MANUAL DE USUARIO DEL SIMULADOR SIMPLEZ

Tecnicas Digitales II

Mercedes Garijo

Gregorio Fernández

Dep. Ingeniería de Sistemas Telemáticos

1. INTRODUCCIÓN

1.1. Objetivos

El objetivo de este manual es aprender paso a paso el uso del simulador de Símplez , ordenador

ficticio que sigue el modelo de la "máquina de von Neumann" y que ha sido diseñado teóricamente en el libro de texto de esta asignatura, "Conceptos básicos de Arquitectura y Sistemas Operativos", de Gregorio Fernández. Este manual puede obtenerse en la Internet en `http://www.gsi.dit.upm.es/~mga/ffoo/manual.html`.

El uso de este entorno de simulación, desarrollado para ordenadores PC-compatibles, permite ejercitar la programación de Símplez , facilitando su comprensión. Aunque en el apéndice se incluye información básica para la programación (formatos y juegos de instrucciones de Símplez y microórdenes de Símplez) no se puede seguir este manual sin haber estudiado antes con detenimiento algunas de las lecciones del libro [Fernández, 98]. En concreto la 1 y la 2 antes de programar en Símplez y las lecciones 9, 10 y 11 para comprender el comportamiento de la ruta de datos de Símplez.

1.2. Componentes del entorno de simulación

El simulador Simplez (`simplez.exe`) posee tres componentes básicos: un editor, un ensamblador y un ejecutor. El editor está adaptado al repertorio de instrucciones de la máquina correspondiente y facilita la escritura de programas en ensamblador, aunque podría utilizarse también cualquier otro editor de texto. El programa ensamblador primero examina el programa fuente y, si es correcto, lo traduce a lenguaje de máquina y, si no, indica sus errores. El programa ejecutor simula la ejecución de los programas en diferentes modalidades: de principio a fin, paso a paso, o hasta un punto marcado. Durante la ejecución se pueden ver y modificar los contenidos de la memoria y de los registros. El simulador de Símplez proporciona además una visión gráfica de la ejecución en el nivel de micromáquina mostrando, para cada ciclo de reloj, los estados de la ruta de datos durante la ejecución de instrucción.

1.3. Ensamblado y ejecución de programas

Para ejecutar en este entorno de simulación un programa fuente (escrito en el lenguaje ensamblador del IEEE) es necesario seguir una secuencia de actividades, descritas gráficamente en la figura 1, con los pasos siguientes:

> **Paso (1)** Se abre una ventana MS-DOS, se pasa al directorio en el que está instalado el simulador, y se carga éste en memoria, escribiendo su nombre (`simplez`). Además, con el editor del simulador (o con cualquier otro editor de texto) se escribe el **programa fuente** (programa en lenguaje ensamblador) y se almacena en el disco. Por convenio, los programas fuente para Símplez se almacenan en ficheros con extensión `".SIM"` .
>
> **Paso (2)** Usando el ensamblador (que forma parte del simulador) se traduce el programa fuente a **código objeto** (programa en lenguaje máquina) que queda guardado en el disco.
>
> **Paso (3)** Se solicita la ejecución simulada del programa, durante la cual el simulador carga en memoria el código objeto y lo ejecuta como si fuese Símplez , es decir, funciona como una "máquina virtual".

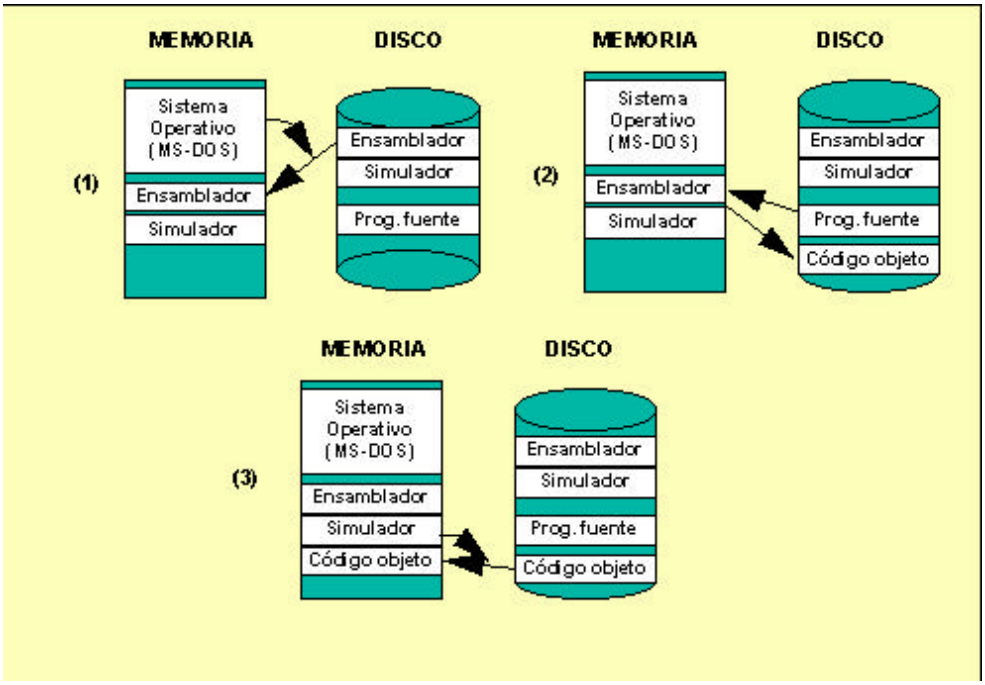

Figura 1. Pasos para la ejecución de un programa.

1.4. Limitaciones del entorno

Este entorno ha sido desarrollado para una versión anterior de los ordenadores Símplez . Aunque las diferencias no son esenciales deben ser tenidas en cuenta.

Las más importantes son:

- No existe el biestable del registro de estado que diferencia entre modo usuario y supervisor (y tampoco la interrupción por violación del modo usuario ni las instrucciones privilegiadas).
- Algunos códigos de operación binarios se han reorganizado (pero no tiene mayor importancia porque trabajaremos esencialmente en el nivel de ensamblador). El repertorio de esta versión y los códigos binarios aparecen en las tablas 10-13 del Apéndice.
- El ensamblador no admite módulos, ni las directivas IMPORT y EXPORT
- El ensamblador no es reubicante
- La rutina de consulta de interrupciones comienza en una dirección predeterminada (D'266) como puede verse en el Programa 6 de este manual.

1.5. Bibliografía

[Fernández, 98] *Conceptos básicos de arquitectura y sistemas operativos. Curso de Ordenadores*. (2ª edición). Ed. Syserso. Madrid 1998.

2. PROGRAMACIÓN DE SIMPLEZ

2.1. Ejemplo de programación con ensamblado y ejecución

Para mostrar el funcionamiento del entorno de simulación de Símplez, usaremos como ejemplo el Programa 1 (Programa 2.1, página 59 de [Fernández, 98]) que suma los dos números contenidos de las direcciones 10 y 11 (en decimal) y lleva el resultado a la dirección 12. Suponemos que este programa ya está escrito y guardado en un fichero de nombre SUMADOS.SIM en el mismo directorio que el simulador de Símplez (SIMPLEZ.EXE).

```
LD   /10   ;carga el primer sumando en el acumulador
ADD  /11   ;suma al acumulador el segundo sumando
ST   /12   ;guarda el resultado en la palabra de
dirección 12
HALT       ;detiene la ejecución
```

Programa 1. SUMADOS.SIM

2.1.1. Arranque

Desde el sistema operativo MS-DOS, y en el directorio en el que está instalado el simulador, teclear `simplez` y `<RC` . Aparecerá la pantalla de la interfaz dividida en tres áreas: **`EJECUCION`**, **`MEMORIA`** y `Registros`, y un menú en su parte inferior:

`E``ditar/Ensamblar` **`O`**`tro programa` **`S`**`alir`

Superpuesta, aparece una ventana de bienvenida que al teclear `<RC` cambia a una petición de nombre del fichero fuente, tal como se ve en la figura 2.

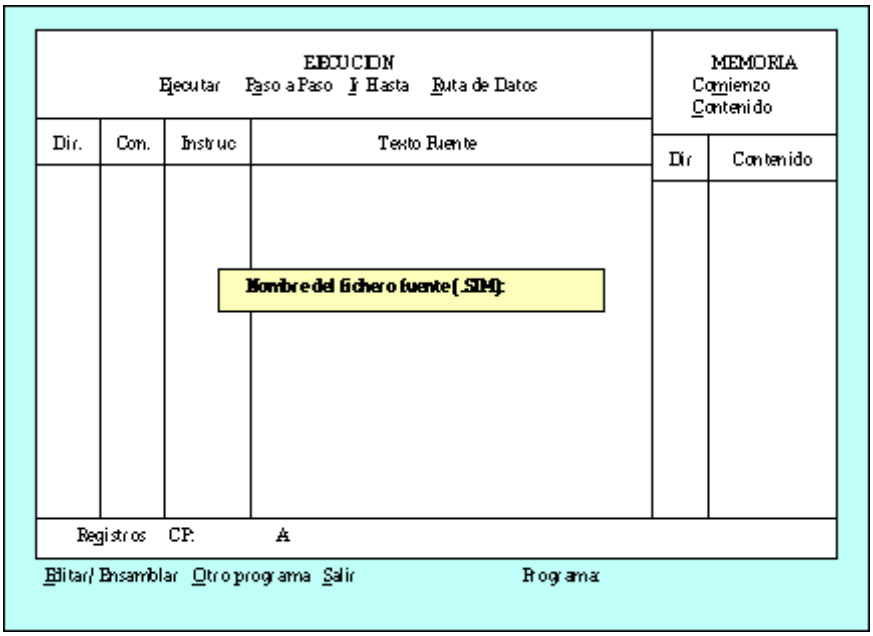

Figura 2. Primera pantalla del simulador.

Después de teclear `sumados <RC`, el entorno comprueba si el programa ya ha sido ensamblado y existe un fichero objeto, `sumados.sim`. Si no es así, propone seleccionar la opción Editar/Ensamblar para generarlo. El programa objeto se carga en la memoria simulada de Símplez, a partir de la posición "[0]". La figura 3 muestra la situación del entorno.

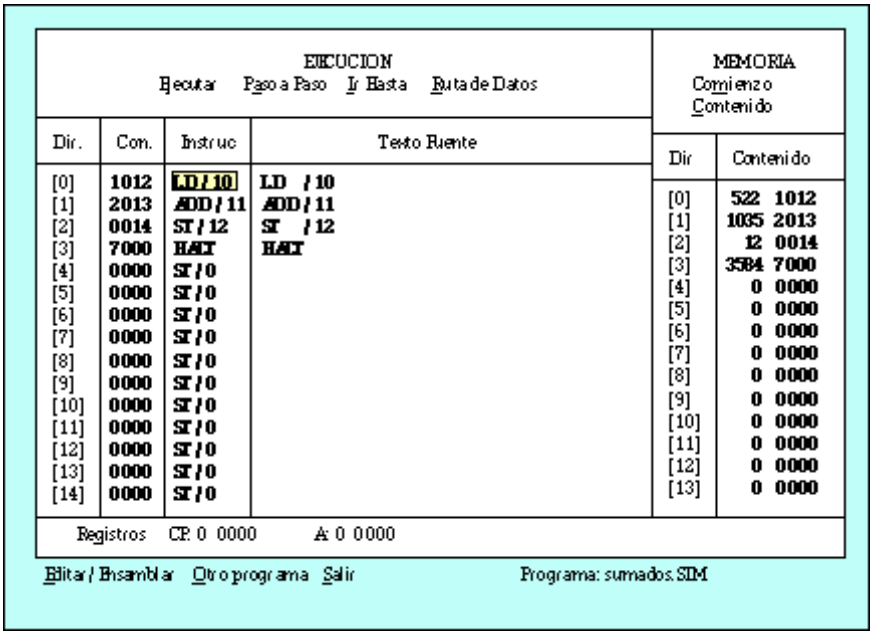

Figura 3. Programa "SUMADOS.SIM".

En el área **`EJECUCIÓN`** hay un menú y cuatro columnas. El menú indica las cuatro formas posibles de realizar la ejecución de un programa en Símplez:

E**j**ecutar P**a**so a Paso **I**r Hasta **R**uta de Datos

es decir instrucción por instrucción, fragmentos de programa, el programa completo o por ciclos de reloj, respectivamente. Como en el resto de menús del entorno, la elección de una opción se hace tecleando la letra resaltada (indistintamente en mayúscula o minúscula).

Las columnas contienen la siguiente información: **Dir.** muestra, en decimal y entre corchetes, las direcciones de memoria donde se encuentra el programa a ejecutar; **Con.** indica el contenido de cada posición de memoria en sistema octal; **Instruc.** da el contenido de dicha posición interpretado como una instrucción; y **Texto Fuente** muestra el programa original en lenguaje ensamblador. La próxima instrucción a ejecutar (`LD /10`) aparece resaltada en vídeo inverso.

En el área **MEMORIA**, hay también un menú con dos opciones y dos columnas: **Dir.** que muestra las direcciones de memoria y **Contenido** que lo muestra en decimal y octal. La dirección que aparece en vídeo inverso es la seleccionada. Inicialmente es la [0], pero puede cambiarse a otra usando las teclas de flechas arriba/abajo del cursor o eligiendo la opción Co**m**ienzo del menú e indicando directamente el valor de dirección que se quiere seleccionar. La otra opción del menú, **C**ontenido, permite modificar el contenido de la palabra de memoria almacenada en la dirección seleccionada.

En el área Re**g**istrosse visualizan los contenidos de los registros, que en Símplez sólo son dos, el contador de programa (CP), que contiene, en decimal y octal, la dirección de la próxima instrucción a ejecutar, y el acumulador (A). Inicialmente el contenido de ambos es 0.

2.1.2. Introducción de datos

Para probar el funcionamiento del programa hemos de introducir dos valores en las posiciones 10 y 11, ejecutar el programa y ver que en la posición 12 aparece la suma de estos valores.

Para introducir en memoria estos valores, seleccionamos la dirección 10 con la flecha abajo (la dirección seleccionada aparece resaltada en brillo en la columna **Dir.**del área de memoria). Luego, se pulsa **C** (opción **C**ontenidodel menú de memoria) y, cuando el simulador solicita el nuevo contenido, se teclea el valor a introducir seguido de <RC . El valor introducido aparece en la columna **Contenido**, en decimal y octal. Se repite la operación para la dirección 11.

2.1.3. Ejecución

Para ver cuál es el resultado de ejecutar cada una de las instrucciones seleccionamos repetidamente la opción P**a**so a Pasodel menú de **EJECUCION**, hasta que se ejecute la instrucción HALT, que hace que el programa se pare.

Al ejecutar cada instrucción (la que aparece en vídeo inverso), es interesante observar como van cambiando los contenidos de CP y A y como, al ejecutar la instrucción `ST /12`, se modifica el contenido de la dirección [12]. Si se han introducido en las posiciones 10 y 11 los valores 1 y 2, respectivamente, tras terminar la ejecución quedará en pantalla lo que muestra la figura 4.

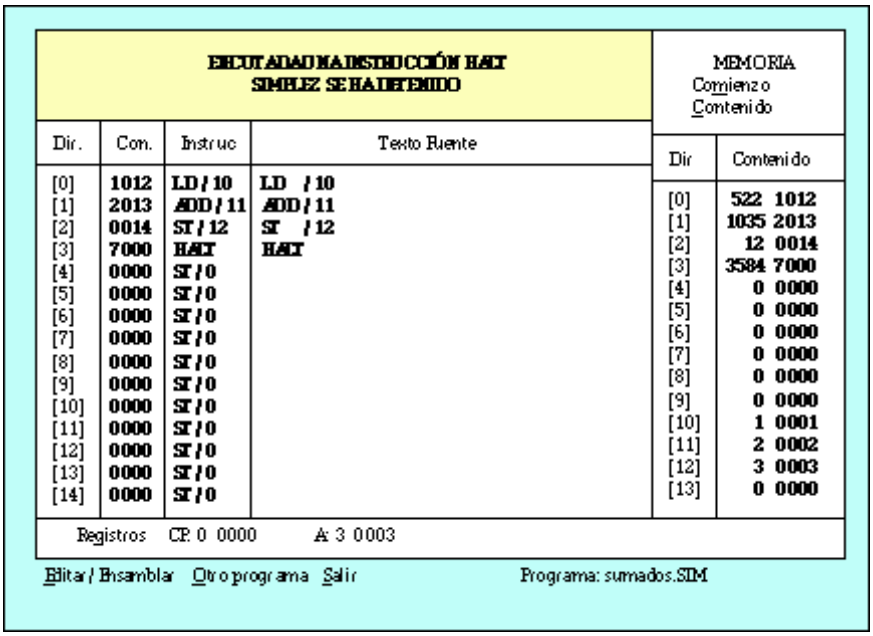

Figura 4.Final de la ejecución de "SUMADOS.SIM".

Una vez ejecutada `HALT`, se puede elegir del menú general, la opción **`O`**`tro programa` para reiniciar la simulación, indicando el nombre de un programa fuente (el mismo de antes u otro), o se puede elegir **`S`**`alir` del entorno.

2.2. Ejemplo de programación con edición

Como segundo ejemplo y para ilustrar el uso del editor se ha seleccionado el Programa 2 que suma los 10 primeros términos de la sucesión de Fibonacci (Programa 2.2, página 61 de [Fernández, 98]).

2.2.1. Edición

Una vez que se ha entrado en el entorno de simulación de Símplez (pantalla de la figura 2), se responde a la petición de nombre de programa escribiendo `fibonacc`. El sistema presenta el mensaje:

Archivo nuevo. ¿Desea entrar al editor (s/n)?

pulsando `s` el simulador crea el fichero "FIBONACC.SIM", y muestra una pantalla con sólo un menú inferior con las opciones:

F1-IMPRIME F2-SALVA F3-ENSAMBLA ESC-SALIR FIBONACC.SIM Lin. 1 Col. 1

El usuario puede empezar a introducir por teclado el programa elegido (Programa 2) sirviéndose de las teclas especiales del editor, especificadas en la tabla 1.

Tecla	Funcionamiento

Tabulador	Sitúa el cursor en la posición del primer carácter no nulo de la línea previa, contando a partir de la posición actual del cursor.
RC	Termina la línea actual, crea una nueva y sitúa el cursor al inicio de la misma.
Inicio	Sitúa el cursor en la primera posición de la línea donde está situado.
Fin	Sitúa el cursor en el último carácter de la línea donde está situado.
Arriba	Sitúa el cursor al final de la línea precedente.
Abajo	Sitúa el cursor al final de la línea siguiente.
AvPag	Refresca la pantalla con la página siguiente.
RePag	Refresca la pantalla con la página anterior.
Izquierda	Mueve el cursor una posición hacia la izquierda en la línea donde se encuentra.
Derecha	Mueve el cursor una posición hacia la derecha en la línea donde se encuentra.
Supr	Borra la línea donde se encuentra el cursor.
Retroceso	Borra el carácter situado a la izquierda del cursor en la línea donde se encuentra.

Tabla 1. Teclas particulares del editor.

Una vez terminada la edición, se guarda el archivo en disco con F2 (es una buena costumbre salvar además varias veces durante la edición). El resultado debe ser un listado como el que se muestra en el Programa 2.

```
;*************************************************************
********
; ARCHIVO FIBONACC.SIM
; Este programa suma los diez primeros términos de la
sucesión de
; Fibonacci. Si t(n) es el término de posición n en la
sucesión, la
; definición matemática sería la siguiente:
; t(0)=0; t(1)=1; t(n)=t(n-1)+t(n-2), n1.
;*************************************************************
********

          BR     /PRINCI ;Se bifurca al comienzo del programa
```

```
UNO       DATA   1       ;En esta palabra guardamos el valor
1
OCHO      DATA   8       ;Y en esta el valor 8
CONT      RES    1       ;Contador que controla las
iteraciones del
                         ;bucle
PEN       RES    1       ;Penúltimo termino de la sucesión
ULT       RES    1       ;Ultimo termino de la sucesión
SIG       RES    1       ;Siguiente termino de la sucesion
SUM       RES    1       ;Suma de todos los terminos de la
sucesion

PRINCI    CLR
          ST     /PEN    ;Al comienzo PEN es t(0), hacemos
t(0)=0
          LD     /UNO
          ST     /ULT    ;ULT ahora es t(1), t(1)=1
          ST     /SUM    ;SUM=t(1)+t(0)=1,sumamos los dos
primeros
          LD     /OCHO
          ST     /CONT   ;Ya tenemos t(0) y t(1). Hemos de
sumar desde
                         ; t(2) a t(9), hay que hacer ocho
sumas.

BUCLE     LD     /PEN    ;Calculamos el siguiente termino de
la
          ADD    /ULT    ;sucesion, t(n)=t(n-1)+t(n-2)
          ST     /SIG    ;Guardamos t(n) en SIG
          LD     /SUM    ;Calculamos la suma, que sera la
suma
          ADD    /SIG    ;anterior mas el ultimo termino
calculado
          ST     /SUM    ;Guardamos la suma
          LD     /ULT    ;Actualizamos para proxima iteracion
          ST     /PEN    ;ULT pasa a ser PEN
          LD     /SIG
          ST     /ULT    ;SIG pasa a ser ULT
          LD     /CONT   ;Decrementamos el contador de
iteraciones
          DEC
          BZ     /FIN    ;Si hemos hecho las ocho sumas
acabamos
          ST     /CONT   ;Si no, guardamos el contador
          BR     /BUCLE  ;Sumamos el siguiente termino
FIN       HALT
          END
```

Programa 2. Suma de los 10 primeros términos de la sucesión de Fibonacci

Para escribir este programa en ensamblador se han usado etiquetas en vez de valores de direcciones. Para más explicaciones sobre el uso de las mismas puede consultarse el punto 4.5, página 114 de [Fernández, 98].

2.2.2. Ensamblado y depuración

La llamada al ensamblador se hace pulsando F3. El ensamblador analiza el texto línea a línea y va traduciendo a código máquina. Si se detectaran errores (por alguna errata al copiar el programa que afecte a las reglas del lenguaje), el proceso se detiene, y se muestran los errores detectados para corregirlos, informando del número de línea donde se ha producido y el tipo de error.

El proceso de corrección y ensamblado hay que repetirlo tantas veces como sea necesario, hasta que finalmente no se detecten errores, y aparezca una pantalla con el mensaje:

Ensamblado correcto. <ESC para volver a editar.

Pulsando <ESC volvemos al editor. Si volvemos a pulsarlo, se nos da la opción de salvar el archivo (siempre es conveniente hacerlo), y volver al simulador, a lo que responderemos afirmativamente tecleando S. Como ya tenemos el código traducido, nuestro programa se habrá cargado en memoria, tal como muestra la figura 5.

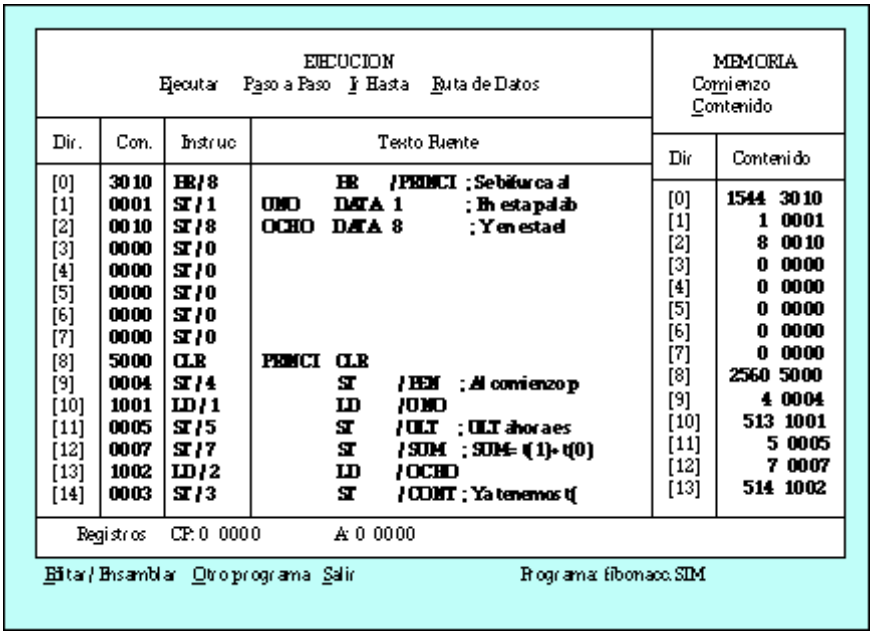

Figura 5. Ejecución de "FIBONACC.SIM".

Comparando el texto fuente con la instrucción correspondiente, se puede ver el valor que han tomado las etiquetas al traducirse. También puede observarse como las directivas y las pseudoinstrucciones "RES" desaparecen, ya que únicamente tienen vigencia durante el proceso de ensamblado.

2.2.3. Ejecución

Con la opción **P**aso a Pasoque ya conocemos, podemos seguir el programa. Como hay un bucle, puede ser útil la opción **I**r Hasta. Pulsando **I** el entorno pide introducir la dirección de la memoria donde queremos que el programa se pare (en decimal). Para comprobar los valores de "ULT", "PEN" y "SUM" en cada iteración, podemos indicar que queremos parar en la dirección 15, que corresponde a la etiqueta "BUCLE".

También se puede ejecutar el programa completo, sin parar, con la opción `Ejecutar`, hasta que se encuentre una instrucción `HALT`. Para comprobar que el programa termina de forma correcta miramos que el valor de `SUM` (posición de memoria 7) sea 88, número que corresponde a la suma de los diez primeros términos de la sucesión, como muestra la figura 6.

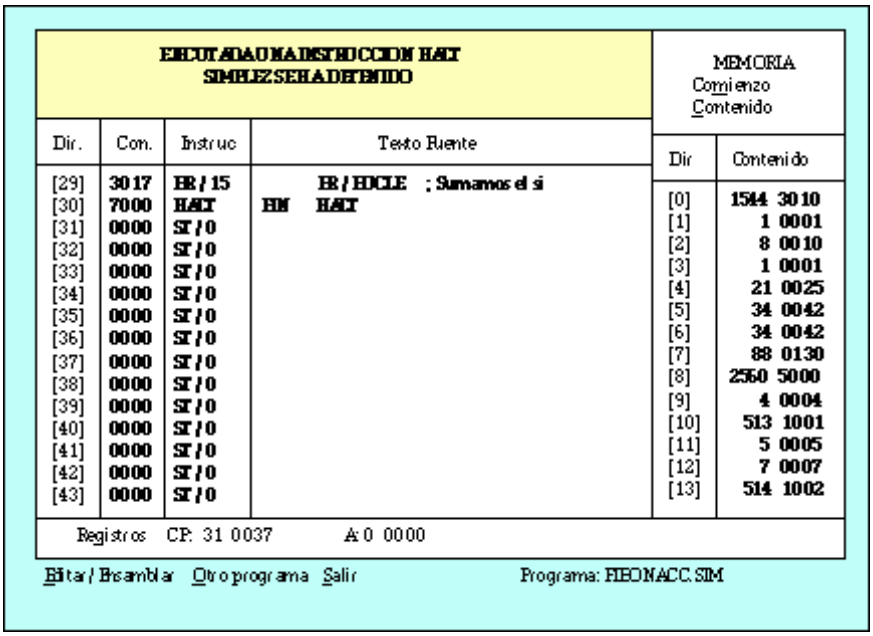

Figura 6. Fin de ejecución de "FIBONACC.SIM".

En cualquier momento de la ejecución con la opción `Editar/Ensamblar`se puede volver a entrar en el editor. Es útil para modificar sobre la marcha los errores encontrados en la ejecución, volver a compilar y repetir la ejecución.

También se permite modificar el contenido de los registros durante la ejecución del programa. Con la opción `Registros`,aparecerá una ventana en la que se pedirá el registro a modificar y el nuevo valor que se le quiere dar. Los registros posibles son `A` , con un rango de valores a introducir entre 0 y 4095, y `CP` cuyo rango va de 0 a 511. En caso de error, invita a reintentar la operación.

Esta opción es útil para ejecutar varias veces un mismo programa. Basta con parar justo en la instrucción `HALT` y en ese momento modificar el `CP` introduciendo el valor cero. También es útil para modificar el acumulador en tiempo de ejecución, probando diferentes valores para la operación que estemos realizando, sin necesidad de programar una instrucción `LD` para ello.

3. SIMULACION EN EL NIVEL DE MICROMÁQUINA.

3.1. Ejercitación de la Ruta de Datos

El simulador también dispone de la opción de ver la ejecución de las instrucciones en el nivel de

ruta de datos. Lo veremos con un ejemplo, el Programa 7, contenido en el fichero SUMADIEZ.SIM, que suma diez números previamente introducidos en memoria (tomado del programa 2.2 de [Fernández, 98]).

```
;*************************************************
*************
; ARCHIVO SUMADIEZ.SIM
; Este programa realiza la suma de 10 números que han
sido
; previamente almacenados en memoria entre las
posiciones 100 y 109.
;*************************************************
*************

          BR        /PRINCI   ;Se bifurca al comienzo
del programa.
DIEZ      DATA      10        ;En esta palabra guardamos
el valor 10.
CONT      RES       1          ;Contador de iteraciones
de bucle.
DIREC     EQU       109       ;Constante 109, que es la
direccion del
                               ;ultimo valor a sumar.
SUMA      RES       1         ;Aqui guardamos la suma de
los numeros.

PRINCI    LD        /DIEZ     ;Inicializamos el
contador a valor 10.
          ST        /CONT
          CLR
BUCLE     ADD       /DIREC    ;Sumamos uno de los
numeros.
          ST        /SUMA     ;Guardamos la suma
parcial.
          LD        /CONT     ;Decrementamos el
contador.
          DEC
          BZ        /FIN      ;Si el contador es 0,
terminamos.
          ST        /CONT     ;Si no, actualizamos el
contador.
          LD      /BUCLE    ;Modificamos la instruccion
de suma para
          DEC                 ;que vaya cambiando el
numero a sumar,
          ST        /BUCLE    ;desde la posicion 109 a
la 100.
          LD        /SUMA     ;Cargamos en ac. la suma
parcial
          BR        /BUCLE    ;Y sumamos el siguiente
numero.
FIN       HALT
          END
```

Programa 7. Suma de diez números almacenados en memoria.

3.1.1. Ejecución

Arrancando el simulador de Símplez, tras editar el fichero SUMADIEZ.SIM, compilarlo y volver al simulador para ejecutar, aparece una pantalla como la que muestra la figura 12.

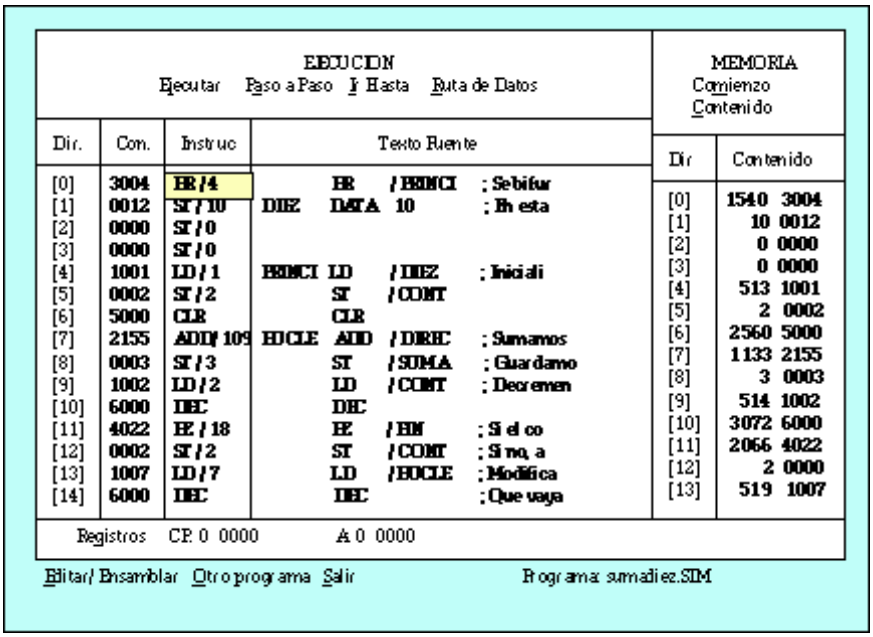

Figura 12. Ejecución de "SUMADIEZ.SIM".

Antes de empezar la ejecución deben introducirse en memoria los números a sumar, en las posiciones 100-109. Para ello hemos de situar nuestra ventana de memoria en la posición 100 y podemos hacerlo con la opción `Comienzo`del menú general, y respondiendo 100 a su solicitud. Luego se van introduciendo los números con la opción **`C`**`ontenido`, como vimos en el primer ejemplo.

La primera instrucción a ejecutar es "`BR /4`". Para ejecutar elegimos la opción **`R`**`uta de Datos`, que va paso a paso según ciclos de reloj y permite ver las microórdenes y los trasvases de información entre los elementos de la ruta de datos. Al pulsar esta opción, se mostrará una pantalla análoga a la de la figura 13.

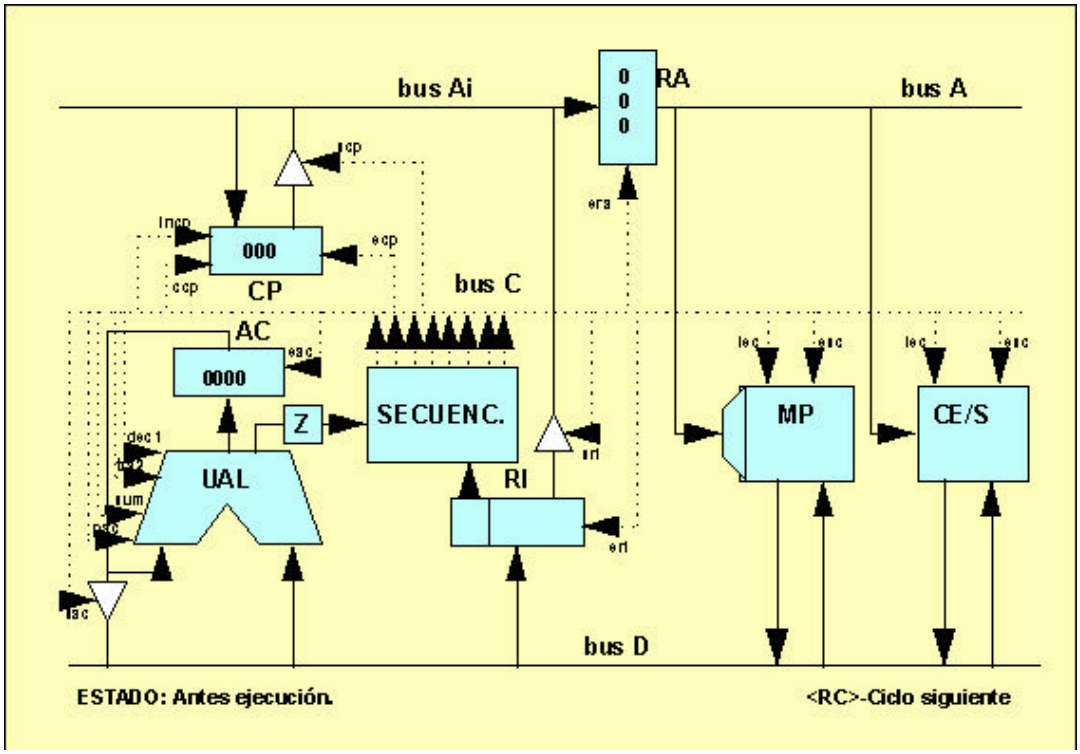

Figura 13. Ruta de datos al comienzo de ejecución.

Para ver la ejecución de esta instrucción debe tenerse presente el cronograma de la instrucción BZ (figura 14) y el contenido de los registros para ese cronograma (tabla 5). Al empezar la ejecución tenemos en CP y RA la dirección de la siguiente instrucción, en este caso la 0, y el acumulador también está a 0. Todos estos contenidos aparecen en sistema octal.

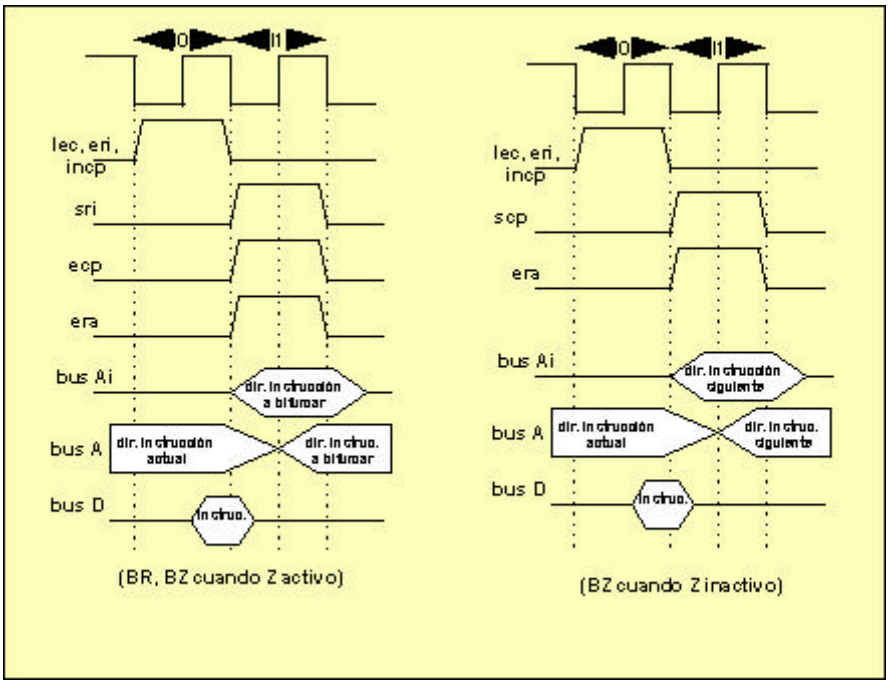

Figura 14.Cronogramas de BR y BZ.

CP	Dir. Instrucción actual	Dir. instrucción siguiente	Dir. instrucción a bifurcar (BR, BZ con AC=0). Dir. instrucción siguiente (BZ con AC no 0).
RA	Dir. Instrucción actual	Dir. instrucción actual	Dir. instrucción a bifurcar (BR, BZ con AC=0). Dir. instrucción siguiente (BZ con AC no 0).
RI	Instrucción anterior	Instrucción actual	Instrucción actual
AC	?	?	?

Tabla 5. Contenido de registros en BR y BZ.

Pulsando <RC se ejecuta el primer ciclo de reloj. Aparecerán en trazo grueso las microórdenes que estuvieron activas durante el ciclo (en este caso **lec**, **eri**, **incp**). También en trazo grueso los buses por los que ha habido trasvase de información, en este caso el bus D, que ha pasado la palabra leída de memoria (contenido octal 3004, equivalente a la instrucción, "BR /4") a RI. Se ve también el contenido de los registros tras el flanco de bajada, en este caso se ha incrementado CP al valor 1, RI tiene 3004 (instrucción actual), AC y RZ no se han modificado (figura 15).

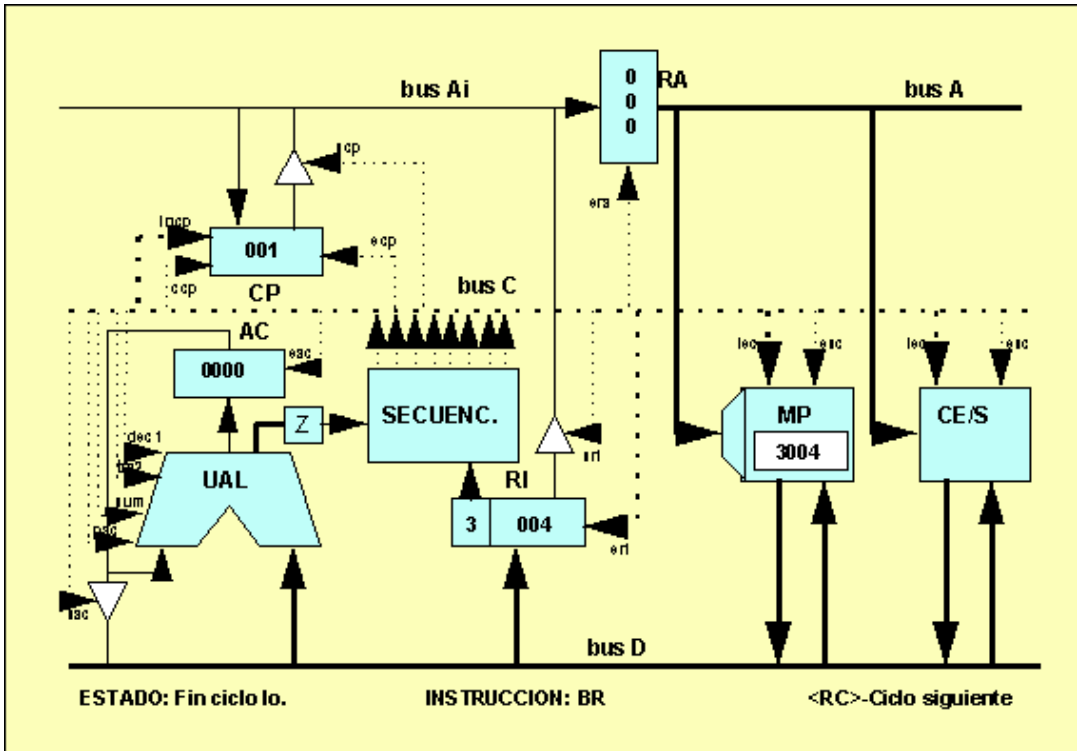

Figura15. Ruta de datos al final del primer ciclo de reloj.

Pulsando de nuevo <RC se pasa al ciclo I1, donde se puede ver como el contenido del campo de dirección (004 en octal), pasa a los registros CP y RA, ya que se activaron las microinstrucciones **sri**, **ecp** y **era** (figura 16).

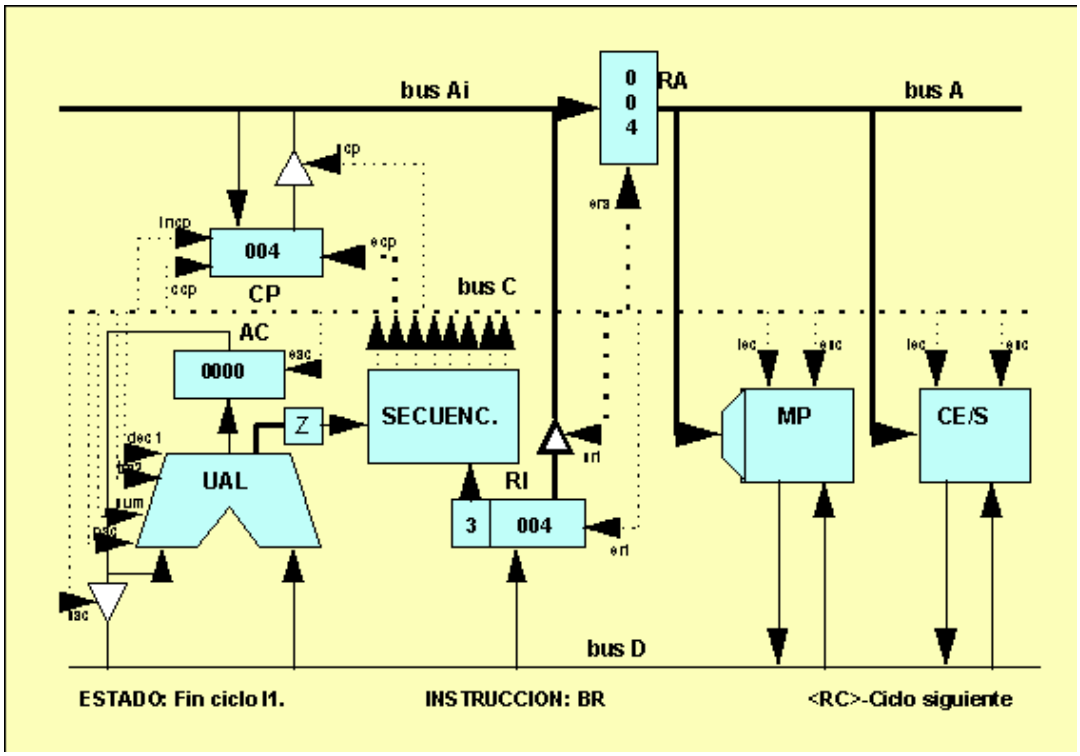

Figura 16. Ruta de datos al final del segundo ciclo de reloj.

Pulsando <RC se retorna a la pantalla del simulador, donde se puede observar que la próxima instrucción a ejecutar es la de posición de memoria 4 (se ha ejecutado "`BR /4`").

Se puede examinar la ejecución sobre la ruta de datos de cualquier instrucción presente en el programa. Lo más didáctico es tratar de reproducir manualmente lo que pasará en la ruta de datos, tomando del simulador los contenidos de CP y AC, los cronogramas de las instrucciones y el contenido de la memoria, si fuera necesario, y luego comprobar que lo que hace la ruta de datos es lo esperado.

Para terminar se puede optar, como siempre, entre **O**`tro programa` o **S**`alir`.

APÉNDICE.
CARACTERÍSTICAS BÁSICAS DE SÍMPLEZ

A.1. Modelo funcional de Símplez.

Se incluye el formato (figura 17) y el repertorio de instrucciones (tabla 6) de Símplez.

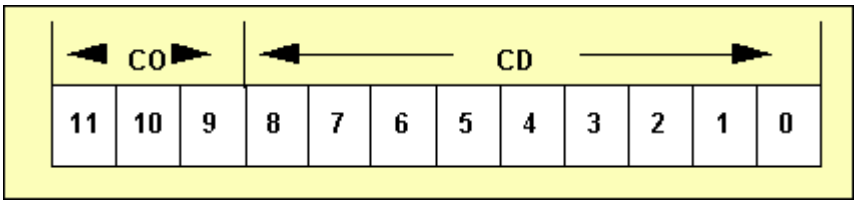

Figura 17. Formato de las instrucciones de Símplez.

CO(bin.)	CO(oct.)	CO(nemo.)	Significado
000	0	ST	Almacena ("STORE") el contenido del acumulador en la palabra cuya dirección se encuentra en CD.
001	1	LD	Carga ("LOAD") en el acumulador el contenido de la palabra de memoria cuya dirección se encuentra en CD
010	2	ADD	Suma ("ADD") al acumulador el contenido de la palabra de memoria cuya dirección se encuentra en CD, dejando el resultado en el acumulador.
011	3	BR	Bifurca ("BRANCH") a la dirección indicada en CD, es decir, la próxima instrucción a ejecutar es la que se encuentra en la palabra cuya dirección está contenida en CD
100	4	BZ	("BRANCH IF ZERO") Bifurca a la dirección indicada en CD si el resultado de la última operación efectuada por la UAL es 0. Si no, continúa secuencialmente.
101	5	CLR	("CLEAR") Pone a cero el acumulador.
110	6	DEC	Decrementa en una unidad el contenido del acumulador.
111	7	HALT	Hace que la máquina se pare.

Tabla 6. Repertorio de instrucciones de Símplez.

A.3. Microórdenes de Símplez

En la tabla 14 se describe las microórdenes de Símplez.

Microorden	Descripción
eac	Entrada en el acumulador desde la salida de la UAL
eri	Entrada en el registro de instrucción desde el bus D
incp	Incremento del contador de programa
ecp	Entrada en el contador de programa desde el bus Ai
ccp	Puesta a cero del contador de programa
era	Entrada en el registro de dirección desde el bus Ai
pac	Puesta a cero de la salida de la UAL
sum	Suma en la UAL de las entradas 1 y 2, resultado en la salida
tra2	Transferencia de la entrada 2 de la UAL a la salida
dec1	Decremento de la entrada 1 de la UAL, resultado en la salida
lec	Lectura en la MP o en el puerto de entrada
esc	Escritura en la MP o en el puerto de salida
sac	Salida del acumulador al bus D
sri	Salida del CD del registro RI al bus Ai
scp	Salida del contador de programa al bus Ai
cbf	Inhibición del funcionamiento
sco	Salida del código de operación
bbz	Para la instrucción BZ (UC microprogramada)
ebi	Entrada en el biestable BI (UC cableada)

Tabla 14. Microórdenes de Símplez.

Recortado por:
Daniel José Scokin

E-mail : nexum@usa.net
nexum@lanet.losandes.com.ar

para la catedra de Técnicas Digitales II de la Universidad Tecnológica Nacional, Regional Mendoza

---

Back to Top ▲