Con las instrucciones que soporta este simulador se pueden escribir una gran cantidad de programas para resolver muchos problemas diferentes.

Las describiré así:

XX - INST [parámetro]

Donde:

XX significa el Código de la Instrucción

INST es la instrucción

[parámetro] es el parámetro si esta tiene o [parametro1,parametro2] si el parámetro es doble Acá pondré algunos ejemplos

Estas son las instrucciones soportadas por la versión actual:

01 - LDA [mem]

Cargue en AX el contenido de la dirección de Memoria especificada. Digamos que en la posición de memoria 1F esta el valor 10111, después de ejecutada la instrucción LDA 1F se obtiene que AX=10111

Es equivalente a usar la instrucción MOV AX,1F

Hay casos donde es mejor usar MOV si se desea pasar datos sin tener que pasarlos por AX.

02 - STA [mem]

Guarde el contenido de AX en la dirección de Memoria especificada. Supongamos que tengo el valor 1010110 en el registro AX y quiero llevarlo a la posición de memoria 3C, la instrucción es STA 3C

Es equivalente a usar la instrucción MOV 3C,AX

Es mejor usar MOV debido a que si quiero pasar algún dato de una dirección de memoria a otra usando LDA y STA serian dos instrucciones: LDA mem1 y luego STA mem2, mientras que con MOV será así: MOV mem2,mem1

03 - XAB

Intercambia los valores de los registros AX y BX Esta instrucción no necesita parámetros.

04 - CLA

Hace AX = 0

06 - PUSH [registro]

Envía el valor del registro especificado a la pila

07 - POP [registro]

Trae de la Pila el ultimo Valor llevado por PUSH (indicado por el registro SP) y lo almacena en el registro especificado.

Nota: En todas las instrucciones desde 08 hasta la 18, Dest puede ser una dirección de Memoria o un Registro

08 - INC [dest]

Incrementa en 1 el destino especificado, el parámetro puede ser una dirección de memoria o un registro. Si en la posición de memoria EB esta el valor 1001, al ejecutar INC EB se obtiene que ahora el valor de EB es 1010.

09 - DEC [dest]

Decremento en 1 el destino especificado, Si el destino queda = 0, se vuelve Z = 1

10 - MOV [dest,orig]

Copia el valor almacenado en el origen al destino. El destino y/o origen pueden ser registros o direcciones de memoria o combinación de estos. Para copiar lo que esta en la posición de memoria 12E a la posición D2 se usa la instrucción MOV D2,12E

11 - AND [dest,orig]

Y lógico, hace un Y lógico entre todos los bits de los dos operándos escribiendo el resultado en el destino. Los parámetros pueden ser direcciones de memoria o Registros. La siguiente regla aplica:

1 AND 1 = 1

1 AND 0 = 0

0 AND 1 = 0

0 AND 0 = 0 Si en AX tengo el numero 1001101 y en la pos 3F tengo el numero 11011. al ejecutar la instrucción AND AX,3F obtendré en AX el resultado 1001.

El Y lógico lo que hace es dejar los bits en común que tengan los dos números.

12 - NOT [destino]

NO lógico, invierte los bits del operando formando el complemento del primero.

NOT 1 = 0

NOT 0 = 1 Si en AX tengo 10011 al ejecutar NOT AX obtengo AX=1111111111101100

13 - OR [dest,orig]

O inclusive lógico, todo bit activo en cualquiera de los operándoos será activado en el destino. La siguiente regla aplica:

1 OR 1 = 1

1 OR 0 = 1

0 OR 1 = 1

0 OR 0 = 0 Si en 3A tengo el numero 1001101 y en la pos 3B tengo el numero 11011. al ejecutar la instrucción OR 3A,3B obtendré en 3A el resultado 1011111.

14 - XOR [dest,orig]

O exclusivo, realiza un O exclusivo entre los operándoos y almacena el resultado en destino. La siguiente regla aplica:

1 XOR 1 = 0

1 XOR 0 = 1

0 XOR 1 = 1

0 XOR 0 = 0 Si en 3A tengo el numero 1001101 y en la pos 3B tengo el numero 11011. al ejecutar la instrucción XOR 3A,3B obtendré en 3A el resultado 1010110.

15 - ROL [dest,veces]

Rota los bits a la izquierda las veces especificadas(en decimal), los bits que salen por la izquierda re-entran por la Derecha. En el Carry Flag queda el ultimo bit rotado. Supongamos que en la posición 7E tengo el numero 101110

Al Ejecutar... obtengo en 7E C=

ROL 7E,2 10111000 0

ROL 7E,7 1011100000000 0

ROL 7E,13 1100000000000101 1

16 - ROR [dest,veces]

Rota los bits a la derecha las veces especificadas(en decimal), los Bits que salen por la derecha re-entran por la izquierda. El Carry Flag guarda el ultimo bit rotado.

17 - SHL [dest,veces]

Desplaza los bits a la izquierda el numero de veces especificado(en decimal), agregando ceros a la derecha, el Carry Flag guarda ultimo bit desplazado.

18 - SHR [dest,veces]

Desplaza los bits a la Derecha el numero de veces especificado(en decimal), agregando ceros a la izquierda, el Carry Flag guarda ultimo bit desplazado. Supongamos que en la posición 1A tengo el numero 101110

Al Ejecutar... obtengo en 1A C=

SHR 1A,2	1011	1
SHR 1A,6	0	1
SHR 1A,11	0	0

20 - ADD [mem]

Sumar:

AX = AX + el contenido de la dirección de memoria. Si el resultado de la suma supera los 16 bits, el resultado queda asi: en BX los bits mas significativos y en BX los menos, tambien se activa el Overflow flag.

21 - SUB [mem]

Restar:

AX = AX - el contenido de la dirección de memoria.

22 - MUL [mem]

Multiplicar:

AX = AX * el contenido de la dirección de memoria.

Si el numero resultante supera su longitud en binario de 16 bits, este resultado se parte almacenando los bits mas significativos en el Registro BX. Si el resultado de una operación fuera 101101000111100010111 (21 bits) Los registros quedarían así:

A = 1000111100010111 (los últimos 16bits)

B = 10110 (los primeros Bits (los mas significativos))

También se activa el Flag Overflow, para indicar que en la ultima operación sucedió esto.

23 - DIV [mem]

Dividir:

AX = AX / el contenido de la dirección de memoria, BX=AX % el contenido de la dir de memoria (BX = modulo o residuo).

24 - CLN

Limpia el Negative Flag. N = 0

25 - CLC

Limpia el Carry Flag. C = 0

26 - STC

Pone el Carry Flag. C = 1

27 - CMC

Complementa (invierte) el Carry Flag. Si C = 1 vuelve C = 0 y viceversa.

29 - LOOP [mem]

Decrementa CX y salta a la Pos de memoria si CX no es cero.

30 - JMP [mem]

Salto incondicional. PC = dirección de memoria donde esta la siguiente instrucción a ejecutar"

31 - JEQ [mem]

Saltar si son iguales.

Si Z = 1, PC = contenido de la memoria. Función equivalente en C:

if (AX == mem)

32 - CMP [mem]

Compara AX con [mem], si AX es mayor, Z=0 N=0, si es igual Z=1 N=0, si es menor Z=0 N=1 Supongamos que en AX tengo el numero 10110 y en la posición de memoria 4G tengo el numero 1100, al ejecutar la instrucción CMP 4G obtengo que como el numero almacenado en AX es mayor entonces los Flags de Control me quedan así: Z=0 y N=0

Nota: Solo en las versiones 1.3.6.2 y anteriores En AX quedaba el resultado de la resta (de la comparación), Ahora ya no. Sugerencia: si necesita el valor original de AX puede usar la pila para almacenarlo temporalmente.

33 - JME [mem]

Saltar si es Menor.

Si N = 1, PC = contenido de la memoria. Supongamos que ejecuto esta instrucción así JME 3F inmediatamente después de ejecutar la instrucción del ejemplo que coloque en la instrucción 32, al ejecutar JME 3F se verifica el Flag N, y como en este caso se encuentra en 0 (porque el numero no es menor) entonces no se realiza dicho Salto a 3F porque el valor de PC no se modifica, el programa sigue su ejecución.

Función equivalente en C:

if (AX < mem)

y Si necesitas hacer un:

if (AX <= mem)

inmediatamente despues de la instruccion JME colocas una JEQ

34 - JMA [mem]

Saltar si es Mayor.

Si Z = 0 y N = 0, PC = contenido de memoria. Supongamos que ejecuto esta instrucción así JMA 2B inmediatamente después de ejecutar la instrucción del ejemplo que coloque en la instrucción 32, al ejecutar JMA 2B se verifican los Flag N y Z, y como en este caso los dos son 0 (porque el numero es menor) entonces si se realiza dicho Salto a 2B ya que el valor del PC se modifica, el programa sigue su ejecución saltando a la dir de mem especificada.

Función equivalente en C:

if (AX > mem)

35 - JC [mem]

Saltar si el Carry Flag esta activado.

Si C = 1, PC = contenido de memoria.

36 - JNC [mem]

Saltar si el Carry Flag no esta activado.

Si C = 0, PC = contenido de memoria

37 - JO [mem]

Saltar si el Overflow Flag esta Activado.

Si O = 1, PC = contenido de memoria

38 - JNO [mem]

Saltar si el Overflow Flag no esta activado.

Si O = 0, PC = contenido de memoria

39 - JNE [mem]

Saltar si no son iguales.

Si Z = 0, PC = contenido de memoria. Función equivalente en C:

if (AX != mem)

40 - LDT

Lee un valor del Teclado y lo lleva al registro AX Esta instrucción es para comunicarse con el usuario, pidiéndole que entre un Dato; Puede colocar una descripción del dato que pide, que se mostrará en tiempo de ejecución.

41 - EAP

Escribe en Pantalla el contenido del registro AX Esta instrucción también es para comunicarse con el usuario; Puede colocar una descripción del dato que se entrega, este se mostrará en tiempo de ejecución.

42 - MSG

Muestra un mensaje en pantalla Ej: MSG "Hola Mundo !!"

50 - LDB [mem]

La instrucción carga en AX el contenido de memoria almacenado en [mem] + BX

ej: Digamos que BX=10 ; LDB 1A carga el contenido de 1C en AX

51 - STB [mem]

guarda el contenido de AX en la dirección [mem] + BX ej: Digamos que BX=101 ; STB 3A guarda AX en 3F

55 - LDF [mem]

Carga en BX y AX un numero de 32 bits (IEEE) que esta almacenado en la dir [mem] y mem+1. En BX quedan los digitos mas Significativos Ej:Supongamos que el número 01000010110010001000000000000000 está cargado en memoria así:

02A 0100001011001000 (Los dígitos mas significativos)

02B 1000000000000000 (Los dígitos menos significativos)

Un LDF 2A dejaría el siguiente resultado:

BX: 0100001011001000

AX: 1000000000000000

Nota: Para pedirle al usuario o mostrar estos numeros IEEE 754 en pantalla, usar puerto 1, con las instrucciones IN AX,1 y OUT 1,AX

56 - STF [mem]

Guarda en [mem] y mem+1 el contenido de BX y AX Ej: siendo AX y BX = al ejemplo anterior, un STF 2A deja la memoria como el ejemplo anterior.

60 - ADDF [mem]

Suma números de 32 bits: En BX y AX queda el resultado de la suma de estos mas el contenido de [mem] y mem+1 Estos numeros IEEE 754 son numeros que pueden ser de punto flotante, o enteros desde -2147483647 hasta 2147483647, si en cualquier operación de estas aritmeticas, se sobrepasa este valor, se activa el Overflow flag.

61 - SUBF [mem]

Resta el numero de 32 bits: BX y AX = BX y AX - [mem]y mem+1 Puedes utilizar esta instrucción como un CMP para numeros de 32 bits.

62 - MULF [mem]

Multiplicación:

BX y AX = BX y AX * [mem]y mem+1 Si el resultado es > 2147483647, Resultado = 2147483647 y Overflow Flag = 1

63 - DIVF [mem]

Division:

BX y AX = BX y AX / [mem]y mem+1, en CX queda el residuo de la division en entero de 16 bits

64 - ITOF

Conversión de Entero a Real:

Convierte un número entero (16bits) almacenado en AX al mismo numero pero representado en Real IEEE754(32bits), el Resultado de la conversión queda en BX (bits mas significativos) y AX

Los registros de control cambian de acuerdo al numero convertido: "Z" si el numero es cero, "N" si el numero es negativo.

65 - FTOI

Conversión de Real a Entero:

Convierte un número Real(32bits) a su equivalente en entero BX y AX en un entero (16bits), el Resultado queda en AX.

Los registros de control cambian de acuerdo al numero convertido: "Z" si el numero es cero, "N" si el numero es negativo, "O" si el numero real es mayor de 65535.

80 - IN registro,puerto

Lleva al Registro el valor retornado por el puerto especificado. Ej: IN AX,8 ;lleva a AX el valor retornado por el puerto 8 (Reloj: los segundos del sistema).

81 - OUT puerto,registro

Escribe en el puerto especificado, el valor del registro.

90 - NOP

Esta operación no hace nada. Útil para cuando se modifica la memoria para parchar código y desactivar instrucciones.

99 - HLT

Terminar Programa Todo Programa lleva esta instrucción para indicarle al simulador que el programa ha terminado su ejecución.