UNIVERSIDAD DE PUERTO RICO
RECINTO DE HUMACAO
DEPARTAMENTO DE FÍSICA Y ELECTRÓNICA
Glendalys Figueroa Freytes e Isaura Rivera Meléndez

LABORATORIO #7: OPERACIONES ARITMÉTICAS EN EL 8085.

Mediante este experimento se practicarán las instrucciones de aritmética y lógica del 8085 y algunas de las instrucciones de jump condicional de éste.

I. Introducción

Todo microprocesador posee un set de instrucciones el cual puede ejecutar. La colección de instrucciones es implementada como patrones de bit, de los cuales cada uno posee un significado diferente cuando son cargados en los registros. Estos patrones de bit están definidos como un set de palabras cortas las cuales son llamadas el lenguaje de "assembly" del procesador. Un "assembler" puede traducir las palabras en sus patrones de bit fácilmente colocando la salida del "assembler" en memoria para que el microprocesador la ejecute. Parte de estas instrucciones lo son las de aritmética (ADD, ADI, SUI, SUB, INX, INR, DCX, DCR) y las de "jump" incondicional (JMP).

II. Experimentación

Se trabajo con un "módulo" en donde se guardo en los espacios de memoria los siguientes datos:

Address	Datos
0020	51
0021	11
0022	32

Luego se entraron los programas de los problemas 1-2 y se corrieron para ver los resultados. Estos fueron luego comparados con los resultados del análisis hecho a mano para corroborar los resultados presentados en el "módulo".

III. Asignación Pre-Lab

Se ejecutó el programa que está a continuación, a mano y se realizó un análisis de cada uno de los Flags luego de cada una de las instrucciones aritméticas. También se encontró el estado de cada registro al finalizar el programa.

MVI A, 30-hace que se grabe 30 en el registro A.
ADI 50- suma 50 al contenido de A (30) el resultado es 80 (1000 0000), por lo tanto:

- Zero Flag =0 (el resultado no es cero)
- Carry Flag =0 (no hubo acarreo)
- Sign Flag =1 (el MSB =1)
- Parity Flag =0 (hay un número impar de 1's)
- Auxiliary Carry =0 (no hubo acarreo del bit 3 al 4)

MVI A, 80-mueve inmediatamente el 80 al acumulador y lo guarda allí.
MVI B, 78-mueve inmediatamente el 78 en B y lo guarda allí.
MVI C, 90- mueve inmediatamente el 90 en C y lo guarda allí.
INR A- incrementa A.

- Zero Flag = 0
- Carry Flag = 0

- Sign Flag = 1
- Parity Flag = 1
- Auxiliary Carry =0

SUB B- le resta B (78) al acumulador.

- Zero Flag = 0
- Carry Flag = 0
- Sign Flag = 0
- Parity Flag = 1
- Auxiliary Carry =1

ADD C- le suma C al acumulador.

- Zero Flag = 0
- Carry Flag = 1
- Sign Flag = 1
- Parity Flag = 1
- Auxiliary Carry = 0

ADD B- le suma B al acumulador.

- Zero Flag = 0
- Carry Flag = 0
- Sign Flag = 1
- Parity Flag = 1
- Auxiliary Carry =0

OUT 00- muestra el resultado.
HLT- Detiene el programa.

Resultado de las operaciones es FF.

III. Análisis y datos
Parte 1
Programa #1

Address	Assembly	Hex Code	Comments
0000 0001 0002	LDA 0020	3A 20 C0	Cargar lo que está en el address 0020 al Acumulador.
0003	MOV B, A	47	Mover a B lo que está en el Acumulador.
0004 0005 0006	LDA 0021	3A 21 C0	Cargar lo que está en el address 0021 al Acumulador.
0007	SUB B	90	Restar a lo que está en B lo que está en el Acumulador
0008	JC	DA	Brincar al Label
0009 000A	LABEL 1	0E C0	1 si hay carry.
000B 000C 000D	LDA 0022	3A 22 C0	Cargar lo que está en el address 0022 al Acumulador.
000E 000F	LABEL 1: ADI 20	C6 20	Añadir inmediatamente 20 al Acumulador.
0010 0011	ANI 20	E6 20	Comparar lo que hay en el Acumulador con el dato 20.
0012 0013 0014	JZ LABEL 2	CA 1A C0	Brincar al Label 2 si el resultado es cero.
0015 0016	MVI A, AA	3E AA	Mover inmediatamente AA al Acumulador.
0017 0018 0019	JMP DISPLAY	C3 1C C0	Brincar incondicionalmente al display.
001A 001B	LABEL 2: MVI A, BB	3E BB	Mover inmediatamente BB al Acumulador.
001C 001D	DISPLAY: OUT 00	D3 00	Mostrar el resultado en el display al darle reset.
001E	HLT	76	Detener el programa.

Estado de los Flags luego de cada instrucción lógica y aritmética:
SUB B- Esta instrucción ocasionó que se restara 11-51.

0001 0001 → 0001 0001
- 0101 0001 → +1010 1111
0|1100 0000

Numero complementario de 1100 0000 es: 0100 0000 → -40
Flags: Z=0 : C=0 ; P=0 ; S=0 ; AC=1

ANI 20 – Comparar lo que esta en el acumulador con 20

0011 0010
(AND)0010 0000
0010 0000 → 20
Flags: Z=0 ; C= 0 ; P= 0 ; S= 0 ; AC= 0

Contenido de registros y memoria al finalizar el programa.
A=AA; B=51; 0020 = 51; 0021 = 11; 0022 = 32

Observaciones: Al correr el programa observamos que no se llevaron a cabo ninguno de los jumps establecidos ni la instrucción que este debía hacer en el programa.

Parte 2
Programa #2

Address	Assembly	Hex Code	Comments
0000 0001 0002	LXI H, 0020	21 20 C0	Colocar inmediatamente lo que está en el address 0020 en Memoria.
0003	MOV A, M	7E	Mover al Acumulador lo que está en memoria.
0004	INX H	23	Incrementar memoria del address 0020 al 0021.
0005	MOV B, M	46	Mover a B lo que está en memoria.
0006 0007	IN 00	DB 00	Busque la primera instrucción.
0008 0009	ANI 08	E6 08	Comparar el 08 con la primera instrucción.
000A 000B 000C	JZ LABEL 1	CA 11 C0	Si el resultado es cero brincar al label 1.
000D	SUB B	90	Restar B.
000E 000F 0010	JMP DISPLAY	C3 12 C0	Brinque incondicionalmente al display.
0011	LABEL1: ADD B	80	Sumar B a 0.
0012 0013	DISPLAY : OUT 00	D3 00	Muestre el resultado en el display al darle reset
0014	HLT	76	Detener el programa.

Estado de los Flags luego de cada instrucción lógica y aritmética.

ANI 08 – Comparar 08 con el contenido del acumulador.
0101 0001
(AND)0000 1000
0000 0000

Flags: Z= 1 : C= 0 ; P= 0 ; S=0 ; AC= 0 ;

ADD B – sumar los contenidos del acumulador con los contenidos del registro B
0000 0000
+0001 0001
0001 0001 → 11
Flags: Z= 0 : C= 0 ; P= 1 ; S=1 ; AC= 0

Contenido de registros y memoria al finalizar el programa.
A=11; B=11; 0020 = 51; 0021 = 11; 0022 = 32

Observaciones: Al correr el programa observamos que este ejecuta un jump cuando encuentra un cero en el acumulador y ejecuta la instrucción correspondiente a la dirección que lo envía el jump.

IV. Conclusión

A través de este experimento, pudimos ver como es que funcionan las operaciones aritméticas y lógicas del microprocesador 8085 y aprendimos a utilizar las instrucciones de jump y además se utilizaron las demás operaciones ya vistas en el laboratorio.

V. Referencias

1)http://computer.howstuffworks.com