Programación II - 2014

Repaso Simulador

El Simulador

Registros **AX, BX, CX y DX** : uso general, 16 bits de longitud, se pueden dividir en 2 partes de 8 bits cada uno. Ejemplo: AX en AH y AL.

Registro **IP** (instrucción pointer) contiene la dirección de memoria de la próxima instrucción a ser ejecutada.

Registro **SP** (stack pointer) contiene la dirección de memoria del tope de la pila.

Registro de flags: muestra el estado de las banderas o flags luego de cada operación.

- Bandera de cero: identificada por la letra **Z**.
- Bandera de overflow: identificada por la letra **O.**
- Bandera de carry/borrow: identificada por la letra **C**.
- Bandera de signo del número: identificada por la letra **S**.

Variables

nombre_variable especificador_tipo valor_inicial

Especificador	Tipo	Tamaño
DB	Byte	8 bits
DW	Word	16 bits

Var1 DB 10
Var2 DW 0A000h

Podemos observar que los valores numéricos se interpretan en decimal, a menos que terminen con una letra 'h', que en cuyo caso se interpretarán como valores en hexadecimal. Además, como los números deben comenzar con un dígito decimal, en el caso del A000h, se **antepone un cero para evitar que se la confunda con una variable que se pueda llamar A000h.**

Modos de Direccionamiento

INMEDIATO

MOV AX, 1000h

El operando contiene la información sobre la que hay que operar. (útil para inicializar registros)

DIRECTO DE MEMORIA O ABSOLUTO

MOV BL, var_byte

La instrucción contiene la dirección de memoria exacta donde se encuentra el operando. El operando se encuentra en memoria.

DIRECTO DE REGISTRO

MOV BX, AX

El operando se encuentra contenido en un registro

INDIRECTO CON REGISTRO

MOV AX, [BX]

La instrucción contiene una dirección que se emplea para leer en memoria una dirección intermedia que será la verdadera dirección del objeto buscado. El operando se encuentra en memoria.

INDIRECTO CON DESPLAZAMIENTO

MOV AX, 20h+[BX]

Similar al anterior al que se le agrega un desplazamiento para obtener la dirección final donde se encuentra el operando.

Instrucción MOV

MOV destino, origen

ORG 1000h ***Variables:*** *almacenadas a partir del valor 1000h de la memoria*

var_byte DB 20h

ORG : permite cambiar la dirección a partir de la cual se colocará lo que está a continuación de la misma

ORG 2000h ***Programa principal:*** *a partir de la dirección 2000h(valor por defecto)*

MOV AX, 1000h *Modo de direccionamiento INMEDIATO*

MOV BX, AX *Modo de direccionamiento DIRECTO DE REGISTRO*

MOV BL, var_byte *Modo de direccionamiento DIRECTO DE MEMORIA*

HLT

END

Instante	AX		BX	
	AH	**AL**	**BH**	**BL**
0	00	00	00	00
1	**10**	**00**	00	00
2	10	00	**10**	**00**
3	10	00	10	**20**

OFFSET

```
ORG 1000h
var_otro DW  0ABCDh
ORG 2000h
MOV BX, OFFSET  var_otro
HLT
END
```

Se mueve a BX OFFSET var_otro . Esto indica que se debe cargar en BX la dirección de var_otro **y NO** el contenido de dicha variable.

Solamente uso OFFSET para obtener direcciones de memoria de variables declaradas con anterioridad.

Instrucciones Aritméticas e Incremento/ Decremento

```
ORG 1000h
dato1 DW 10
dato2 DW 20
resultado DW ?

ORG 2000h
MOV AX, dato1
ADD AX, dato 2
MOV resultado, AX
END
```

ADD operando1, operando2
SUB operando1, operando2

Utilizo el registro AX como variable auxiliar, de manera de no afectar a dato1 .

Existen dos instrucciones ,similares a las anteriores, que tienen en cuenta al flag de carry/borrow :

ADC operando1, operando2
SBB operando1, operando2

Existen dos instrucciones que Incrementan o Decrementan al operador en 1 unidad

INC operando1
DEC operando1

Instrucciones Lógicas

Utilizan 2 operadores

AND operando1, operando2
OR operando1, operando2
XOR operando1, operando2

Bit a bit (resultado en operando 1)

Utilizan 1 operador

NOT operando — Complemento a1

NEG operando — Complemento a2

Instrucción de Comparación

CMP es esencialmente equivalente en funcionamiento a **SUB** pero el resultado de la resta no se almacena en ninguna parte, por lo que ninguno de los operandos se ve. **Si se modifican los flags los cuales puedo usar para saber cosas.**

flag Z= 1 (el resultado de la resta fue cero) -> Numeros iguales

flag S= 1 (signo) con operandos en CA2 -> El segundo operando mayor que el primero

Instrucciones de Salto

Las instrucciones de **salto** permiten realizar saltos alterando el flujo de control a lo largo de la ejecución de un programa. Estas instrucciones tienen un operando que indica la dirección que se le asignará al registro IP.

Existen dos tipos de :

- **Saltos incondicionales: S**e producen siempre que se ejecuta la instrucción
- **Saltos condicionales. D**ependen de alguna condición para que se produzca ,o no ,dicho salto.

Condicionales

JMP dirección ; Salta siempre (**Incondicional**)
JZ dirección ; Salta si el flag Z=1
JNZ dirección ; Salta si el flag Z=0
JS dirección ; Salta si el flag S=1
JNS dirección ; Salta si el flag S=0
JC dirección ; Salta si el flag C=1
JNC dirección ; Salta si el flag C=0
JO dirección ; Salta si el flag O=1
JNO dirección ; Salta si el flag O=0

Z= Cero
O= Overflow
C= Carry/Borrow
S= Signo

Etiquetas para Saltos

Al igual que lo que ocurría con las variables, el ensamblador nos facilita indicar la dirección a la que se quiere saltar mediante el uso de una etiqueta, la cual se define como un nombre de identificador válido seguido de dos puntos .

```
ORG 2000h
MOV AX, 10
Lazo: ... ;
... ; <Instrucciones a repetir>
... ;
DEC AX
JNZ Lazo
HLT  Instrucción para detener la ejecución
END
```

•¿Cuál es el contenido final del registros AL?

b) En el programa reemplaze la instrucción JNZ por las siguientes e indique en cada caso el contenido final del registro AL:

1. JS → Salto Condicional : Salta si S(signo)= 1
2. JZ → Salto Condicional: Salta si Z(signo)= 1
3. JMP → Salto incondicional

```
        ORG 1000H
INI     DB  0
FIN     DB  15
        ORG 2000H
        MOV AL, INI
        MOV AH, FIN
SUMA:   INC AL
        CMP AL, AH
        JNZ SUMA
        HLT
        END
```

Salto Condicional : Salta Si el flag Z(signo)=0

Instrucciones Push y Pop

Trabajan solo con registros de 16 bits

PUSH fuente	Carga a fuente en el tope de la pila
POP destino	Desapila el tope de la pila y lo carga en destino

Ejemplos de la práctica

```
MOV byte ptr [BX ], 1  ; escribe 1 en byte

MOV word ptr [BX], 1  ; escribe 1 en word

MOV dword ptr [BX ], 1 ; escribe 1 en dword
```

Le indico al ensamblador como tiene que interpretar el valor que estoy moviendo a BX

¿Preguntas?