

Ensamblador del simulador de una máquina Von Neumann

1 Introducción

Este documento describe las características del lenguaje ensamblador para el simulador de una máquina Von Neumann y del programa ensamblador que pasa de este lenguaje a ficheros con formato *.aje*. La extensión preferida para los ficheros en ensamblador es *.ens*.

2 Lenguaje ensamblador

2.1 Aspectos léxicos

Todo lo que siga a un punto y coma hasta el fin de línea será considerado comentario.

Las siguientes palabras están reservadas y no podrán utilizarse como identificadores de etiquetas:

| | | | |
|---------------|-----------|----------|-------|
| @CODIGO | @DATOS | @PILA | ADD |
| AND | BRC | BRNC | BRNO |
| BRNS | BRNZ | BRO | BRs |
| BRZ | BYTEALTO | BYTEBAJO | CALL |
| CLI | CODIGO | COMP | DATOS |
| DEC | DIRECCION | EQU | FIN |
| FINP | INC | INICIO | INT |
| IRET | JMP | MOV | MOVH |
| MOVL | NEG | NOP | NOT |
| OR | ORIGEN | PILA | POP |
| PROCEDIMIENTO | PUSH | R0 | R1 |
| R2 | R3 | R4 | R5 |
| R6 | R7 | RET | STI |
| SUB | VALOR | VECES | XOR |

Las palabras reservadas pueden estar tanto en mayúsculas como en minúsculas. En los identificadores, sin embargo, se tendrán en cuenta las mayúsculas y minúsculas. Por lo tanto, **MOV** y **mov** serán la misma palabra reservada pero **etiqueta1** y **Etiqueta1** serán dos etiquetas distintas.

Las constantes pueden tener los siguientes formatos:

- Hexadecimal: La constante deberá empezar por un número y acabar en **h** o **H**. Ej.: **0F5h**.
- Binario: La constante deberá acabar en **b** o **B**. Ej.: **1110b**.
- Constantes decimales: Resto de constantes formadas por números. Sólo pueden ser enteras. Ej.: **25**.
- Constantes de carácter: Serán un carácter encerrado entre comillas simples. Ej.: **'a'**.
- Cadenas: Serán una serie de caracteres encerrados entre comillas dobles. Ej.: **"hola"**.
- Pueden obtenerse constantes numéricas al aplicar los operadores **BYTEALTO** y **BYTEBAJO** a constantes de 16 bits.
- Puede aplicarse el operador **DIRECCION** a una etiqueta. El valor obtenido será una constante de 16 bits con la dirección a la que apunte la etiqueta. Lo habitual será utilizar esta directiva en conjunto con **BYTEALTO** y **BYTEBAJO**. Por ejemplo:

| |
|---------------------------------------|
| MOVL R0, BYTEBAJO DIRECCION etiqueta1 |
|---------------------------------------|

- Existen tres constantes predefinidas:
 - **@DATOS**: Es la dirección donde comienzan los datos.
 - **@CODIGO**: Es la dirección donde comienza el código.
 - **@PILA**: Es la dirección más baja perteneciente a la pila.

2.2 Estructura del fichero de entrada

El fichero de entrada estará formado por una serie de líneas que podrán contener directivas, instrucciones o ambas cosas. Se utiliza el retorno de carro como separador; por lo tanto, una instrucción no podrá extenderse por varias líneas.

Los ficheros de entrada al ensamblador tendrán la siguiente estructura:

| |
|-------------------------------------|
| Definición de constantes simbólicas |
| Definición del origen |
| Definición del inicio del programa |
| Definición de la pila |
| Definición de los datos |
| Definición del código |

Sólo será obligatoria la sección de definición de código. El orden de las secciones debe ser el indicado. A continuación se explica qué puede ir en cada sección.

2.2.1 Definición de constantes simbólicas

En esta sección se pueden definir constantes simbólicas con la siguiente sintaxis:

| |
|--|
| nombre_símbolo EQU valor_numérico |
|--|

Estas constantes se podrán utilizar en cualquier otro lugar donde pueda ir un valor numérico.

2.2.2 Definición del origen

En esta sección sólo puede ir la directiva **ORIGEN** seguida de una constante. Indica la dirección a partir de la cual se cargarán los datos y el código del programa (el ensamblador colocará primero los datos y luego el código). Si no aparece esta directiva, la dirección de origen será **100H**.

2.2.3 Definición del inicio del programa

En esta sección sólo puede ir la directiva **INICIO** seguida de una etiqueta. Indica la dirección de la primera instrucción que se ejecutará del programa. Si no aparece esta directiva, la dirección de inicio del programa es la de la primera instrucción del código.

2.2.4 Definición de la pila

En esta sección sólo puede aparecer la directiva **.PILA** seguida opcionalmente de una constante. La constante indica el tamaño de la pila. Si no aparece esta directiva o no aparece la constante, el tamaño de la pila será de 256 palabras¹.

2.2.5 Definición de los datos

Esta sección debe comenzar por la directiva **.DATOS**. A continuación en la misma línea puede llevar una constante que indicará el número de palabras reservadas para los datos del programa.

¹ En este documento el término una palabra se utilizará como sinónimo de 2 bytes.

En líneas sucesivas se pueden definir valores para estos datos. Para ello se utiliza la siguiente sintaxis:

```
[etiqueta] VALOR constante1[, constante2, constante3...]
```

Es decir, puede haber una etiqueta que permitirá acceder a la dirección del primer dato de la lista de datos que seguirá a la directiva **VALOR**. Estos datos deben ser constantes. Nótese que aplicando el operador **DIRECCION** a una etiqueta se obtiene una constante.

Se pueden definir *arrays* mediante la directiva **VECES**:

```
[etiqueta] VALOR num_veces VECES constante
```

Por ejemplo:

```
lista VALOR 9 VECES 3
```

La línea anterior define una lista de 9 elementos, todos ellos con el valor 3.

2.2.6 Definición del código

La sección de código debe comenzar con la directiva **.CODIGO**, que puede ir seguida de una constante que indicará el tamaño en palabras del código. A continuación debe ir una serie de instrucciones con la siguiente sintaxis:

```
[etiqueta:] mnemónico operandos
```

Los mnemónicos pueden ser cualquiera de los del juego de instrucciones de la CPU teórica. Dentro del código también se puede hacer una definición de datos, lo que puede utilizarse para codificar directamente instrucciones, por ejemplo, utilizando el código de una macro que el usuario haya creado en el Simulador Von Neumann y que, por lo tanto, no tenga mnemónico reconocido en el ensamblador.

Se puede incluir la definición de procedimientos en esta sección. Los procedimientos deben comenzar por la directiva **PROCEDIMIENTO** seguida del nombre del procedimiento y deben acabar con la directiva **FINP**.

Esta sección debe acabar con la directiva **FIN**. Todo lo que haya tras esta directiva será ignorado.

2.3 Fichero de salida

El ensamblador generará un fichero de salida con la siguiente estructura:

- Un número que indicará a partir de qué dirección se cargarán los datos.
- Un número que indicará con qué valor se cargará inicialmente el PC.
- Un número que indicará el valor inicial de R7 (puntero de pila). El ensamblador calculará este número sumándole a la dirección donde termina el código el tamaño de la pila, ya que se supone que la pila está situada en memoria después del código.
- Una lista de números con la codificación de los datos. Si se ha reservado espacio para más datos de los que se han definido, se rellenará con ceros.
- Una lista de números con la codificación de las instrucciones.

3 Uso de la herramienta

La herramienta funciona en modo consola (bajo una interfaz de comandos). La sintaxis de invocación es la siguiente:

```
ensambla nombre_fich
```

El fichero de salida tendrá el mismo nombre que el fichero de entrada pero cambiando la extensión a **.aje**. Si el fichero de entrada no tenía extensión, se añadirá al fichero de salida el **.aje**.

Si el ensamblado se pudo llevar a cabo sin problemas, la herramienta no generará ningún mensaje. Mostrará mensajes de aviso o de error (según la gravedad) cuando encuentre problemas en el fichero de entrada, indicando la línea en la que los detectó.