Operaciones Aritméticas en el 8085

Jorge L. Morales Ortiz
Héctor M. Solís Villodas

Abstracto

Este informe es una continuación sobre la introducción del microprocesador Intel 8085, que ya comenzamos en el informe anterior. Este microprocesador es un dispositivo que lleva a cabo una serie de instrucciones. En este experimento trabajaremos con las operaciones de aritmética y lógica del 8085 y las instrucciones de jump condicional.

I. Introducción

El microprocesador 8085 está compuesto de unidades de control, de aritmética y lógica y de registros. En este experimento nos centralizamos en la unidad de aritmética y lógica (ALU). Su diagrama lógico se presenta en la Figura 1. Esta lleva acabo todas las operaciones aritméticas y lógicas de las que es capaz el microprocesador. Es alimentada por un registro de acumulador y un registro temporal. El microprocesador también tiene un registro de "status flags", que almacena las condiciones resultantes de cada operación que ejecuta el ALU. Este se conoce como el **Status Register**.

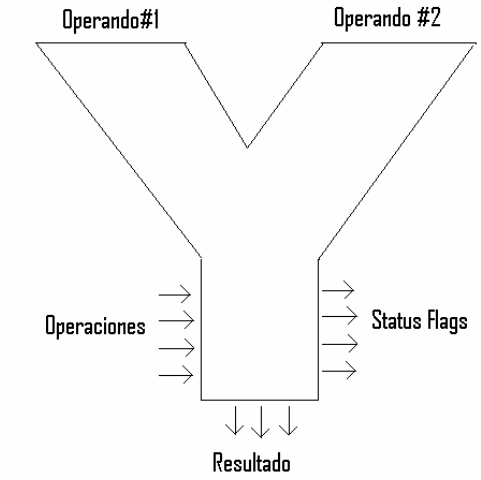

Figura 1 – Diagrama lógico de ALU

El ALU del 8085 sabe llevar a cabo dos tipos de operaciones:

A. **Aritméticas –** estas instrucciones llevan a cabo la función de sumar, restar, incrementar o decrementar datos en los registros o en la memoria. Algunas de las instrucciones son: ADD, ADI, SUB, SUI, DCR, DCX, INR y INX.

B. **Lógicas –** este grupo de instrucciones ejecuta operaciones lógicas (booleanas) en los datos de los registros o la memoria y en las condiciones de los "flags". Algunas de sus instrucciones son: ANA, ANI, ORA, OR, XRA y XRI.

El **Status Register** del 8085 es un registro de 8bits. Este almacena cada una de las condiciones resultantes de una operación del ALU en un bit separado. Los "status flags" que contiene este registro son las siguientes:

Z (zero flag) – indica si el resultado es o no es cero.
CY (carry flag) – indica si hubo "carry".
S (sign flag) – indica si el resultado es positivo o negativo.
P (parity flag) – indica si la suma de unos (1) en el resultado es par o impar.
AC (auxiliary carry) – indica si se produjo un carry del bit 3 al bit 4.

II. Experimento

Asignación pre-lab:

Ejecute el siguiente programa a mano y especifique el estado de cada uno de los "Flags" luego de cada una de las

instrucciones de aritmética. También especifique el contenido de cada uno de los registros al finalizar el programa.

Programa pre-lab:

Mnemonic	Hex	Comentarios
MVI A, 30	3E 30	Carga al registro A 30.
ADI 50	C6 50	Suma 50 al acumulador (80).
MVI A, 80	3E 80	Carga al registro A 80.
MVI B, 78	06 78	Carga al registro B 78.
MVI C, 90	0E 90	Carga al registro C 90.
INR A	3C	Incrementa registro A (81).
SUB B	90	Resta 78 al acumulador (09).
ADD C	81	Suma 90 al acumulador (99).
ADD B	80	Suma 78 al acumulador (111).
OUT 00	D3 00	Muestra lo que está en el acumulador (11).
HLT	76	Detenerse.

Análisis de los "status flags" en las instrucciones de aritmética para el programa pre-lab:

Assembly	Z	C	S	P	AC
MVI A, 30	No altera flags				
ADI 50	0	0	0	0	0
MVI A, 80	No altera flags.				
MVI B, 78	No altera flags.				
MVI C, 90	No altera flags.				
INR A	No altera flags.				
SUB B	0	1	0	1	0
ADD C	0	0	0	1	0
ADD B	0	1	0	1	1
OUT 00	No altera flags.				
HLT	No altera flags.				

Después de ejecutar el programa, el registro A debe tener 11, el registro B debe tener 78 y el registro C debe tener 90. El resultado del programa es 11.

Durante el experimento:

Para todos los programas que va a analizar, asuma que la memoria contiene inicialmente los siguientes datos:

Tabla 1 – Datos iniciales para el 8085

Address Dec	Address Hex	Dato Hex
32	0020	51
33	0021	11
34	0022	32

Parte A:

Corra el siguiente programa a mano. Determine lo que hace cada instrucción y el estado del los "status flags" luego de cada instrucción de lógica o aritmética. Determine el estado de cada registro y de las direcciones de memoria 0020 a 0022 al terminar el programa. Diga qué debemos observar al correr el programa.

Programa #1 –

Address	Assembly	HexCode
C000 C001 C002	LDA 0020	3A 20 00
C003	MOV B, A	47
C004 C005 C006	LDA 0021	3A 21 00
C007	SUB B	90
C008 C009 C00A	JC LABEL1	DA 0E←XX C0←XX
C00B C00C C00D	LDA 0022	34 22 00
C00E C00F	LABEL1: ADI 20	LABEL1: C6 20
C010 C011	ANI 20	E6 20

C012 C013 C014	JZ LABEL2	CA 1A←XX C0←XX
C015 C016	MVI A, AA	3E AA
C017 C018 C019	JMP DISPLAY	C3 1C←XX C0←XX
C01A C01B	LABEL2: MVI A, BB	LABEL2: 3E BB
C01C C01D	DISPLAY: OUT 00	DISPLAY: D3 00
C01E	HLT	76

Traduzca el programa a lenguaje de máquina y éntrelo al módulo, así como los datos en las direcciones 0020 a 0022, para verificar sus resultados.

Parte B:

Repita el procedimiento descrito en la Parte A con el siguiente programa:

Programa #2 –

Address	Assembly	HexCode
C000 C001 C002	LXI H, 0020	21 20 C0
C003	MOV A, M	7E
C004	INX H	23
C005	MOV B, M	46
C006 C007	IN 00	DB 00
C008	ANI 08	E6 08
C009 C00A C00B	JZ LABEL1	CA XX→11 XX→C0
C00C	SUB B	90
C00D C00F C010	JMP DISPLAY	C3 XX→12 XX→C0
C011	LABEL1: ADD B	LABEL1: 80
C012	DISPLAY:	DISPLAY:
C013 C014	OUT 00	D3 00
C015	HLT	76

III. Análisis de Datos

Parte A:

Programa #1 –

Assembly	Comentarios
LDA 0020	Carga en el acumulador 51.
MOV B, A	Carga en el registro B 51.
LDA 0021	Carga en el acumulador 11.
SUB B (1)	Restar 51 al acumulador (-40)
JC LABEL1	No se ejecuta porque no hubo un “carry”.
LDA 0022	Carga en el acumulador 32.
LABEL1: ADI 20	No se ejecuta.
ANI 20 (2)	Operación de AND entre A (32) y 20. Guarda 20 en el acumulador.
JZ LABEL2	No se ejecuta porque el resultado no fue 0.
MVI A, AA	Carga en el acumulador AA.
JMP DISPLAY	Brinca a DISPLAY.
LABEL2: MVI A, BB	No se ejecuta.
DISPLAY: OUT 00	Mostrar lo que está en el acumulador (AA).
HLT	Detenerse.

Flags:
(1) Z = 0 C = 0 P = 0 S = 1 AC = 0
(2) Z = 0 C = 0 P = 0 S = 0 AC = 0

Después de ejecutar el programa, el registro A debe tener AA y el registro B debe tener 51. El resultado del programa es AA. Al correr el programa observamos que JC

brinca sólo si el carry flag es 1, y JZ sólo brinca si el zero flag es 1.

Parte B:

Programa #2 –

Assembly	Comentarios
LXI H, 0020	Carga en el registro HL 51.
MOV A, M	Guarda 51 en el acumulador.
INX H (1)	Incrementa el registro HL (carga 11).
MOV B, M	Guarda 11 en el registro B.
IN 00	Entra 00 al acumulador.
ANI 08 (2)	Operación de AND entre el acumulador (00) y 08. Guarda 00 en A.
JZ LABEL1	Salta a la dirección C011 porque el resultado fue 0.
SUB B	No se ejecuta.
JMP DISPLAY	No se ejecuta.
LABEL1: ADD B (3)	Le suma 11 al acumulador.
DISPLAY: OUT 00	Muestra lo que está en el acumulador (11).
HLT	Fin del programa.

(1) Flags:
No ocurre ningún cambio en los "flags" (todo permanece en el estado anterior).

(2) Flags:
Z = 1 C = 0 P = 1 S = 0 AC = 0

(3) Flags:
Z = 0 C = 0 P = 1 S = 0 AC = 0

Al finalizar el programa, el registro A tiene 11 y el registro B tiene 11. El resultado del programa es 11. Las instrucciones de SUB B y JMP DISPLAY no se ejecutan porque la ejecución de la instrucción JZ LABEL1 causa que se salten esas instrucciones.

IV. Conclusión

En este experimento trabajamos nuevamente con el módulo 8085. Además de utilizar las instrucciones de transferencia de datos ya aprendidas en el experimento anterior, practicamos algunas de las instrucciones de aritmética y de lógica del microprocesador. A través de esta práctica, pudimos observar cómo se altera el contenido de los "flags" almacenados en el Status Register cada vez que se ejecuta una de estas instrucciones. Las instrucciones de jump condicional del 8085 también fueron utilizadas en este experimento. Estas condiciones se ejecutan dependiendo de algunos de los estados de los "flags" en el Status Register.

V. Referencias

(1) http://www.insoluz.com/Micro/Micro.html

(2) www.book-books.org/intel-microprocessors:-hardware-software-and-applications:-**8085**-8086-88-80486.html