Macroensamblador

- Pseudoinstrucciones para definir segmentos:

SEGMENT: Define el inicio de un nuevo segmento. Su formato es:
nombre SEGMENT alineación combinación clase

ENDS: Define el final de un segmento. Su formato es:
nombre ENDS

Los parámetros del SEGMENT son información para el ligador:

- **Alineación**: Define la dirección a partir de donde puede colocarse el segmento:

PARA: La dirección inicial del segmento es un múltiplo de 16 (10h).

PAGE: La dirección inicial del segmento es donde empieza una página (múltiplo de 100h).

WORD: La dirección inicial del segmento es una dirección par.

BYTE: EL segmento inicia donde sea.

- **Combinación**: Define la forma en que el segmento puede combinarse con otros segmentos para que se tenga el mismo nombre y clase.

OMITIRLA: Segmento privado, es decir, no puede combinarse.

STACK: Segmento para usarse con el stack.

PUBLIC: Este segmento puede unirse con todos los segmentos del mismo y la misma clase para formar una sola.

COMMON: Todos los segmentos del mismo nombre y clase se colocan a partir de la misma dirección.

Cuando se tienen dos segmentos con el mismo nombre y clase y son públicos, al ligar se unen en un solo segmento no importando que estén en archivos distintos.

Cuando se usa la psudoinstrucción COMMON van a utilizar el mismo espacio de memoria, si son de diferente tamaño en memoria, se toma el tamaño del mayor bloque.

- **Clase**: Indica el tipo de datos que contiene el segmento, siempre se ponen entre comillas y pueden definirse propios.
  'DATA': Datos.

  'CODE': Código.

  'STACK': Pila.

- Pseudoinstrucciones para reservar memoria y definir constantes:

DB: Sirve para reservar un byte en la memoria con un valor determinado. Su formato es:

[nombre] DB val_1 [, val_2, ..., val_n]

DW: Reserva un dato de dos bytes (una palabra) con un valor inicial. Su formato es:
[nombre] DW val_1 [, val_2, ..., val_n]

DD: Reserva un dato de cuatro bytes (doble palabra) con un valor inicial. Su formato es:
[nombre] DD val_1 [, val_2, ..., val_n]

DQ: Reserva un dato de ocho bytes (cuádruple palabra) con un valor inical. Su formato es:
[nombre] DQ val_1 [, val_2, ..., val_n]

DT: Reserva un dato de diez bytes con un valor inicial. Su formato es:
[nombre] DT val_1 [, val_2, ..., val_n]

val_i representa una expresión formada por números en cualquiera de las siguientes bases:

XXXXb	Binaria
XXXXo	Octal
XXXXd	Decimal
XXXX	Decimal
XXXXh	Hexadecimal

También se pueden tener expresiones de esos números con los operadores de:

+	Suma
-	Resta
–	Negación (C'2)
*	Multiplicación
/	División

También pueden ser etiquetas o expresiones aritméticas que involucren etiquetas o bien cadenas de caracteres, entre apóstrofes.

EQU: Permite definir constantes. Su formato es:

etiq EQU val

ORG: Define un desplazamiento inicial para ensamblar las siguientes líneas. Su formato es:

ORG val

- Pseudoinstrucciones para definir procedimientos:

PROC: Define el inicio de una subrutina.

nombre PROC tipo

ENDP: Define el final de una subrutina.

nombre ENDP

El tipo de la subrutina puede ser:

- **NEAR**: Cercano.
- **FAR**: Lejano.
- **OMITIRLO**: Se define por omisión de tipo NEAR:

Un ensamblador de archivo, revisa errores de sintaxis, es decir, revisa que el programa esté bien escrito, más no que funcione.

Para poner comentarios dentro del programa se inician con un ';' y todo lo que este a la derecha será un comentario sobre el mismo renglón.

La estructura del archivo quedaría:

```
Datos SEGMENT PARA 'DATA'

        ; Definición de variables y constantes

Datos ENDS

Pila SEGMENT PARA STACK 'STACK'
    DW 100 DUP (0)    ; Indica que se tiene que repetir la instrucción n-veces con el
                      ; valor que aparece en los paréntesis
Pila ENDS

Codigo SEGMENT PARA 'CODE'
    ASSUME DS:Datos, CS:Codigo, SS:Pila, ES:NOTHING
                      ; Sirve para indicarle al macroensamblador cuales segmentos
```

```
                                ; son usados por los registros

        subrutina1 PROC
                ; Código de la rutina uno
        subrutina1 ENDP

        subrutina2 PROC
                ; Código de la rutina dos
        subrutina2 ENDP


        subrutina-n PROC
                ; Código de la rutina-n
        subrutina-n ENDP

        ; Programa principal
        Main PROC FAR
                PUSH  DS            ; Sirve para cuando se termine el programa regrese al
                XOR   AX, AX        ; debug o al sistema operativo según sea el caso.
                PUSH  AX
                MOV   AX, Datos     ; Actualiza los registros de segmentos de datos y extra
                MOV   DS, AX
                MOV   ES, AX

                ; Código del programa principal.
        Main ENDP
Codigo ENDS
END Main                        ; Le indica al macroensamblador que el ensamble terminó
```

Las rutinas pueden ir antes o después del programa principal, el orden se puede intercambiar.