyo
                    ; funcionamiento está explicado en el ejercicio.
 JMP ports
.org 500h
LXI H, pila         ; HL <= pila
SPHL                ; SP <= HL
CALL ports
NOP
HLT                 ; detener procesador
ports:
PUSH PSW            ; Push AF
IN 04h              ; A <= puerto entrada 04h
ANI 00000001        ; A <= A AND 01h
OUT 05h             ; puerto salida 05h <= A
POP PSW             ; pop AF
RET
```

Parte 3

Tarea 4: Diseña un programa asembler que represente en un display de 7 segmentos los números del 0 al 5 introducidos por teclado y que tenga una opción de apagar todos los leds del display si se pulsa la tecla c. Explica brevemente el algoritmo escogido y copia el código en ensamblador comentado.

Para realizar esta tarea he creado una subrutina para saber qué número se está pulsando en el teclado y según qué número detecte va a una subrutina que se ocupa de mostrar en el display ese número. El código comentado y explicado es el siguiente:

```
.define
      out1  44h          ; se definen los datos outi, los cuales representan
      out2  3eh          ; el output requerido para que el display muestre
      out3  6eh          ; el número i
      out4  4dh
      out5  ebh
.org 00h                 ; Comenzamos el programa inicializando el valor del SP y
                         ; luego entramos en el bucle principal de nuestro
                         ; programa
                         ; en la etiqueta loop.
      LXI H, pila        ; HL <= pila
      SPHL               ; SP <= HL
      JMP loop
.org 24h                 ; cada vez que se produce una interrupción se salta
                         ; a este punto del programa y a su vez se salta a la
                         ; subrutina selec, cuyo funcionamiento está explicado
                         ; más adelante. Al volver de la rutina también se vuelve
                         ; a donde se estaba antes de la interrupción (el bucle
                         ; loop).
      CALL selec
      RET
.data 200h
pila:
.org 250h
```

```
loop:                   ; Bucle infinito en el que está el programa a la espera
                        ; de una interrupción
      JMP loop
selec:                  ; La subrutina selec comprueba qué número (o tecla
                        ; clear) se está introduciendo por teclado para sacarlo
                        ; por el display
      PUSH PSW
num1:                   ; en cada parte numi se comprueba si el puerto in 04h
                        ; es el número indicado (o la c en el último caso), en
                        ; cuyo caso se introduce el número correspondiente en el
                        ; acumulador y se salta a la etiqueta mostrar en la que
                        ; el contenido del acumulador se pone en el puerto de
                        ; salida para que se muestre el número indicado. En caso
                        ; de que el número del puerto de entrada no sea el
                        ; indicado se salta a la etiqueta end en la que se sale
                        ; de la subrutina selec
IN 04h
      SBI 31h
      JNZ num2
      MVI A, out1
      JMP  mostrar
num2:
      IN 04h
      SBI 32h
      JNZ num3
      MVI A, out2
      JMP mostrar
num3:
      IN 04h
      SBI 33h
      JNZ num4
      MVI A, out3
      JMP mostrar
num4:
      IN 04h
      SBI 34h
      JNZ num5
      MVI A, out4
      JMP mostrar
num5:
      IN 04h
      SBI 35h
      JNZ clear
      MVI A, out5
      JMP mostrar
clear:
      IN 04h
      SBI 43h           ; la tecla c se corresponde con 43h en ASCII
      JNZ end
      MVI A, 00h
mostrar:
      OUT 07h
end:
      POP PSW
      RET
```

Conclusiones

Tras finalizar la práctica creo que he aumentado mis conocimientos sobre el i8085, especialmente sobre rutinas y cómo utilizarlas para crear programas más complejos, así como sus interacciones con la pila. También he entendido mejor cómo se organiza la memoria de este simulador y cómo se utilizan los dispositivos de entrada y salida.