ENSAMBLADOR PARA EL SIMULADOR DE CPU v1.1

Un ensamblador es una utilidad que traduce programas escritos para una CPU, en lenguaje ensamblador, a código máquina.

Permite escribir programas usando los mnemotécnicos de las instrucciones de la CPU, y evita que el programador tenga que realizar la traducción de cada instrucción a código máquina.

Un ejemplo de programa sencillo, escrito en lenguaje ensamblador, apto para ser traducido a código máquina con el ensamblador, es el siguiente:

```
;Programa muy sencillo que suma dos números
;Los números se leen de memoria, en las posiciones N1 y N2
;El resultado de la suma se guarda en la posicion RES

        org 0x0000
        ent comienzo

n1:     dw 37   ;el primer número.
n2:     dw 27   ;el segundo número.
res:    ds 1    ;aquí estará el resultado.

comienzo:       movh r0,hi(n1)
                movl r0,lo(n1)  ;R0=posición de N1
                movh r1,hi(n2)
                movl r1,lo(n2)  ;R1=posición de N2
                mov r2,[r0]     ;R2=primer número
                mov r3,[r1]     ;R3=segundo número
                add r4,r2,r3    ;R4=N1+N2
                movh r5,hi(res)
                movl r5,lo(res) ;R5=posición de RES
                mov [r5],r4     ;RES=R4
```

Un ensamblador maneja números y etiquetas.

Un número puede estar en decimal o hexadecimal. Los números en decimal se ponen tal cual. Los números hexadecimales van precedidos de “0x”, al igual que en el lenguaje C.

Las etiquetas sirven para dar un nombre a una dirección de memoria. Por ejemplo, “n1” es una etiqueta que designa a la posición de memoria donde hay guardado un 38, y según veremos, dicha posición de memoria es la 0000. La etiqueta “comienzo” contiene el valor de la posición de memoria donde comienza el código. Las etiquetas se definen como se ve en el ejemplo; se pone el nombre de la etiqueta, seguido del símbolo de los dos puntos “:” . Una vez que la etiqueta existe, puede usarse allí donde sea necesario. Son muy útiles en las instrucciones de salto, ya que podemos usar una etiqueta como destino de un salto condicional o incondicional, y el ensamblador se encarga de calcular el valor de desplazamiento que le corresponde.

Un programa en ensamblador puede tener comentarios. Todo texto que comience con el símbolo del punto y coma “;” será tratado como un comentario hasta el final de la línea, y será ignorado en la traducción del código.

En este programa hay instrucciones que no parecen pertenecer a la CPU del simulador, como son: ORG, ENT, DW, DS. Estas “instrucciones” son en realidad pseudo-instrucciones, o directivas del ensamblador. Son instrucciones que le dicen al ensamblador cómo se debe traducir el programa, o cómo manejar la memoria del programa.

- ORG: va seguido de un número. Este número indica dónde comienza en memoria el programa que viene a continuación. En el fichero “memoria.txt”, es el primer valor que aparece. ORG debe ser la primera instrucción que exista en un programa en ensamblador para nuestro simulador.
- ENT: va seguido de un número o una etiqueta, e indica en qué dirección de memoria comienza a ejecutarse el código. ENT es la segunda instrucción que debe aparecer en un programa en ensamblador.
- DW: es la abreviatura de “Define Word”. Va seguido de un número. Simplemente, indica que en la posición de memoria actual se guardará dicho número.
- DS: es la abreviatura de “Define Space”. Va seguido de un número N. Define un bloque de N “huecos” o posiciones de memoria, y las inicializa a 0. Es equivalente a escribir N veces la instrucción DW 0.
- HI(x): es una pequeña función que devuelve los 8 bits altos del argumento que se le pase en x.
- LO(x): lo mismo que la anterior, pero devuelve los 8 bits bajos.

Estas dos últimas funciones se usan casi siempre en las instrucciones MOVH y MOVL. Por ejemplo, en el código propuesto, las dos primeras instrucciones cargan el registro R0 con la dirección de memoria del número N1.

Una vez traducido por el ensamblador, se obtiene un fichero "memoria.txt" con este contenido (en el fichero, cada número está en una línea por separado. Aquí lo reproducimos de izquierda a derecha y de arriba a abajo):

```
0000  0003  0025  001B
0000  2800  2000  2900
2101  0A00  0B20  4113
2D00  2502  1580  0000
```

El primer número, 0000, es el valor que había en ORG.

El segundo número, 0003 es el que corresponde a la etiqueta "comienzo". Si contamos desde 0 (el valor de ORG), veremos que el primer número ocupa la posición 0000, el segundo, la posición 0001, el resultado la posición 0002, así que "comienzo" que acompaña a la primera instrucción, está en la posición 0003.

El tercer y cuarto números son 0025 y 001B. Si traducimos esos valores a decimal obtenemos 37 y 27, que son los valores puestos ahí mediante sendas instrucciones DW.

El quinto número, 0000, es el "hueco" que crea en memoria la instrucción DS. Como en el programa hemos pedido un hueco, pues éste es el que deja. Nótese que podríamos haber conseguido el mismo efecto si en lugar de usar DS 1 hubiéramos usado la instrucción DW 0. En el fichero "resultados.txt" que se genere tras la ejecución del programa por parte del simulador, en esta posición tendremos el resultado de la suma de los dos números dados.

A partir del sexto número comienza la parte de código de nuestro programa.

Modo de uso

El ensamblador es un programa muy sencillo, hecho con C-Free. Se suministra el código fuente para quien quiera ver cómo está hecho. Para usar el ensamblador no es necesario tener instalado C-Free ni tener accesible el código fuente. Tan sólo es necesario el fichero ejecutable "ensamblador.exe".

Primero, con el bloc de notas (no uséis el Word para esto) escribid el programa ensamblador que quereis ensamblar ("ensamblar" es el acto de traducir un programa ensamblador a código máquina). Grabadlo con un nombre que no tenga espacios, eñes o acentos. Es importante que no llaméis a este fichero "memoria.txt" ya que ese será el fichero que produzca el ensamblador como resultado, y si vuestro programa ensamblador se llamase así, se borraría su contenido.

IMPORTANTE: tras la última línea de vuestro programa, dejad una línea en blanco *(es decir, pulsar **Intro** tras escribir la última línea)***.**

Supongamos que habéis llamado al fichero con vuestro programa ensamblador "prueba.txt". Copiáis dicho fichero a donde tengáis el ensamblador, y ejecutáis el ensamblador haciendo doble clic en "ensamblador.exe":

Preguntará el nombre del fichero ensamblador: se lo ponéis y pulsáis *Intro*. Si todo va bien, aparecerá un mensaje diciendo que el fichero "memoria.txt" se ha creado. Para salir del ensamblador, pulsar una tecla.

El fichero "memoria.txt" es el fichero que usaremos con el simulador de CPU. Contiene el código máquina del programa que hemos escrito, en el formato adecuado para el simulador.

Esto es lo que aparece en pantalla cuando ejecutamos el ensamblador con este programa de prueba:

Contador de posiciones: indica en qué dirección de memoria se alojará el código ensamblado en esa línea. Se establece con ORG

Dirección de memoria donde comienza la ejecución (valor inicial de PC).

Datos de 16 bits definidos usando dw o ds.

Código máquina que se corresponde con la instrucción ensamblada en cada línea.

```
Fichero fuente? prueba.txt
Ensamblando prueba.txt...
Primera pasada... OK
Segunda pasada.
    1  ;Programa muy sencillo que suma dos n·meros
    2  ;Los numeros se leen de memoria, en las posiciones N1 y N2
    3  ;El resultado de la suma se guarda en la posicion RES
    4
    5  0000              org 0x0000
    6         0003       ent comienzo
    7
    8  0000   0026  n1:  dw 38    ;el primer numero.
    9  0001   001A  n2:  dw 26    ;el segundo numero.
   10  0002   ....  res: ds 1     ;aqui estara el resultado.
   11
   12  0003   2800  comienzo:    movh r0,hi(n1)
   13  0004   2000               movl r0,lo(n1)   ;R0=posicion de N1
   14  0005   2900               movh r1,hi(n2)
   15  0006   2101               movl r1,lo(n2)   ;R1=posicion de N2
   16  0007   0A00               mov r2,[r0]      ;R2=primer numero
   17  0008   0B20               mov r3,[r1]      ;R3=segundo numero
   18  0009   4113               add r4,r2,r3     ;R4=N1+N2
   19  000A   2D00               movh r5,hi(res)
   20  000B   2502               movl r5,lo(res)  ;R5=posicion de RES
   21  000C   1580               mov [r5],r4      ;RES=R4
   22
Fichero memoria.txt creado.

Programa ENSAMBLADOR terminado. Pulse una tecla para salir.
```

Si cometemos algún error, por ejemplo, escribiendo mal una instrucción, veremos algo como esto (aquí se ha escrito mal la instrucción ADD de la línea 18):

```
   ...
   ...
   16  0007  0A00               mov r2,[r0]      ;R2=primer numero
   17  0008  0B20               mov r3,[r1]      ;R3=segundo numero
   18  ERROR en linea 18. No se reconoce la instruccion.

Programa ENSAMBLADOR terminado. Pulse una tecla para salir.
```