



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Estructura y Programación de Computadoras

Grupo: 02 - Semestre: 2021-1

Tarea 6: Ensamblador de una pasada

FECHA DE ENTREGA: 20/01/2021

Alumno:
Téllez González Jorge Luis



Índice

1. Planteamiento del problema	2
1.1. Consideraciones	3
2. Procedimiento	3

1. Planteamiento del problema

Se requiere realizar el ensamblado de una pasada, de acuerdo con lo visto en clase, del siguiente bloque de código en lenguaje ensamblador.

```
.data
tecla_enter equ 0Dh
N_mayus equ 'N'
N_minus equ 'n'
caracter db 0
.code
inicio:
mov ax,@data
mov ds,ax
lee_teclado:
mov ah,08h
int 21h
mov [caracter],al
cmp [caracter],tecla_enter
je salir
cmp [caracter],N_mayus
je salir
cmp [caracter],N_minus
je salir
jmp lee_teclado
salir:
mov ah,4Ch
mov al,0
int 21h
end inicio
```

Figura 1: Código a ensamblar.

Las tablas de instrucciones y símbolos utilizadas serán las siguientes:

Tabla de Instrucciones

Instrucción	Código de operación	Longitud
mov ax,imm16	B8 ?? ??	3 bytes
mov ds,ax	8E D8	2 bytes
mov ah,imm8	B4 ??	2 bytes
int imm8	CD ??	2 bytes
mov m8,al	A2 ?? ??	3 bytes
cmp m8,imm8	80 3E ?? ?? ??	5 bytes
je rel8	74 ??	2 bytes
je rel16	0F 84 ?? ??	4 bytes
jmp imm8	EB ??	2 bytes
mov al,imm8	B0 ??	2 bytes

Figura 2: Tabla de instrucciones.

Tabla de símbolos

Nombre	Segmento	Tamaño	Valor	Tipo
tecla_enter	(Constante)	Byte	0Dh	D
N_mayus	(Constante)	Byte	'N'	D
N_minus	(Constante)	Byte	'n'	D
caracter	Datos	Byte	0000h	D

Figura 3: Tabla de símbolos.

1.1. Consideraciones

- Se asume que `@data=48B0h`.
- Las constantes en el programa (definidas con la directiva **equ** se consideran símbolos, aunque éstos no ocupan espacio en memoria (no se les asigna una localidad de memoria). El ensamblador resuelve el uso de constantes en tiempo de ensamblado sustituyéndolas por su valor en donde se usan.
- Las referencias futuras se resuelven cuando se encuentra la definición del símbolo al que hacen referencia. En el primer uso, se utiliza la misma localidad de la instrucción, posteriormente, se utiliza el valor que se tenga dentro de la tabla de símbolos para el símbolo referenciado. En cada uso, el valor del símbolo se actualiza con el valor de LC actual.

2. Procedimiento

Una vez que se genera la tabla de símbolos tras analizar el segmento **.data**, se continúa al segmento de código (**.code**), donde el valor de $LC = 0$ y se lee la primera línea de código, la cuál se trata de una etiqueta denominada **inicio**:

```
.code                ; LC=0
inicio:            ; LC=0
    mov ax, @data
    mov ds, ax
```

Figura 4: Declaración de etiqueta.

Esta etiqueta es encontrada por primera vez, por tanto, esta etiqueta será agregada a la tabla de símbolos, considerando que su valor será el mismo al de LC, que es igual a 0 o `0000h`. La tabla de símbolos resulta de la siguiente forma:

Nombre	Segmento	Tamaño	Valor	Tipo
tecla_enter	(Constante)	Byte	0Dh	D
N_mayus	(Constante)	Byte	'N'	D
N_minus	(Constante)	Byte	'n'	D
carácter	Datos	Byte	0000h	D
inicio	Código	Word	0000h	D

Figura 5: Tabla de símbolos con la nueva etiqueta.

La línea que contiene a **inicio** es ensamblada, pero dado que no contiene ninguna instrucción, se procede a leer la siguiente línea ya que esta no genera código objeto; quedando el valor de *LC* inalterado y referenciando a la línea de código siguiente. Continuamos con la siguiente línea de código que se observa a continuación:

```

inicio:
    mov ax, @data ;<--- Línea actual. ;LC=0
    mov ds,ax
  
```

Figura 6: Lectura de línea.

Asumiendo que *@data* = 48B0h, se ensambla la línea considerando el código de operación de la tabla de instrucciones y completando el valor, lo cual resulta como:

Instrucción	Código de Operación	Longitud
mov ax, imm16	B8 B0 48	3 bytes

Figura 7: Valor completado.

Esta instrucción generará el código objeto **B8 B0 48**. *LC* será incrementado de acuerdo a la longitud de la instrucción. Entonces $LC = 0 + 3 = 3$. Continuamos con la siguiente línea escaneada:

```

inicio:
    mov ax, @data
    mov ds,ax ;<--- Línea actual. ;LC=3
  
```

Figura 8: Siguiendo línea.

Considerando que no usa símbolos, se procede a ensamblar la línea, utilizando su código de operación correspondiente y generando el código objeto correspondiente, el cual puede observarse a continuación.

<code>mov ds, ax</code>	<code>8E D8</code>	<code>2 bytes</code>
-------------------------	--------------------	----------------------

Figura 9: Valor completado.

El código objeto resulta: **B8 B0 48 8E D8**. Tras esto y siguiendo el diagrama de flujo del ensamblador, LC se incrementa en dos unidades. Por tanto, $LC = 5$. A continuación, tenemos la siguiente línea:

```

mov ax, @data
mov ds, ax
lee_teclado:    ;<--- Línea actual. ;LC=5

```

Figura 10: Siguiete línea.

Atendiendo al diagrama de flujo, se revisa si la etiqueta ya existe en la tabla de símbolos. Debido a que se trata de una primer declaración, se procede a insertarla en la tabla de símbolos asignándole el valor de $LC = 0005h$. Dado que se trata de una instrucción vacía, esta no genera código objeto y el valor de LC no se ve modificado.

Nombre	Segmento	Tamaño	Valor	Tipo
tecla_enter	(Constante)	Byte	0Dh	D
N_mayus	(Constante)	Byte	'N'	D
N_minus	(Constante)	Byte	'n'	D
carácter	Datos	Byte	0000h	D
inicio	Código	Word	0000h	D
lee_teclado	Código	Word	0005h	D

Figura 11: Tabla de símbolos con la nueva etiqueta.

Se procede con la siguiente línea que puede verse a continuación:

```

lee_teclado:
    mov ah, 08h    ;<--- Línea actual. ;LC=5
    int 21h

```

Figura 12: Siguiete línea.

En esta línea no se está ocupando un símbolo, así que se ensambla y se genera el código objeto de la siguiente forma: **B8 B0 48 8E D8 B4 08**. LC se incrementa en 2 unidades, lo que resulta $LC = 7$.

Continuando a la siguiente línea leída, se observa que no tiene símbolos, por lo que se ensambla, se genera el código objeto y se escribe sobre el archivo objeto como: **B8 B0 48 8E D8 B4 08 CD 21**.

```
lee_teclado:
    mov ah, 08h
    int 21h ;<--- Línea actual. ;LC=7
```

Figura 13: *Siguiente línea.*

Finalmente, se incrementa LC en dos unidades, por tanto, $LC = 9$. Ahora, se procede a leer la siguiente línea de código.

```
int 21h
mov [caracter], al ;<--- Línea actual. ;LC=9
cmp [caracter], tecla_enter
```

Figura 14: *Siguiente línea.*

Esta línea de código utiliza el símbolo *caracter*, el cual si se encuentra definidos en nuestra tabla de símbolos. Como no hay conflicto entre el tamaño de *caracter* y el registro *al*, se ensambla la línea utilizando la tabla de instrucciones. El código objeto resulta: **B8 B0 48 8E D8 B4 08 CD 21 A2 00 00**.

mov m8, al	A2 00 00	3 bytes
------------	----------	---------

Figura 15: *Valor completado.*

LC se incrementa en 3 unidades, entonces $LC = 12$. Se procede a escanear la siguiente línea:

```
mov [caracter], al
cmp [caracter], tecla_enter ;<--- Línea actual. ;LC=12
je salir
```

Figura 16: *Siguiente línea.*

Esta línea utiliza símbolos, así que se procede a revisar si estos existen. Como se encuentran presentes en la tabla de símbolos, se ensambla la línea y se inserta el código objeto al archivo **obj**, lo cual resulta como: **B8 B0 48 8E D8 B4 08 CD 21 A2 00 00 80 3E 00 00 0D**, recordando que *caracter*=0000h y *tecla_enter*=0Dh.

cmp m8, imm8	80 3E 00 00 0D	5 bytes
--------------	----------------	---------

Figura 17: *Valor completado.*

Con lo anterior, $LC = 17$ incrementándose 5 unidades. Tras esto, se escanea la siguiente línea que puede verse a continuación:

```
cmp [caracter], tecla_enter
je salir           ;<--- Línea actual. ;LC=17
cmp [caracter], N_mayus
```

Figura 18: *Siguiente línea.*

En este caso se ha encontrado un símbolo (la etiqueta **salir**) que no existe en la tabla de símbolos, por lo que se procederá a insertar su nombre, el valor de LC actual como referencia a sí misma y con el tipo **U**. Como $LC = 17$, se le insertará el valor **0011h** en la tabla de símbolos.

Nombre	Segmento	Tamaño	Valor	Tipo
tecla_enter	(Constante)	Byte	0Dh	D
N_mayus	(Constante)	Byte	'N'	D
N_minus	(Constante)	Byte	'n'	D
carácter	Datos	Byte	0000h	D
inicio	Código	Word	0000h	D
lee_teclado	Código	Word	0005h	D
salir	Código	Word	0011h	U

Figura 19: *Tabla de símbolos actualizada.*

Ahora, procedemos a ensamblar la línea de código. Como se mencionó en clase, el ensamblador no conoce en este punto la longitud del salto que se está realizando, y por tanto, debe de considerar el peor caso posible en caso de un salto considerable que se pudiese presentar.

Como el peor caso que puede presentarse es el salto de tipo **near**, se genera el código objeto en el caso cuando el salto **je** sea de tipo **near**. Por tanto, el valor que se genera es el siguiente:

je rel16	0F 84 11 00	4 bytes
-----------------	--------------------	----------------

Figura 20: *Valor completado.*

Posteriormente, se inserta el código objeto en el archivo, y se incrementa $LC = 21$. El código objeto resulta: **B8 B0 48 8E D8 B4 08 CD 21 A2 00 00 80 3E 00 00 0D 0F 84 11 00**. La instrucción quedará pendiente de resolverse.

Ahora, se continúa con la siguiente línea:

```
je salir
cmp [caracter], N_mayus ;<--- Línea actual. ;LC=21
je salir
```

Figura 21: *Siguiente línea.*

Esta línea utiliza dos símbolos. Dado que ambos se encuentran en la tabla de símbolos, se ensambla la línea y se inserta el código objeto de la siguiente forma: **B8 B0 48 8E D8 B4 08 CD 21 A2 00 00 80 3E 00 00 0D 0F 84 11 00 80 3E 00 00 4E**, considerando el valor hexadecimal de 'N' como **4E** de acuerdo a la tabla ASCII proporcionada al inicio del curso.

<code>cmp m8, imm8</code>	80 3E 00 00 4E	5 bytes
---------------------------	-----------------------	---------

Figura 22: *Valor completado.*

Finalmente, se incrementa el valor de LC en 5 unidades, lo que resulta en $LC = 26$. Ahora, continuamos con la siguiente línea de código.

```
cmp [caracter], N_mayus
je salir ;<--- Línea actual. ;LC=26
cmp [caracter], N_minus
```

Figura 23: *Siguiente línea.*

Similar al caso anterior, estamos ante el uso de una etiqueta sin haberla declarado previamente. Para realizar el ensamblado y dejar esa sección pendiente, esta segunda línea de código ensamblada hará referencia hacia la primer línea de código en donde se utilizó la etiqueta **salir**. Entonces, se genera el siguiente código objeto: **B8 B0 48 8E D8 B4 08 CD 21 A2 00 00 80 3E 00 00 0D 0F 84 11 00 80 3E 00 00 4E 0F 84 1A 00**. Pero ahora, el valor que se encuentra en la tabla hará referencia al valor LC en donde se encontró esta línea de código.

salir	Código	Word	001Ah	U
--------------	--------	------	--------------	---

Figura 24: *La etiqueta **salir** ahora apunta a la segunda instrucción que la utiliza.*

Nótese una vez más que esta línea de código hace referencia a la primera línea de código que contiene **salir**, mientras que la primer línea hace referencia a sí misma. Hecho esto, se incrementa LC en cuatro unidades. Por tanto, $LC = 30$.

En la siguiente línea se encuentra el uso de dos símbolos declarados de forma similar a un caso anterior. Entonces, se ensambla la línea y se inserta el código objeto al archivo **.obj**, lo que resulta como: **B8 B0 48**

```
je salir
cmp [caracter], N_minus ;<--- Línea actual. ;LC=30
je salir
```

Figura 25: *Siguiente línea.*

8E D8 B4 08 CD 21 A2 00 00 80 3E 00 00 0D 0F 84 11 00 80 3E 00 00 4E 0F 84 1A 00 80 3E 00 00 6E, considerando **6E** como el valor hexadecimal de 'n' de acuerdo a la tabla ASCII. Finalmente, se incrementa una vez más *LC* en 5 unidades. Por tanto, $LC = 35$. Se procede a leer la siguiente línea:

```
cmp [caracter], N_minus
je salir ;<--- Línea actual. ;LC=35
jmp lee_teclado
```

Figura 26: *Siguiente línea.*

Esta línea de código hace uso por tercera vez de la etiqueta **salir**, por lo que esta hará referencia a la segunda línea de código que hace uso de la etiqueta y sobrescribirá el valor de la etiqueta en la tabla de símbolos.

salir	Código	Word	0023h	U
-------	--------	------	-------	---

Figura 27: *Valor actualizado.*

Se ensambla la línea y se inserta el código objeto al archivo **.obj**, lo que resulta como: **B8 B0 48 8E D8 B4 08 CD 21 A2 00 00 80 3E 00 00 0D 0F 84 11 00 80 3E 00 00 4E 0F 84 1A 00 80 3E 00 00 6E 0F 84 23 00**. Se incrementa el valor de *LC* en cuatro unidades nuevamente, entonces, $LC = 39$. Se procede a leer la siguiente línea:

```
jmp lee_teclado ;<--- Línea actual. ;LC=39
salir:
mov ah, 4Ch
```

Figura 28: *Valor actualizado.*

Se detecta el uso de un símbolo ya definido en la tabla de símbolos. Para ensamblar la línea, el ensamblador debe de calcular el desplazamiento entre la línea actual y en la que se definió el símbolo. Entonces:

$$[T.S] - (LC + long) = 0005h - (39d + 2) = -5d - 41d = -36d = DCh$$

Este valor calculado es parte del código objeto, y se escribe en **.obj** como: **B8 B0 48 8E D8 B4 08 CD 21 A2 00 00 80 3E 00 00 0D 0F 84 11 00 80 3E 00 00 4E 0F 84 1A 00 80 3E 00 00 6E 0F 84 23 00 EB DC**. Incrementando en dos unidades, $LC = 41$. Se continúa leyendo la siguiente línea:

```
salir:          ;<--- Línea actual. ;LC=41;
    mov ah, 4Ch
    mov al, 0
```

Figura 29: Valor actualizado.

Se detecta una definición de etiqueta, la cuál ya está definida en la tabla de símbolos como tipo **U**. El valor de la tabla de símbolos apunta a la última instrucción que utilizó esa instrucción.

La línea que se encuentra en la localidad 17d (0F 84 11 00) apunta a sí misma. Para resolver la referencia, se calcula:

$$41d - 17d - long = 24d - long = 18h - long = 18h - 2 = 16h$$

Para escribir, se utiliza el código de operación 74 16 de la siguiente forma:

je rel8	74 16	2 bytes
---------	-------	---------

Figura 30: Valor actualizado.

Para evitar tener que recalcular las líneas anteriores de código, las localidades vacías son *llenadas* utilizando instrucciones NOP, las cuales tienen un código de operación 90h. De esta forma, el código objeto resulta como: **B8 B0 48 8E D8 B4 08 CD 21 A2 00 00 80 3E 00 00 0D 74 16 90 90 80 3E 00 00 4E 0F 84 1A 00 80 3E 00 00 6E 0F 84 23 00 EB DC**

Ahora, con la segunda instrucciones que usa *salir* en la localidad 26h (0F 84 1A 00), se calcula la referencia:

$$41d - 26d - long = 15d - long = 0Fh - long = 0Fh - 2 = 0Dh$$

Utilizando el mismo código de operación y las instrucciones NOP para rellenar los espacios vacíos, el código objeto resulta: **B8 B0 48 8E D8 B4 08 CD 21 A2 00 00 80 3E 00 00 0D 74 16 90 90 80 3E 00 00 4E 74 0D 90 90 80 3E 00 00 6E 0F 84 23 00 EB DC**.

Finalmente, la última instrucción que utiliza la etiqueta *salir* se encuentra ubicada en la localidad 35d (0F 84 23 00). Calculando la referencia:

$$41d - 35d - long = 6d - long = 6h - long = 6h - 2 = 4h$$

Usando el mismo código de operación y las instrucciones NOP para rellenar los espacios vacíos, el código objeto resulta: **B8 B0 48 8E D8 B4 08 CD 21 A2 00 00 80 3E 00 00 0D 74 16 90 90 80 3E 00 00 4E 74 0D 90 90 80 3E 00 00 6E 74 04 90 90 EB DC**.

Después del cálculo de referencias en las instrucciones pendientes, se escribe en la tabla de símbolos el valor de LC en el símbolo resuelto y se cambia a tipo **D**. Dado que se resolvió en $LC = 41d$, se escribe el valor hexadecimal **29h**.

Nombre	Segmento	Tamaño	Valor	Tipo
tecla_enter	(Constante)	Byte	0Dh	D
N_mayus	(Constante)	Byte	'N'	D
N_minus	(Constante)	Byte	'n'	D
carácter	Datos	Byte	0000h	D
inicio	Código	Word	0000h	D
lee_teclado	Código	Word	0005h	D
salir	Código	Word	0029h	D

Figura 31: *Tabla de símbolos actualizada.*

Esta línea es ensamblada, pero no genera código objeto ni incrementa LC debido a que carece de instrucción alguna. Procediendo a leer la siguiente línea de código:

```
salir:
    mov ah, 4Ch ;<--- Línea actual. ;LC=41;
    mov al, 0
```

Figura 32: *Siguiente línea.*

La línea carece de símbolos, así que se ensambla, se genera el código objeto y se escribe sobre el archivo **.obj**, incrementando $LC = 43$. El código objeto resulta: **B8 B0 48 8E D8 B4 08 CD 21 A2 00 00 80 3E 00 00 0D 74 16 90 90 80 3E 00 00 4E 74 0D 90 90 80 3E 00 00 6E 74 04 90 90 EB DC B4 4C**. Se continúa a la siguiente línea:

```
mov ah, 4Ch
mov al, 0 ;<--- Línea actual. ;LC=43;
int 21h
```

Figura 33: *Siguiente línea.*

La línea carece de símbolos, así que se ensambla, se genera el código objeto y se escribe sobre el archivo **.obj**, incrementando $LC = 45$. El código objeto queda de la siguiente forma: **B8 B0 48 8E D8 B4 08 CD 21 A2 00 00 80 3E 00 00 0D 74 16 90 90 80 3E 00 00 4E 74 0D 90 90 80 3E 00 00 6E 74 04 90 90 EB DC B4 4C B0 00**. Se procede a leer la siguiente línea:

```
mov al, 0
int 21h      ;<--- Línea actual. ;LC=45;
end inicio
```

Figura 34: *Siguiente línea.*

Esta línea de código no contiene símbolos, así que se ensambla, genera código objeto, y se escribe sobre el archivo objeto, incrementando LC en dos unidades. Entonces, $LC = 47$ y el código objeto resulta: **B8 B0 48 8E D8 B4 08 CD 21 A2 00 00 80 3E 00 00 0D 74 16 90 90 80 3E 00 00 4E 74 0D 90 90 80 3E 00 00 6E 74 04 90 90 EB DC B4 4C B0 00 CD 21**. Pasando a la última línea del programa:

```
mov al, 0
int 21h
end inicio ;<--- Línea actual. ;LC=47;
```

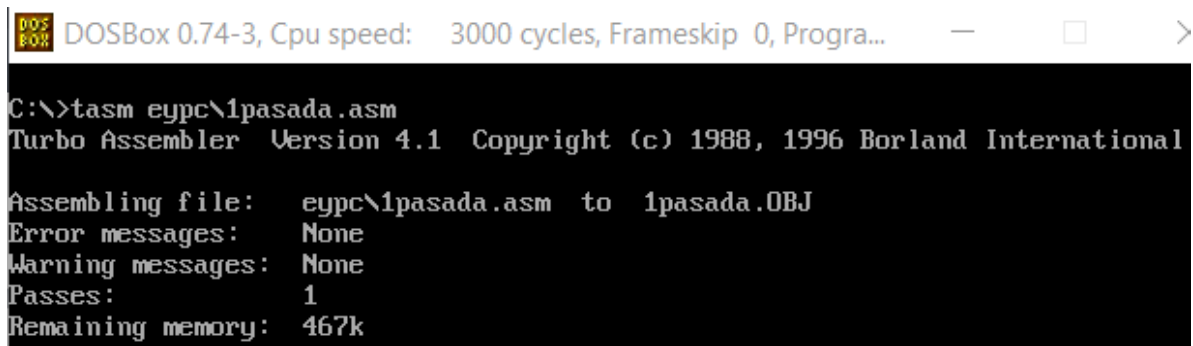
Figura 35: *Última línea.*

En este punto el ensamblador detecta la directiva **end** que indica que se trata del fin del archivo (EOF). Entonces, se revisa la tabla de símbolos para verificar que no haya símbolos indefinidos.

Nombre	Segmento	Tamaño	Valor	Tipo
tecla_enter	(Constante)	Byte	0Dh	D
N_mayus	(Constante)	Byte	'N'	D
N_minus	(Constante)	Byte	'n'	D
carácter	Datos	Byte	0000h	D
inicio	Código	Word	0000h	D
lee_teclado	Código	Word	0005h	D
salir	Código	Word	0029h	D

Figura 36: *Tabla de símbolos actualizada.*

Debido a que todos los símbolos en la tabla son de tipo **D** y no existe ningún conflicto entre los tipos de datos en las instrucciones del programa, este podrá ser ensamblado sin errores a la salida. Escribiendo el programa nuevamente y ensamblándolo con DOSBox, se obtiene la salida sin errores esperada como puede observarse en la siguiente página:



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra... — □ >

```
C:\>tasm eypc\1pasada.asm
Turbo Assembler Version 4.1 Copyright (c) 1988, 1996 Borland International

Assembling file:  eypc\1pasada.asm  to  1pasada.OBJ
Error messages:   None
Warning messages: None
Passes:           1
Remaining memory: 467k
```

Figura 37: Salida sin errores al ensamblar el programa.