

Tema 3: Ensambladores y macroensambladores

Objetivo: El alumno explicará el funcionamiento y llevará a cabo el diseño de un programa ensamblador capaz de procesar un lenguaje simbólico y las directivas comúnmente empleadas; asimismo, a través de la extensión de los conocimientos adquiridos diseñará un macroensamblador.

3.1: El lenguaje de máquina y el lenguaje humano: Necesidad de un traductor

- El **código o lenguaje máquina** es una **colección de dígitos binarios** que la computadora es capaz de leer e interpretar.
- El código máquina es el único lenguaje que una computadora es capaz de interpretar (...por ahora).

Ejemplo de cadena de texto en lenguaje máquina

- “Hola mundo”
- 0100 1000 0110 1111 0110 1100 0110 0001 0010 0000 0110 1101 0111 0101
0110 1110 0110 0100 0110 1111

Necesidad de un traductor

- Los **lenguajes de programación** surgen por la **necesidad** del ser humano de **dar instrucciones a una computadora** en un lenguaje que sea comprensible.
- Se estandariza el idioma inglés al ser un idioma globalizado. La mayoría de lenguajes de programación utilizan el idioma inglés como base.
- El **lenguaje ensamblador** utiliza **mnemónicos** para representar instrucciones de una manera más entendible para el ser humano.

Necesidad de un traductor

- Una instrucción en lenguaje ensamblador representa una instrucción en código máquina. Una instrucción en código máquina es una cadena de bits que representan ciertas acciones para el procesador.

Tipos de traductor

- **Intérprete:** Es un programa capaz de analizar y ejecutar otros. Realiza la traducción a medida que sea necesario (traduce en tiempo de ejecución).
- **Compilador:** Programa que analiza un código fuente de alto nivel y lo traduce en código máquina que un CPU puede ejecutar (traduce en tiempo de compilación).
- **Ensamblador:** Programa que analiza un código fuente de bajo nivel y lo traduce en código máquina que un CPU puede ejecutar (traduce en tiempo de ensamblado).

Traducción (ensamblado)

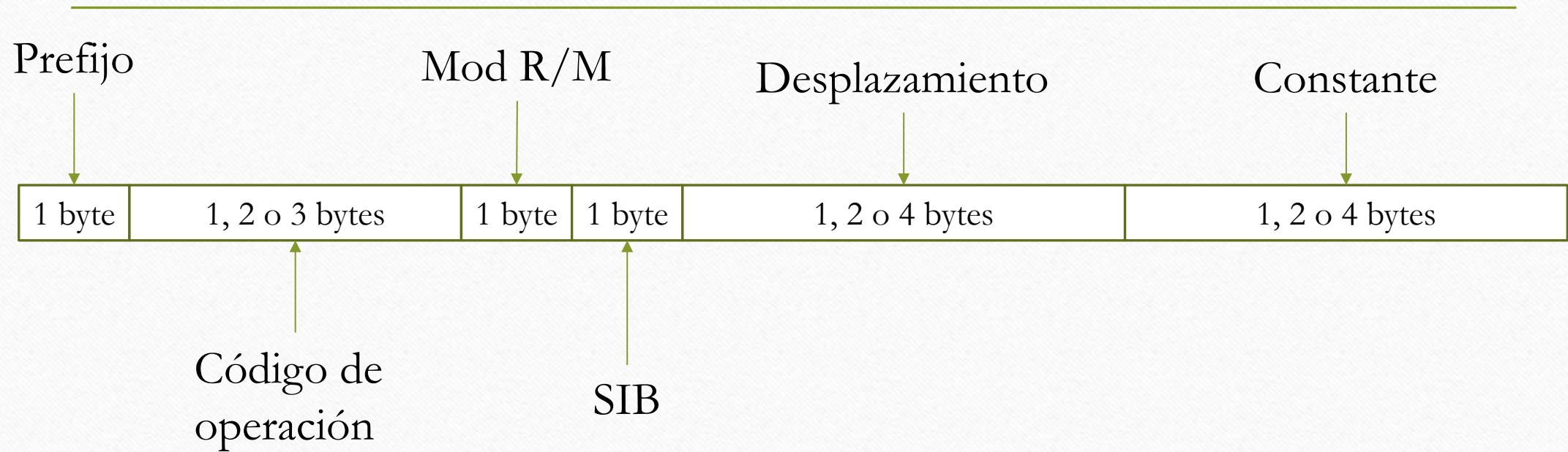
- El **ensamblador** se encarga de traducir el código fuente (lenguaje ensamblador) a código máquina.
- La traducción depende, entre otras cosas, del código de operación de la instrucción. También dependerá de los operandos y de los modos de direccionamiento utilizados en la misma instrucción.

Traducción (ensamblado)

- Instrucción MOV: Esta instrucción puede tener diferentes códigos de operación según su implementación.

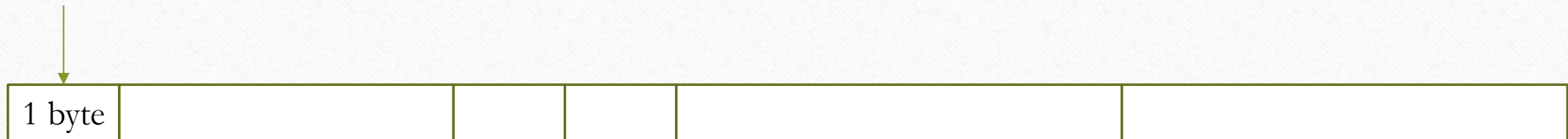
Código de operación	Instrucción
88h	mov r/m8,r8
89h	mov r/m16,r16
8Ah	mov r8,r/m8
8Bh	mov r16,r/m16
...	...

Formato de una instrucción en arquitectura Intel



Formato de una instrucción en arquitectura Intel

Prefijo



Se utiliza en modo de 64 bits

***Byte opcional**

Formato de una instrucción en arquitectura Intel

Código de operación



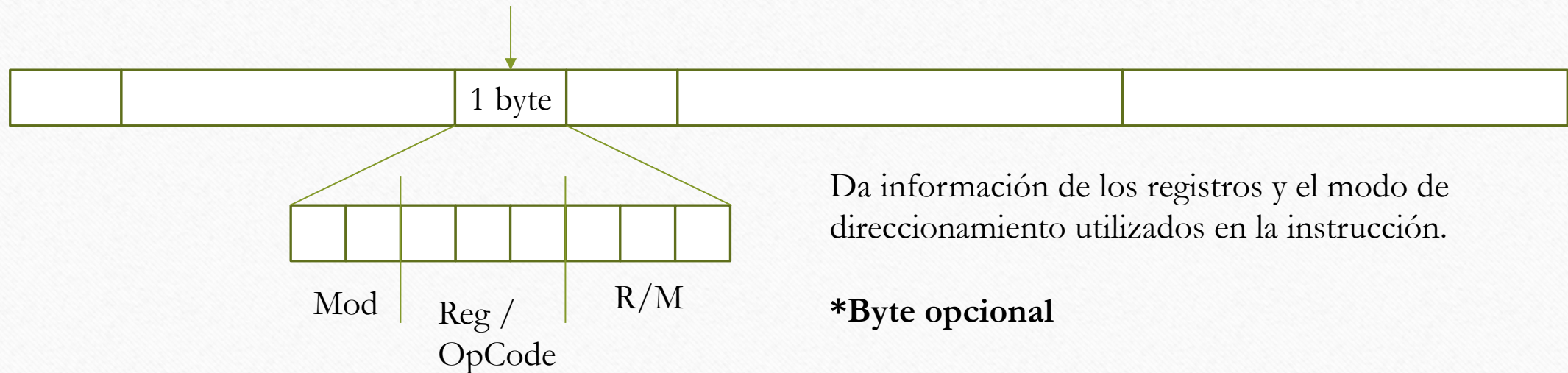
Sirve al procesador para indicar la instrucción de la que se trata.

Con el código de operación mismo, el procesador determina la longitud total de la instrucción.

El código de operación puede ser de 1, 2 o 3 bytes de longitud.

Formato de una instrucción en arquitectura Intel

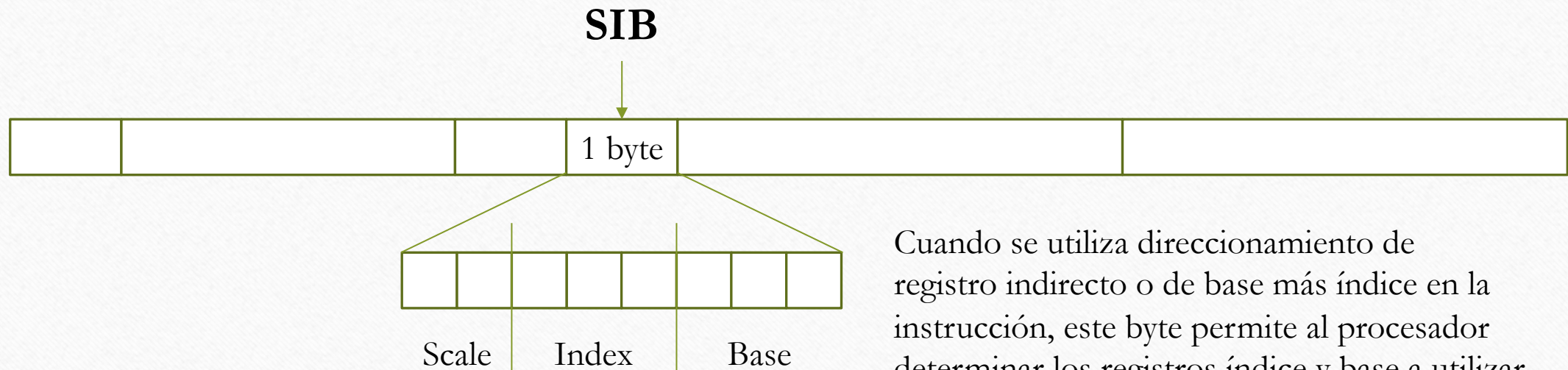
Mod R/M



Da información de los registros y el modo de direccionamiento utilizados en la instrucción.

***Byte opcional**

Formato de una instrucción en arquitectura Intel

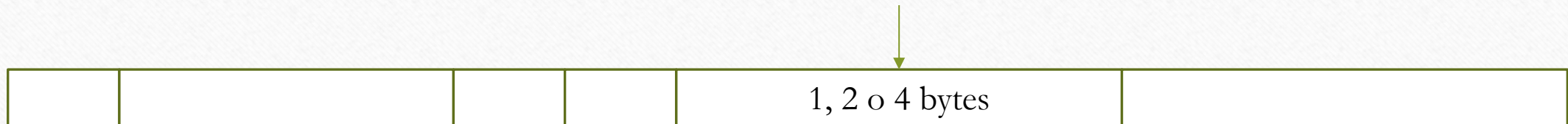


Cuando se utiliza direccionamiento de registro indirecto o de base más índice en la instrucción, este byte permite al procesador determinar los registros índice y base a utilizar.

***Byte opcional**

Formato de una instrucción en arquitectura Intel

Desplazamiento

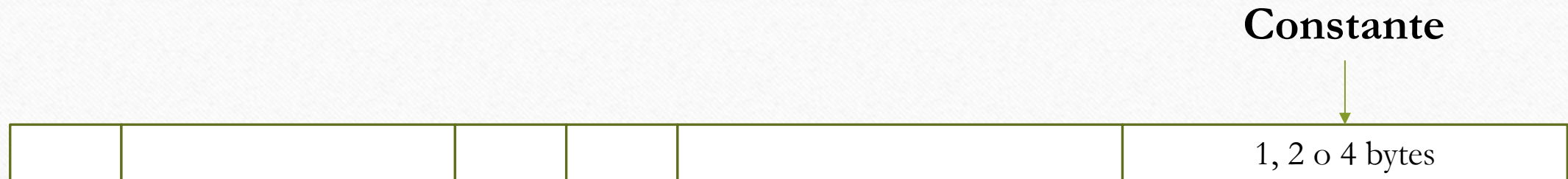


Determina el desplazamiento para instrucciones que lo requieran.

Dependiendo la instrucción, el desplazamiento se va a aplicar de diferente manera.

***Bytes opcionales, la longitud puede ser 0 si la instrucción no requiere desplazamiento**

Formato de una instrucción en arquitectura Intel



Especifica un valor constante si la instrucción lo especifica.
La constante puede ser de tamaño de 8 bits, 16 bits o 32 bits.

***Bytes opcionales, la longitud puede ser 0 si la instrucción no requiere un valor constante.**

Traducción (ensamblado)

- Un ensamblador reconoce los mnemónicos del lenguaje. Además, reconoce los operandos que siguen al mnemónico para generar el código máquina correspondiente a la instrucción.
- Al final, el ensamblador generará una cadena de bits por cada instrucción, utilizando el formato de instrucción que corresponda al procesador.

3.2: Características de un lenguaje simbólico

- En programación, **un lenguaje simbólico** es un lenguaje que utiliza caracteres o símbolos para representar conceptos, tales como operaciones, y las entidades (operandos) sobre los que se aplican estas operaciones.
- **Lenguaje formal:** Lenguaje cuyos símbolos primitivos y reglas para unir esos símbolos están formalmente especificados.
 - **Símbolos primitivos:** alfabeto.
 - **Conjunto de reglas:** gramática.

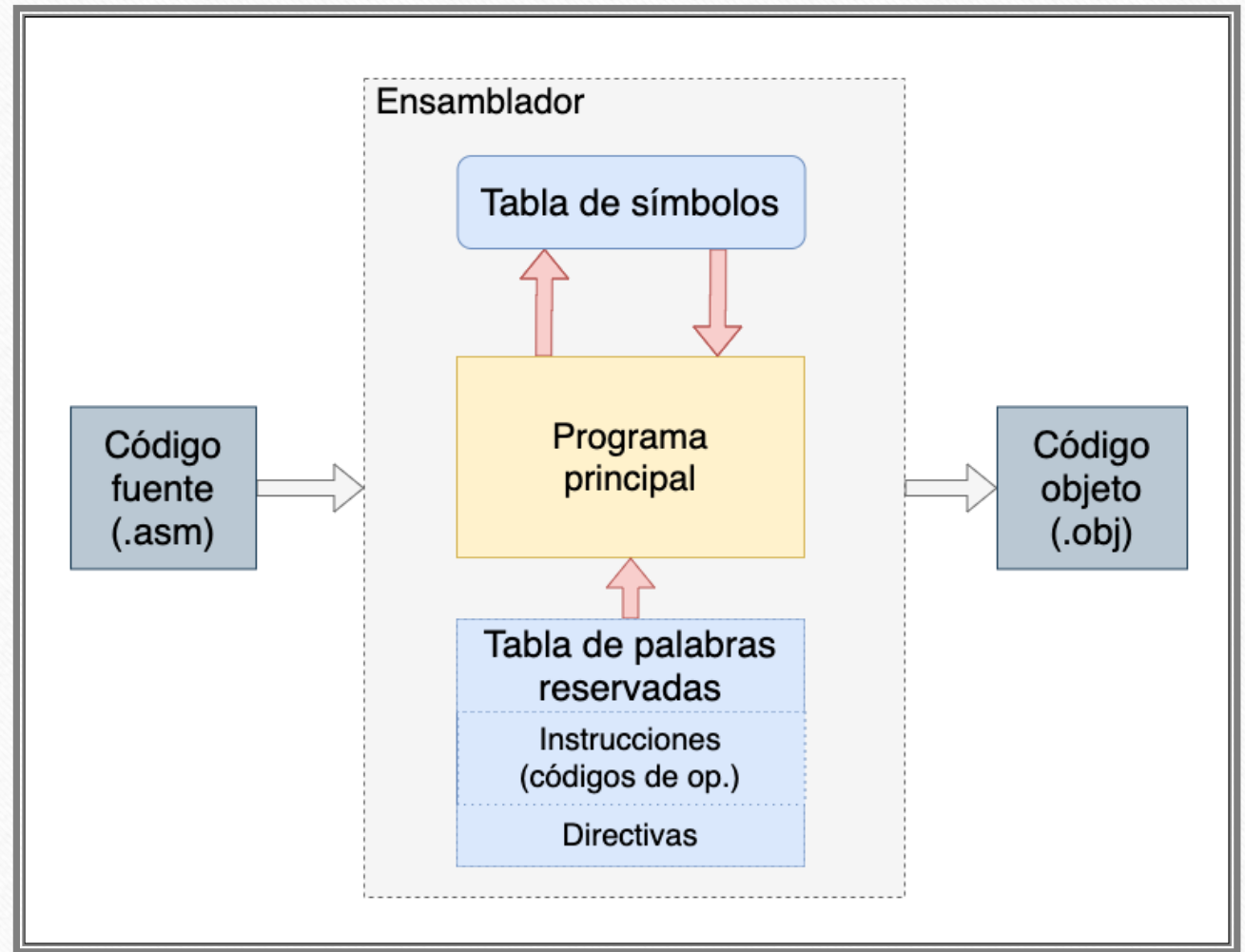
Lenguaje simbólico

- **Comprensible**
- **Bien definido**
- **Estandarizado**
- En programación, un lenguaje simbólico **debe poder traducirse en lenguaje máquina.**

3.3: Funciones y características de un ensamblador

- El ensamblador es un traductor de código fuente en lenguaje ensamblador a código máquina.
- Por la forma en la que trabaja un ensamblador, puede funcionar de dos maneras:
 - **Ensamblador de línea:** recibe una sola línea de programa y la ensambla independientemente del resto del programa.
 - **Ensamblador de archivo:** aquel que ensambla todo un programa almacenado en un archivo.

Funcionamiento de un ensamblador



Funcionamiento de un ensamblador

- El ensamblador es un programa que se encarga de realizar la traducción del código fuente a un código objeto. El código objeto es una secuencia de bits que corresponde al código fuente ensamblado.
- Mientras realiza la traducción, el ensamblador debe consultar ciertas tablas para el análisis del código. Estas tablas son:
 - Tabla de palabras reservadas, que incluye instrucciones y directivas, al igual que nombres de registros, operadores, símbolos predefinidos (como @data).
 - Tabla de símbolos.

Tabla de palabras reservadas

- Son palabras reconocidas por el ensamblador y que no deberían ser utilizadas como identificadores para variables, etiquetas y/o procedimientos debido a que puede causar problemas al traducir el código.
- Las palabras reservadas pueden ser mnemónicos de instrucciones o pueden ser directivas.

Tabla de palabras reservadas

- **Tabla de instrucciones (códigos de operación):** Contiene información de las instrucciones disponibles del procesador para el cual se realiza la traducción. Cada instrucción contiene su correspondiente código de operación que le servirá al ensamblador para generar el código objeto correspondiente a la instrucción, y también contiene la longitud de la instrucción considerando sus operandos.

Tabla de palabras reservadas

- **Tabla de directivas:** Contiene una lista de palabras que señalan operaciones específicas al ensamblador.
- Las directivas son comandos para el ensamblador que le indican cómo ensamblar el programa.
- Las directivas tienen una sintaxis válida en lenguaje ensamblador pero no corresponden a instrucciones del procesador.

Tabla de símbolos

- Es una tabla generada por el ensamblador y contiene las palabras utilizadas para identificar variables, etiquetas, macros o procedimientos (identificadores), que no representan constantes o palabra reservada alguna.
- La tabla de símbolos es una tabla interna y dinámica que se genera y se usa por el ensamblador.

Ejemplo de tabla de símbolos

Nombre	Segmento	Tamaño	Valor	Tipo (D,U)
var1	Datos	Word	0000h	D
var2	Datos	Byte	0001h	D
contador	Datos	Byte	0002h	D
inicio	Código	Word	0000h	D
et1	Código	Word	002Ch	D
salir	Código	Word	005Ch	D

Tabla de símbolos

- El manejo de símbolos consiste de dos partes: definir el símbolo y usarlo. Un símbolo sólo se puede definir una sola vez pero puede usarse cuantas veces sea necesario.
- Definición de datos
var1 db 0
- Definición en código
et1:

Tabla de símbolos

- Para determinar una localidad de memoria definida por una etiqueta, el ensamblador utiliza un contador de localidades (LC).

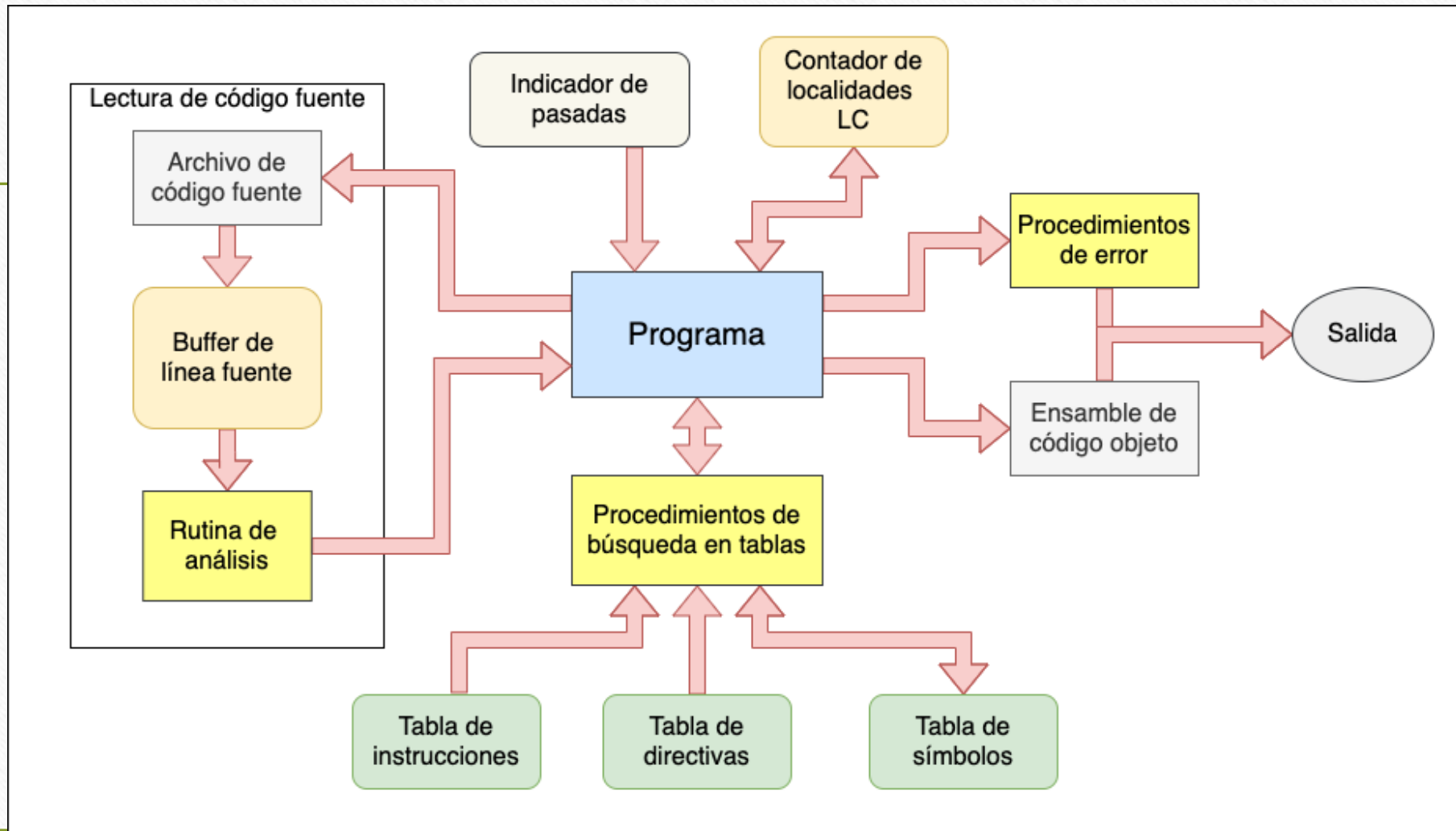
		LC=0
mov ax,@data	3 bytes	LC=3
mov ds,ax	3 bytes	LC=6
mov al,6	2 bytes	LC=8
mov bl,5	2 bytes	LC=10
mov cx,5	3 bytes	LC=13
loop1:		
add al,bl	2 bytes	LC=15
loop loop1	3 bytes	LC=18

Tabla de símbolos

En el ejemplo anterior, cuando se define la etiqueta “loop1”, el valor de LC es 13 (000Dh). Por lo tanto, en la tabla de símbolos, el ensamblador registra el símbolo “loop1” con un valor de 000Dh.

Nombre	Segmento	Tamaño	Valor	Tipo (D,U)
...
loop1	Código	Word	000Dh	D
...

Componentes de un programa ensamblador



Tipos de ensamblador

- **Ensamblador propio:** Ensambla programas escritos en lenguaje del procesador con que trabaja la máquina en la que se ensambla.
- **Ensamblador cruzado:** Ensambla programas escritos en lenguaje de un procesador distinto al de la computadora de trabajo, pero no necesariamente se ejecuta en la misma.
- **Ensamblador residente:** Es un ensamblador que siempre se encuentra cargado en memoria y reside en ROM.

Tipos de ensamblador

- **Desensamblador:** Es el opuesto de un ensamblador. Traduce código máquina en un código fuente en lenguaje ensamblador.
- **Metaensamblador:** Aquel ensamblador que puede lidiar con varios sets de instrucciones.
- **Macroensamblador:** También llamado ensamblador modular. Es aquel que permite el uso de macroinstrucciones (macros).

Pasadas en un ensamblador

- Una característica de los ensambladores es el número de pasadas que realiza.
- El número de pasadas se refiere al número de veces que se va a recorrer el código fuente para ensamblarlo.
- Un ensamblador puede ser de una o de múltiples pasadas:
 - **Una pasada:** recorre el código fuente en una sola ocasión y genera el código objeto al mismo tiempo que se recorre.
 - **Múltiples pasadas:** recorre el código fuente en más de una ocasión para resolver referencias y al final generar un código objeto.

Ensamblador de una pasada

- Lee el archivo de código fuente sólo una vez.
- En un paso, el ensamblador manipula la definición de un símbolo y el ensamblado. Si el ensamblador encuentra un símbolo que aún no está definido, lo marca como tipo **U** (Undefined) dentro de la tabla de símbolos

Ensamblador de una pasada

- Ejemplo:

000Ah	je et1	=> et1	Código	?	U
...					
001Eh	jnz et1				
...					
003Fh	jmp et1				
...					
004Ah	et1: mov ax,bx	=> et1	Código	004Ah	D

Ensamblador de dos pasadas

- El ensamblador recorre el código fuente en 2 ocasiones.
- En el primer recorrido (primera pasada) el ensamblador encuentra todos los símbolos y se le asignan valores, entonces se ponen en la tabla de símbolos. Ninguna instrucción se ensambla en la primera pasada.
- En el segundo recorrido (segunda pasada) se vuelven a leer las instrucciones y se ensamblan utilizando la tabla de símbolos.

Ensamblador de dos pasadas

- Las principales características que resuelve un ensamblador de dos pasadas son:
 - Referencias futuras (uso de símbolos antes de su definición).
 - Archivos objeto reubicables.

Ensamblador de una y media pasadas

- Es una técnica de ensamblado intermedia entre una pasada y dos pasadas. La utilizaban algunos ensambladores antiguos.
- En la primera pasada se genera un archivo objeto pero incompleto. Las instrucciones con referencias hacia adelante están parcialmente ensambladas y van al código objeto con un espacio en ellas. Cada vez que se usa un símbolo futuro se genera un entrada en tabla de símbolos. Al final de la primera pasada, la tabla de símbolos estará completa.

Ensamblador de una y media pasadas

- En la segunda pasada (media pasada), el ensamblador recorre el código fuente en sentido inverso (de adelante hacia atrás). Recorre instrucción por instrucción y utiliza la información de la tabla de símbolos para completar las instrucciones en el código objeto.

3.4: Diseño de un ensamblador

- Ensamblador de una pasada
- https://drive.google.com/file/d/1Qp_bubuEDlv3FtS1fDHQqBG1M-dbeCVG/view

Ejemplo: Ensamblador de una pasada

Se tiene el siguiente código fuente de un programa en lenguaje ensamblador para arquitectura Intel x86.

En este ejemplo práctico se omitirá el ensamblado de las directivas.

De la tabla de instrucciones sólo se utilizarán los códigos de operación para las instrucciones necesarias.

```
.data
x      db      2
y      db      1
z      dw      0

.code
inicio:
    mov ax,@data
    mov ds,ax
for:
    mov al,[x]
    mov bl,[y]
    mul bl
    mov [z],ax
    cmp [z],10
    jg salir
    inc [w]
    jmp for
salir:
    mov ah,4Ch
    mov al,0
    int 21h
end inicio
```

Instrucción	Código de operación	Longitud
mov ax,imm16	B8 ?? ??	3 bytes
mov ds,ax	8E D8	2 bytes
mov al,m8	A0 ?? ??	3 bytes
mov bl,m8	8A 1E ?? ??	4 bytes
mul bl	F6 E3	2 bytes
mov m16,ax	A3 ?? ??	3 bytes
cmp m16,imm8	83 3E ?? ?? ??	5 bytes
jg rel8	7F ??	2 bytes
jg rel16	0F 8F ?? ??	4 bytes
inc m8	FE 06 ?? ??	4 bytes
jmp rel8	EB ??	2 bytes
mov ah,imm8	B4 ??	2 bytes
mov al, imm8	B0 ??	2 bytes
int imm8	CD ??	2 bytes

De la tabla de instrucciones, se tiene:

```

.data
x      db      2
y      db      1
z      dw      0

.code
inicio:
    mov ax,@data
    mov ds,ax
for:
    mov al,[x]
    mov bl,[y]
    mul bl
    mov [z],ax
    cmp [z],10
    jg salir
    inc [w]
    jmp for
salir:
    mov ah,4Ch
    mov al,0
    int 21h
    end inicio

```


Ejemplo: Ensamblador de una pasada

```

      .data
x      db      2
y      db      1
z      dw      0

```

Antes de comenzar a ensamblar el código del programa, el ensamblador revisa las líneas anteriores a la sección **.code**. Se hace la revisión de las directivas y búsqueda de símbolos. En el código fuente, se puede observar que hay símbolos definidos en la sección **.data**, los cuales se insertan en la tabla de símbolos.

Tabla de símbolos

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D

Ejemplo: Ensamblador de una pasada

```
z      dw      0
.code
inicio:
mov ax,@data
```

Cuando el ensamblador llega a la línea que contiene la directiva `.code`, eso le indica que comienza el ensamblado de la sección del código de programa. Se inicializa LC en 0.

LC = 0

Ejemplo: Ensamblador de una pasada

```
.code  
inicio:  
    mov ax,@data  
    mov ds,ax  
-
```

Se lee la primera línea, siguiendo la estructura del diagrama de flujo.

LC = 0

Ejemplo: Ensamblador de una pasada

```
.code
inicio:
    mov ax,@data
    mov ds,ax
```

Se lee la primera línea, siguiendo la estructura del diagrama de flujo.

Se trata de la definición de una etiqueta. Por lo tanto, se insertará en la tabla de símbolos.

Tabla de símbolos

LC = 0

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D

Ejemplo: Ensamblador de una pasada

```
.code
inicio:
    mov ax,@data
    mov ds,ax
```

Se lee la primera línea, siguiendo la estructura del diagrama de flujo.

Se trata de la definición de una etiqueta. Por lo tanto, se insertará en la tabla de símbolos.

Tabla de símbolos

LC = 0

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D

Ejemplo: Ensamblador de una pasada

```
.code  
inicio:  
    mov ax,@data  
    mov ds,ax  
-
```

Se procede a ensamblar la línea, sin embargo no contiene instrucción, así que se procede a leer la siguiente línea de código y LC no se afecta.

LC = 0

Ejemplo: Ensamblador de una pasada

inicio:

mov ax,@data

mov ds,ax

for:

- - -

Se lee la siguiente línea del código fuente.

LC = 0

Ejemplo: Ensamblador de una pasada

inicio:

mov ax,@data

mov ds,ax

for:

- - -

Se lee la siguiente línea del código fuente.

Asumiremos que el valor de @data es 48B0.

LC = 0

Ejemplo: Ensamblador de una pasada

inicio:

mov ax,@data

mov ds,ax

for:

- - -

Se lee la siguiente línea del código fuente.

Asumiremos que el valor de @data es 48B0.

Se ensambla la línea tomando el código de operación de la tabla de instrucciones y completando con el valor.

LC = 0

Tabla de instrucciones

Instrucción	Código de operación	Longitud
mov ax,imm16	B8 ?? ??	3 bytes

Ejemplo: Ensamblador de una pasada

inicio:

mov ax,@data

mov ds,ax

for:

- - -

Se ensambla la línea tomando el código de operación de la tabla de instrucciones y completando con el valor.

El código objeto que se genera es: B8 B0 48.

LC = 0

Archivo Objeto

...

B8 B0 48

Ejemplo: Ensamblador de una pasada

inicio:

mov ax,@data

mov ds,ax

for:

- - -

Se ensambla la línea tomando el código de operación de la tabla de instrucciones y completando con el valor.

El código objeto que se genera es: B8 B0 48.

LC se debe incrementar según la longitud de la instrucción.

$$\text{LC} = 0 + 3 = 3$$

Ejemplo: Ensamblador de una pasada

```
mov ax,@data  
mov ds,ax  
for:  
mov al,[x]
```

Se lee la siguiente línea del código fuente.

LC = 3

Ejemplo: Ensamblador de una pasada

```
mov ax,@data
mov ds,ax
for:
mov al,[x]
```

Se lee la siguiente línea del código fuente.

Se ensambla la línea, utilizando su código de operación.

Instrucción	Código de operación	Longitud
mov ds,ax	8E D8	2 bytes

LC = 3

Ejemplo: Ensamblador de una pasada

```
...  
    mov ax,@data  
    mov ds,ax  
for:  
    mov al,[x]
```

Se lee la siguiente línea del código fuente.

Se ensambla la línea, utilizando su código de operación.

Se genera el código objeto y se imprime el archivo objeto.

LC = 3

Archivo Objeto

...

B8 B0 48 **8E D8**

Ejemplo: Ensamblador de una pasada

```
112000:
    mov ax,@data
    mov ds,ax
for:
    mov al,[x]
```

Se ensambla la línea, utilizando su código de operación.

Se genera el código objeto y se imprime el archivo objeto.

Se incrementa LC.

LC = 5

Ejemplo: Ensamblador de una pasada

```
mov ax, 0x0000  
mov ds, ax  
for:  
mov al, [x]  
mov bl, [v]
```

Se lee la siguiente línea de código fuente.

LC = 5

Ejemplo: Ensamblador de una pasada

```
mov ax, 0000h  
mov ds, ax  
for:  
mov al, [x]  
mov bl, [y]
```

Se lee la siguiente línea de código fuente.

Al igual que en la primera línea, se trata de la definición de una etiqueta.

Se inserta en la tabla de símbolos utilizando el valor de LC.

LC = 5

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D

Ejemplo: Ensamblador de una pasada

```
mov ax, 0000h  
mov ds, ax  
for:  
mov al, [x]  
mov bl, [y]
```

Se lee la siguiente línea de código fuente.

Al igual que en la primera línea, se trata de la definición de una etiqueta.

Se inserta en la tabla de símbolos utilizando el valor de LC.

LC = 5

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D

Ejemplo: Ensamblador de una pasada

for:

```
mov al, [x]  
mov bl, [y]  
mul bl
```

Se lee la siguiente línea de código fuente.

LC = 5

Ejemplo: Ensamblador de una pasada

for:

```
mov al, [x]  
mov bl, [y]  
mul bl
```

Se lee la siguiente línea de código fuente.

En este caso se está utilizando un símbolo. Se revisa si el símbolo existe en la tabla de símbolos.

LC = 5

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D

Ejemplo: Ensamblador de una pasada

for:

```
mov al, [x]  
mov bl, [y]  
mul bl
```

Se lee la siguiente línea de código fuente.

En este caso se está utilizando un símbolo. Se revisa si el símbolo existe en la tabla de símbolos.

El símbolo existe y es de tipo D. Por lo tanto, se ensambla la línea.

LC = 5

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D

Ejemplo: Ensamblador de una pasada

for:

mov *al*, [x]

mov *bl*, [y]

mul *bl*

Se lee la siguiente línea de código fuente.

En este caso se está utilizando un símbolo. Se revisa si el símbolo existe en la tabla de símbolos.

El símbolo existe y es de tipo D. Por lo tanto, se ensambla la línea.

El código objeto que se genera es: A0 00 00

LC = 5

Archivo Objeto

...

B8 B0 48 8E D8 **A0 00 00**

Ejemplo: Ensamblador de una pasada

for:

mov *al*, [x]

mov *bl*, [y]

mul *bl*

En este caso se está utilizando un símbolo. Se revisa si el símbolo existe en la tabla de símbolos.

El símbolo existe y es de tipo D. Por lo tanto, se ensambla la línea.

El código objeto que se genera es: A0 00 00

Se incrementa LC

LC = 8

Archivo Objeto

...

B8 B0 48 8E D8 **A0 00 00**

Ejemplo: Ensamblador de una pasada

101 .

mov *al*, [x]

mov *bl*, [y]

mul *bl*

mov [z].*ax*

Se lee la siguiente línea de código fuente.

LC = 8

Ejemplo: Ensamblador de una pasada

```
101 .  
    mov al, [x]  
    mov bl, [y]  
    mul bl  
    mov [z].ax
```

Se lee la siguiente línea de código fuente.

Sucede algo similar que en el paso anterior, donde se utiliza un símbolo que existe en la tabla de símbolos y es de tipo D.

LC = 8

Nombre	Segmento	Tamaño	Valor	Tipo
y	Datos	Byte	0001h	D

Ejemplo: Ensamblador de una pasada

```
101 .  
    mov al, [x]  
    mov bl, [y]  
    mul bl  
    mov [z], ax
```

Se lee la siguiente línea de código fuente.

Sucede algo similar que en el paso anterior, donde se utiliza un símbolo que existe en la tabla de símbolos y es de tipo D.

El código objeto que se genera es: 8A 1E 01 00

Se incrementa LC

LC = 12

Archivo Objeto

...

B8 B0 48 8E D8 A0 00 00 8A 1E 01 00

Ejemplo: Ensamblador de una pasada

```
mov bl, [y]  
mul bl  
mov [z], ax  
cmp [z], 10
```

Se lee la siguiente línea de código fuente.

LC = 12

Ejemplo: Ensamblador de una pasada

```
mov bl, [y]  
mul bl  
mov [z], ax  
cmp [z], 10
```

Se lee la siguiente línea de código fuente.

No hay símbolos, así que se ensambla y genera el código objeto, que se escribe en el archivo y luego se incrementa LC

LC = 14

Archivo Objeto

...

B8 B0 48 8E D8 A0 00 00 8A 1E 01 00 **F6 E3**

Ejemplo: Ensamblador de una pasada

```
mov ax, 100  
mul bl  
mov [z], ax  
cmp [z], 10  
ja salir
```

Se lee la siguiente línea de código fuente.

LC = 14

Ejemplo: Ensamblador de una pasada

```
mov ax, 100  
mul bl  
mov [z], ax  
cmp [z], 10  
ja salir
```

Se lee la siguiente línea de código fuente.

Utiliza símbolos, se busca en la tabla. Como existe, la línea se ensambla.

LC = 14

Ejemplo: Ensamblador de una pasada

```
mov [z],ax  
mul bl  
mov [z],ax  
cmp [z],10  
ja salir
```

Se lee la siguiente línea de código fuente.

Utiliza símbolos, se busca en la tabla. Como existe, la línea se ensambla.
Se incrementa LC.

LC = 17

Archivo Objeto

...

B8 B0 48 8E D8 A0 00 00 8A 1E 01 00 F6 E3 A3 02 00

Ejemplo: Ensamblador de una pasada

```
mul  dx  
mov  [z],ax  
cmp  [z],10  
jg   salir
```

Se lee la siguiente línea de código fuente.

LC = 17

Ejemplo: Ensamblador de una pasada

```
mul dl  
mov [z],ax  
cmp [z],10  
jq salir
```

Se lee la siguiente línea de código fuente.

Se utiliza símbolo, se ensambla la línea y se inserta el código objeto al archivo además de incrementar LC

LC = 22

Archivo Objeto

...

B8 B0 48 8E D8 A0 00 00 8A 1E 01 00 F6 E3 A3 02 00 **83 3E 02 00 0A**

Ejemplo: Ensamblador de una pasada

```
mov [z], 10  
cmp [z], 10  
jg salir  
inc [w]  
...
```

Se lee la siguiente línea de código fuente.

LC = 22

Ejemplo: Ensamblador de una pasada

```
mov [z], dx  
cmp [z], 10  
jg salir  
inc [w]  
...
```

Se lee la siguiente línea de código fuente.

En este caso, se encontró un símbolo que no existe en la tabla de símbolos.
Se inserta el nombre, el valor de LC y con tipo U.

LC = 22

Nombre	Segmento	Tamaño	Valor	Tipo
...
for	Código	Word	0005h	D
salir	Código	Word	0016h	U

Ejemplo: Ensamblador de una pasada

```
mov [z], dx  
cmp [z], 10  
jg salir  
inc [w]  
...
```

Se lee la siguiente línea de código fuente.

En este caso, se encontró un símbolo que no existe en la tabla de símbolos.

Se inserta con el nombre y el valor de LC y con tipo U.

Se ensambla la línea.

En todos los saltos condicionales de tipo **Jcc (Jump if...)**, el código de operación y la longitud de la instrucción depende del operando. El salto puede ser de tipo **short (8 bits)**, **near (16 bits)** o **far (32 bits)**. Debido a que en este punto el ensamblador no conoce la longitud del salto, debe estar preparado para el peor caso.

LC = 22

Ejemplo: Ensamblador de una pasada

```
mov [z], dx  
cmp [z], 10  
jg salir  
inc [w]  
...
```

Se lee la siguiente línea de código fuente.

En este caso, se encontró un símbolo que no existe en la tabla de símbolos.

Se inserta con el nombre y el valor de LC y con tipo U.

Se ensambla la línea.

En una arquitectura con registros de 16 bits, el peor caso es un salto tipo **near**. Por lo tanto, al ensamblar la línea, se genera el código objeto para cuando el salto JG es de tipo near.

Es decir, el código objeto que se genera es **0F 8F 16 00** donde 0016h es el valor de LC actual.

LC = 22

Ejemplo: Ensamblador de una pasada

```
mov [z], 0x  
cmp [z], 10  
jg salir  
inc [w]  
...
```

Se lee la siguiente línea de código fuente.

En este caso, se encontró un símbolo que no existe en la tabla de símbolos.

Se inserta con el nombre y el valor de LC

Se ensambla la línea.

Se inserta el código objeto en el archivo, además de incrementar LC.

LC = 26

Archivo Objeto

Sin embargo, la instrucción queda pendiente de resolver...

...

B8 B0 48 8E D8 A0 00 00 8A 1E 01 00 F6 E3 A3 02 00 83 3E 02 00 0A **0F 8F 16 00**

Ejemplo: Ensamblador de una pasada

```
cmp [2], 10  
jg salir  
inc [w]  
jmp for
```

Se lee la siguiente línea de código fuente.

LC = 26

Ejemplo: Ensamblador de una pasada

```
cmp [2], 10  
jg salir  
inc [w]  
jmp for
```

Se lee la siguiente línea de código fuente.
Se encuentra un símbolo que no existe.

LC = 26

Ejemplo: Ensamblador de una pasada

```
cmp [2], 10
jg salir
inc [w]
jmp for
```

Se lee la siguiente línea de código fuente.

Se encuentra un símbolo que no existe.

Se registra el símbolo en tabla de símbolos

LC = 26

Nombre	Segmento	Tamaño	Valor	Tipo
...
salir	Código	Word	0016h	U
w	Datos	?	001Ah	U

Ejemplo: Ensamblador de una pasada

```
cmp [2], 10
jg salir
inc [w]
jmp for
```

Se lee la siguiente línea de código fuente.

Se encuentra un símbolo que no existe.

Se registra el símbolo en tabla de símbolos.

**Se ensambla la línea con el valor de LC como apuntador,
genera su código objeto y se escribe en el archivo .obj.**

Se incrementa LC.

LC = 30

Archivo Objeto

...

B8 B0 48 8E D8 A0 00 00 8A 1E 01 00 F6 E3 A3 02 00 83 3E 02 00 0A **0F 8F 16 00 FE 06 1A 00**

Ejemplo: Ensamblador de una pasada

```
    jmp for  
salir:
```

Se lee la siguiente línea de código fuente.

LC = 30

Ejemplo: Ensamblador de una pasada

```
    jmp for
salir:
```

Se lee la siguiente línea de código fuente.

Se detecta un símbolo que se encuentra definido en la tabla de símbolos.

LC = 30

Nombre	Segmento	Tamaño	Valor	Tipo
...
for	Código	Word	0005h	D
salir	Código	Word	0016h	U
w	Datos	?	0018h	U

Ejemplo: Ensamblador de una pasada

```
    jmp for  
salir:
```

Se lee la siguiente línea de código fuente.

Se detecta un símbolo que se encuentra definido en la tabla de símbolos.

Al ensamblar la línea, el ensamblador calcula el desplazamiento entre la línea actual y en la que se definió el símbolo (valor en T.S.).

$$(\text{Línea de definición}) - (\text{Línea de uso} + \text{Longitud de la instrucción}) \quad \mathbf{LC = 30}$$

$$[\text{Valor en T. S.}] - (\text{LC} + \text{long. de instrucción actual}) = 0005h - (30d + 2d) = 5d - 32d = -27d = E5h$$

Se trata de un **salto short relativo**

Ejemplo: Ensamblador de una pasada

```
    jmp for
salir:
```

Se lee la siguiente línea de código fuente.

Se detecta un símbolo que se encuentra definido en la tabla de símbolos.

Al ensamblar la línea, el ensamblador calcula el desplazamiento entre la línea actual y en la que se definió el símbolo (valor del símbolo).

El valor calculado es parte del código objeto, se escribe en .obj y acumula LC.

LC = 32

Archivo Objeto

...

B8 B0 48 8E D8 A0 00 00 8A 1E 01 00 F6 E3 A3 02 00 83 3E 02 00 0A 0F 8F 16 00 FE 06 1A 00
EB E5

Ejemplo: Ensamblador de una pasada

```
inc [w]  
jmp for  
salir:  
mov ah, 4Ch
```

Se lee la siguiente línea de código fuente.

LC = 32

Ejemplo: Ensamblador de una pasada

```
inc [w]
jmp for
salir:
mov ah, 4Ch
```

Se lee la siguiente línea de código fuente.

Se detecta una definición de etiqueta. La etiqueta existe en la tabla de símbolos y es de tipo U. El valor de la tabla de símbolos apunta a la instrucción en esa posición (posición 0016h = 22d)

LC = 32

Nombre	Segmento	Tamaño	Valor	Tipo
...
salir	Código	Word	0016h	U
w	Datos	?	0018h	U

Ejemplo: Ensamblador de una pasada

```
inc [w]
jmp for
salir:
mov ah, 4Ch
```

Se lee la siguiente línea de código fuente.

Se detecta una definición de etiqueta. La etiqueta existe en la tabla de símbolos y es de tipo U. El valor de la tabla de símbolos apunta a la instrucción en esa posición (posición 0016h = 22d).

Se resuelven las instrucciones pendientes que utilizan el símbolo.

La línea en la localidad 22d es 'jg salir' (0F 8F 16 00) y apunta a sí misma.

LC = 32

Si ésta apuntara a otra instrucción también deberá resolverse.

Para resolver la referencia, se calcula:

(Línea de definición) – (Línea de uso) – (Longitud de la instrucción)

$32d - 22d - long = 10d - long = 0Ah - long \rightarrow$ Salto relativo de 8 bits (longitud: 2 bytes) $\rightarrow 0Ah - 2 = 08h$

Este valor indica que se trata de un **salto short relativo**. Para este salto se utiliza el código de operación 7F 08

Ejemplo: Ensamblador de una pasada

```
inc [w]  
jmp for  
salir:  
mov ah, 4Ch
```

Se lee la siguiente línea de código fuente.

Se detecta una definición de etiqueta. La etiqueta existe en la tabla de símbolos y es de tipo U. El valor de la tabla de símbolos apunta a la instrucción en esa posición (posición 0016h = 22d).

Se resuelven las instrucciones pendientes que utilizan el símbolo.

Debido a que el salto es de tipo short, el código de operación generado por la instrucción 'jg salir' en el archivo objeto cambia de 0F 8F 16 00 a 7F 08

LC = 32

El código de operación 7F 08 ocupa menos localidades de memoria que 0F 8F 16 00 pero si se eliminaran las que sobran, el ensamblador debería reajustar el código objeto y volver a calcular la localidad para cada instrucción y símbolo, lo cual no es lo ideal.

Ejemplo: Ensamblador de una pasada

```
inc [w]
jmp for
salir:
mov ah, 4Ch
```

Se lee la siguiente línea de código fuente.

Se detecta una definición de etiqueta. La etiqueta existe en la tabla de símbolos y es de tipo U. El valor de la tabla de símbolos apunta a la instrucción en esa posición (posición 0016h = 22d).

Se resuelven las instrucciones pendientes que utilizan el símbolo.

LC = 32

Archivo Objeto

...

B8 B0 48 8E D8 A0 00 00 8A 1E 01 00 F6 E3 A3 02 00 83 3E 02 00 0A **7F 08 16 00** FE 06 **1A 00**
EB E5

Ejemplo: Ensamblador de una pasada

```
inc [w]
jmp for
salir:
mov ah, 4Ch
```

Se lee la siguiente línea de código fuente.

Se detecta una definición de etiqueta. La etiqueta existe en la tabla de símbolos y es de tipo U. El valor de la tabla de símbolos apunta a la instrucción en esa posición (posición 0016h = 22d).

Se resuelven las instrucciones pendientes que utilizan el símbolo.

Lo que hace el ensamblador con las localidades restantes en la instrucción ‘**lg salir**’ es convertirlas en instrucciones “vacías”. Estas instrucciones realizan el ciclo de instrucción (búsqueda-decodificación-ejecución) como normalmente lo haría una instrucción común, la diferencia es que esta operación no realiza ninguna acción sobre memoria o sobre los registros, incluyendo el de estados. A esta instrucción se le denomina **NOP** (mnemónico de **No Operation**). Su código de operación es **90h**

LC = 32

Ejemplo: Ensamblador de una pasada

```
inc [w]
jmp for
salir:
mov ah, 4Ch
```

Se lee la siguiente línea de código fuente.

Se detecta una definición de etiqueta. La etiqueta existe en la tabla de símbolos y es de tipo U. El valor de la tabla de símbolos apunta a la instrucción en esa posición (posición 0016h = 22d).

Se resuelven las instrucciones pendientes que utilizan el símbolo.

LC = 32

Archivo Objeto

...

B8 B0 48 8E D8 A0 00 00 8A 1E 01 00 F6 E3 A3 02 00 83 3E 02 00 0A 7F 08 90 90 FE 06 1A 00
EB E5

Ejemplo: Ensamblador de una pasada

```
inc [w]
jmp for
salir:
mov ah, 4Ch
```

Se lee la siguiente línea de código fuente.

Se detecta una definición de etiqueta. La etiqueta existe en la tabla de símbolos y es de tipo U. El valor de la tabla de símbolos apunta a la instrucción en esa posición (posición 0016h = 22d).

Se resuelven las instrucciones pendientes que utilizan el símbolo.

Se escribe en tabla de símbolos el valor de LC **LC = 32**
en el símbolo resuelto y cambia a tipo D.

Nombre	Segmento	Tamaño	Valor	Tipo
...
salir	Código	Word	0020h	D
w	Datos	?	0018h	U

Ejemplo: Ensamblador de una pasada

```
inc [w]  
jmp for  
salir:  
mov ah, 4Ch
```

Se lee la siguiente línea de código fuente.

Se detecta una definición de etiqueta. La etiqueta existe en la tabla de símbolos y es de tipo U. El valor de la tabla de símbolos apunta a la instrucción en esa posición (posición 0016h = 22d).

Se resuelven las instrucciones pendientes que utilizan el símbolo.

Se escribe en tabla de símbolos el valor de LC
en el símbolo resuelto.

LC = 32

Se ensambla pero no hay instrucción.

No genera código objeto y no incrementa LC.

Ejemplo: Ensamblador de una pasada

```
    jmp 10h  
salir:
```

Se lee la siguiente línea de código fuente.

```
    mov ah,4Ch
```

```
    mov al,0
```

```
    int 0Ah
```

LC = 32

Ejemplo: Ensamblador de una pasada

```
    jmp 101  
salir:
```

```
    mov ah, 4Ch
```

```
    mov al, 0
```

```
    int 0Ah
```

Se lee la siguiente línea de código fuente.

No hay símbolos. La línea se ensambla, genera código objeto y se escribe en el archivo objeto.

Se incrementa LC.

LC = 34

Archivo Objeto

...

B8 B0 48 8E D8 A0 00 00 8A 1E 01 00 F6 E3 A3 02 00 83 3E 02 00 0A 7F E5 90 90 FE 06 1A 00
EB E5 B4 4C

Ejemplo: Ensamblador de una pasada

```
salir:
    mov ah, 4Ch
    mov al, 0
    int 21h
    . . .
```

Se lee la siguiente línea de código fuente.

LC = 34

Ejemplo: Ensamblador de una pasada

```
5a111:
    mov ah,4Ch
    mov al,0
    int 21h
    . . . .
```

Se lee la siguiente línea de código fuente.

No hay símbolos. La línea se ensambla, genera código objeto y se escribe en el archivo objeto.

Se incrementa LC.

LC = 36

Archivo Objeto

...

B8 B0 48 8E D8 A0 00 00 8A 1E 01 00 F6 E3 A3 02 00 83 3E 02 00 0A 7F E5 90 90 FE 06 1A 00
EB E5 B4 4C B0 00

Ejemplo: Ensamblador de una pasada

```
mov al, 0  
int 21h  
end inicio
```

Se lee la siguiente línea de código fuente.

LC = 36

Ejemplo: Ensamblador de una pasada

```
mov al,0  
int 21h  
end inicio
```

Se lee la siguiente línea de código fuente.

No hay símbolos. La línea se ensambla, genera código objeto y se escribe en el archivo objeto.

Se incrementa LC.

LC = 38

Archivo Objeto

...

B8 B0 48 8E D8 A0 00 00 8A 1E 01 00 F6 E3 A3 02 00 83 3E 02 00 0A 7F E5 90 90 FE 06 1A 00
EB E5 B4 4C B0 00 CD 21

Ejemplo: Ensamblador de una pasada

```
int 21h  
end inicio
```

Se lee la siguiente línea de código fuente.

LC = 38

Ejemplo: Ensamblador de una pasada

```
int 21h  
end inicio
```

Se lee la siguiente línea de código fuente.

El ensamblador detecta la directiva 'end' que le indica que se trata de fin de archivo (EOF).

Ejemplo: Ensamblador de una pasada

```
int 21h  
end inicio
```

Se revisa la tabla de símbolos por símbolos indefinidos.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D
salir	Código	Word	0020h	D
w	Datos	?	0018h	U

Ejemplo: Ensamblador de una pasada

```
int 21h  
end inicio
```


Se revisa la tabla de símbolos por símbolos indefinidos.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D
salir	Código	Word	0020h	D
w	Datos	?	0018h	U

Ejemplo: Ensamblador de una pasada

Se encuentra el símbolo indefinido w en tabla de símbolos.

Por consiguiente, el ensamblador devuelve “Error de símbolo indefinido” a la salida.



```
C:\>tasm eypc\ensamb
Turbo Assembler Version 4.1 Copyright (c) 1988, 1996 Borland International

Assembling file:  eypc\ensamb.ASM to  ensamb.OBJ
**Error** eypc\ensamb.ASM(20) Undefined symbol: W
*Warning* eypc\ensamb.ASM(20) Argument needs type override
Error messages:      1
Warning messages:    1
Passes:              1
Remaining memory:    467k

C:\>_
```


Diseño de un ensamblador

- Ensamblador de dos pasadas
- <https://drive.google.com/open?id=1Qo0Gza5yEEEEex7cpeOAKM9003DCOawT>

Ejemplo: Ensamblador de dos pasadas

Se utilizará el mismo código fuente del ejemplo de ensamblador de una pasada.

En este ejemplo práctico se omitirá el ensamblado de las directivas, así como la creación del archivo intermedio que se ocupa entre pasadas.

De la tabla de instrucciones sólo se utilizarán los códigos de operación para las instrucciones necesarias.

```
.data
x      db      2
y      db      1
z      dw      0

.code
inicio:
    mov ax,@data
    mov ds,ax
for:
    mov al,[x]
    mov bl,[y]
    mul bl
    mov [z],ax
    cmp [z],10
    jg salir
    inc [w]
    jmp for
salir:
    mov ah,4Ch
    mov al,0
    int 21h
end inicio
```


Instrucción	Código de operación	Longitud
mov ax,imm16	B8 ?? ??	3 bytes
mov ds,ax	8E D8	2 bytes
mov al,m8	A0 ?? ??	3 bytes
mov bl,m8	8A 1E ?? ??	4 bytes
mul bl	F6 E3	2 bytes
mov m16,ax	A3 ?? ??	3 bytes
cmp m16,imm8	83 3E ?? ?? ??	5 bytes
jg rel8	7F ??	2 bytes
jg rel16	0F 8F ?? ??	4 bytes
inc m8	FE 06 ?? ??	4 bytes
jmp rel8	EB ??	2 bytes
mov ah,imm8	B4 ??	2 bytes
mov al, imm8	B0 ??	2 bytes
int imm8	CD ??	2 bytes

De la tabla de instrucciones:

```

.data
x      db      2
y      db      1
z      dw      0

.code
inicio:
    mov ax,@data
    mov ds,ax
for:
    mov al,[x]
    mov bl,[y]
    mul bl
    mov [z],ax
    cmp [z],10
    jg salir
    inc [w]
    jmp for
salir:
    mov ah,4Ch
    mov al,0
    int 21h
    end inicio

```

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

En la primera pasada, el ensamblador se encarga de llenar la tabla de símbolos y generar un archivo intermedio de apoyo en la segunda pasada, para ello recorre el archivo línea por línea buscando **definiciones de símbolos**, ya sea de variables o de etiquetas.

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
1  .data
2  x      db      2
```

Se omite la revisión de las directivas y la generación del archivo intermedio. Aunque cabe mencionar que por cada línea de código fuente, incluyendo directivas y comentarios, se inserta información al respecto en el archivo intermedio.

Tabla de símbolos

Nombre	Segmento	Tamaño	Valor	Tipo

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```

.data
x      db      2
y      db      1

```

Lee la siguiente línea.

Nombre	Segmento	Tamaño	Valor	Tipo

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
.data
x      db      2
y      db      1
```

Lee la siguiente línea.

Es la definición de un símbolo. Se inserta el símbolo en tabla de símbolos y se inserta info en archivo intermedio. Su valor es la localidad de memoria relativa al segmento en el que se encuentra.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

data
x db 2
y db 1
z dw 0

Lee la siguiente línea.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
data
x      db      2
y      db      1
z      dw      0
```

Lee la siguiente línea.

Es la definición de un símbolo. Se inserta el símbolo en tabla de símbolos y se inserta info en archivo intermedio.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
x      db      0
y      db      1
z      dw      0
      .code
```

Lee la siguiente línea.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
x      db      0
y      db      1
z      dw      0
      .code
```

Lee la siguiente línea.

Es la definición de un símbolo. Se inserta el símbolo en tabla de símbolos y se inserta info en archivo intermedio.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
y      ud      1
z      dw      0
.code
inicio:
```

Lee la siguiente línea.

Detecta directiva .code. Inicializa LC en 0

LC = 0

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```

L      uw      0
.code
inicio:
mov ax,@data

```

Lee la siguiente línea.

Es la definición de un símbolo, en este caso, una etiqueta. Se inserta el símbolo en tabla de símbolos con valor LC.

LC = 0

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
    .code
inicio:
    mov ax, @data
```

Lee la siguiente línea.

Es la definición de un símbolo, en este caso, una etiqueta. Se inserta el símbolo en tabla de símbolos con valor LC.

Escribe info en archivo intermedio y calcula la longitud de la instrucción para incrementar LC.

LC = 0

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
      .code  
inicio:  
      mov ax,@data  
      mov ds,ax  
      .end
```

Lee la siguiente línea.

LC = 0

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
      .code  
inicio:  
      mov ax,@data  
      mov ds,ax  
      .end
```

Lee la siguiente línea.

No hay definición de símbolo, así que sólo se escribe información en archivo intermedio y se calcula la longitud de la instrucción para incrementar LC.

LC = 3

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

inicio:

mov ax,@data

mov ds,ax

Lee la siguiente línea.

for:

- - -

LC = 3

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
inicio:
    mov ax,@data
    mov ds,ax
for:
    - - -
```

Lee la siguiente línea.

No hay definición de símbolo. Escribe en archivo intermedio e incrementar LC con la longitud de la instrucción.

LC = 5

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
mov ax, 0000h  
mov ds, ax  
for:  
mov al, [x]  
mov bx, [y]
```

Lee la siguiente línea.

LC = 5

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
mov ax, 0000h  
mov ds, ax  
for:  
mov al, [x]  
mov bx, [y]
```

Lee la siguiente línea.

Es la definición de una etiqueta, se inserta en tabla de símbolos.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D

LC = 5

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
mov ax, 0000h  
mov ds, ax  
for:  
mov al, [x]  
mov bl, [y]
```

Lee la siguiente línea.

Es la definición de una etiqueta, se inserta en tabla de símbolos.

Inserta info en archivo intermedio. No hay instrucción, no incrementa LC.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D

LC = 5

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
for:
    mov al, [x]
    mov bl, [y]
    mul bl
```

Lee la siguiente línea.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D

LC = 5

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
for:
    mov al, [x]
    mov bl, [y]
    mul bl
```

Lee la siguiente línea.

No hay definición de símbolo. Inserta info en archivo intermedio e incrementa LC.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D

LC = 8

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
mov al, [x]
```

```
mov bl, [y]
```

```
mul bl
```

Lee la siguiente línea.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D

LC = 8

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
mov al, [x]
```

```
mov bl, [y]
```

```
mul bl
```

Lee la siguiente línea.

No hay definición de símbolo. Inserta info en archivo intermedio e incrementa LC.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D

LC = 12

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
mov bl, [y]  
mul bl  
mov [z], ax
```

Lee la siguiente línea.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D

LC = 12

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
mov bl, [y]  
mul bl  
mov [z], ax
```

Lee la siguiente línea.

No hay definición de símbolo. Inserta info en archivo intermedio e incrementa LC.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D

LC = 14

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
mov ax, 10  
mul bl  
mov [z], ax  
cmp [z], 10  
je exit
```

Lee la siguiente línea.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D

LC = 14

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
mov ax, 10  
mul bl  
mov [z], ax  
cmp [z], 10  
je exit
```

Lee la siguiente línea.

No hay definición de símbolo. Inserta info en archivo intermedio e incrementa LC.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D

LC = 17

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
mov [z],ax  
cmp [z],10  
jg salir
```

Lee la siguiente línea.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D

LC = 17

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
mov [z],ax  
cmp [z],10  
jg salir
```

Lee la siguiente línea.

No hay definición de símbolo. Inserta info en archivo intermedio e incrementa LC.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D

LC = 22

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
mov [z], 00  
cmp [z], 10  
jg salir  
inc [w]  
inc [w]
```

Lee la siguiente línea.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D

LC = 22

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
mov [z], 00  
cmp [z], 10  
jg salir  
inc [w]  
...
```

Lee la siguiente línea.

No hay definición de símbolo. Inserta info en archivo intermedio e incrementa LC. En este caso, como la longitud de la instrucción puede variar, es posible que la info del archivo interm. considere todos los casos.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D

LC = 24

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
cmp [x], 10  
jg salir  
inc [w]  
jmp for  
salir:
```

Lee la siguiente línea.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D

LC = 24

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
cmp [x], 10  
jg salir  
inc [w]  
jmp for  
salir:
```

Lee la siguiente línea.

No hay definición de símbolo. Inserta info en archivo intermedio e incrementa LC.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D

LC = 28

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
    jmp for
salir:
```

Lee la siguiente línea.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D

LC = 28

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
inc [w]  
jmp for  
salir:  
mov ah, 4Ch
```

Lee la siguiente línea.

No hay definición de símbolo. Inserta info en archivo intermedio e incrementa LC.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D

LC = 30

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
inc [w]
jmp for
salir:
mov ah, 4Ch
```

Lee la siguiente línea.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D

LC = 30

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
inc [w]
jmp for
salir:
mov ah, 4Ch
```

Lee la siguiente línea.

Es la definición de una etiqueta, se inserta en tabla de símbolos.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D
salir	Código	Word	001Eh	D

LC = 30

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
inc [w]
jmp for
salir:
mov ah, 4Ch
```

Lee la siguiente línea.

Es la definición de una etiqueta, se inserta en tabla de símbolos.

Inserta info en archivo intermedio. No hay instrucción, no incrementa LC.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D
salir	Código	Word	001Eh	D

LC = 30

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
jmp for  
salir:
```

Lee la siguiente línea.

```
mov ah, 4Ch
```

```
mov al, 0
```

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D
salir	Código	Word	001Eh	D

LC = 30

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
jmp for  
salir:
```

```
mov ah, 4Ch
```

```
mov al, 0
```

Lee la siguiente línea.

No hay definición de símbolo. Inserta info en archivo intermedio e incrementa LC.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D
salir	Código	Word	001Eh	D

LC = 32

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

50011:

mov ah, 4Ch

mov al, 0

int 21h

Lee la siguiente línea.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D
salir	Código	Word	001Eh	D

LC = 32

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

50011:

mov ah, 4Ch

mov al, 0

int 21h

Lee la siguiente línea.

No hay definición de símbolo. Inserta info en archivo intermedio e incrementa LC.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D
salir	Código	Word	001Eh	D

LC = 34

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
mov al, 0  
int 21h  
end inicio
```

Lee la siguiente línea.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D
salir	Código	Word	001Eh	D

LC = 34

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
mov al, 0  
int 21h  
end inicio
```

Lee la siguiente línea.

No hay definición de símbolo. Inserta info en archivo intermedio e incrementa LC.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D
salir	Código	Word	001Eh	D

LC = 36

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
int 21h  
end inicio
```

Lee la siguiente línea.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D
salir	Código	Word	001Eh	D

LC = 36

Ejemplo: Ensamblador de dos pasadas (Primera pasada)

```
int 21h  
end inicio
```

Lee la siguiente línea.

Es fin de archivo (EOF). Se cierra el archivo intermedio y hace la segunda pasada.

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D
salir	Código	Word	001Eh	D

LC = 36

Ejemplo: Ensamblador de dos pasadas

(Segunda pasada)

En la segunda pasada, el ensamblador lee el contenido del archivo intermedio, generado en la primera pasada.

El **archivo intermedio** contiene información de cada una de las líneas de código fuente, incluyendo directivas, comentarios y líneas inválidas. También **puede contener información del LC** en cada una de las líneas del código fuente y **apuntadores a las tablas de directivas e instrucciones** para más rápido acceso a ellas. Por último, el archivo intermedio **podrá contener una copia del código fuente** que servirá para ensamblar la línea de código directamente.

Ejemplo: Ensamblador de dos pasadas (Segunda pasada)

Conociendo la tabla de símbolos, el ensamblador puede resolver cualquier referencia directamente, siempre y cuando el símbolo exista en la tabla

Nombre	Segmento	Tamaño	Valor	Tipo
x	Datos	Byte	0000h	D
y	Datos	Byte	0001h	D
z	Datos	Word	0002h	D
inicio	Código	Word	0000h	D
for	Código	Word	0005h	D
salir	Código	Word	001Eh	D

Ejemplo: Ensamblador de dos pasadas

(Segunda pasada)

	fuelle	.obj
LC = 0d	inicio:	
LC = 0d	mov ax,@data	B8 B0 48
LC = 3d	mov ds,ax	8E D8
LC = 5d	for:	
LC = 5d	mov al,[x]	A0 00 00
LC = 8d	mov bl,[y]	8A 1E 01 00
LC = 12d	mul bl	F6 E3
LC = 14d	mov [z],ax	A3 02 00
LC = 17d	cmp [z],10	83 3E 02 00 0A
LC = 22d	jg salir	7F 06
LC = 24d	inc [w]	FE 06 18 00
LC = 28d	jmp for	EB E7
LC = 30d	salir:	
LC = 30d	mov ah,4Ch	B4 4C
LC = 32d	mov al,0	B0 00
LC = 34d	int 21h	CD 21
	end inicio	

En el ejemplo, todas las referencias se resuelven con los valores de la tabla de símbolos, excepto [w], el cual es un símbolo que no existe en la tabla de símbolos. Por lo tanto, el ensamblador deberá devolver el error de símbolo indefinido.

Cabe mencionar que a diferencia del ensamblador de una pasada, en el de dos pasadas las referencias futuras están definidas. En consecuencia, la instrucción 'jg salir' no genera instrucciones NOP al final.

3.5: Diseño de un macroensamblador

- Un **macroensamblador** es aquel ensamblador que permite el uso y la definición de macroinstrucciones (macros) dentro del código fuente en lenguaje ensamblador.
- Una **macroinstrucción** es un grupo de instrucciones del procesador que se comprimen en una forma más simple y que aparecen como una instrucción en código fuente.

Diseño de un macroensamblador

- El macroensamblador reconoce la definición de una macroinstrucción a través de una directiva ejecutada por el ensamblador.
- En el caso de Turbo Ensamblador, la directiva que especifica el inicio de una macroinstrucción es **macro**. La directiva que especifica el fin de la macroinstrucción es **endm**.
- Ejemplo:

```
clear macro  
    mov ax,0003h  
    int 10h  
endm
```


Diseño de un macroensamblador

- Una macro sólo se puede definir una vez. El ensamblador lee la definición del código fuente y la guarda en una tabla especial llamada **Tabla de Definición de Macros** (TDM).
- Al insertar la definición de una macro en la TDM, el ensamblador no revisa errores, sólo guarda la definición.

Nombre	Contenido	Argumentos	Etiquetas locales
clear	mov ax,0003h int 10h	-	-

Diseño de un macroensamblador

- Cuando el ensamblador detecta el uso de una macro en el código fuente, genera una copia de su contenido y la inserta en el programa en lugar de la macro. A este proceso se le denomina **macroexpansión**.
- Una macroexpansión empieza cuando el ensamblador detecta una línea que no es instrucción, ni directiva, ni palabra reservada; se busca la línea en la TDM y, si está en esta tabla, cambia de modo de ensamblado normal a modo de macroexpansión.

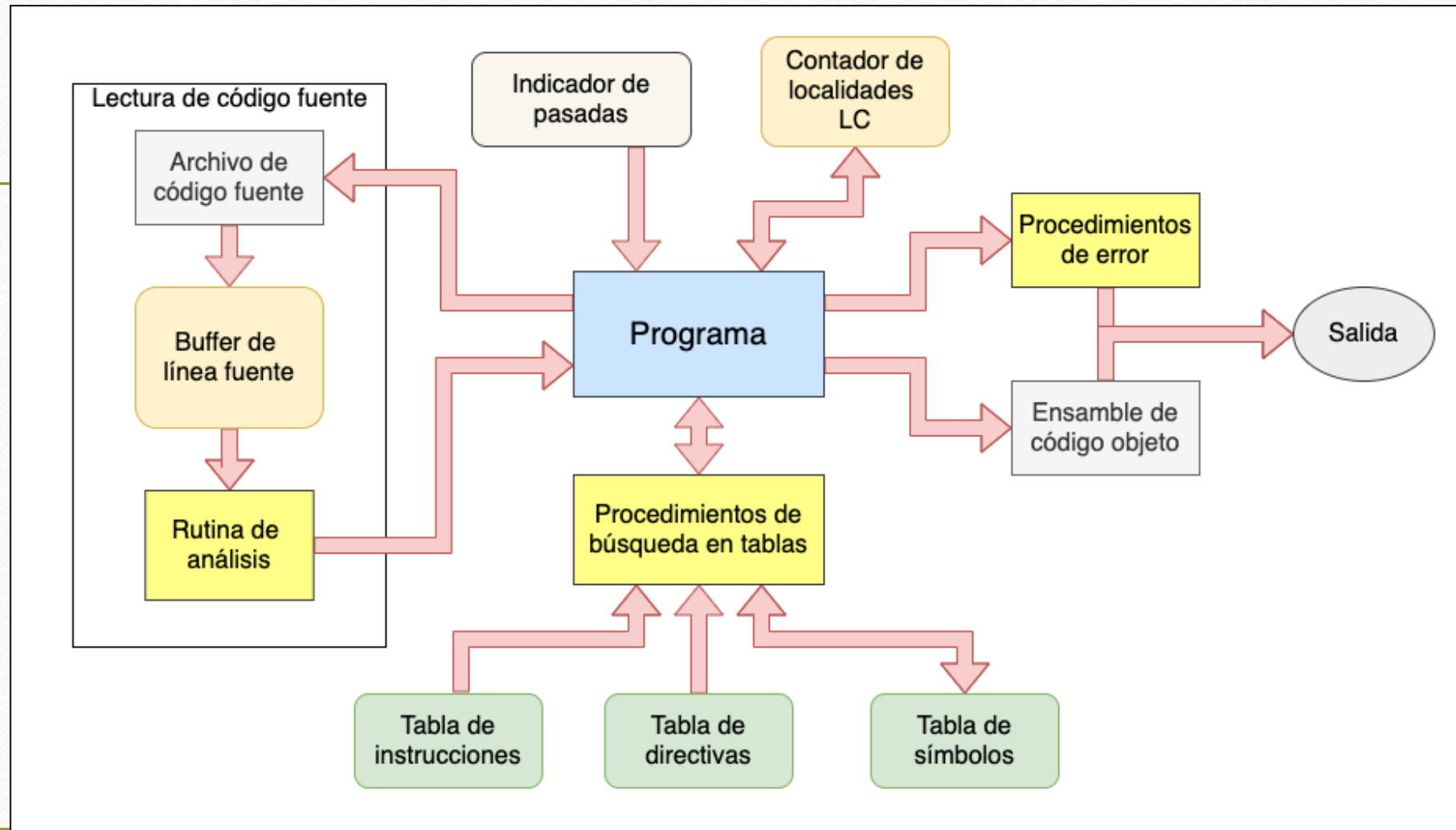
Diseño de un macroensamblador

```
inicio:
    mov ax,@data
    mov ds,ax
    clear
salir:
    mov ax,4C00h
    int 21h
    end inicio
```

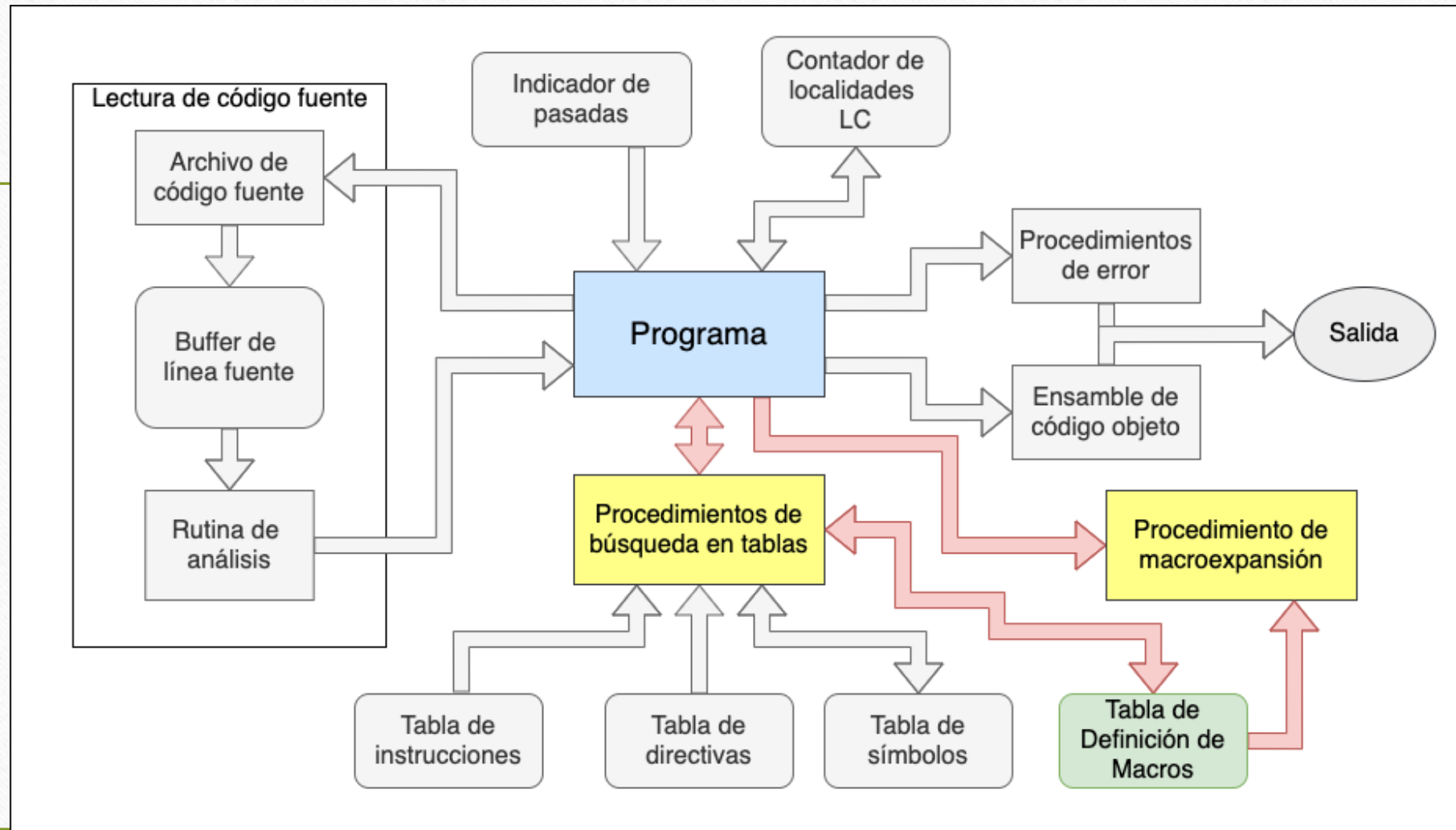
Macroexpansión

```
inicio:
    mov ax,@data
    mov ds,ax
    mov ax,0003h
    int 10h
salir:
    mov ax,4C00h
    int 21h
    end inicio
```

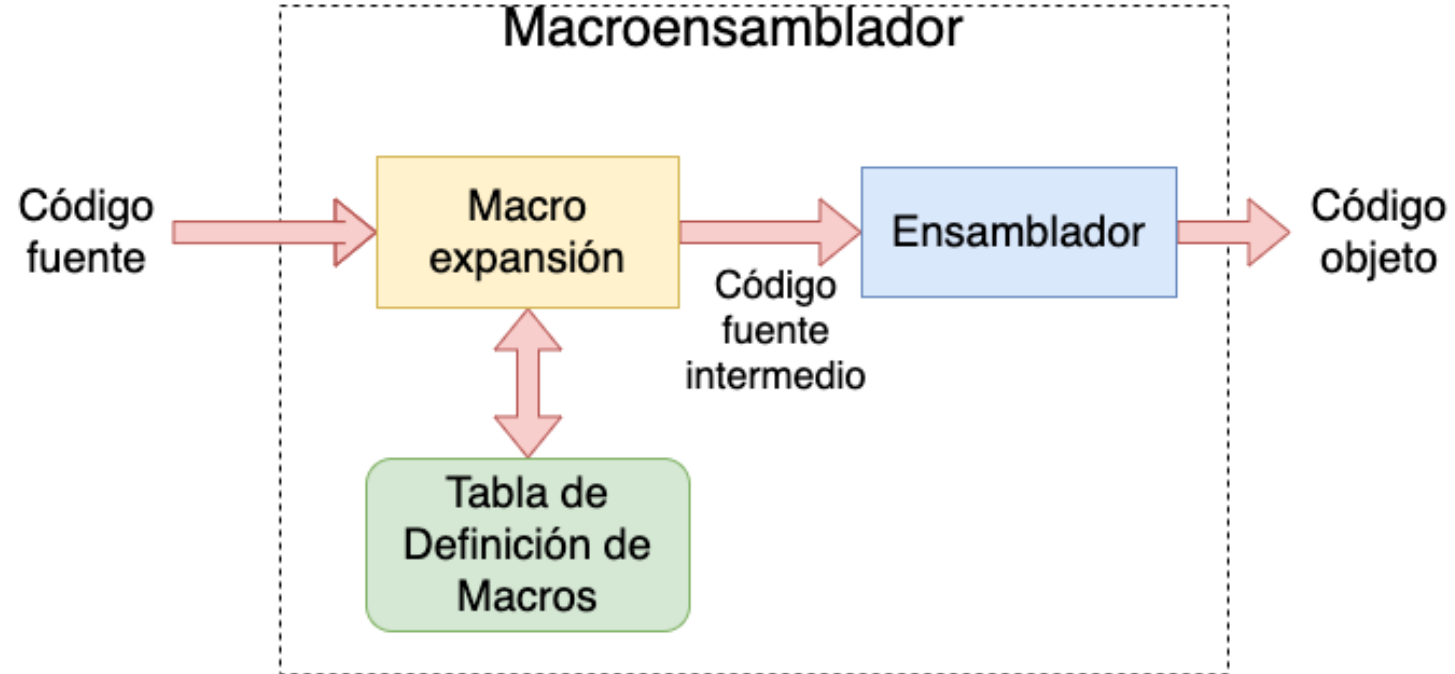
Componentes de un programa ensamblador



Componentes de un macroensamblador



Otra estructura de un macroensamblador



Diseño de un macroensamblador

- Turbo Assembler (TASM) se considera un macroensamblador debido a que permite el uso de macroninstrucciones en el código fuente.
- TASM considera algunos aspectos básicos en una macro. Estos aspectos son:
 - Uso de argumentos
 - Etiquetas locales
 - Macros anidadas
 - Expansión condicional

Diseño de un macroensamblador

- **Argumentos**

Una macro puede tener argumentos los cuales se deberán resolver por el ensamblador. En TASM, los argumentos se sustituyen directamente en donde se utilizan dentro de la macro.

Diseño de un macroensamblador

- **Argumentos (ejemplo)**

Se tiene la siguiente macro con 2 argumentos, separados por coma: renglon y columna.

```
mueve_cursor macro renglon,columna
    mov dh,renglon
    mov dl,columna
    mov bx,0
    mov ax,0200h
    int 10h
endm
```

Diseño de un macroensamblador

- **Argumentos (ejemplo)**

Cuando se usa la macro con argumentos, éstos deben pasarse después de la macro, y separados por coma.

```
mueve_cursor 10,0
```


Diseño de un macroensamblador

- Argumentos (ejemplo)

```
inicio:
    mov ax,@data
    mov ds,ax
    mueve_cursor 10,0
salir:
    mov ax,4C00h
    int 21h
    end inicio
```

Macroexpansión

```
inicio:
    mov ax,@data
    mov ds,ax
    mov dh,10
    mov dl,0
    mov bx,0
    mov ax,0200h
    int 10h
salir:
    mov ax,4C00h
    int 21h
    end inicio
```

Diseño de un macroensamblador

- Etiquetas locales

Las macros pueden tener etiquetas utilizadas dentro de sí misma. Para TASM, se pueden definir con la directiva **local** al inicio de una macro.

```
loop_cadena macro repeticiones,cadena
    local loop1
    mov cx,repeticiones
loop1:
    mov ah,09h
    lea dx,cadena
    int 21h
    loop loop1
endm
```


Diseño de un macroensamblador

- **Etiquetas locales**

Para resolver el uso de etiquetas dentro de una macro y que no se genere la misma etiqueta cuando se vuelve a utilizar la macro, el ensamblador puede concatenar un identificador al nombre de la etiqueta que permita diferenciarla entre usos.

Generalmente, se usan identificadores numéricos.

Diseño de un macroensamblador

- Etiquetas locales

```
inicio:
    mov ax,@data
    mov ds,ax
    loop_cadena 5,[hola]
    loop_cadena 10,[adios]
salir:
    mov ax,4C00h
    int 21h
    end inicio
```



```
inicio:
    mov ax,@data
    mov ds,ax
    mov cx,5
loop1_0001:
    mov ah,09h
    lea dx,[hola]
    int 21h
    loop loop1_0001
    mov cx,10
loop1_0002:
    mov ah,09h
    lea dx,[adios]
    int 21h
    loop loop1_0002
salir:
    mov ax,4C00h
    int 21h
    end inicio
```

loop_cadena 5, [hola]

loop_cadena 10, [adios]

Diseño de un macroensamblador

- **Macros anidadas**

Una macro puede estar anidada dentro de otra.

```
imprime_cadena macro cadena
    mov ah,09h
    lea dx,cadena
    int 21h
endm
loop_cadena macro repeticiones,cadena
    local loop1
    mov cx,repeticiones
loop1:
    imprime_cadena cadena
    loop loop1
endm
```

Diseño de un macroensamblador

- **Macros anidadas**

El ensamblador no debe tener ningún problema al resolver macros anidadas, ya que el proceso de macroexpansión puede ser **secuencial**.

Diseño de un macroensamblador

- Macros anidadas

```
inicio:
    mov ax,@data
    mov ds,ax
    loop_cadena 5,[hola]
salir:
    mov ax,4C00h
    int 21h
end inicio
```



```
inicio:
    mov ax,@data
    mov ds,ax
    mov cx,5
loop1_0001:
    imprime_cadena [hola]
    loop loop1_0001
salir:
    mov ax,4C00h
    int 21h
end inicio
```



```
inicio:
    mov ax,@data
    mov ds,ax
    mov cx,5
loop1_0001:
    mov ah,09h
    lea dx,[hola]
    int 21h
    loop loop1_0001
salir:
    mov ax,4C00h
    int 21h
end inicio
```

Diseño de un macroensamblador

- **Macros anidadas**

Una macro se puede usar dentro de sí misma (recursividad). Sin embargo, esto puede provocar una macroexpansión de código “infinita” y desbordar la memoria de la computadora. Para resolver este problema, es necesario establecer un punto de quiebre dentro de la macro y con ayuda de la **expansión condicional**.

Diseño de un macroensamblador

- **Expansión condicional**

Una expansión condicional se presenta **dentro de una macro** y genera una **macroexpansión** solo **cuando se cumple con una condición específica**.

La expansión condicional **se puede implementar utilizando directivas propias del ensamblador**. En TASM, existen las directivas condicionales: IF, IFE, IF1, IF2, IFDEF, IFNDEF, IFB, IFNB, IFIDN, IFDIF.

También existen las directivas ELSE (directiva opcional cuando se usa una directiva condicional) y ENDIF (obligatoria para finalizar un bloque condicional).

Diseño de un macroensamblador

- **Expansión condicional**

Cuando el ensamblador detecta el uso de una directiva condicional, se valida una condición. Si la condición se cumple, el ensamblador procede a ensamblar el código fuente que se encuentre dentro del bloque condicional.

Diseño de un macroensamblador

- Expansión condicional

```
cadena_recursivo macro iterador,cadena
    ;Si 'iterador' es diferente de 0
    IF iterador
        imprime_cadena cadena
        cadena_recursivo iterador-1,cadena
    ;Si 'iterador' es 0
    ELSE
        EXITM
    ENDIF
endm
```

Diseño de un macroensamblador

- Expansión condicional

```
inicio:
    mov ax,@data
    mov ds,ax
    cadena_recursivo 4,[hola]
salir:
    mov ax,4C00h
    int 21h
    end inicio
```



```
inicio:
    mov ax,@data
    mov ds,ax
    imprime_cadena [hola]
    cadena_recursivo 3,[hola]
salir:
    mov ax,4C00h
    int 21h
    end inicio
```



...

Diseño de un macroensamblador

- Expansión condicional

...



```
inicio:
    mov ax,@data
    mov ds,ax
    imprime_cadena [hola]
    imprime_cadena [hola]
    cadena_recursivo 2,[hola]
salir:
    mov ax,4C00h
    int 21h
end inicio
```



```
inicio:
    mov ax,@data
    mov ds,ax
    imprime_cadena [hola]
    imprime_cadena [hola]
    imprime_cadena [hola]
    cadena_recursivo 1,[hola]
salir:
    mov ax,4C00h
    int 21h
end inicio
```




...

Diseño de un macroensamblador

- Expansión condicional

...



```
inicio:
    mov ax,@data
    mov ds,ax
    imprime_cadena [hola]
    imprime_cadena [hola]
    imprime_cadena [hola]
    imprime_cadena [hola]
    cadena_recursivo 0,[hola]
salir:
    mov ax,4C00h
    int 21h
    end inicio
```



```
inicio:
    mov ax,@data
    mov ds,ax
    imprime_cadena [hola]
    imprime_cadena [hola]
    imprime_cadena [hola]
    imprime_cadena [hola]
salir:
    mov ax,4C00h
    int 21h
    end inicio
```