Práctica 1

Captura, ensamble y ligado de programas en NASM

Introducción

Como ya sabemos, todo programa en ensamblador consta de 3 bloques:

- Segmento o sección de pila.
- Segmento o sección de datos, y
- Segmento o sección de código.

Los bloques de pila y datos son opcionales. Dentro del bloque de código, que es el único obligatorio, debe haber por lo menos un procedimiento.

Todo programa está compuesto de: i) instrucciones, ii) directivas o pseudoinstrucciones y iii) constantes, variables y etiquetas.

En este curso utilizaremos el ensamblador Netwide Assembler, un ensamblador de código abierto (open source). Este ensamblador tiene su propia sintaxis para definir instrucciones, directivas, segmentos, variables, constantes y etiquetas. Es pues necesario conocer la sintaxis de este ensamblador.

El proceso de programación en ensamblador es el siguiente:

i) Capturar el código fuente en un editor de texto (edit o notepad, por ejemplo). Este proceso genera un archivo fuente de ensamblador (con extensión `.asm`)
ii) Ensamblar el código fuente, esto es, traducir a lenguaje máquina. Este proceso genera un archivo objeto (extensión `.obj`)
iii) Ligar el (los) archivo(s) objeto del programa. Esto genera un archivo ejecutable (extensión `.exe`)

Durante el ensamble, el traductor puede generar avisos cuando encuentra errores de sintaxis (un procedimiento no definido, o una etiqueta que no está definida, por ejemplo). En ese caso, la traducción se detiene y es necesario corregir el código fuente con un editor de textos e intentar de nuevo la traducción. También durante el ensamble puede generarse un listado, esto es, un archivo que contiene tanto el código fuente como el código máquina del programa traducido y también puede generarse un archivo de referencias cruzadas, que es útil cuando se manejan programas divididos en varios archivos fuente.

Durante el ligado es posible generar un archivo de mapeo, que indica donde se localiza cada segmento o sección del programa en el archivo ejecutable y que es útil sobre todo para programas divididos en varios archivos fuente, también pueden agregarse librerías (una librería es una colección de procedimientos ya traducidos a código objeto, esto es, en lenguaje máquina).

Objetivo

El objetivo de esta práctica es aprender el proceso de programación (captura, ensamble, ligado y depuración) con el NASM y empezar a conocer y manejar su sintaxis.

Actividades

1.- Captura del programa.

Captura en un editor (edit o notepad) el siguiente programa en ensamblador y llámalo `hola.asm`:

```
; Archivo: hola.asm
; Despliega una cadena en pantalla. Utiliza Int 21h/9
; -------------------------------------------------
segment pila stack
resb 512                    ; Definimos una pila de 512 bytes

segment Datos data
msje  db "Hola Mundo desde Netwide Assembler!!!",10,13,"$"

segment Prog code
..start:
      mov ax,Datos          ; Inicializa DS
      mov ds,ax
      mov ah,9              ; Llamamos al DOS
      mov dx, msje          ; para desplegar una cadena
      int 21h
      mov ax,4c00h          ; Llamamos al DOS
      int 21h               ; para terminar el programa
```

2.- Ensamble del programa

Ejecuta el comando:

nasm -f obj hola.asm

para generar el archivo objeto `hola.obj`

Con este comando invocamos al traductor y le indicamos el archivo fuente a traducir.

En caso de errores de sintaxis, el traductor nos informa sobre el error encontrado y en que línea del código fuente se encuentra. Si el programa no contiene errores de sintaxis, el traductor no despliega mensaje alguno**.**

3.- Ligado del programa

Cuando ya no hay errores en el ensamble, el paso siguiente es el ligado. En este curso usaremos el ligador LINK de Microsoft. Para ligar utilizamos el comando **`link hola.obj`**

El ligador nos pide (y sugiere) el nombre del ejecutable:

```
Microsoft (R) Overlay Linker Version 3.64
Copyright (C) Microsoft Corp 1983-1988. All rights reserved.

Run file [hola.exe]: <Enter>
```

Si tecleamos **<Enter>** aceptamos la sugerencia, que en este caso es `hola.exe`. Si queremos otro nombre para el archivo ejecutable a generar, entonces debemos teclear el nombre deseado, con un máximo de 8 letras. En este caso, vamos a aceptar la sugerencia tecleando **<Enter>**.

Posteriormente el ligador me pregunta si deseo un archivo de mapeado:

```
List file [nul.map]: hola<Enter>
```

En este caso nos sugiere `nul.map`, que, de aceptarlo quiere decir que no queremos un archivo de mapeado. Si queremos un archivo de mapeado, debemos teclear un nombre de máximo 8 letras. En este caso vamos a pedirle que genere el archivo `hola.map`, por lo que a esta petición contestamos

tecleando **`hola<Enter>`**. Nota que no es necesario teclear la extensión `.map`.

Finalmente nos pide las librerías que requiere el programa:

```
Libraries [.lib]: <Enter>
```

Dado que este programa no utiliza librerías, contestamos con **`<Enter>`**.

Si el ligado se realizó correctamente debió generarse el archivo `hola.exe`.

Usando el comando **`dir hola.*`** obtén la información siguiente:

Archivo:	Tamaño	Fecha	Hora
`hola.asm`	_____	_____	_____
`hola.obj`	_____	_____	_____
`hola.lst`	_____	_____	_____
`hola.map`	_____	_____	_____
`hola.exe`	_____	_____	_____

4.- Prueba de ejecución del programa

Para ejecutar el programa sólo es necesario teclear su nombre (en este caso **`hola<Enter>`**).

Ejecútalo y describe la salida que produce: _____

_____

_____

_____

5.- Captura, ensamble, ligado y prueba de ejecución de otro programa

Ahora captura el programa desarrollado en clase (el programa que limpia la pantalla, despliega el mensaje Arquitectura de Computadoras 2, Bloque 3, Sección 2, en una posición determinada de la pantalla, espera por la tecla 'q' o 'Q' y termina)

5.1.- Comando para ensamble: _____

5.2.- Comando de ligado: _____

Genera el archivo de mapeado para este programa.

5.3.- Información de los archivos generados:

Archivo	Tamaño	Fecha	Hora
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____

5.4.- Prueba de ejecución

Comando para ejecutar este programa: _____

Ejecútalo y describe la salida que produce: _____

_____

_____

6.- Comentarios y conclusiones

_____

_____

_____

_____

_____

_____

_____

_____