Índice

Marina de la Cruz
Alfonso Ortega

Esquema básico de un programa NASM

```
segment .data
...
...
segment .bss
...
...
```

Sección de datos

```
segment .text
global main
extern scan_int, scan_float ...
_fun1:
...
...
ret
_fun2:
...
...
ret
main:
...
...
ret
```

Sección de código

La sección de datos

Introducción

- La sección de datos contiene las declaraciones de las variables del programa.
- Dentro de la sección de datos se distinguen dos subsecciones:
  - **segment .data:** en esta subsección se declaran variables inicializadas.
  - **segment .bss**: reservado para la declaración de variables no inicializadas.
- La declaración de las variables se ajusta a una sintaxis diferente en cada una de las subsecciones.

La sección de datos

Variables inicializadas

- La variables incializadas se declaran en la subsección **"segment .data"** según la siguiente sintaxis:

  <nombre variable> <tamaño> <valor inicial>

  Donde:
  - **<nombre variable>** es el identificador de la variable
  - **<tamaño>** es el tamaño de la variable según la siguiente notación:
    - db: 1 byte
    - dw: 2 bytes
    - dd: 4 bytes
    - ...
  - **<valor inicial>** es el valor de inicialización de la variable
- Ejemplos:
  - unbyte db 0
  - unword dw 10
  - undoubleword dd 1000
  - Mensaje1 db "División por cero", 0

La sección de datos

Variables no inicializadas

- La variables no inicializadas se declaran en la subsección **"segment .bss"** según la siguiente sintaxis:

  <nombre variable> <tamaño> <cantidad>

  Donde:
  - **<nombre variable>** es el identificador de la variable
  - **<tamaño>** es el tamaño de la variable según la siguiente notación:
    - resb: byte
    - resw: 2 bytes
    - resd: 4 bytes (este es el tamaño e datos que vamos a utilizar)
    - ...
  - **<cantidad>** es la cantidad de posiciones de tamaño indicado por el campo <tamaño> que se reservan para la variable

- Ejemplos:
  - unbyte resb 1
  - doswords resw 2
  - ochodoublewords resd 8 (indicado para vectores y conjuntos)

La sección de código

- La sección de código contiene:
  - Declaración de símbolos externos y globales.
  - Instrucciones

- Los **símbolos externos**:
  - Son aquellos que no se definen en el fichero, pero que se utilizan y por lo tanto se asume que están definidos en otro fichero. Por ejemplo, las funciones de la librería que se proporcionan al alumno.
  - Para generar correctamente el ejecutable hay que enlazar con el fichero que contenga la definición de los símbolos.
  - La definición de símbolos externos se realiza con la palabra reservada **extern** seguida del símbolo o símbolos separados por comas. Por ejemplo:

    extern scan_int, scan_float, print_int

- Los **símbolos globales**:
  - Son aquellos que se definen en el módulo y que pueden ser utilizados desde el exterior.
  - La definición de símbolos externos se realiza con la palabra reservada **global** seguida del símbolo o símbolos separados por comas. Por ejemplo:

    global main

La sección de código

- Registros de propósito general de 32 bits: eax, ebx, ecx, edx
- Registros para operaciones en coma flotante: st0, st1,..., st7
- Puntero de pila: esp
- Cuando aparezca en una instrucción el nombre de una variable, representa la dirección de memoria de dicha variable. Para acceder al contenido hay que utilizar corchetes alrededor del nombre.
- Los códigos de operación se verán justo en el momento en el que se vayan a utilizar. Por ejemplo, la operación que permite multiplicar valores se verá cuando se estudie la generación de código correspondiente al producto de expresiones.
- Los comentarios son de una línea y comienzan con el carácter ";".

La sección de código

- Convenio de llamadas a funciones: se sigue el convenio de llamadas C, es decir, el llamador inserta en la pila los parámetros de la llamada, a continuación invoca a la función, y cuando termina la ejecución de ésta, restaura la pila, es decir, elimina de la misma los parámetros que insertó antes de realizar la llamada a la función. La función deja el resultado (si es entero) en el registro eax.

- Para crear un ejecutable a partir de un fichero fuente escrito en NASM, se pueden utilizar los siguientes comandos:

  nasm -g –o <fichero objeto (*.o) > –f elf <fichero fuente (*.asm) >

  gcc -o <fichero ejecutable> <fichero objeto 1> <fichero objeto 2> ...