Instrucciones del microprocesador

Iniciaremos el estudio de algunas instrucciones.

Instrucción MOV
Instrucción de almacenamiento.

Sintaxis :
```
MOV   reg, imm
MOV   reg, mem
MOV   reg, reg
MOV   mem, imm
MOV   mem, reg
```

Reg: Registro listados en la tabla de registros
Imm: Es un valor inmediato, hace referencia a un valor numérico decimal o hexadecimal.
Mem: Localidad de memoria

No entre dos localidades de memoria, datos inmediatos a registros de segmento, registro de segmento a registro de segmento

Ejemplos:

```
MOV  AX,54          MOV AL, 999   ──╲  No es posibles
MOV  AX,36H         MOV AL, 3E7H  ──╱
                          ↑        ↑
                    Longitud: 8 bits  12 bits
```

```
MOV  EAX,  EBX
MOV  EBX, DX        No es válido deben de ser del mismo tamaño.
```

En un ensamblador de archivo esto es un número no un registro

```
MOV    AL,   0A_H  ó  MOV  AL,  10
```

Instrucción ADD
Suma

Sintaxis:

ADD reg,imm

Ejemplo: ADD BL, 10
BL= BL +10

ADD reg, reg Sumar el contenido del 2° registro al contenido del 1º.

Ejemplo: ADD BL, AL ; No cambia el valor de AL

BL= BL + AL

ADD reg,mem
ADD mem,reg
ADD mem, mem
ADD mem,imm

Instrucción SUB
Resta

Sintaxis: SUB reg, imm
SUB reg, reg
SUB reg,mem
SUB mem,reg
SUB mem, mem
SUB mem,imm

Ejemplos: SUB BL, 10
BL = BL -10

SUB BL, AL

BL = BL-AL

Instrucción MUL
Multiplicación sin signo

Sintaxis: MUL reg/mem

Instrucción IMUL
Multiplicación con signo

Sintaxis: IMUL reg/mem

El registro puede ser cualquiera de los 24 listados en la tabla de registros (este tiene la función del multiplicador).

En las tablas siguientes se muestra el tipo de multiplicaciones que se pueden realizar de acuerdo a la longitud de los datos, así como los registros que se comprometen antes y después de la operación de multiplicación.

Byte por byte (8 bits)

	AH	AL
Antes	Ignorado	Multiplicando
Después	← AX Producto →	

Palabra por palabra (16 bits)

	DX	AX
Antes	Ignorado	Multiplicando
Después	Parte alta del Producto	Parte baja del producto

Palabra doble por palabra doble

	EDX	EAX
Antes	Ignorado	Multiplicando
Después	Parte alta del producto	Parte baja del producto

Ejemplos:

Si el registro es de 8 bits
MUL BH AX = AL*BH

Números de 16 bits
MUL BX DX : AX = AX*BX

16 bits más altos *16 bits más bajos*

Números de 32 bits
MUL EBX EDX: EAX = EAX* EBX

32 bits más altos *32 bits más bajos*

Instrucción DIV
División sin signo

Sintaxis: DIV reg/mem

Instrucción IDIV
División con signo

Sintaxis: IDIV reg/mem

El registro puede ser cualquiera de los 24 listados en la tabla de registros (este tiene la función del divisor).

En las tabla siguiente se muestra el tipo de divisiones que se pueden realizar de acuerdo a la longitud de los datos, así como los registros que se comprometen antes y después de la operación de dividir.

Bits	Dividendo	Residuo	Cociente
32 bits	EDX : EAX	EDX	EAX
16 bits	DX : AX	DX	AX
8 bits	AX	AH	AL

Ejemplo:

```
Mov  AX, 17
Mov BH, 3
DIV BH.
```

Residuo AH =2
Cociente AL =5

Ejemplo:

```
MOV  AX, 400h
MOV  BH,2
DIV    BH
```

Produce un error de división overflow puesto que el cociente es de más de 8 bits (200h)

OBSERVACIÓN:
Al realizar movimiento de un dato de una zona de memoria del segmento DS a un registro, considerar las siguientes situaciones:

Ejemplo 1:

MOV CX, [0101]

Se almacenará en el registro los datos de la siguiente forma:

CH	CL
[0102]	[0101]

Ejemplo 2:

MOV CH, [0101]

CH
[0101]

Ejemplo 3:

MOV [0101], DX

[0101]	[0102]
DL	DH

Práctica 1 en sala de Cómputo

Dentro del DEBUG realizar:

1. Realizar un programa que tome 2 datos de un byte de la memoria de las direcciones 100 y 101.
2. Sumarlo, restarlos, multiplicarlos y dividirlos.
3. Guardar cada resultado en la memoria en las siguientes direcciones:

   Resultado de la suma en la dirección 110.

   Resultado de la resta en la dirección 112.

   Resultado de la multiplicación en la dirección 114 y 115.

   Resultado de la división en la dirección 117 y 118.

El código generado queda como sigue:

Código de la Suma:

```
mov al, [100]
mov ah, [101]
add al, ah
mov [110], al
```

Código para la Resta:

```
mov bl, [100]
mov bh, [101]
sub  bl, bh
mov [112], bl
```

Código para la Multiplicación a 8 bits:

```
mov al, [100]          ; en AL se deposita el multiplicando
mov ch, [101]          ; cualquier registro almacenará el multiplicador
mul ch
mov [114], al          ; Recuerde que el producto queda en AX
mov [115], ah          ;Estas  dos intrucciones se puden sustituir por:
                               MOV [114], AX
```

Código para la División a 8 bits:

```
mov ah, 00
mov al, [100]          ; en AX se deposita el dividendo
mov ch, [101]
div ch                 ; En cualquier otro registro el divisor
mov [117], al          ;Recuerde que en AH queda el residuo y en AL el cociente
mov [118], ah          ;Estas  dos intrucciones se puden sustituir por:
                               MOV [117], AX
NOP
```

Instrucciones de Control

- ***Saltos Incondicionales***
- ***Saltos Condicionales***
- ***Saltos basándose en una comparación***

Salto Incondicional JMP

Formato: JMP mem/etiqueta

Transfiere la ejecución a la dirección especificada por la etiqueta, la etiqueta es creada por el programador.

NOTA: Algunas instrucciones contienen la misma etiqueta de salto, pero dos líneas no pueden tener la misma etiqueta en el campo de etiqueta. La etiqueta es una cadena de caracteres.

Ejemplo: 2+2+2+2+2..... (Ciclo infinito)

```
        MOV  AX, 0
        MOV BX,  2
Salta : ADD   AX,   BX     ; Agregando 2 al contenido de AX
        JMP   Salta
```

Saltos Condicionales

Es un salto que toma como base un valor de verdad. La información de la cual toma la decisión está basada en el contenido de un bit llamado bandera (flags). Dos banderas importantes son la bandera cero (ZF) y la bandera de signo (SF), se utilizan para verificar el resultado de algunas operaciones, los comandos ADD y SUB no solo afectan sus registros destino si no también a algunas banderas.

Ejemplo: ADD AX, BX

Resultado AX si es igual a cero ZF igual 1
AX diferente de cero ZF igual 0

El valor para la bandera de signo SF después de esta instrucción es el valor del bit más significativo del resultado.

SF= 1 Resultado negativo
SF= 0 Resultado positivo

Instrucciones de salto condicional

Formato	Descripción
JZ etiqueta	Salta si la ZF = 1
JS etiqueta	salta si la SF = 1
JNZ etiqueta	salta si la ZF = 0
JNS etiqueta	salta si SF = 0

Ejercicio 1: Dados dos números enteros que se encuentran almacenados en BX y AX determinar cual es el mayor.

```
        MOV   AX , valA   ; Valor A
        MOV  BX, valB     ; Valor B
        MOV DX, BX        ; Resguardo de  Bx
        SUB BX, AX        ; operación de resta para determinar el número más grande
        JZ IGUALES
        JS  MAYA
        MOV BX,DX         ; Recuperando BX
        ----------        ; Imprimir "Valor  2 mayor"  (BX)
        JMP END
MAYA: ------------        ; Imprimir "Valor  1 mayor"  (AX)
        JMP END
IGUALES:
        ----------
        ----------        ; Imprimir "Valores IGULES"
END:    NOP
```

Ejercicio 2: Multiplicación por sumas sucesivas

```
        MOV Ah,0
        MOV  Bh, 8
        MOV  Ch, 5
RE:     ADD Ah, Bh
        SUB   Ch,1
        JZ END
        JMP RE
END:    ----------              ; Imprime resultado
```

Ejercicio 3: Dados tres números enteros diferentes, hallar el mayor.

Pseudocódigo:

```
if (A>B) then
        if (A>C) then
                        MAX ← A
        else
                        MAX←C
else if (B>C) then
                        MAX←B
        else
                        MAX←C
```

Utilizar los registros Ax, Bx, Cx para los valores a comparar y en Dx almacenar el valor mayor.

```
        Mov ax, val1
        Mov bx, val2
        Mov cx, val3
        Mov dx, ax      ; Resguardando el valor ax
        Sub ax,bx
        Js MAYB
        Mov ax, dx      ; Recuperndo Ax
        Sub ax,cx       ; Mayor A, el valor  de Ax ya esta en Dx
        Js  MAYC
        Jmp FIN
MAYB:   mov dx,bx       ; Resguardando Bx
        sub bx, cx
        Js  MAYC        ; B Mayor, el valor de Bx ya esta en Dx
        Jmp FIN
MAYC:   mov dx,cx       ; C Mayor, el valor  de Cx  se mueve a Dx
FIN:    nop
```

Saltos basándose en una comparación

Instrucción CMP

Compara dos operandos. Realiza una substracción del operando destino menos el fuente, el operando destino no sufre cambios, solo se afecta el estado de las banderas.

SINTAXIS: CMP destino, fuente

Donde uno o ambos operandos están contenidos en registros.

Instrucciones de salto utilizados por números con signo

Instrucción	Descripción	Banderas afectadas
JL	Salta si es menor	SF, OF AX< BX
JLE	Salta si es menor que o igual	ZF, SF, OF AX ≤ BX
JG	Salta si es mayor que	ZF, SF, OF AX >BX
JGE	Mayor que o igual	SF, OF AX ≥ BX
JE	Igual	ZF AX = BX
JNL	No menor que	SF, OF AX ≥ BX
JNLE	No menor que o no igual	ZF, SF, OF AX >BX
JNG	No mayor que	SF, ZF, OF AX ≤ BX
JNGE	No mayor que o no igual	SF, OF AX < BX
JNE	No igual	ZF AX > < BX

Ejemplo 1: Dados dos números diferente con signo. Determinar cual es el mayor

```
        MOV  AX, A
        MOV  BX, B
        CMP  AX, BX
        JG    SAL
        ----------------        ;Imprime BX es mayor
        JMP  END
SAL:    ----------------        ;Imprime AX es mayor
END:    NOP
```

Instrucciones de salto utilizados por números sin signo

Las palabras arriba de y de bajo de, son usados por mayor que o menor que.

Instrucción	Descripción (Salta si)	Banderas afectadas
JB	Debajo de	CF
JBE	Debajo de o igual	CF, AF
JA	Arriba de	CF, ZF
JAE	Arriba de o igual	CF
JE	Igual a	ZF
JNB	No debajo de	CF
JNBE	No debajo de o no igual	CF, ZF
JNA	No arriba de	CF, AF
JNAE	No arriba de o no igual	CF
JNE	No es igual	ZF

Ejemplo 2: Dado un número determinar si es par o impar

```
        MOV   AX, B
        MOV   BH, 2
        DIV BH
        CMP AH,0
        JE SALTA
        ----------          ;   Imprime AX es impar
        JMP FIN
SALTA: --------             ;   Imprime AX es par
  FIN :  NOP
```

Ejemplo 3: Tabla de multiplicar del 5, Byte por Byte con signo

```
        MOV AL, 5
        MOV BL, 1
RE :    IMUL BL             ;  AX = AL *BL
        ------------------  ;   Imprimir resultado
        MOV AL, 5
        INC BL              ; incrementa a uno
        CMP BL, 0Ah
        JLE RE
        NOP
```

Instrucción de control: LOOP

Realiza un ciclo un número especifico de veces o hasta alcanzar una condición particular. Esta instrucción utiliza al registro CX con un valor inicial, el cual disminuye en 1 en cada iteración, si el valor en CX es cero pasa el control a la instrucción que sigue; si el valor en CX no es cero, el control pasa a la dirección del operando.
No afecta a ninguna bandera.

Ejemplo 1:

```
        MOV AX,0
        MOV BX,0
        MOV CX,A
CICLO:  INC AX
        ADD BX,AX
        LOOP CICLO
        NOP
```

Ejemplo 2:

```
        MOV AX,0
        MOV DX,0
        MOV CH,0
        MOV CL,[100]
        MOV BX,101
CICLO:  INC AX
        ADD DX,AX
        MOV [BX],DX
        INC BX
        INC BX
        LOOP CICLO
        NOP
```

Ejemplo 3:

```
        MOV AX,1
        MOV BX,1
        MOV DX,1
        MOV CX,A
CICLO:  INC AX
        ADD BX,AX    ; ACUMULA LA SUMA EN BX
        SHL DX,1     ; MULTIPLICA A DX POR 2
        LOOP CICLO
        NOP
```

Ejemplo 4:

```
        Mov cx,A
Otro:
        Mov al,7
        Mul cl
        Loop Otro
        NOP
```

EJERCICIO: Modificar el ejemplo que realiza la tabla de multiplicar del 5 (Byte por Byte) con signo, almacenando los resultados a partir de la localidad 100 del segmento DS. Utilice la instrucción LOOP

Instrucción de incremento y decremento

Instrucción INC
INC reg Incrementa en 1 reg es cualquier registro de los 24 listados.

Instrucción DEC
DEC reg Decrementa en 1

Usados comúnmente para incremento y decremento en un ciclo.

Instrucciones de transferencia de control

CALL Llamado a subrutina

RET Regreso de subrutina

Práctica 2 en sala de Cómputo

Dentro del DEBUG realizar:

1. Introducir con el comando *E* la cadena "Hola mundo", en la dirección 200.
2. Revisar con el comando D qué datos tiene la memoria en la dirección 200 y 220.
3. Con el comando *A* introducir:

```
LEA SI,[0200]
LEA DI,[0220]
MOV CX,0010
MOV AH,[SI]
MOV [DI],AH
INC SI
INC DI
LOOP 010B
NOP
```

4. Trazar el conjunto de instrucciones con el comando *T*.
5. Revisar la memoria con el comando *D*, comparando la dirección 200 con la 220.
6. Contestar:
   ¿Qué hace el programa?

Práctica 3 en sala de Cómputo

Dentro del DEBUG realizar:

1. Con el comando *A* introducir:

```
            MOV DX, VAL_POTENCIA
            MOV CX, DX
            DEC CX
            MOV AX, VAL_BASE
REGRESA:    MOV BX, VAL_BASE
            MUL BX
            LOOP REGRESA
            NOP
```

2. Trazar el conjunto de instrucciones con el comando *T*.

El código del punto 1 es una solución para obtener la potencia de un número, como este resultado crece rápidamente se decidio tomar longitudes de registro de 16 bits, entonces el resultado del producto quedará en DX y AX como se explicó anteriormente.