

Lectura y escritura de un caracter

00% Símbolo del sistema - debug

```
C:\Users\hilario>debug
```

```
~a
```

```
179C:0100 mov ah,01
```

```
179C:0102 int 21
```

```
179C:0104 mov dl,al
```

```
179C:0106 mov ah,02
```

```
179C:0108 int 21
```

```
179C:010A int 20
```

```
179C:010C
```

```
~g
```

```
AA
```

```
Program terminated normally
```

```
~
```

Trabajando con cadenas en **Debug**

```
Símbolo del sistema - debug

C:\Users\hilario>debug
-e 500 'hola mundo' 0d 0a '$'
-a
179C:0100 mov dx,0500
179C:0103 mov ah,09
179C:0105 int 21
179C:0107 int 20
179C:0109
-g
hola mundo

Program terminated normally
-d 500
179C:0500  68 6F 6C 61 20 6D 75 6E-64 6F 0D 0A 24 00 00 00  hola mundo..$.
179C:0510  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
179C:0520  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
179C:0530  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
179C:0540  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
179C:0550  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
179C:0560  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
179C:0570  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
-
```

Herramientas

Entorno Dos y Windows

Masm es el ensamblador mas usado en **DOS y Windows**, desarrollado por Microsoft, puede utilizarse para crear bibliotecas y controladores de dispositivos., existes otros ensambladores para Dos como son: TASM, NASM, ETC..

Entorno Linux

Para desarrollar **programas ensamblador en Linux**, deberemos elegir el compilador de ensamblador que queremos utilizar. La elección lógica es usar **GAS (as)**, **el estándar que lleva toda distribución Linux**.

GAS tiene la desventaja de utilizar la sintaxis AT&T, que difiere bastante de la sintaxis Intel (algo más sencilla). Una alternativa es utilizar **NASM**, **que ofrece la misma funcionalidad que GAS, pero utiliza la sintaxis de Intel**.

*Sin embargo, puesto que gcc utiliza **as** como ensamblador, conviene conocer ambas sintaxis.*

Así pues, los programas que hagamos en esta parte de las prácticas los desarrollaremos utilizando las herramientas NASM y GAS.

Sintaxis AT&T. Ensamblador de GNU

GAS (Gnu ASsembler) utiliza la sintaxis de AT&T, que tiene pequeñas diferencias con respecto a la sintaxis estándar de Intel (usada en NASM, TASM, MASM, etc). Las principales diferencias se detallan a continuación:

- En AT&T, a los **nombres de los registros se les añade el prefijo %**

AT&T: %eax

I NTEL: eax

	high-byte	low-byte	16-bit	32-bit	default use
	%ah	%al	%ax	%eax	accumulator
	%dh	%dl	%dx	%edx	data
	%ch	%cl	%cx	%ecx	count
	%bh	%bl	%bx	%ebx	base
			%bp	%ebp	frame base pointer
			%si	%esi	source index
			%di	%edi	destination index
			%sp	%esp	stack pointer

31 16/15 8/7 0

- En AT&T, el destino se coloca a la derecha y el fuente a la izquierda (en Intel es al revés). Las siguientes instrucciones cargan en ebx el valor de eax

AT&T: `movl %eax, %ebx`

INTEL: `mov ebx, eax`

- En AT&T, a los valores inmediatos se les añade el prefijo \$ en el siguiente ejemplo, la primera instrucción carga la dirección de la variable en eax; la segunda carga el valor 0F02h en ebx

AT&T: `movl $var, %eax`

`movl $0xf02, %ebx`

INTEL: `mov eax, offset var`

`mov ebx, 0f02h`

- En AT&T, el tamaño del resultado se especifica con sufijos (b, w o l) en las instrucciones (en Intel cuando hay ambigüedad se utiliza *byte ptr*, *word ptr* o *dword ptr*). Si lo omitimos, GAS intentará “adivinar” el tamaño, y es algo que no queremos que haga...

AT&T: movb var, %ah
 movw %bx, %ax
INTEL: mov ah, byte ptr var
 mov ax, bx

AT&T: movb %bl, %al
 movw %bx, %ax
 movl %ebx, %eax
 movl (%ebx), %eax

INTEL: mov al, bl
 mov ax, bx
 mov eax, ebx
 mov eax, dword ptr [ebx]

- **Direccionamiento a memoria:**

Es uno de los aspectos que más cambian. Veamos la sintaxis de Intel para hacer un direccionamiento a base, con índice y desplazamiento:

[base + índice*escala + desplazamiento]

en la sintaxis AT&T esto queda como sigue: desplazamiento (base , índice , escala)

Veamos dos ejemplos:

AT&T: movl array (, %eax, 4), %edx

INTEL: mov edx, array[eax*4]

AT&T: movl (%ebx) , %eax

movl 3(%ebx) , %eax

I NTEL: mov eax , [ebx]

mov eax,[ebx+3]

- **Salto lejano**

AT&T: lcall \$sección, \$offset

ljmp \$sección, \$offset

lret \$V

INTEL: call far sección:offset

jmp far sección:offset

ret far V

- **Nemotécnico. Varían los nemotécnicos de algunas instrucciones**

AT&T: movswl %ax, %ecx

movzbl %ah, %cx

cbl

cwbl

cwtd

cld

INTEL: movsx ecx, ax

movzx cx, ah

cw

cwde

cw

cdq

- **Directivas del compilador.**

Como comentamos más arriba, hay diferencias en cuanto a algunas directivas al programar con el ensamblador GAS o NASM.

- En ambos ensambladores hay que definir las secciones de datos y código utilizando los mismos nombres (*.data .bss .text*). Sin embargo, la directiva utilizada para definir las secciones difiere de un ensamblador a otro:

NASM	GAS
<code>section .data</code>	<code>[.section] .data</code>
<code>section .bss</code>	<code>[.section] .bss</code>
<code>section .text</code>	<code>[.section] .text</code>

- En ambos ensambladores, la etiqueta de entrada al programa ensamblador suele ser *_start*. Sin embargo, la directiva utilizada difiere de un ensamblador a otro:

NASM	GAS
<code>section .text</code> <code>global _start</code> <code>_start:</code>	<code>.text</code> <code>.globl _start</code> <code>_start:</code>

- La definición de datos constantes se lleva a cabo utilizando de la misma forma, pero utilizando palabras reservadas diferentes:

NASM	GAS
<pre>section .data cadena db "un texto" longitud equ \$ - cadena cero dw 0 letra db 'A'</pre>	<pre>.data cadena: .ascii "un texto" longitud = . - cadena cero: .hword 0 letra: .byte 'A'</pre>

;Programa Hola Mundo en ensamblador
;Compilador: **MASM 6.15**
;MASM (Microsoft Assambler)
;Programa para DOS bajo entorno Windows

;declaramos el segmento de pila

Pila segment stack 'stack'

db 128 dup (?)

Pila ends

;segmento de datos

Datos segment 'data'

Saludo db 'Hola mundo!!!Datos ends

;segmento de código

Codigo segment 'code'

assume CS:Codigo, DS:Datos, SS:Pila

Main:

;lo primero es mover el segmento de datos al

;registro de segmento DS

mov ax, seg Datos

mov ds, ax

;Usamos la función 09h de DOS para mostrar una cadena

;La cadena debe estar en DX

mov dx, offset Saludo

mov ah, 9

int 21h

;devuelve el control al sistema operativo mediante la

;función 4Ch del DOS.

mov ah, 4Ch

int 21h

Codigo ends

end Main

Masm (DOS)

```
1 ;hola.asm
2 pila segment stack 'stack'
3     db 256 dup (?)
4 pila ends
5 datos segment 'data'
6     msg db 'hola mundo$'
7 datos ends
8 codigo segment 'code'
9     assume cs:codigo, ds:datos, ss:pila
10 main:
11     mov ax,seg datos
12     mov ds,ax
13     mov dx,OFFSET msg
14     mov ah,09
15     int 21h
16
17     mov ah,4ch
18     int 21h
19 codigo ends
20     END main
```

SS { 2, 3, 4

DS { 5, 6, 7

CS { 13, 14, 15, 16, 17, 18

Compilando masm usando ml

C:\ Símbolo del sistema

```
C:\DOCUMENTOS\ADMINISTRACION\MIS DOCUMENTOS\NOTAS>ml holasm.asm
```

```
Microsoft (R) Macro Assembler Version 6.15.8803
```

```
Copyright (C) Microsoft Corp 1981-2000. All rights reserved.
```

```
Assembling: holasm.asm
```

```
Microsoft (R) Segmented Executable Linker Version 5.31.009 Jul 13 1992
```

```
Copyright (C) Microsoft Corp 1984-1992. All rights reserved.
```

```
Object Modules [.obj]: holasm.obj
```

```
Run File [holasm.exe]: "holasm.exe"
```

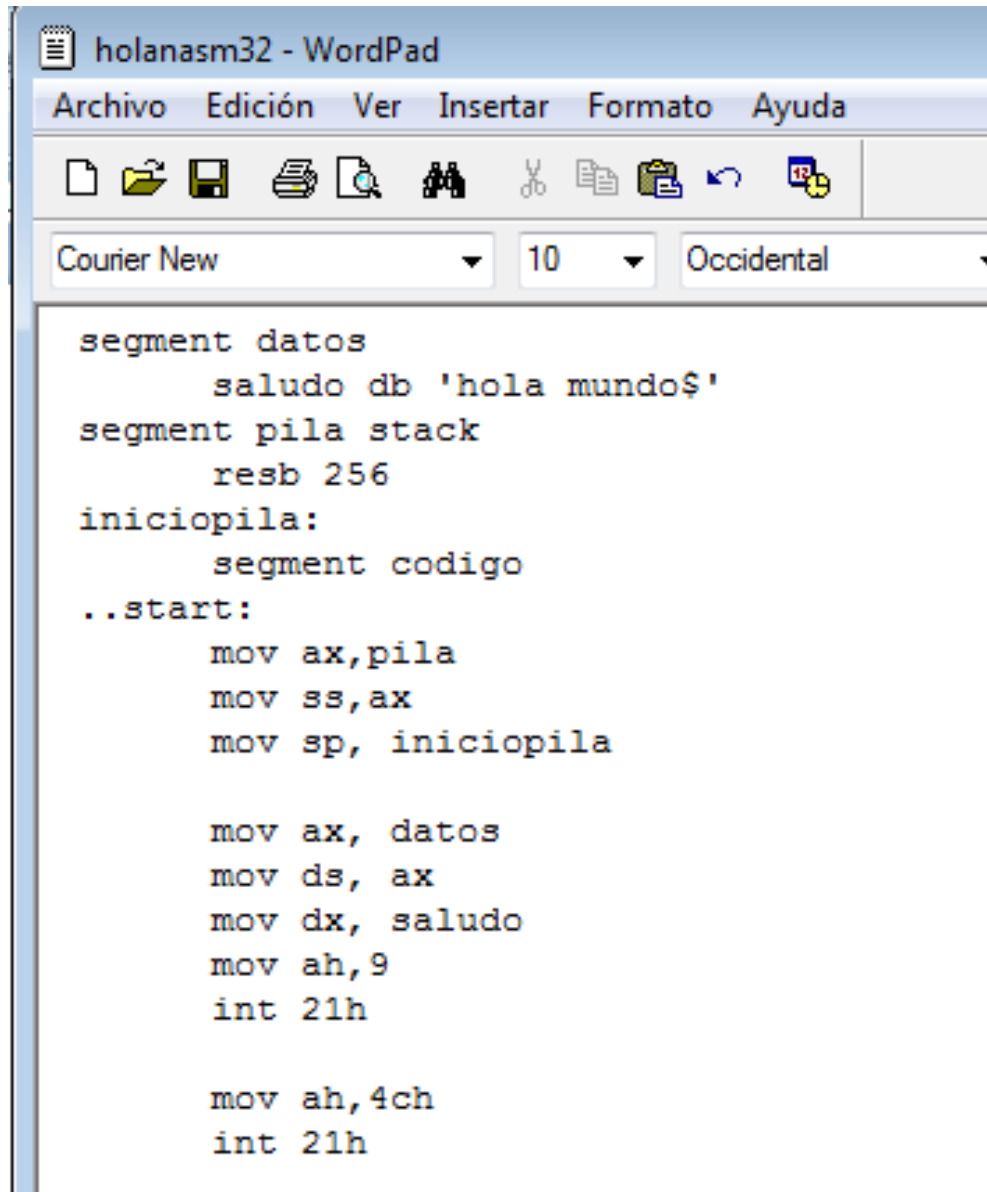
```
List File [nul.map]: NUL
```

```
Libraries [.lib]:
```

```
Definitions File [nul.def]:
```

```
C:\DOCUMENTOS\ADMINISTRACION\MIS DOCUMENTOS\NOTAS>_
```

holanasm32.asm usando NASM en windows



```
segment datos
    saludo db 'hola mundo$'
segment pila stack
    resb 256
iniciopila:
    segment codigo
..start:
    mov ax,pila
    mov ss,ax
    mov sp, iniciopila

    mov ax, datos
    mov ds, ax
    mov dx, saludo
    mov ah,9
    int 21h

    mov ah,4ch
    int 21h
```

Compilando y ejecutando con Nasm en Windows

```
C:\Nueva carpeta\nasm32>dir/w
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 843A-73AD

Directorio de C:\Nueva carpeta\nasm32

[.]                [...]                ALINK.EXE           ALINK.TXT
al_al.zip          [contrib]             [example]           example.zip
history.txt        holanasm.asm             holanasm.exe        holanasm.obj
holanasm32.asm     holanasm32.o            holanasm32.obj      LICENSE
nasm.exe           nasm.ico                nasmdoc.pdf         nasmpath.bat
ndisasm.exe        [rdoff]                rtn.asm             rtn.RC
rtn.res            t2.asm                  tcoff.asm           tdll.asm
TEST.ASM           TEST.OBJ                Uninstall.exe
                   26 archivos                1.884.388 bytes
                   5 dirs  104.048.074.752 bytes libres

C:\Nueva carpeta\nasm32>nasm -fobj holanasm32.asm

C:\Nueva carpeta\nasm32>alink holanasm32.obj -oEXE
ALINK v1.6 (C) Copyright 1998-9 Anthony A.J. Williams.
All Rights Reserved

Loading file holanasm32.obj
matched Externs
matched ComDefs
```

```
C:\Nueva carpeta\nasm32>dir/w
```

```
El volumen de la unidad C no tiene etiqueta.
```

```
El número de serie del volumen es: 843A-73AD
```

```
Directorio de C:\Nueva carpeta\nasm32
```

[.]	[..]	ALINK.EXE	ALINK.TXT
al_al.zip	[contrib]	[example]	example.zip
history.txt	holanasm.asm	holanasm.exe	holanasm.obj
holanasm32.asm	holanasm32.exe	holanasm32.o	holanasm32.obj
LICENSE	nasm.exe	nasm.ico	nasmdoc.pdf
nasmpath.bat	ndisasm.exe	[rdoff]	rtn.asm
rtn.RC	rtn.res	t2.asm	tcoff.asm
tdll.asm	TEST.ASM	TEST.OBJ	Uninstall.exe
	27 archivos	1.884.764 bytes	
	5 dirs	104.048.074.752 bytes libres	

```
C:\Nueva carpeta\nasm32>holanasm32
```

```
hola mundo
```

```
C:\NUEVAC~1\nasm32>_
```


hola.asm usando NASM en linux

```
holanasm.asm ✕  
section .data  
    msg db "hola mundo", 0xA  
    len equ $ - msg  
  
section .text  
    global _inicio  
_inicio:  
    movl edx, len  
    movl ecx, msg  
    movl ebx, 1  
    movl eax, 4  
    int 0x80  
  
    movl ebx, 0  
    movl eax, 1  
    int 0x80  
  
.data
```

Instalando NASM en Linux

```
hilario@hilario-Inspiron-530: ~/ensam-prog
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

hilario@hilario-Inspiron-530:~/ensam-prog$ ls
holagas holagas.asm holagas.o holanasm.asm
hilario@hilario-Inspiron-530:~/ensam-prog$ nasm -f holanasm.asm
El programa «nasm» no está instalado actualmente. Puede instalarlo escribiendo:
sudo apt-get install nasm
hilario@hilario-Inspiron-530:~/ensam-prog$ sudo apt-get install nasm
[sudo] password for hilario:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes NUEVOS:
  nasm
0 actualizados, 1 se instalarán, 0 para eliminar y 385 no actualizados.
Necesito descargar 1031kB de archivos.
Se utilizarán 2863kB de espacio de disco adicional después de esta operación.
Des:1 http://mx.archive.ubuntu.com/ubuntu/ maverick/main nasm i386 2.08.01-1 [10
31kB]
Descargados 1031kB en 3s (287kB/s)
Seleccionando el paquete nasm previamente no seleccionado.
(Leyendo la base de datos ... 00%
123116 ficheros y directorios instalados actualmente.)
Desempaquetando nasm (de .../nasm_2.08.01-1_i386.deb) ...
Procesando disparadores para man-db ...
Procesando disparadores para doc-base ...
Processing 31 changed 1 added doc-base file(s)...
Registering documents with scrollkeeper...
Procesando disparadores para install-info ...
Configurando nasm (2.08.01-1) ...
hilario@hilario-Inspiron-530:~/ensam-prog$
```



holanasm - WordPad

Archivo Edición Ver Insertar Formato Ayuda



```
section .data
    msg db "hola mundo",0xA
    len equ $ - msg
```

```
section .text
    global _start
_start:
    mov edx,len
    mov ecx,msg
    mov ebx,1
    mov eax,4
    int 0x80

    mov ebx,0
    mov eax,1
    int 0x80
```

Compilando y ejecutando con Nasm en Linux

```
hilario@hilario-Inspiron-530: ~/ensam-prog
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
hilario@hilario-Inspiron-530:~/ensam-prog$ ls
holagas holagas.asm holagas.o holanasm.asm
hilario@hilario-Inspiron-530:~/ensam-prog$ nasm -f elf holanasm.asm
hilario@hilario-Inspiron-530:~/ensam-prog$ ld -s -o holanasm holanasm.o
hilario@hilario-Inspiron-530:~/ensam-prog$ ls
holagas holagas.asm holagas.o holanasm holanasm.asm holanasm.o
hilario@hilario-Inspiron-530:~/ensam-prog$ ./holanasm
hola mundo
hilario@hilario-Inspiron-530:~/ensam-prog$
```

hola.asm usando GAS

holagas.asm ✕

```
.text
.global _start

_start:
    movl $len,%edx
    movl $msg,%ecx
    movl $1,%ebx
    movl $4,%eax

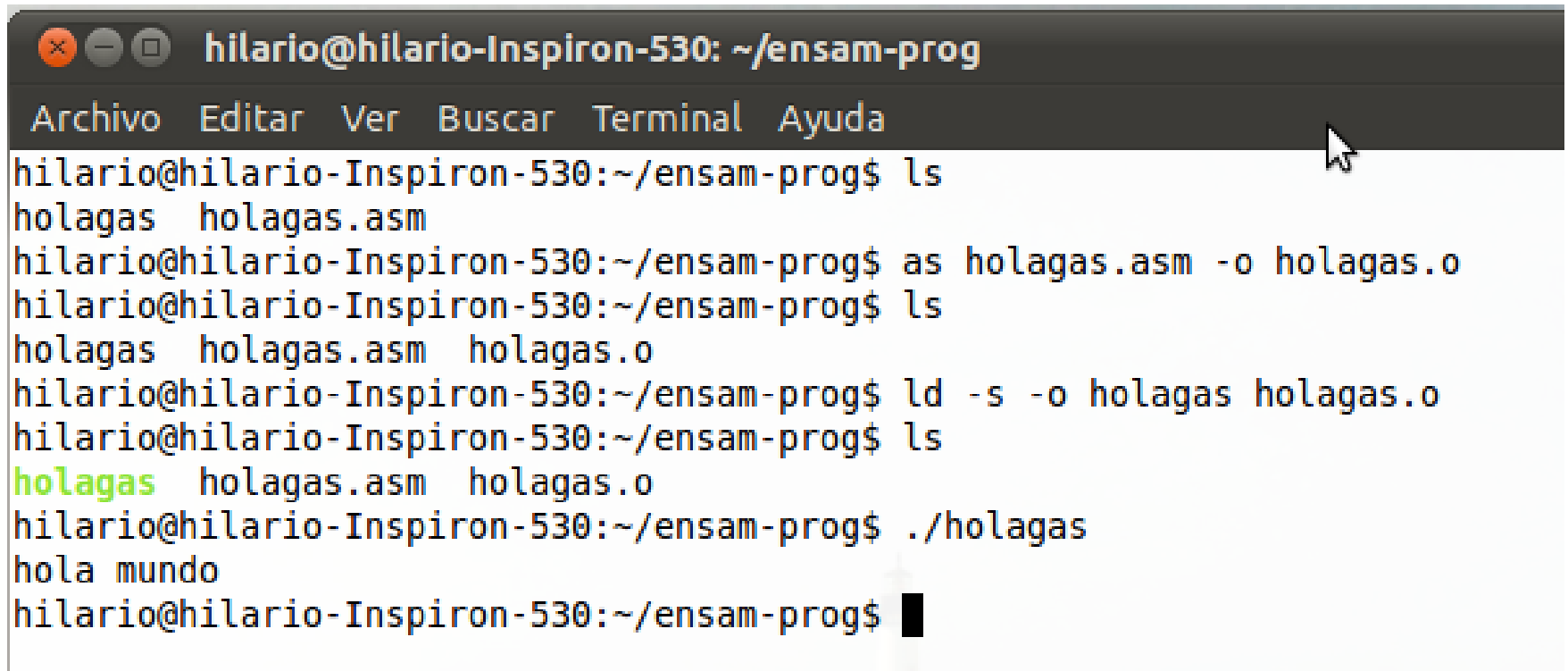
    int $0x80

    movl $0,%ebx
    movl $1,%eax

    int $0x80

.data
    msg: .ascii "hola mundo \n"
    len= 12
```

Compilando y ejecutando con GAS



```
hilario@hilario-Inspiron-530: ~/ensam-prog
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
hilario@hilario-Inspiron-530:~/ensam-prog$ ls
holagas  holagas.asm
hilario@hilario-Inspiron-530:~/ensam-prog$ as holagas.asm -o holagas.o
hilario@hilario-Inspiron-530:~/ensam-prog$ ls
holagas  holagas.asm  holagas.o
hilario@hilario-Inspiron-530:~/ensam-prog$ ld -s -o holagas holagas.o
hilario@hilario-Inspiron-530:~/ensam-prog$ ls
holagas  holagas.asm  holagas.o
hilario@hilario-Inspiron-530:~/ensam-prog$ ./holagas
hola mundo
hilario@hilario-Inspiron-530:~/ensam-prog$
```

W nasm - debug holanasm32.exe

Microsoft Windows [Versión 6.0.6000]

Copyright (c) 2006 Microsoft Corporation. Reservados todos los der

C:\Nueva carpeta\nasm32>dir/w

El volumen de la unidad C no tiene etiqueta.

El número de serie del volumen es: 843A-73AD

Directorio de C:\Nueva carpeta\nasm32

[.]	[..]	ALINK.EXE	ALINK.TXT
al_al.zip	[contrib]	[example]	example.zip
history.txt	holanasm.asm	holanasm.exe	holanasm.obj
holanasm32.asm	holanasm32.exe	holanasm32.o	holanasm32.obj
lect-escrip.asm	lect-escrip.exe	lect-escrip.obj	LICENSE
nasm.exe	nasm.ico	nasmdoc.pdf	nasmpath.bat
ndisasm.exe	[rdoff]	rtn.asm	rtn.RC
rtn.res	t2.asm	tcoff.asm	tdll.asm
TEST.ASM	TEST.OBJ	Uninstall.exe	
	30 archivos	1.885.627 bytes	
	5 dirs	103.868.989.440 bytes libres	

C:\Nueva carpeta\nasm32>debug holanasm32.exe

```
-u
1809:0000 0E          PUSH    CS
1809:0001 1F          POP     DS
1809:0002 BA0E00    MOV     DX,000E
1809:0005 B409        MOV     AH,09
1809:0007 CD21        INT     21
1809:0009 B8004C      MOV     AX,4C00
1809:000C CD21        INT     21
1809:000E 54          PUSH    SP
1809:000F 68          DB      68
1809:0010 69          DB      69
1809:0011 7320        JNB     0033
1809:0013 7072        JO      0087
1809:0015 6F          DB      6F
1809:0016 67          DB      67
1809:0017 7261        JB      007A
1809:0019 6D          DB      6D
1809:001A 207265     AND     [BP+SI+65],DH
1809:001D 7175        JNO     0094
1809:001F 69          DB      69
```

-

Esqueleto básico

```
[section .text]
```

```
..start:
```

```
    ;colocar código
```

```
    mov ah,4ch
```

```
    int 21h
```

```
[section .data]
```

```
    ;variables inicializadas
```


Pseudo-op

- `ORG 100h`
 - Define la dirección de “origen”. En el caso de programas **com** la dirección es **100h**
- `SECTION .text`
 - Define el inicio de un grupo de instrucciones para ser ensambladas en un código ejecutable.
- `SECTION .data`
 - Define el inicio de un grupo de declaraciones que son inicializadas en el segmento de datos.
- `SECTION .bss`
 - Define el inicio de un grupo de declaraciones no inicializadas.

El uso de la directiva `SECTION` puede ser intercalada en cualquier orden, ya que cuando el código se compila todas las secciones son concatenadas.

Primero `.text`, luego `.data` y `.bss`. (la sección `.bss` no genera código en realidad pero el proceso de reservación de bloques de memoria provoca que las etiquetas asociadas tomen los valores correctos).

Declaración de constantes

- No olvidar que comunmente en la declaración también se usan constantes. Para ello se emplea la instrucción EQU
- La sintaxis de EQU es similar a la de DB, solo que en lugar del db o dw se pone EQU, ejemplo:

Etiqueta EQU valor

- En conjunto a la declaración de constantes, el nasm define dos pseudo operaciones que permiten el cálculo de direcciones estas dos constantes especiales son \$ y \$\$, a continuación se explican

Constantes especiales⁽¹⁾

- El \$ define la posición de la dirección de memoria al inicio de donde lo coloquemos, ejemplo:

```
msj db "pepito"  
tamaño EQU $ - msj
```

Ya que \$ regresa la posición en donde éste se encuentra; si realizamos una resta del valor que tiene \$ menos la posición de la etiqueta msj, lo que obtendremos como resultado es la longitud de la declaración msj. En este caso el resultado lo asignamos a la variable tamaño. Tamaño es igual a 6.

Constantes especiales⁽²⁾

- De forma similar, el uso de la constante \$\$ regresa la dirección segmento en donde incluyamos esta constante. Por ejemplo:

```
var1 db "variable 1$"  
var2 db "variable 2$"  
valor db 10  
espacio EQU $ - $$
```

- En este caso lo que se obtiene como resultado es la longitud de espacio ocupada por todo el segmento hasta antes la declaración de la variable espacio. Para este ejemplo el valor es **23d** o **17h**. Un byte por cada caracter

Compilación

- Para compilar en un ambiente de windows se opta por tener dos formatos de salida:
 - Los **obj** que representan aplicaciones típicas .exe de 16 o 32 bits
 - Los **bin** que representan programas típicos como los .com
- Para compilar desde la línea de comandos se hace tecleando:
`nasm -f obj archivo.asm`

Ligado

- Un proceso necesario para la generación de archivos ejecutables es el ligado a través del cual se agregan todos los elementos necesarios para el ambiente de ejecución del programa.
- La compilación en línea de comandos se hace tecleando:
 alink archivo.obj
- Como resultado obtenemos un archivo con extensión .exe

Diferencia de los com y exe(1)

- En los ejemplos anteriores las diferencias principales aparecen en negritas y con fuente color azul.
- Los com inician en la posición 100h
- Los exe emplean la directiva `..start:` para marcar el punto de inicio que puede ser 0
- Los com finalizan el programa mediante la `int 20`
- Los exe emplean la función `4ch` de la interrupción 21

Campos simples

La sintaxis para definir campos en el segmento de datos depende en parte del ensamblador usado, si bien casi siempre guardan cierta similitud, en el caso de NASM

Identificador db | dw | dd valor

Donde:

Identificador es una etiqueta o nombre que vamos a asignar a esa posición en el segmento de datos, nasm no distingue entre mayúsculas y minúsculas.

db (define byte) en hexadecimal max +7F en decimal +127 a -128

dw (define word)) en hexadecimal max +7FFF en decimal +32,767 a -32,768

dd (define double word) en hexadecimal max +7FFFFFFF en decimal +2,147,483,647 a -2,147,483,648

Conjunto de campos

Cuando se necesitan múltiples campos de un mismo tamaño y asociados entre sí, lo que en otros lenguajes se conoce como matrices, arreglos o vectores, tenemos varias opciones, se pueden usar cualquiera de los campos simples (**db**, **dw**, o **dd**) con el repetidor **times** de nasm. Este precede al indicador y va seguido de un entero que comunica el número de veces que se repetirá.

Ejem:

caracteres times 256 db '?'

Si no necesitamos dar un valor inicial a cada uno de los bytes reservados, podemos usar la sintaxis siguiente

caracteres resb 256

Además de **resb**, para reservar bytes, también podemos usar **resw** y **resd** para reservar un cierto número de palabras o dobles palabras.

Conjunto campos

matriz.asm

```
1 stack segment para stack 'stack'
2     dw 10 dup(?)
3 stack ends
4 ;-----
5 data segment para 'data'
6     mesa db 'Proporciona la matriz A$'
7     mes db 'Diste esta matriz ?$'
8     impfila db 'Esta es la 2da. fila$'
9     matriza db 3 dup(3 dup(' '))
10 data ends
11 .
```

```
C:\ENSAMB~1>matriz
Proporciona la matriz A
1  2  3
4  5  6
a  b  c

Diste esta matriz ?
1  2  3
4  5  6
a  b  c

Esta es la 2da. fila
4  5  6
C:\ENSAMB~1>
```

lect-escri - Bloc de notas

Archivo Edición Formato Ver Ayuda

```
segment datos
    saludo db 'hola mundo$'
segment pila stack
    resb 256
iniciopila:
    segment codigo
..start:
    mov ax,pila
    mov ss,ax
    mov sp, iniciopila

    mov ax, datos
    mov ds, ax
    ;mov ah,01
    ;int 21h

    mov dl,'A'
    mov ah,02
    int 21h
```

W nasm

C:\NUEUAC~1\nasm32>lect-escri.exe

A

C:\NUEUAC~1\nasm32>

```

1 segment datos
2     fin db 'Fin del programa$'
3 segment pila stack
4     resb 256
5 iniciopila:
6     segment codigo
7 ..start:
8     mov ax, datos
9     mov ds, ax
10    mov ax, pila
11    mov ss, ax
12    mov sp, iniciopila
13
14    mov cx, 10    ;decimal
15    mov cx, 10q   ;octal
16    mov cx, 10b   ;binario
17    mov cx, 10h   ;hexadecimal
18
19
20    mov dx, fin
21    mov ah, 9
22    int 21h
23
24    mov ah, 4ch
25    int 21h

```

C:\Nueva carpeta\nasm32>nasm -fobj bases.asm

C:\Nueva carpeta\nasm32>alink bases.obj -oEXE
ALINK v1.6 (C) Copyright 1998-9 Anthony A.J. Williams.
All Rights Reserved

Loading file bases.obj
matched Externs
matched ComDefs

C:\Nueva carpeta\nasm32>bases.exe

Fin del programa

C:\NUEVAC~1\nasm32>debug bases.exe

~u

17D3:0000	B8C117	MOV	AX,17C1
17D3:0003	8ED8	MOV	DS,AX
17D3:0005	B8C317	MOV	AX,17C3
17D3:0008	8ED0	MOV	SS,AX
17D3:000A	BC0001	MOV	SP,0100
17D3:000D	B90A00	MOV	CX,000A
17D3:0010	B90800	MOV	CX,0008
17D3:0013	B90200	MOV	CX,0002
17D3:0016	B91000	MOV	CX,0010
17D3:0019	BA0000	MOV	DX,0000
17D3:001C	B409	MOV	AH,09
17D3:001E	CD21	INT	21

~q

C:\NUEVAC~1\nasm32>_

Menu.asm

```
menu.asm
1 segment datos
2     inicio db '  Menu $'
3     op1 db '1.- suma $'
4     op2 db '2.- resta $'
5     termina db '3.- Fin $'
6
7
8 segment pila stack
9     resb 256
10 iniciopila:
11     segment codigo
12 ..start:
13     mov ax, datos
14     mov ds, ax
15     mov ax, pila
16     mov ss, ax
17     mov sp, iniciopila
18
19     mov dx, inicio
20     call escribecadena
21     mov dx, op1
22     call escribecadena
23     mov dx, op2
24     call escribecadena
25     mov dx, termina
26     call escribecadena
27     call leecar
```

```

26      call escribecadena
27      call leecar
28
29      mov dh,20h
30      mov dl,22h
31      cmp al,'1'
32      jne resta
33      add dl,dh
34      call escribecar
35      jmp fin
36 resta: cmp al,'2'
37      jne fin
38      sub dl,dh
39      call escribecar
40 fin:
41      mov ah,4ch
42      int 21h
43 ;-----
44 escribecadena:
45      mov ah,9
46      int 21h
47      ret
48 ;-----
49 leecar:
50      mov ah,01
51      int 21h
52      ret
53 ;-----
54 escribecar:
55      mov ah,02
56      int 21h
57      ret

```

W nasm

C:\NUEVAC~1\nasm32>nasm -fobj menu.asm

C:\NUEVAC~1\nasm32>alink menu.obj -oEXE
 ALINK v1.6 (C) Copyright 1998-9 Anthony A.J. Williams.
 All Rights Reserved

Loading file menu.obj
 matched Externs
 matched ComDefs

C:\NUEVAC~1\nasm32>menu.exe
 Menu 1.- suma 2.- resta 3.- Fin 1B
 C:\NUEVAC~1\nasm32>