Ensamblador del Computador Teórico

1. Introducción

En este documento se describen brevemente los elementos del lenguaje ensamblador del Computador Teórico.

Un programa en lenguaje ensamblador está formado por una secuencia de sentencias. Cada sentencia ocupa una sola línea y tiene la siguiente estructura:

```
[etiqueta] [operacion] [operandos] [;comentarios]
```

Los cuatro campos de una sentencia son opcionales, si no aparece ninguno de ellos (una línea en blanco) se tendría una sentencia vacía.

Las sentencias se dividen en dos tipos:

- **Instrucciones.** Estas sentencias representan órdenes al procesador y tras el proceso de compilación generan código ejecutable.
- **Directivas.** Estas sentencias dirigen el proceso de compilación o construcción del programa ejecutable. No generan código ejecutable. Normalmente se utilizan para aumentar la legibilidad del código fuente.

El fichero creado con todas las sentencias que constituyen un programa se denomina fichero fuente. Este tipo de fichero tiene formato ASCII. Se recomienda que el fichero de código fuente tenga por extensión `.ens`.

El fichero fuente se compila utilizando el programa ensamblador, que generará un nuevo fichero, denominado fichero ejecutable, de igual nombre y de extensión `.eje`. Este fichero tiene formato ASCII y contiene el código máquina de las instrucciones que componen el programa, así como cierta información adicional para realizar la carga del programa en el simulador del Computador Teórico.

2. Elementos del lenguaje

En este apartado se repasan brevemente los elementos que forman el lenguaje ensamblador del Computador Teórico. A partir de ellos, siguiendo la sintaxis adecuada, se construirán las sentencias que forman los programas.

2.1. Comentarios

Todo lo que siga a un punto y coma (`;`) hasta el final de la línea se considera un comentario y será ignorado por el compilador.

2.2. Palabras reservadas

Las palabras que se muestran en la tabla 1 son palabras reservadas del lenguaje y no podrán usarse como identificadores de etiquetas.

Las palabras reservadas pueden estar tanto en mayúsculas como en minúsculas.

.CODIGO	.DATOS	.PILA	ADD	AND	BRC	BRNC
BRNO	BRNS	BRNZ	BRO	BRS	BRZ	BYTEALTO
BYTEBAJO	CALL	CLI	CODIGO	CMP	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

Tabla 1: Palabras reservadas del lenguaje ensamblador de la CPU elemental

2.3. Identificadores

Se pueden definir nombres simbólicos o identificadores dentro de un programa, que reciben el nombre genérico de etiquetas. Dependiendo de la zona de programa donde se definan, su significado será diferente. Así se tendrán:

- Si el nombre simbólico aparece en las primeras líneas del programa, antes de cualquier otra directiva, será una constante simbólica. Posteriormente, cuando aparezca en el código la **constante simbólica**, será reemplazada por el valor asociado. Un ejemplo será:

  ```
  escala EQU 8
  ```

  Siempre que aparezca en el código fuente el nombre `escala`, será equivalente a colocar el valor 8.

- Si el identificador aparece en la parte de definición de datos, recibe el nombre de **etiqueta de datos** y marca la dirección de memoria en la que reside un dato. Su significado es equivalente a un nombre de variable. Un ejemplo:

  ```
  dato1 VALOR 8
  ```

  Cuando aparezca en el código fuente el identificador `dato1`, se estará indicando la dirección de memoria en la que está almacenado el valor 8.

- Si el identificador aparece en la parte de código, recibe el nombre de **etiqueta de código**. En este caso el identificador marcará la dirección en la que está el código de la instrucción. Un ejemplo:

```
                JMP   destino
                ...
                ...
destino:        ADD   R2, R3, R4
```

  Cuando aparece en el código seguida del carácter «dos puntos» (`:`), la etiqueta `destino` hace referencia a la dirección de memoria donde está el código de la instrucción. Desde otro punto del código se puede hacer referencia a la instrucción marcada, empleando únicamente el nombre de la etiqueta de código.

Un identificador o nombre simbólico válido estará formado por cualquier secuencia de caracteres o dígitos siempre que el primero de ellos sea un carácter alfabético. A diferencia de las palabras reservadas del lenguaje, en el caso de los identificadores el compilador distinguirá entre las mayúsculas y las minúsculas. Por ejemplo, el identificador `Etiqueta1` será distinto del identificador `etiqueta1`.

2.4. Constantes

Las constantes empleadas pueden ser de los siguientes tipos:

- Decimal: cualquier número natural o entero formado con dígitos decimales dentro de los siguientes rangos: naturales [0 a 65 535], enteros de [−32 768 a 32 767].
- Hexadecimal: es un valor constante formado por hasta 4 dígitos hexadecimales y terminado con la letra `h` o `H`. La constante ha de empezar obligatoriamente por número, en caso de que el primer dígito hexadecimal sea una letra se antepone un 0.
- Binario: La constante será una secuencia de hasta 16 dígitos 0 ó 1 terminada con la letra `b` o `B`.
- Caracteres: será un carácter ASCII encerrado entre comillas simples. Ejemplo: `'a'`.
- Cadenas de caracteres: están formadas por una serie de caracteres ASCII encerrados entre comillas dobles. Ejemplo: `"Cadena"`.

2.5. Operadores

Existe una serie de operadores que se pueden aplicar a constantes e identificadores. El resultado que devuelve un operador será siempre un valor numérico entero. Los operadores se muestran en la tabla 2.

Es posible encadenar el uso de los operadores. Así por ejemplo, será posible escribir la siguiente instrucción:

```
MOVL R0, BYTEBAJO DIRECCION etiqueta1
```

Operador	Resultado
`BYTEALTO`	Aplicado sobre una cantidad de 16 bits, devuelve el valor del byte más significativo.
`BYTEBAJO`	Aplicado sobre una cantidad de 16 bits, devuelve el valor del byte menos significativo.
`DIRECCION`	Aplicado sobre una etiqueta o un identificador, devuelve como resultado la dirección de la posición de memoria que marca esa etiqueta.

Tabla 2: Operadores del lenguaje ensamblador del Computador Teórico

2.6. Instrucciones

El conjunto de instrucciones que se puede emplear en el lenguaje ensamblador de la CPU elemental está recogido en el documento *Juego de instrucciones del Computador Teórico*.

2.7. Directivas

Las directivas sirven para definir las secciones del programa y realizar la definición de datos.

A continuación se listan las directivas que se pueden usar en un programa escrito en lenguaje ensamblador de la CPU elemental, así como su significado. Existen cuatro grupos de directivas:

- Directivas orientadas a dirigir la carga del programa en la memoria del simulador del Computador Teórico. Se muestran en la tabla 3.
- Directivas de definición de datos, que se utilizan para declarar etiquetas y crear datos con los que puede trabajar el programa. Se muestran en la tabla 4.
- Directivas para la definición de las secciones del programa ensamblador con las que se definen las distintas partes en las que se divide un programa en lenguaje ensamblador. Se muestran en la tabla 5.
- Directivas de definición de procedimientos. Se muestran en la tabla 6.

3. Estructura de un fichero en lenguaje ensamblador

Los ficheros de código fuente escritos en lenguaje ensamblador se organizan en líneas. Cada una de las líneas del fichero puede contener una directiva, una instrucción o ambas cosas a la vez en los casos en que sea posible. El carácter separador de líneas es el retorno de carro, por lo que una instrucción no podrá ocupar más de una línea en el fichero fuente.

Todos los ficheros fuente tienen que adecuarse a una estructura fija dividida en secciones. La estructura a seguir se muestra a continuación:

Directiva	Significado
`ORIGEN [dirección]`	Indica la dirección de la memoria a partir de la cual se cargará el programa (primero los datos, luego el código y por último la pila). Si no se indica un valor para `dirección` se cargará por defecto a partir de la dirección `100h`.
`INICIO [etiqueta]`	`etiqueta` hace referencia a la primera instrucción del código del programa que se ejecutará. La dirección de memoria correspondiente a esta instrucción, será el valor con el que se inicialice el registro `PC`, una vez cargado el programa en memoria. Si esta directiva no aparece en un programa, se considerará la primera instrucción de código del fichero fuente como la primera instrucción a ejecutar.

Tabla 3: Directivas de carga del programa

Directiva	Significado
`Etiqueta EQU num`	Define `Etiqueta` como una constante numérica cuyo valor es `num`.
`Etiqueta VALOR num`	Esta directiva asigna el valor `num` a una posición de memoria dentro de la zona de datos del programa. El nombre `Etiqueta` se puede utilizar para acceder a la posición en la que se encuentra el dato. Esta directiva también se puede utilizar para definir una lista de datos consecutivos en memoria. En ese caso `Etiqueta` hace referencia al primero de ellos
`Eti VALOR n VECES d`	Repite la acción de `VALOR` el número de veces indicado por `n`. Se usa para definir vectores. El resultado es similar a repetir `n` veces la sentencia `Eti VALOR d`. Si está presente, `Eti` corresponderá a la dirección del primero de los datos.

Tabla 4: Directivas de Definición de Datos

Directiva	Significado
`.PILA [tamaño]`	Esta directiva indica el tamaño en palabras que va tener la pila. El tamaño será el que indique el parámetro `tamaño`. Si en un programa no aparece esta directiva, o se omite el valor de `tamaño`, la pila tendrá un tamaño de 256 palabras. Las posiciones de memoria reservadas para la pila se ubican a continuación de la sección de código.
`.DATOS`	Define el inicio de la sección del programa en la que se definen los datos del programa. En la sección de datos se reservan tantas posiciones como ocupen los datos definidos.
`.CODIGO`	Define el inicio de la sección del programa en la que se encuentra el código. En la sección de código se reservan tantas posiciones como ocupe el código.
`FIN`	Marca el final del programa. Todo lo que aparezca después de esta directiva será ignorado.

Tabla 5: Directivas de definición de secciones

Directiva	Significado
`PROCEDIMIENTO nombre`	Directiva que define el comienzo de un trozo de código correspondiente a un procedimiento. La etiqueta `nombre` hace referencia a la dirección de la primera instrucción del código del procedimiento.
`FINP`	Marca el fin del bloque de código correspondiente a un procedimiento.

Tabla 6: Directivas de definición de procedimientos

```
escala equ 1000 ; Definición de constantes simbólicas
origen 7F40h    ; Definición del origen de carga del programa
inicio ini      ; Definición de la etiqueta que marca la
                ; primera instrucción a ejecutar del programa

.pila  100h     ; Definición de la pila
.datos          ; Definición de los datos del programa
  dato1  valor 12h
  .....

.codigo         ; Definición del código del programa

ini:
  mov r5, r4
  .....
fin
```

De todas las secciones posibles que pueden formar parte de una programa en lenguaje ensamblador, sólo será obligatoria la sección de definición del código del programa. Todas las demás secciones serán opcionales pudiendo estar presentes o no. Si aparece alguna de las secciones del esquema, tendrá que estar colocada en el orden indicado.

4. Estructura de un programa en memoria

Cuando un programa se carga en memoria se divide en tres secciones: datos, código y pila. Estas tres secciones se cargan de manera consecutiva a partir de la dirección indicada en la directiva `ORIGEN`, tal y como se muestra en la figura 1. Después de cargar el programa en memoria se inicializan los registros para comenzar su ejecución. El registro `PC` se carga con la dirección de la primera instrucción a ejecutar, indicada por la directiva `INICIO`. Además, el registro `R7` se carga con la dirección que se encuentra justo por debajo de la sección de pila, al estar ésta vacía inicialmente. El tamaño de la pila vendrá definido por el número de posiciones reservadas mediante la directiva `.PILA`.

5. Estructura de un fichero ejecutable

El compilador de lenguaje ensamblador genera, a partir de los ficheros de código fuente, ficheros ejecutables (extensión `.eje`). Los ficheros ejecutables son ficheros de texto ASCII con la siguiente estructura:

- Cuatro dígitos hexadecimales que indican la dirección de la memoria a partir de la cual se cargará el programa, comenzando por la sección de datos. Corresponde al valor asociado a la directiva `ORIGEN`, o `0100h` si se omitió dicha directiva.

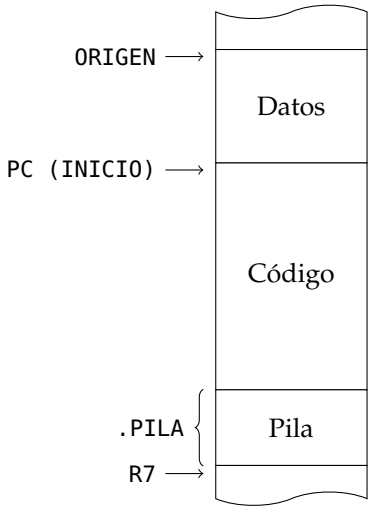

Figura 1: Estructura de un programa en memoria

- Cuatro dígitos hexadecimales con el valor inicial del registro `PC`. Este valor corresponde a la dirección de memoria en la que se encuentra el código de la instrucción indicada por la etiqueta asociada a la directiva `INICIO`.

  Si se omite la etiqueta de la directiva `INICIO`, el registro `PC` se inicializará con la dirección de la primera instrucción del código.

- Cuatro dígitos hexadecimales con el valor inicial que tomará el registro `R7` (puntero de pila). El ensamblador calcula este número sumándole a la dirección de memoria en la que se carga la última instrucción de código el valor suministrado para el tamaño de la pila, ya que supone que la pila se encuentra a continuación del código.

- Una lista de dígitos hexadecimales con la codificación de los datos. Si se ha reservado espacio para más datos de los que se han inicializado, se rellenan con el valor cero.

- Una lista de dígitos hexadecimales con la codificación de las instrucciones. Si se ha reservado espacio para más instrucciones de las que se han utilizado, se rellenan con el valor cero.