



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Estructura y Programación de Computadoras

Grupo: 02 - Semestre: 2021-1

Tarea 7:
Ensamblador de dos pasadas

FECHA DE ENTREGA: 24/01/2021

Alumno:
Téllez González Jorge Luis



Índice

1. Planteamiento del problema	2
1.1. Consideraciones	3
2. Procedimiento	3
2.1. Primer pasada	3
2.2. Segunda pasada	8

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:

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.

2. Procedimiento

2.1. Primer pasada

1. Iniciando desde el segmento de código, se lee la definición de un símbolo, concretamente, de una etiqueta. Entonces, se inserta el símbolo en la tabla de símbolos con el valor de $LC = 0$ o `0000h`.

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

Figura 4: Primer línea leída.

Se escribe información en el archivo intermedio y calcula la longitud de la instrucción para incrementar LC . Dado que no hay instrucción, el valor de $LC = 0$ se mantiene.

2. Se lee la siguiente línea. Dado que no contiene símbolos y se trata únicamente de una instrucción, únicamente se escribe información en el archivo intermedio y se calcula la longitud de la instrucción; incrementando $LC = 0 + 3 = 3$.
3. Se continúa con la siguiente línea. Tampoco contiene símbolos, por lo que se escribe en el archivo intermedio y se incrementa $LC = 3 + 2 = 5$.

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 actualizada.*

```

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

```

Figura 6: *Segunda línea leída.*

```

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

```

Figura 7: *Tercer línea leída.*

4. La siguiente línea contiene una etiqueta, así que se inserta esta en la tabla de símbolos con el valor actual de $LC = 5$.

```

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

```

Figura 8: *Cuarta línea leída.*

El valor de LC no se ve afectado al tratarse de una instrucción vacía y se actualiza 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

Figura 9: *Tabla de símbolos actualizada.*

5. La siguiente línea no contiene símbolos, así que se procede a escribir sobre el archivo intermedio y se incrementa $LC = 5 + 2 = 7$.

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

Figura 10: *Quinta línea leída.*

6. La siguiente línea tampoco contiene símbolos, así que se escribe sobre el archivo y se incrementa $LC = 7 + 2 = 9$.

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

Figura 11: *Sexta línea leída.*

7. Como esta línea no presenta definiciones de símbolos, se inserta sobre el archivo y se incrementa $LC = 9 + 3 = 12$.

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

Figura 12: *Séptima línea leída.*

8. Esta línea no presenta definiciones de símbolos, se inserta sobre el archivo y se incrementa $LC = 12 + 5 = 17$.

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

Figura 13: *Octava línea leída.*

9. La siguiente línea hace uso de un símbolo no declarado, por lo que se inserta información en el archivo intermedio y se incrementa $LC = 17 + 2 = 19$. Para incrementar LC , se hace uso del código de operación de 8 bits como en el ejemplo de la clase.

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

Figura 14: *Novena línea leída.*

10. Esta línea no presenta definiciones de símbolos, se inserta sobre el archivo y se incrementa $LC = 19 + 5 = 24$.

```
je salir  
cmp [caracter], N_mayus ;<--- Línea actual. ;LC=19;  
je salir
```

Figura 15: *Décima línea leída.*

11. Esta siguiente línea hace uso de un símbolo no declarado que fue usado previamente, por lo que se inserta información en el archivo intermedio y se incrementa $LC = 24 + 2 = 26$.

```
cmp [caracter], N_mayus  
je salir ;<--- Línea actual. ;LC=24;  
cmp [caracter], N_minus
```

Figura 16: *Onceava línea leída.*

12. La siguiente línea no presenta definiciones de símbolos, así que se escribe sobre el archivo y se incrementa $LC = 26 + 5 = 31$.

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

Figura 17: *Doceava línea leída.*

13. Esta siguiente línea hace uso de un símbolo no declarado que fue usado previamente, por lo que se inserta información en el archivo intermedio y se incrementa $LC = 31 + 2 = 33$.

```
cmp [caracter], N_minus  
je salir ;<--- Línea actual. ;LC=31;  
jmp lee_teclado
```

Figura 18: *Treceava línea leída.*

14. La línea no contiene definición de símbolo, inserta en el archivo intermedio e incrementa $LC = 33 + 2 = 35$.

```

    jmp lee_teclado ;<--- Línea actual. ;LC=33;
salir:
    mov ah, 4Ch

```

Figura 19: Catorceava línea leída.

15. Se detecta la definición de un símbolo, por lo que se inserta en la tabla de símbolos y se actualiza usando el valor de $LC = 35$ o $0023h$. No se incrementa LC .

```

    jmp lee_teclado
salir:                ;<--- Línea actual. ;LC=35;
    mov ah, 4Ch

```

Figura 20: Quinceava línea leída.

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	0023h	D

Figura 21: Tabla de símbolos actualizada.

16. La línea no contiene definición de símbolo, inserta en el archivo intermedio e incrementa $LC = 35 + 2 = 37$.

```

salir:
    mov ah, 4Ch ;<--- Línea actual. ;LC=35;
    mov al, 0

```

Figura 22: Dieciseisava línea leída.

17. La línea no contiene definición de símbolo, inserta en el archivo intermedio e incrementa $LC = 37 + 2 = 39$.

```
mov ah, 4Ch
mov al, 0 ;<--- Línea actual. ;LC=37;
int 21h
```

Figura 23: Diecisieteava línea leída.

18. La línea no contiene definición de símbolo, inserta en el archivo intermedio e incrementa $LC = 39 + 2 = 41$.

```
mov al, 0
int 21h ;<--- Línea actual. ;LC=39;
end inicio
```

Figura 24: Dieciochoava línea leída.

19. Se ha llegado al fin del archivo (EOF). Se cierra el archivo intermedio y comienza la segunda pasada.

```
mov al, 0
int 21h
end inicio ;<--- Línea actual. ;LC=41;
```

Figura 25: Diecinueveava línea leída.

2.2. Segunda pasada

Teniendo la tabla de símbolos generada en la primera pasada, se pueden resolver las referencias de forma directa siempre y cuando se encuentren en la tabla.

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	0023h	D

Figura 26: Tabla de símbolos final generada en la primera pasada.

Como en el ejemplo anterior no hay símbolos que no se encuentren en la tabla de símbolos, este procederá a ensamblarse de forma exitosa. Por otra parte, como las referencias futuras están definidas, las

```

.code
inicio:
    mov ax, @data                ;B8 B0 48      LC=0
    mov ds,ax                    ;8E D8          LC=3
lee_teclado:
    mov ah, 08h                  ;B4 08          LC=5
    int 21h                      ;CD 21          LC=7
    mov [caracter], al           ;A2 00 00      LC=9
    cmp [caracter], tecla_enter  ;80 3E 00 00 0D LC=12
    je salir                     ;74 12          LC=17
    ;35d-17d-long -> 12h

    cmp [caracter], N_mayus      ;80 3E 00 00 4E LC=19
    je salir                     ;74 0B          LC=24
    ;35d-24d-long -> 0Bh

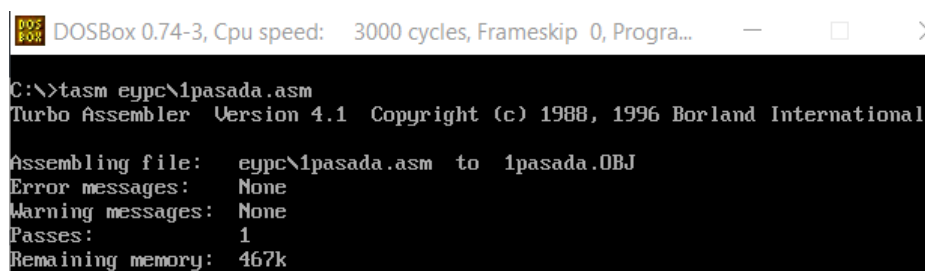
    cmp [caracter], N_minus      ;80 3E 00 00 6E LC=26
    je salir                     ;74 04          LC=31
    ;35d-31d-long -> 04h
    jmp lee_teclado              ;EB E2          LC=33
    ;0005h-(35)=-30d=E2h
salir:
    mov ah, 4Ch                  ;B4 4C          LC=35
    mov al, 0                    ;B0 00          LC=37
    int 21h                      ;CD 21          LC=39
end inicio                      ;           LC=41

```

Figura 27: Código generado final.

instrucciones **je salir** no generan NOP al final. Su valor se calcula haciendo uso de la fórmula definida en el ensamblador de 1 pasada.

El código generado al final no muestra ningún error al ensamblarlo ni tiene ningún error de sintaxis que provoque un error a la salida, como puede verse a continuación (Como el código es idéntico al de la tarea anterior, la salida es la misma y se usa el mismo archivo).



```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
C:\>tasm eyyc\1pasada.asm
Turbo Assembler Version 4.1 Copyright (c) 1988, 1996 Borland International

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

```

Figura 28: Ensamblado del código.