Ensambladores

Un Ensamblador es un programa encargado de traducir un programa fuente escrito en Lenguaje ensamblador (nemónicos) en otro programa equivalente escrito en Lenguaje máquina (binario).

Clasificación

En la ***forma en que trabajan***:

De Línea: Ensamblan una sola línea a la vez del programa fuente. Ejemplo Comando A de Debug.

De Archivos: Ensamblan todo un programa fuente previamente almacenado en un archivo

De a cuerdo ***al tipo de información que procesan***:

Propios o residentes: Ensamblan programas escritos en el mismo lenguaje que el procesador de la máquina de trabajo. La ventaja de estos ensambladores es que permiten ejecutar inmediatamente el programa; la desventaja es que deben mantenerse en la memoria principal tanto el ensamblador como el programa fuente y el programa objeto.

Cruzados (Cross- Assembler): Ensamblan programas escritos en un lenguaje distinto al del procesador de trabajo.

El empleo de este tipo de traductores permite aprovechar el soporte de medios físicos (discos, impresoras, pantallas, etc.), y de programación que ofrecen las máquinas potentes para desarrollar programas que luego los van a ejecutar sistemas muy especializados en determinados tipos de tareas.

Macroensambladores: Ensambladores propios o cruzados que permiten la definición y expansión de MACROS. Debido a su potencia, normalmente son programas robustos que no permanecen en memoria una vez generado el programa objeto. Puede variar la complejidad de los mismos, dependiendo de las posibilidades de definición y manipulación de las macroinstrucciones.

Programando con *Debug*

La programación de los procesadores en bajo nivel resulta siempre muy complicada en un inicio, como primer acercamiento a la programación de bajo nivel se utilizará el antiguo sistema *Debug.*

El *Debug* es un programa que formaba parte del sistema operativo *MS DOS* y servía para realizar y revisar programas muy sencillos, bajo arquitecturas de 16 bits. Cumple con varias funciones: ensamblar, desensamblar y depurar (*debug*) programas con código directo (sin encabezados), cuya extensión era ".com" (de *command*). Al tener una arquitectura basada en 16 bits, sólo podían direccionar segmentos de memoria de 64Kb (2^16).

Es importante señalar que algunas de las versiones más recientes de Windows a 32 bits 0 64 podrían no incluyen el programa *Debug*, aunque se puede descargar. Los sistemas Windows a 64 bits no son compatibles con los programas de 16 bits, si bien, a esto se le puede dar la vuelta utilizando el sistema *DOSBox.*
DOSBox es un emulador que recrea un entorno similar al sistema *MS-DOS* con el objetivo de poder ejecutar programas y videojuegos originalmente escritos para dicho sistema en computadoras más modernas o en diferentes arquitecturas.

DOSBox proporciona un entorno DOS completo, pero de forma predeterminada no tiene acceso a ninguno de sus archivos y carpetas de Windows. Por lo tanto, antes de poder ejecutar un programa en *DOSBox*, debe montar su carpeta dentro de *DOSBox.* El comando *mount* hace que la carpeta especificada esté disponible como una letra de unidad dentro de *DOSBox.*

Pasos para montar la carpeta de trabajo.

1. Busque la carpeta del programa en el Explorador de Windows. Tenga en cuenta la ruta completa a la carpeta del programa, que aparece en la barra de ubicación en la parte superior de la ventana del Explorador de Windows.
2. Escriba el siguiente comando en la ventana de *DOSBox*, reemplazando "C: Usuarios Nombre Carpeta de ejemplo" con la ruta a la carpeta del programa:

   mount C "C: Usuarios Nombre Carpeta de ejemplo"

   para nuestro caso:

   > ***mount C C:***

3. Acceda a la unidad del programa montado dentro de *DOSBox* escribiendo "C:" (sin comillas) y presionando "Enter".

   para nuestro caso:

   > ***C:***

4. Escriba el nombre del archivo EXE del programa para ejecutarlo. Por ejemplo, si su programa se llama "Example.exe", escriba "Ejemplo" (sin comillas) y presione "Enter".

   para nuestro caso:

   > ***debug***

El *debug* se invoca con el comando, siempre que esté instalado y accesible en *MS DOS*. Todos los valores y direcciones expresadas en *debug* estarán en hexadecimal, y no deberá agregarse 'h' al final.

Una vez invocado, se entra en un modo interactivo de comando y respuesta.

Debug

Es un depurador de instrucciones que ayuda a probar programas ejecutables, realizandolo en modo sencillo (inst. por inst.). Características:

1. Prueba y depura programas escritos en lenguaje máquina y en lenguaje ensamblador
2. Proporciona un conjunto de comandos para desplegar, introducir y trazar
3. No distingue entre mayúsculas y minúsculas
4. Todos los números están en formato hexadecimal
5. No permite el uso de etiquetas ni comentarios

Comandos de Debug

Comando R: Muestra o modifica los contenidos de los registros.

-r

Salida:

```
AX  BX  CX  DX    Punteros índices
CS   SS   ES   BS    IP   Banderas
XXXX : XXXX      Siguiente instrucción   hexadecimal  y ASCII
```

Dir. Segmento: Dir. IP

Dir. real o absoluta

NOTA: La primera acción a realizar antes de ensamblar es destralapar los segmentos CS, DS, SS, ES

- r DS
  0100

- r SS
  0200

- r CS
  0300

- r ES
  0400

Comando A*:* Ensamblar instrucciones

Como el IP siempre apunta al desplazamiento 100 se sugiere iniciar el ensamblado en este desplazamiento

-a 100

```
xxxx : 0100  mov   cx, 1
xxxx : 0103   mov ax, 0
xxxx : 0106   add ax, cx
xxxx : 0108   inc cx
xxxx :0109   CMP CX, 64
xxxx :010C   JBE   0106
xxxx :010E   NOP
```

Para modificar una instrucción la instrucción debe de ser de la misma longitud, para no modificar la dirección relativa siguiente.

Comando U: Desensambla instrucciones

-U 100 10c (rango)
-U Desensambla 32 bytes desde el último desplazamiento
-U 100 desensambla 32 bytes a partir del IP

Pasos para guardar un programa a disco

1. Nombre : Nombre.com-------- ***Comando N:*** Da nombre al archivo
2. Obtener la longitud de las instrucciones----***Comando H***: Suma y resta de la dir. final e inicial especificadas
3. Almacenar la longitud del archivo, el tamaño de un archivo se almacena en la pareja de registros BX, CX
   - Almacenar en BX el valor cero------**Comando R**
   - Almacenar la longitud en Cx------**Comando R**
4. Escribir el archivo a disco ----***Comando W***: Escribe

En nuestro caso de ejemplo haremos lo siguiente:

```
-n conteo.asm
-h 10E  100
-r  CX
        000E
-r  BX
        0000
-w
```

Pasos para cargar un programa a memoria

1. Nombre : Nombre.com-------- ***Comando N:*** Da nombre al archivo
2. Carga archivo a memoria ------***Comando L 0100:*** Carga un archivo a memoria

```
-n conteo.com
-L 0100   carga en una localidad específica omisión
```

Comando T : Ejecutar las instrucciones una por una, mostrando el contenido de los registros.

Ejemplos de uso:

- r IP Primero ajustar el IP a la dirección de desplazamiento 0100
  100
- t 10 Ejecute 16(10h) instrucciones a partir del IP
- t =100 [5]
  dirección # de instrucciones opcional
- t Ejecuta una sola instrucción a partir de la dirección apuntada por el IP

Comando G **:** Ejecuta un bloque de instrucciones

-G = Dir inicial Dir final
- G = 110 11A Ejecuta desde la 110 hasta la 11A

Comando E **:** Introduce datos a los segmentos por lista o secuencial. Por omisión al segmento DS.

- E Dirección [lista]

  Donde: Dirección esta dada por: Dir. Segmento: Dir. Relativa p.e. ES:200

- E 100 1 2 3 4 5

  Introduce una Lista: Un conjunto de valores de un byte separados por espacios.

- E 100 “Esto es una cadena$”

  Introduce una lista: Una cadenas entre comillas o apostrofes.

- E CS: 211 21 2A

  Almacena apartir del desplazamiento 211 del segmento CS la lista de valores.

- E 100 Mostrara el dato almacenado y permitirá cambiar por otro.

Comando D: Despliega los datos o el contenido de la memoria. Por omisión despliega al segmento DS.

-D dirección	Despliega 80h bytes a partir de la dirección indicada
-D 100 130	Despliega un rango
-D	Despliega 80 bytes a partir de la última dirección de DS.
-D CS:150	Despliega 80 bytes a partir de CS:150h
-D DS200 5	Despliega 5 bytes desde DS:200

Ejercicio Práctico en Sala de Computo:

Dentro del DEBUG realizar:

I. Realizar el siguiente ejercicio, siga los puntos solicitados:

1. Con el comando *A* introducir (recuerde destrazlapar los segmentos):

```
MOV AX, 1234
MOV BL, 23
DIV BL
NOP
```

2. Guardar el programa con el nombre divide.asm
3. Trazar el conjunto de instrucciones con el comando *T*.
4. Si el comando *DIV* realiza la división de dos números, contestar:
   ¿Cuál es el dividendo?
   ¿Cuál es el divisor?
   ¿Dónde quedaron los resultados (cociente y residuo)?
5. Escribir sus resultados para discutirlos en clase

II. Realizar el siguiente ejercicio, siga los puntos solicitados:

1. Cargue en memoria a partir de la dirección 0120 de desplazamiento su nombre completo en mayúsculas finalizando con el carácter $
2. Cargue en memoria a partir de la dirreción final +1 del ejercicio anterior los digitos AA AB AC AD BC