**LENGUAJE ENSAMBLADOR**

**Definición:** Es una notación legible al ser humano, del código de máquina y es especifico de cada arquitectura de computadora (ES PROPIO DE CADA uP).

Debido a que el lenguaje de maquina es un patrón de bits, el ensamblador se diseñó para reemplazar esos bits por símbolos denominados, mnemotécnicos o mnemónicos, y así no programar directamente en el código binario.

Este códigosimbólico (no es de alto nivel) no puede ejecutarse en una PC, por tanto es necesario convertirlo mediante ciertos programas (interpretes, traductores, compiladores) para obtener el código de maquina.

En el diseño, de un programa se tienen las sgtes fases:

Edición => Archivo fuente\_nombre.asm

Ensamble =>Software ej TASM.exe

Enlace => TLINK.exe

Ejecución =>Verificación

Depuración =>

Coloque en la raíz del disco duro llamada ASM

Colocar el TASM.exe, y el TLINK.exe.

El archivo fuente colóquelo en la misma carpeta.

Se edita con cualquier editor de texto como por ejemplo el emu8086.

Para el segundo paso C:ASM>TASM nombre.asm, aquí verifica sintaxis.

Ahora para el tercer paso C:ASM>TLINK nombre

es aquí cuando crea el ejecutable

Para ejecutarlo nombre

En el lenguaje ensamblador las líneas constan de dos partes, nombre de instrucción y los parámetros del comando o los operadores del mismo.

Acción /destino/fuente

Siempre se va a ejecutar en esta dirección<-

es decir de fuente a destino.

Ejemplo:

MOV Permite transferir datos desde y hacia el uP(los registros) y/o memoria.

Ejemplo:

MOV AX, 4C00H

Acción destino/fuente

Nota: Vea que el 4C00h es de 16 bits entonces el registro a debe ser de 16 bits.

Nota: Hay que indicar el sistema numérico, si no lo indica asume como decimal.

Nota: Si un número empieza con letra se antepone un 0:

MOV AL, F4h ; MAL

MOV AL, 0F4h ;CORRECTO

El nombre de una instrucción es de 2,3 o 4 letras:

ADD ; Suma.

MOV ;Movimiento.

CMP ;Compara

IN ;ingresar a un puerto

LOOP ;crear lazo

Hay comandos que no requieren parámetros para su operación. Otras solo requieren de un parámetro:

CLD ; Activa la direcciónflag

NOP ; No opere.

DEC AX ; Decreméntele 1 al registro AX.

INC AX ; Increméntele 1 al registro AX.

**LENGUAJE ENSAMBLADOR**

Maquina

Ensamblador

C

Pascal

Visual Basic

Visual C++

.Net

Lab View

Bajo nivel

....

......

Alto nivel

**Lenguaje de bajo nivel:**

Ventajas

**-**No tiene instrucciones avanzadas o complejas.

-Permite optimizar el manejo del hardware.

Desventajas:

-Mucho código para tareas simples.

**Lenguaje de alto nivel:**

Ventajas

**-**Instrucciones avanzadas o complejas, para facilitar programacion.

-Orientada a objetos.

-Es gráficos o visuales.

Emplea programas que permiten obtener un código tipo objeto, el cual debe enlazarse; para obtener un código binario, o de maquina o .exe

La finalización puede ser de comandos del programa y del código fuente.

NOP; No opera

CLD; Activa bandera de dirección

Pueden usarse corchetes, indicando con ello que se va a trabajar con el término de la posición de memoria que están apuntando.

MOV AX,[200] ;Cargar en AX el contenido de la posición de memoria 200.

MOV AX, 200 ;Cargar en AX el valor numérico 200.

MOV [200], AX;

MOV 200, AX; Incorrecto, no se le puede colocar un valor a un numero.

Las instrucciones en LE se dividen en grupos según su accionar. Los principales grupos:

1. Aritméticas

ADD

DIV

MUL

SUB

ADC ; Suma con acarreo

SBB ; Resta c préstamo

2. Lógicas

AND

OR

NOT

XOR

3. Transferencia

A)

Interior de la PC:

uP => memoria

MOV reg, {reg/memoria/inmediato}

PUSH ; Ingresar a la pila

PUSH AX

POP ; extraer.

Ej:

PUSH AX

POP BX

o

MOV BX,AX

B)

IN Lee una dirección.

OUT Escribe una dirección.

IO/M'

Si está haciendo out =1

Si está haciendo Mov=0

8086 64K de direcciones

4. Control: Permiten variar la secuencia del prog haciendo que partes del código o no se ejecuten o se ejecuten varias veces.

A) **SALTOS**

A.1) **Incondicionales**:Varían el flujo del prog sin ninguna condición (bandera).

JMP etiqueta

**Etiqueta**: Hace la referencia a una porción en particular del programa.

Ej:

JMP salir

----

----

-----

Salir: ; las etiquetas de salto siempre tienen un carácter de **:** al final

-----

---

---

---

A.2) **Condicionales**:

Dependen de un valor, usualmente el estado de una bandera (REG de banderas).

Depende de la ALU, entonces, en la instrucción anterior debe colocarse una instrucciónlógica o aritmética, que permita actualizar el contenido del registro de banderas, haciendo que la evaluación tenga sentido:

Ej: DEC, ADD, INC, etc.

Ej:

JNE etiqueta ; Jump not equal

JNZetiqueta ; Jump not zero

JZ etiqueta ; Jump if zero

JE etiqueta; Jump if equal

Para lazos:

LOOP etiqueta, opera con CX.

Loop CX le asigna CX-1 y salta si CX es diferente de 0.

Ej:

MOV BX, Puerto ; Dirección de un puerto

MOV AX, 0Fh ;

MOV CX, 10 ;

otro: OUT DX,AX ; Envié el contenido de AX a DX.

loop otro

B) **LLAMADOS**

Se hace un llamado a subrutina:

CALL etiqueta

Esta lleva el flujo del programa a un procedimiento cuyo inicio se maneja mediante una etiqueta.

Ej:

CALL tecla

---

---

---

--

Tecla proc

---

---

ret ; es un retorno

Endp ; Endprocedure

El procedimiento es un grupo de instrucciones que son agrupadas bajo un mismo nombre y pueden ser llamadas las veces que sean dentro del programa. Debe terminar con la instrucción **ret** de lo contrario pueden tomarse instrucciones de un procedimiento ajeno.

5) Control del uP:

Permiten modificar ciertos valores del uP como tal, como la modificación de banderas, generar tiempos de espera, o detener la acción del uP.

Ej:

CLC

CLD

NOP

WAIT

**Formado general del código fuente**

Formato interno->sintaxis

Formato externo->Comandos que establecen el entorno operativo del programa->**Declaración de segmentos:**

CS

DS

SS

ES

Declaración de segmentos:

-Simplificada

-Ampliada

Todo programa en ensamblador debe contener la declaracion de las "regiones" del programa que estarán en memoria, esto es necesario fisicamente para ubicar al programa como un ente completo a ser ejecutado.

La declaración simplificada se basa en diferentes modelos de memoria que pueden usarse. Y se basa en los segmentos que requieren el programa.

Ej:

.model<tabla pagina 11>

.stack<tamano>

Indica cuanta memoria se utilizara para la pila (lifo) usualmente es suficiente con un tamano de 256B

Tabla en pagina 11.

Ej:

.Stack 100h;

.data ; segmento de datos

Variables/ctes/textos/mensajes a pantalla.

Debe cargarse manualmente en el registro DS y requiere para su uso una serie de desplazamientos. (offset) dados por la instruccion LEA y OFFSET.

.code

Este da inicio al programa como tal, aca se escriben todas las lineas de codigo del archivo. Solo el progrma, puede iniciar con una etiqueta y debe terminar con un END.