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: Son ensambladores residentes que permiten definición 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.

ENSAMBLADOR DE ARCHIVO

- TASM
- MASM
- NASM
- GAS

Características de un Ensamblador de Archivo

- Traduce de Leng. Ensamblador a Lenguaje máquina
- Permite definición de etiquetas
- Reserva memoria para variables octales del programa y asigna un valor inicial
- Permite evaluar expresiones aritméticas (+,- **, /,negación).**

  Mov al,-3
  Mov al, 80*40/5*2
- Permite definición de números en otras bases:

XxxxH	Hexadecimal
XxxxO	Octal
XxxxD	Decimal
Xxxx	Decimal
XxxxB	Binario

Cuerpo de un programa en MASM

Se integra de directivas (pseudoinstrucciones o pseudooperadores) e instrucciones. La primera no genera código ejecutable, sólo se utilizan para llevar acabo el ensamble, las instrucciones si generan código ejecutable.

El cuerpo de un programa queda de la siguiente forma:

1. [Comentarios]
2. Definición de segmentos, por lo menos se deben definirse CS y el SS
   2.1 Area de código

Definición de Segmentos

Para realizar esta acción se utilizan las denominadas ***directivas de segmento***. Se pueden utilizar las directivas largas o las simplificadas.

Directivas largas.

Directivas de segmento: SEGMENT, ENDS y END

SEGMENT : Directiva para definir un segmento, el tamaño máximo de un segmento es de 64kb
ENDS: Directiva para finalizar un segmento

Formato:

Nombre Directiva alineación [combinación] [clase]
.
.
.
Nombre ENDS

Alineación: Indica el límite en el que inicia el segmento, por omisión alinea el segmento al límite de un párrafo, de manera que la dirección inicial es divisible entre 16 0 10h, el requerimiento típico es PARA.

Combinación: Indica si el segmento se combina con otros segmentos cuando son enlazados después de ensamblar, los tipos son STACK, COMMON, PUBLIC y la expresión AT.

Para la pila es:
Nombre SEGMENT PARA STACK
Se puede utilizar PUBLIC y COMMON en donde tenga el propósito de combinar de forma separada programas ensamblados cuando los enlaza. En otros casos, donde no utilice combinación, se puede omitir o escribir NONE.

Clase: Es encerrada entre apóstrofes, es utilizada para agrupar segmentos cuando se enlazan.

Se utilizan ´code´ con CS, ‘data’ para DS, ‘stack’ para SS.

END: : Directiva para definir el fin del programa

Ejemplo:

```
STACKSG segment para stack 'stack'
     Definición de la pila
STACKSG ENDS

DATASG SEGMENT PARA 'DATA'
     Definición de datos
DATASG ENDS

CODESG SEGMENT PARA 'CODE'
     Definición de codigo
CODESG ENDS

END
```

Directiva de segmento: ASSUME

Se utiliza para asignar el nombre de los segmentos definidos a los registros correspondientes , se codifica en el segmento de código utilizando el siguiente formato:

ASSUME SS: nombre_SEGSS, DS: nombre_SEGDS, CS:nombre_SEGCS, ...

Directivas simplificadas

Son formas abreviadas para definir segmentos, para usarlas primero se debe definir el modelo de memoria antes de definir algún segmento

La directiva simplificada .MODEL crea por omisión a los segmentos y los enunciados ASSUME y GROUP necesarios. Formato de definición de memoria:

.MODEL modelo de mem

MODELO	NUM DE SEG DE CODIGO	NUM DE SEG DE DATOS
TYNY	*	*
SMALL	1	1
MÉDIUM	MAS DE 1	1
COMPACT	1	MAS DE 1
LARGE	MAS DE 1	MAS DE 1

Se puede utilizar cualquiera de estos modelo para un programa autónomo (esto es , un programa que no esta enlazado con algún otro). El modelo TYNY esta destinado para

uso exclusivo de programas .COM, los cuales tienen sus datos, código y pila en el mismo segmento.
El modelo SMALL exige que el código quepa en un segmento de 64K y los datos en otro de 64K, la directiva .MODEL genera de forma automática la directiva ASSUME.

Los formatos generales para las directivas de segmento son:

```
.STACK      [tamaño]
.DATA
.CODE       [nombre]
```

Cada una de estas directivas hace que el ensamblador genere el enunciado SEGMENT necesario y su correspondiente ENDS. El tamaño de la pila es de 1024 bytes, el cual puede pasarse por alto. Los nombres de los segmentos (que no se tienen que definir) por omisión son STACK, DATA y TEXT (código).

Ejemplo:

```
.MODEL SMALL
.STACK 120
.DATA
        [Elementos de datos]
.CODE
        [instrucciones]
END
```

Uso y explicación de la sección de protocolo en un código ensamblador

Al utilizar directivas largas se prtesenta de la siguiente manera:

PUSH DS	; Guarda en la pila la dirección del DS actual
SUB AX,AX	; Limpia el registro AX
PUSH AX	; Guarda el valor actual de AX
MOV AX,SEG DATASG	; Almacena la dirección del segmento de datos en AX, esta es determinada por el sistema cuando el código objeto esta enlazado y cargado para su ejecución, ya que el cargador puede ubicar cualquier programa en cualquier parte de la memoria y el ensamblador no la conoce.
MOV DS,AX	;Mueve el contenido del registro AX al registro DS, para inicializar el DS, el cargador de DOS inicializa automáticamente el SS y el CS cuando se carga un programa para su ejecución, pero es responsabilidad del programador inicializar el DS y el ES.

Al utilizar directivas simplificadas el protocolo se presenta de la siguiente manera:

Para inicializar la dirección del segmento de datos en el DS se utilizan las instrucciones:

```
MOV AX,@data  ; ASIGNA A AX EL NOMBRE DEL SEGMENTO DE
                DATOS
MOV DS,AX     ;MUEVE EL CONTENIDO DE AX AL REGISTRO DS
```

Ensamblar y ligar usando MASM611

1. ***ml nombre-prog.asm***

Ejemplo para Ensamblar con Directivas Largas

```
;programa ejemplo para masm

; DEFINICION SEGMENTO DE PILA
STACKSG segment para stack 'stack'
      DB 20 DUP (0)                          ; Define espacio en la pila
STACKSG ENDS

;DEFINICION DE AREAS DE TRABAJO

 ; DEFINICION  SEGMENTO DE DATOS
DATASG SEGMENT PARA 'DATA'
MEN1 DB ' HOLA MUNDO SOY ESTUDIANTE DE INGENIERIA DE LA UTM$'
SALTA db 13, 10,'$'
MEN2 DB 'ADIOS$'
DATASG ENDS

; DEFINICION  SEGMENTO DE CODIGO
CODESG SEGMENT PARA 'CODE'

PRINCI PROC FAR                                  ;Inicia proceso
    ASSUME SS:STACKSG, DS:DATASG,CS:CODESG ;     ;Alineación de Segmentos
                                                 ;PROTOCOLO
     PUSH DS
      SUB AX,AX
      PUSH AX
      MOV AX,SEG DATASG
      MOV DS,AX

        ;INICIA PROGRAMA
        mov ah,01                    ; Funcion: Leer caracter
        int 21h                      ; Servicio: 01 int 21h alto nivel DOS
        mov dx, offset SALTA         ; Obtiene direccion del mensaje en dx
        mov ah,09                    ; Funcion: Visualizar cadena
        int 21h                      ; Servicio: 09 int 21h alto nivel DOS
        mov dx, offset MEN1
        mov ah,09
        int 21h
        mov dx, offset SALTA
        mov ah,09
        int 21h
        mov dx, offset MEN2
        mov ah,09
        int 21h
        mov   ax,4c00h               ;Funccion :Quit with exit code (EXIT))
       int   21h                     ;Servicio:4c int 21 DOS

   RET
   PRINCI ENDP                                   ;Fin proceso
CODESG ENDS                                      ;Fin segemento de códico
END PRINCI                                       ;Fin de programa
```

Ejemplo para Ensamblar con Directivas Cortas

```
.model small
.stack
.data
  saludo   db "Hola Mundo soy Estudiante de ingenieria de la UTM", "$"

.code

main  proc            ;Inicia proceso
  mov   ax,@data   ;PROTOCOLO resguarda dirección del segmento de datos
  mov   ds,ax        ;Protocolo ds = ax

  mov   ah,09        ; Function (print string)
  lea   dx,saludo    ;DX = String terminated by "$"
  int   21h          ;Interruptions DOS Functions

;mensaje en pantalla

  mov   ax,4c00h      ;Function (Quit with exit code (EXIT))
  int   21h          ;Interruption DOS Functions

main  endp            ;Termina proceso
end main
```

Otras directivas

Dentro de la definición de un segmento se define su contenido, para ello se utilizan otros tipos de directivas las denominadas **directivas para definición de datos**.

Directivas para definición de datos

Estas se utilizan para establecer áreas de trabajo y constantes, la definición se puede llevar a cabo por el contenido o el tamaño del área.

Formato :

[Nombre] Directiva expresión

Nombre: Es opcional y ayuda a referenciar a un elemento en el programa fuente.

Directiva: Indican la longitud del elemento definido. DB (byte), DW (palabra), DD (palabra doble), DF (palabra larga), DQ (palabra cuádruple) y DT (diez bytes).

Expresión: Es un operando que pude contener: constantes, sucesiones o listas constantes separadas por comas, cadenas de caracteres, duplicaciones de constantes, definición sin valor inicial (?)

Ejemplos:

Dat1	db 20	Almacena en un byte el valor 20
Dat2	db 3,4,5,6	Almacena en un byte cada valor
Dat3	db 'ensamblador$'	Asigna un byte para cada carácter
Dat 4	db ?	Elemento no inicializado
Dat5	dw 0	Almacena en una palabra el valor 0
Dat6	db 'D'	Almacena en un byte el caracter 'D'
Lista	db 'E','n','s','a','m','b','l','a','d','o','r', '$'	Asigna un byte para cada carácter

Para duplicar constantes se utiliza el siguiente formato:

*[Nombre] Directiva **DUP** expresión*

Datos DB 10 DUP 0	Duplica el valor 0 en 10 localidades de un byte.
DW 200 DUP 2040	Duplica el valor 2040 en 200 localidades de una palabra.

Para definir valores constantes se utiliza la directiva EQU, con el siguiente formato:

*Nombre **EQU** constante*

Datocte EQU 14h

Directiva de procedimientos: PROC y ENDP

La definición de código se hace en el segmento de código; tanto el programa principal como las subrutinas se definen como procedimientos, para esto se utilizan las directivas PROC y ENDP.

PROC: Define un procedimiento
ENDP: Finaliza un procedimiento

Formato:

```
Nombre  PROC  operando
        .
        .
        .
        ret
Nombre ENDP
```

Operando: Este puede ser FAR o NEAR

Estos informan al sistema que la dirección indicada es el punto de entrada para la ejecución del programa.

FAR: Código escrito en un segmento de código diferente
NEAR: Código escrito en el mismo segmento de código