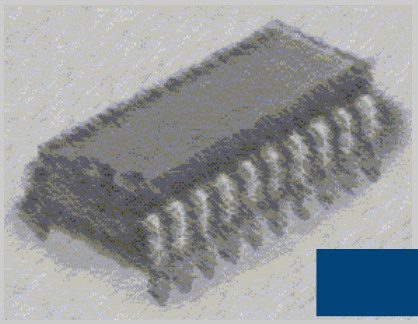
Enunciados de prácticas

Práctica 7. Lenguaje máquina y lenguaje ensamblador



Laboratorio de Estructura de Computadores

I. T. Informática de Gestión / Sistemas

Curso 2008-2009

PRÁCTICA 7:
Lenguaje máquina y lenguaje ensamblador

Objetivos:

El objetivo es que el alumno pueda relacionar el código máquina con las instrucciones en lenguaje ensamblador y con los modos de direccionamiento, diferenciando entre la dirección física y la dirección efectiva.

Medios:

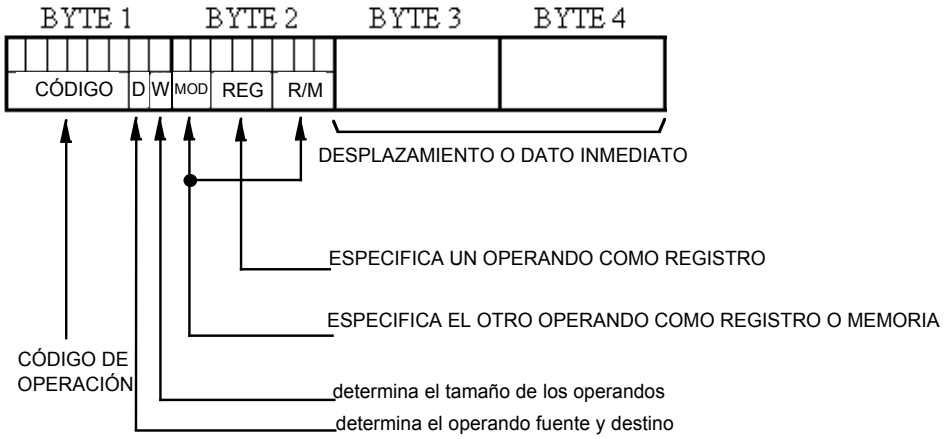
Para la programación se emplea el Microsoft Assembler 5.1. El software viene acompañado de un programa ensamblador (MASM) y su correspondiente enlazador (LINK) que generará un fichero ejecutable (.EXE) que será el que podrá ser ejecutado paso a paso mediante el simulador o ejecutado de golpe como se hace habitualmente. El desarrollo de la práctica consistirá básicamente en el manejo del programa Code View para alterar la ejecución del programa y modificar el código máquina de las instrucciones del mismo.

INTRODUCCIÓN

En esta práctica se trabajará con el código máquina del i80x86 y con el formato de instrucción. El objetivo es que el alumno pueda relacionar el código máquina con las instrucciones en lenguaje ensamblador y con los modos de direccionamiento, diferenciando entre la dirección física y la dirección efectiva.

FORMATO DE INSTRUCCIONES

El juego de instrucciones del i80x86 admite una gran cantidad de formatos. El que se muestra a continuación es una simplificación del mismo, que servirá, únicamente, en algunos casos, las instrucciones de registro-registro o de registro-memoria.



REG	W=0	W=1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Tabla codificación del operando REG

MOD = 11			CÁLCULO DE LA DIRECCIÓN EFECTIVA			
R/M	**W = 0**	**W = 1**	**R/M**	MOD = 00	**MOD = 01**	**MOD =10**
000	AL	AX	000	[BX]+[SI]	[BX]+[SI] + Desplaz.8	[BX]+[SI] + Desplaz.16
001	CL	CX	001	[BX]+[DI]	[BX]+[DI] + Desplaz.8	[BX]+[DI] + Desplaz.16
010	DL	DX	010	[BP]+[SI]	[BP]+[SI] + Desplaz.8	[BP]+[SI] + Desplaz.16
011	BL	BX	011	[BP]+[DI]	[BP]+[DI] + Desplaz.8	[BP]+[DI] + Desplaz.16
100	AH	SP	100	[SI]	[SI] + Desplaz.8	[SI] + Desplaz.16
101	CH	BP	101	[DI]	[DI] + Desplaz.8	[DI] + Desplaz.16
110	DH	SI	110	Dirección directa	[BP] + Desplaz.8	[BP] + Desplaz.16
111	BH	DI	111	[BX]	[BX] + Desplaz.8	[BX] + Desplaz.16

Tabla de codificación para el operando R/M en función del modo de direccionamiento MOD

ACTIVIDADES PARA LA PRÁCTICA 7

<table>
<tr><th>Núm Ejer.</th><th>Ejercicio</th></tr>
<tr><td>1</td><td>Introduce, ensambla y ejecuta con el Code View el código siguiente:
dosseg
.model small.
.stack 100h
.data
Texto DB ‘Introduce un número hexadecimal de cómo mucho dos cifras entre 0 y 9$’
.code
Inicio:
mov ax, @data
mov ds, ax

mov ah, 9
lea dx, Texto
int 21h
xor bl, bl
mov ah, 1
int 21h

mov cl, 4
mov bl, al
sub bl, 30h
shl bl, cl

int 21h
sub al, 30h
add bl, al

mov ah, 4Ch
int 21h
END Inicio</td></tr>
<tr><td>2</td><td>A continuación se presenta lo que se vería en el programa Code View cuando damos la opción de Assembly. Ésta será la vista con la que trabajaremos.

45B0:0010 B8B345 MOV AX,45B3
45B0:0013 8ED8 MOV DS,AX
45B0:0015 B409 MOV AH,09
45B0:0017 8D160600 LEA DX,Word Ptr [0006]
45B0:001B CD21 INT 21
45B0:001D 32DB XOR BL,BL (Continua)</td></tr>
</table>

2	A continuación se presenta lo que se vería en el programa Code View cuando damos la opción de Assembly. Ésta será la vista con la que trabajaremos. 45B0:0010 B8B345 MOV AX,45B3 45B0:0013 8ED8 MOV DS,AX 45B0:0015 B409 MOV AH,09 45B0:0017 8D160600 LEA DX,Word Ptr [0006] 45B0:001B CD21 INT 21 **45B0:001D 32DB XOR BL,BL** **(Continua)** **(Continua)** 45B0:001F B401 MOV AH,01 45B0:0021 CD21 INT 21 45B0:0023 B104 MOV CL,04 45B0:0025 8AD8 MOV BL,AL 45B0:0027 80EB30 SUB BL,30 45B0:002A D2E3 SHL BL,CL 45B0:002C CD21 INT 21 45B0:002E 2C30 SUB AL,30 45B0:0030 02D8 ADD BL,AL 45B0:0032 B44C MOV AH,4C 45B0:0034 CD21 INT 21 La información de la columna de la izquierda es Segmento:Efectiva Código máquina, es decir, la última línea indica 45B0:0034 CD21 quiere decir que el registro CS=45B0h, la dirección efectiva de la instrucción es 0034h (el valor de IP al ejecutar la instrucción) y que a partir de esa posición de memoria 45B0x10h + 0034h = 45B34h se encuentra el código máquina de la instrucción INT 21h que es el CD21h Para esta actividad ejecutar el programa paso a paso hasta llegar a la instrucción del programa XOR BL, BL (observar como varía el valor del registro IP según procedemos) ¿Qué valor tendría que tener IP para ejecutar la instrucción anterior? Modifica el valor del registro IP con el contenido que tendría que tener para ejecutar de nuevo la instrucción INT21h que la precede (*R IP nuevo valor*) Vuelve a ejecutar la instrucción. ¿Qué ocurre? (puedes comprobarlo pulsando F4 para ver la salida por pantalla)
3	Si ahora quisiésemos salir del programa sin leer ningún número. ¿Qué valor tendríamos que tener en IP para que saltásemos a las instrucciones de finalización del programa?
4	Si te fijas en el código máquina de la instrucción LEA DX, Texto

4	Si te fijas en el código máquina de la instrucción LEA DX, Texto (8D160600) ¿Cuál sería la dirección efectiva de la variable texto? ¿Cuál sería la dirección efectiva para que empezase en la palabra *número*?
5	Calcula el valor de la dirección efectiva de manera que el texto aparezca a partir de la palabra *como.*
6	Aunque el programa no funcione correctamente vamos a alterar el origen y el destino de la instrucción MOV BL, AL 8 A D 8 1000 1010 1101 1000 Código de operación = 100010 **(Continua)** **(Continua)** Bit D = 1 (destino reg) Bit W = 0 (8 bits) Mod = 11 (registro-registro) Reg = 011 (BL) R/M = 000 (AL) ¿Cómo cambiaríamos la instrucción anterior por MOV AL, BL cambiando el segundo byte? (Introdúcelo como en el caso anterior EB 0x45B0:0x0026 *nuevo valor*)
7	¿Cómo cambiaríamos la instrucción anterior por MOV AL, BL cambiando el primer byte? (Introdúcelo como en el caso anterior EB 0x45B0:0x0025 *nuevo valor*)

PRÁCTICA 7
Lenguaje máquina y lenguaje ensamblador

El siguiente programa realiza la solicitud de acceso a un sistema multiusuario. Se pide introducir el programa y realizar los cambios que se solicitan sobre el código máquina.

```
DOSSEG
.MODEL SMALL
.STACK 100h
.DATA
      Usuarios      DB "ANTONIO GOMEZ GOMEZ%A23$"
                    DB "LUISA ALONSO LOPEZ%A1SA3$"
                    DB "FERNANDO PEREZ MINGUEZ%2W45$"
                    DB "JOSEFA RUIZ SANCHEZ%ASQ12$"
                    DB "MIGUEL GARCIA GARCIA%S1A$"
      LonUsua       EQU $-Usuarios
      LonParc       DB 5 DUP()        ;Permite indexar la tabla de
                                      ;usuarios guardando la longitud
                                      ;de cada entrada
      Mensaje1      DB "Nombre de usuari@: $"
      Mensaje2      DB "Por favor, introduzca su clave de acceso: $"
      Mensaje3      DB "Bienvenid@ al sistema.$"
      Mensaje4      DB "Nombre de usuari@ o palabra clave incorrectos.$"
      Mensaje5      DB "Ha sobrepasado el numero de intentos de entrada."
                    DB "El sistema se bloquea.",10,13,"$"
      BufUsu        DB 31         ;Buffer para guardar el usuario
      LonUsu        DB 0
                                                  Usuari        DB 31 DUP (?)

      LonCla        DW 0          ;Buffer para guardar la clave
      Clave         DB 5 DUP (?)
      NumInt        DB 0          ;Aqu¡ guardamos intentos fallidos
```

(**Continua)**

(**Continua)**

```
.CODE
;Macro que pone el modo de v¡deo
ModoVideo MACRO modo
        mov ah,0
        mov al,modo
        int 10h
        ENDM

;Macro que situa el cursor
PonCursor MACRO fil,col
        mov ah,2
        xor bx,bx
        mov dh,BYTE PTR fil
        mov dl,BYTE PTR col
        int 10h
        ENDM

;Procedimiento que escribe una cadena
SacaMens  PROC
        push bp
        mov bp,sp
        push dx
        push ax
        mov ah,9
        mov dx,[bp+4]
        int 21h
        pop ax
        pop dx
        pop bp
        RET  2
SacaMens  ENDP


Inicio: mov ax,@DATA
       mov ds,ax
;Presentar pantalla, pedir y leer nombre usuario y palabra clave
otrave: ModoVideo 3
       PonCursor 10,15
       lea ax, Mensaje1
       push ax
       call SacaMens
       lea dx, BufUsu
       mov ah,0Ah
       int 21h
       PonCursor 12,15
       lea ax, Mensaje2
       push ax
       call SacaMens
       lea bx, Clave     ;leer clave sin eco
       xor si,si
```

(**Continua)**

(**Continua)**

```
lazo1:  mov ah,8
        int 21h
        mov [bx+si],al
        cmp al,13
        je salir
        mov ah,0Eh
        mov al,'*'
        int 10h
        inc si
        cmp si,5
        jbe lazo1
salir:  mov LonCla,si          ;guardamos la longitud de la clave
;Indexar tabla de usuarios
        xor si,si
        xor di,di
        mov dl,1               ;Cuenta la longitud de cada entrada de usuario
        lea bx, Usuarios
        mov cx,LonUsua
        mov al,'$'
lazo2:  cmp al,[bx+si]
        jne seguir
        mov LonParc[di],dl
        xor dl,dl
        inc di
seguir: inc si
        inc dl
        loop lazo2
;Comparar nombre usuario con los de la tabla
        lea bx, Usuarios
        xor di,di
        xor si,si
        xor cx,cx
        mov cl,LonParc              ;Longitud primera entrada
otrous: xor si,si
lazo3:  mov al,Usuari[si]
        cmp al,20h                  ;Si es espacio no convertir
        je saltar
        and al,11011111b            ;Pasar a mayusculas el nombre leido
saltar: cmp al,[bx+si]
        jne salfin
        inc si
        loop lazo3
salfin: cmp BYTE PTR[bx+si],'%'
        je nomcorr
        mov cl,LonParc[di]
        add bx,cx
        inc di
        cmp bx,LonUsua         ;Termina
        jb otrous
        jmp nominc
```

(**Continua)**

(**Continua)**

```
;Nombre correcto, comparar clave
nomcorr:
        xor cx,cx
        mov cl,LonParc[di]      ;Calculamos longitud clave
        sub cx,si               ;(LonParc[di]-si)-2
        dec cx
        dec cx
        cmp cx,LonCla           ;Detecta clave demasiado larga aunque
        jb nominc               ;coincidan los primeros caracteres
        inc si
        xor di,di
lazo4:  mov al,Clave[di]
        cmp al,[bx+si]
        jne nominc
        inc si
        inc di
        loop lazo4
;Mensaje de Bienvenida
        PonCursor 14,15
        lea ax, Mensaje3
        push ax
        call SacaMens
        jmp final
;Nombre o clave incorrectos, bloquear
nominc: PonCursor 14,15
        lea ax, Mensaje4
        push ax
        call SacaMens
        mov al,NumInt
        inc al
        cmp al,3
        jae bloquea
        mov NumInt,al
        jmp otrave
bloquea:PonCursor 16,4
        lea ax, Mensaje5
        push ax
        call SacaMens
        jmp bloquea
final:  mov ah,4Ch
        int 21h
        END Inicio
```

Se pide:

1.- Cambiar la macro *Modovideo* en un procedimiento al que se le pase el modo por medio del registro *BL*.

2.- Cambiar el procedimiento *SacaMens* en una macro en la que se pase como parámetro la dirección de la cadena

3.- El código de operación de la instrucción JE es 74h y el de JNE es 75h. El siguiente byte es la dirección efectiva que hay que sumar al IP para que se efectúe el salto de manera correcta. Modifica el código máquina anterior para que salte a la siguiente instrucción a la de *Salir*: *XOR SI, SI.* Ejecuta el programa y comprueba qué ocurre

4.- Después de la etiqueta *Salfin* cambia el primer byte del código máquina de la instrucción *add bx,cx* para que se inviertan los operandos fuente y destino.

5.- Realiza el mismo cambio que en el punto 4 pero codificando el segundo byte del código máquina de la instrucción

6.- Después de la etiqueta *Salfin* cambia el primer byte del código máquina de la instrucción *je nomcorr* **para que si no es correcta la clave**, se salte al mensaje de bienvenida.

7.- Después de la etiqueta *Salfin* cambia el segundo byte del código máquina de la instrucción *je nomcorr* para que **si es correcta** la clave, se salte al mensaje de clave incorrecta.

8.- Cuando se va a escribir el mensaje que solicita el nombre de usuario mediante la llamada del procedimiento *Sacamens*, cambia la dirección efectiva de la cadena en el código máquina de la instrucción para que el texto comience en *usuari@*

9.- Cuando se va a leer el nombre de usuario mediante el empleo de la función 0Ah de la interrupción 21h, cambia la dirección efectiva de la cadena en el código máquina de la instrucción para que apunte a *Usuari* y no a *BufUsu*