Stack y Subrutinas en el 8085

Maite Orama Miranda, William Neris Diaz
Departamento de Física y Electrónica

Abstracto

En este experimento estaremos practicando las instrucciones de tiempo del 8085 y comenzaremos a utilizar las instrucciones para hacer subrutinas.

I. Teoría

El stack en el 8085 puede ser descrito como un conjunto de localizaciones en memoria en R/W controlado por el programador en el programa principal. El comienzo del stack es definido por la instrucción LXI SP, la cual carga una dirección de 16 bits en el registro "stack point" del procesador.

El "Stack" es un dispositivo para guardar la direccion de un CALL. Este es un almacén temporal para los datos. La CPU puede EMPUJAR (PUSH) datos importantes sobre el "Stack", mientras está procesando otros datos. Cuando acaba esta tarea, QUITA (POP) los datos ahorrados del "Stack". Se puede ver esto como una pila de platos. El plato inferior es el primer pedacito de los datos que fueron empujados sobre el "Stack". La placa superior es el último dato que ha sido ingresado. La placa superior es el primer dato que será quitado (Poped) y la placa inferior será el último dato que será sacado. Es un "LAST IN, FIRST OUT stack".(1) El la figura 1, *X* es el primer dato que se empujará, luego *Y* y finalmente *A*. La CPU sale a procesar otros datos. Una vez termina esta tarea vuelve a tirar los datos salvados. Primero *A* es tirado, luego *Y* y finalmente *X*. La instrucción para empujar los datos es PHA. Solamente los datos en el acumulador se pueden empujar sobre el "stack". Otros datos pueden ser empujados si se transfieren primero al acumulador. La instrucción para tirar los datos del "Stack" es PLA. Los datos en el "Stack" se transfieren entonces al acumulador.

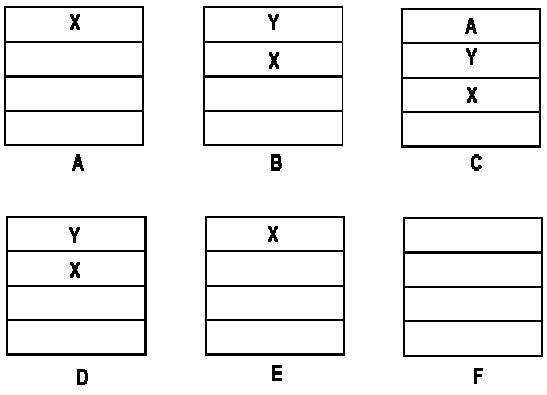

Fig 1. Procedimiento para un PUSH y POP

Hay que notar que el subprograma o subrutina que es llamado termine con una instrucción de RETURN. Esta instrucción es similar a un JUMP pero esta vuelve a la instrucción que esta después de la que la llamó. Por lo tanto el "Stack" se actualiza cuando usted LLAMA y vuelve del subprograma. El PC (program counter el cual señala la instrucción siguiente de ser ejecutado) es empujado al "Stack" y la dirección del CALL se carga en el PC. Al volver, la PC se carga con el dato de la parte superior del "Stack".

Algunas de las instrucciones relacionadas al stack son las siguientes:

PUSH RP	***POP RP***
PUSH B	***POP B***
PUSH D	***POP D***
PUSH H	***POP H***
PUSH PSW	***POP PSW***
CALL	***RET***

II. **Experimento**

En este laboratorio hicimos 4 programas en total los cuales realizaban subrutinas pero cada una de ellas era diferentes los programas fueron los siguientes:

Programa 1:

Direc	Instrucción	Opcode	Comentario
C000	LXI SP,C800	3100C8	Carga el stack pointer con esta dirección
C003	MVI C,28	0E 28	Mueve inmediatamente 28 a C
C005	TOP: Call Beep	CD9000	Llamar a la bocina
C008	DCR C	0D	Quitarle uno a C
C009	JNZ TOP	C205C0	Brinca a esa dirección si C no es cero
C00C	HLT	76	Terminar

El programa 2 fue el siguiente:

Direc	Instrucción	Opcode	Comentario
C000	LXI SP, C099	3199C0	Carga el valor que esta en esa dirección
C003	MVI L, 00	2E 00	Mueve inmediatamente 00 al registro L
C005	PUSH H	E5	Escribe el valor de H
C006	POP PSW	F1	Lee el estado del procesador
C007	POP H	E1	Le el valor del registro H
C008	MOV A, L	7D	Mueve lo que esta en L al acumulador
C009	OUT PORT	D3 00	Muestra el resultado en los seven segment
C00B	MVI A,00	3E 00	Mueve 00 al acumulador
C00D	ORA A	B7	Haz un inclusive Or con el acumulador
C00E	PUSH PSW	F5	Escribe lo que hay en el procesador
C00F	POP H	E1	Lee lo que hay en H
C010	MOV A, L	7D	Mueve lo que esta en L al registro A
C011	ANI 40	E6 40	Haz un and con el acumulador
C013	OUT PORT	D3 00	Muestra el resultado en los seven segment
C015	HLT	76	Terminar

El programa 3 es el siguiente:

Direc	Instrucción	Opcode	Comentario
C000	LXI SP, C099	3199C0	Carga el valor que esta en esa dirección
C003	MVI A, 20	3E 20	Mueve inmediatamente 20 al registro A
C005	OUT PORT	D3 00	Muestra el resultado en los seven segment
C007	MVI B, 0A	06 0A	Mueve inmediatamente 0A al registro B
C009	CALL	CD50C0	Llama a la

	DELAY		subrutina delay
C00C	MVI A, 30	3E 30	Mueve 30 al acumulador
C00E	OUT PORT	D3 00	Muestra el resultado en los seven segment

La subrutina de delay es la siguiente:

Direc	Instrucción	Opcode	Comentario
C050	PUSH D	D5	Escribe el contenido de DE
C051	PUSH PSW	F5	Escribe el contenido deL procesador
C052	LXI D, 28	11 28	Carga el registro D con 28
C054	DCX D	1B	Decrementa el registro D
C055	MOV A, D	7A	Mueve lo que esta en D al acumulador
C056	ORA E	B3	Haz un inclusive Or con E
C057	JNZ LOOP	C254C0	Brinca a esa dirección si el registro no es cero
C05A	DCR B	05	Decrementa el registro B
C00B	JNZ SEGUNDO	C252C0	Brinca a esa dirección si el registro no es cero
C05E	POP PSW	F1	Lee lo que hay en el procesador
C05F	POP D	D1	Lee lo que hay en D
C060	RET	C9	Regresa al programa principal

El ultimo programa que se realizo en este experimento fue un semáforo utilizando el microprocesador y la secuencia de instrucciones fue la siguiente:

Direc	Instrucción	Opcode	Comentario
C000	LXI SP, C099	3199C0	Carga el valor que esta en esa dirección
C003	MVI A, 41	3E 41	Mueve inmediatamente 20 al registro A
C005	OUT PORT	D3 00	Muestra el resultado en los seven segment
C007	MVI B, 0F	06 0F	Mueve inmediatamente 0F al registro B
C009	CALL DELAY	CD50C0	Llama a la subrutina delay
C00C	MVI A, 84	3E 84	Mueve 84 al acumulador
C00E	OUT PORT	D3 00	Muestra el resultado en los seven segment
C010	MVI B, 05	06 05	Mueve inmediatamente 05 al registro B
C011	CALL DELAY	CD50C0	Llama a la subrutina delay
C014	MVI A, 90	3E 90	Mueve 90 al acumulador
C016	OUT PORT	D3 00	Muestra el resultado en los seven segment
C019	MVI B, 05	06 05	Mueve inmediatamente 05 al registro B
C01B	CALL DELAY	CD50C0	Llama a la subrutina delay
C01E	JMP	C303C0	Brinca a esta dirección incondicional

La secuencia de delay utilizada para este programa del semáforo fue la misma que para el programa 3.

Direc	Instrucción	Opcode	Comentario
C050	PUSH D	D5	Escribe el contenido de DE
C051	PUSH PSW	F5	Escribe el contenido deL procesador
C052	LXI D, 28	11 28	Carga el registro D con 28
C054	DCX D	1B	Decrementa el registro D
C055	MOV A, D	7A	Mueve lo que esta en D al acumulador
C056	ORA E	B3	Haz un inclusive Or con E
C057	JNZ LOOP	C254C0	Brinca a esa dirección si el registro no es cero
C05A	DCR B	05	Decrementa el registro B
C00B	JNZ SEGUNDO	C252C0	Brinca a esa dirección si el registro no es cero
C05E	POP PSW	F1	Lee lo que hay en el procesador
C05F	POP D	D1	Lee lo que hay en D
C060	RET	C9	Regresa al programa principal

III. Análisis y Discusión

En el primer programa hacíamos sonar la bocina con una subrutina por aproximadamente un segundo. En los seven segment podíamos ver como el registro iba decrementando. En el segundo programa utilizamos mucho las instrucciones pop y push las cuales tienen que entran y salen del stack pointer, el resultado de este programa es:

40

En el tercer programa utilizamos varias de las instrucciones del segundo pero esta vez utilizamos un delay de un segundo el resultado de este programa fue el siguiente:

40 30

El ultimo programa fue el semáforo y este continuaba sin interrupción. En nuestro programa sabíamos que cambiaba de color utilizando números los números representativo fueron los siguientes:

41- luz roja

90- luz amarilla

84- luz verde

También en nuestros cálculos estaba el cruce de peatones mientras la luz estuviera roja y amarilla los peatones no podrían pasar y cuando estuviera verde si podían pasar. Los delays entre las luces eran distintos y esto lo lográbamos hacer cargando un valor diferente en el registro B.

IV. Conclusión

En conclusión pudimos aprender como el microprocesador tiene la capacidad de crear delay y de hacer subrutinas esto es sumamente útil cuando tu necesitas que el microprocesador espere por un cierto tiempo y las subrutinas nos permite grabar pequeños programas en diferentes partes de memoria. En este experimento no tuvimos problemas todo nos salio como debía.

V. Referencias

(1) Knott, Graham. THE STACK, http://ourworld.compuserve.com/homepages/g_knott/elect298.htm