UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INFORMÁTICA

TRABAJO FIN DE GRADO

IMPLEMENTACIÓN DE UNA RUTA DE DATOS SIN ADELANTAMIENTO EN EL SIMULA3MS.

María Isabel Hernández Mejía

Febrero, 2015

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

Departamento de Sistemas Informáticos

TRABAJO FIN DE GRADO

IMPLEMENTACIÓN DE UNA RUTA DE DATOS SIN ADELANTAMIENTO EN EL SIMULA3MS.

Autor: María Isabel Hernández Mejía

Director: Francisco José Alfaro Cortes

Pedro Javier García García

Febrero, 2015

Resumen

Simula3MS es un simulador de una arquitectura básica de procesador (basada en MIPS) en sus versiones monociclo, multiciclo y segmentado. Este simulador es utilizado en la docencia de aquellas asignaturas que tratan la arquitectura y organización del computador, ya que permite observar el comportamiento del procesador y aplicar los conocimientos adquiridos en las clases de teoría. Tiene un entorno de trabajo gráfico y de fácil manejo que permite depurar cómodamente los programas, y observar la evolución de la memoria, así como la ejecución de las instrucciones sobre distintos caminos de datos. La posibilidad de usar los distintos cauces en la misma herramienta permite observar las diferencias de ejecución de un mismo código según cuales sean las características del procesador.

En esta memoria se describen principalmente las modificaciones que hemos introducido en el simulador Simula3MS para añadir algunas funcionalidades de las que no disponía. Concretamente, se ha añadido la posibilidad de que el camino de datos segmentado básico pueda funcionar con adelantamiento o sin adelantamiento, permitiendo así que los alumnos puedan simular ambas opciones. Ahora los alumnos podrán ver la diferencia del comportamiento del simulador cuando puede adelantar los datos que las instrucciones necesitan y cuando el simulador no adelanta los datos a las instrucciones y espera hasta que el dato o registro está libre.

Agradecimientos

Este trabajo ha sido realizado gracias al esfuerzo para poder lograr los objetivos planteados.

Agradecer a mis tutores Francisco José Alfaro y Pedro Javier García por su dedicación y comprensión a lo largo de este trabajo.

También agradecer a mi familia y a mis amigos por aguantarme en los momentos difíciles. En especial a mis padres por el esfuerzo que han realizado para poder formarme como persona y a mi abuela por creer siempre en mí.

María.

ÍNDICE

CAPÍTULO 1. INTRODUCCIÓN 1
1.1 MOTIVACIÓN 1
1.2 OBJETIVOS Y METODOLOGÍA 2
1.3 ESTRUCTURA DE LA MEMORIA 3

CAPÍTULO 2. DESCRIPCIÓN DEL ENTORNO 5
2.1 ARQUITECTURA BÁSICA MIPS 5
2.2 CONSTRUCCIÓN DE UN CAMINO DE DATOS 8
2.2.1 PROCESADOR MONOCICLO 9
2.2.2 PROCESADOR MULTICICLO 10
2.2.3 PROCESADOR SEGMENTADO 13
2.3 MODELADO MIPS EN SIMULA3MS 19

CAPÍTULO 3. DISEÑO E IMPLEMENTACIÓN DEL CAUCE MIPS SIN ADELANTAMIENTO 25
3.1 MAPA DE CLASES 25
3.2 PLANIFICACIÓN DE LAS MODIFICACIONES 27
3.3 IMPLEMENTACIÓN DE LA VENTANA DE INICIO 27
3.4 IMPLEMENTACIÓN INTERNA PARA EL CAUCE SIN ADELANTAMIENTO 28
3.5 IMPLEMENTACIÓN PARA LA VISUALIZACIÓN DEL CAUCE SIN ADELANTAMIENTO 29
3.6 MODIFICACIONES DE LAS IMÁGENES DEL CAMINO DE DATOS 30
3.7 CASOS ERRÓNEOS DEL CAUCE MIPS CON ADELANTAMIENTO EN EL SIMULA3MS ORIGINAL 31

CAPÍTULO 4. PRUEBAS DE SISTEMA 35
4.1 DEPENDENCIA DE DATOS ENTRE DOS INSTRUCCIONES ARITMÉTICAS ENTERAS 36
4.2 DEPENDENCIA DE DATOS ENTRE DOS INSTRUCCIONES ARITMÉTICAS DE PUNTO FLOTANTE 38
4.3 DEPENDENCIA DE DATOS ENTRE UNA INSTRUCCIÓN DE CARGA SEGUIDA DE UNA INSTRUCCIÓN ARTIMÉTICA 42
4.4 DEPENDENCIA DE DATOS ENTRE UNA INSTRUCCIÓN DE CARGA Y UNA INSTRUCCIÓN DE ALMACENAMIENTO EN EL PRIMER OPERANDO DEL ALMACENAMIENTO 45

4.5 DEPENDENCIA DE DATOS ENTRE UNA INSTRUCCIÓN DE CARGA Y UNA INSTRUCCIÓN DE ALMACENAMIENTO EN EL SEGUNDO OPERANDO DEL ALMACENAMIENTO 50
4.6 DEPENDENCIA DE DATOS ENTRE UNA INSTRUCCIÓN ARITMÉTICA Y UNA INSTRUCCIÓN DE ALMACENAMIENTO 53
4.7 CONCLUSIONES 58

CAPÍTULO 5. CONCLUSIONES Y TRABAJO FUTURO 59
5.1 CONCLUSIONES 59
5.2 TRABAJO FUTURO 60

BIBLIOGRAFÍA 61
LIBROS Y ARTICULOS 61
ENLACES INTERNET 61

CONTENIDO DEL CD 62

ANEXO A. CÓDIGO FUENTE 63

ÍNDICE DE FIGURAS

Figura 2.1. Modelo básico de la arquitectura MIPS. 6
Figura 2.2. Formatos básicos de instrucciones MIPS. 7
Figura 2.3. Componentes básicos de un procesador. 8
Figura 2.4. Camino de datos monociclo. 9
Figura 2.5. Camino de datos multiciclo. 11
Figura 2.6. Ejemplo segmentación en lavandería. 14
Figura 2.7. Etapas de la ejecución segmentadas de las instrucciones. 15
Figura 2.8. Diferencias entre la ejecución de un procesador secuencial y un segmentado. 15
Figura 2.9. Camino de datos segmentado. 16
Figura 2.10. Tipos de riesgos de dependencia de datos. 17
Figura 2.11. Ejemplo introducción de burbujas. 18
Figura 2.12. Esquema simplificado de la ruta de datos con anticipación. 18
Figura 2.13. Anticipación de los resultados leídos de memoria. 19
Figura 2.14. Ventana editor del Simula3MS. 20
Figura 2.15. Procesador monociclo ejecutando el código *Ejemplo 3.s*. 21
Figura 2.16. Procesador multiciclo ejecutando el código *Ejemplo 3.s*. 22
Figura 2.17. Procesador segmentado ejecutando el código *Ejemplo 3.s*. 23
Figura 2.18. Diagrama multiciclo para la ejecución del código *Ejemplo 3.s*. 23
Figura 2.19. Diagrama monociclo para la ejecución del código *Ejemplo 3.s*. 24

Figura 3.1. Mapa de clases del Simula3MS. 26
Figura 3.2. Ventana inicio del simulador original. 28
Figura 3.3. Ventana inicio del simulador tras la modificación realizada. 28
Figura 3.4. Nueva imagen del camino de datos segmentado con adelantamiento. 30
Figura 3.5. Nueva imagen del camino de datos segmentado sin adelantamiento. 31
Figura 3.6. Cambios realizados en la etapa EX. 32
Figura 3.7. Ejemplo 5.s en el simulador. 33
Figura 3.8. Adelantamiento con una instrucción `sw`. 34

Figura 4.1. *Ejemplo 1.1.s,* código utilizado para evaluar una dependencia aritmética entera. 36
Figura 4.2. *Ejemplo 1.1.s* ejecutando en el camino de datos con adelantamiento de $t2 de `add` a `sub`. 37
Figura 4.3. Diagrama multiciclo para el *Ejemplo 1.1.s* sin adelantamiento. 37

Figura 4.4. Comparación de las estadísticas aplicando adelantamiento (izquierda) y sin aplicarlo (derecha). 38
Figura 4.5. *Ejemplo 1.s*, código utilizado para evaluar una dependencia aritmética..39
Figura 4.6. Diagrama multiciclo para el *Ejemplo 1.s* con adelantamiento. 40
Figura 4.7. Diagrama multiciclo *Ejemplo 1.s* sin adelantamiento. 40
Figura 4.8. Comparación de las estadísticas aplicando adelantamiento (izquierda) y sin aplicarlo (derecha). 41
Figura 4.9. *Ejemplo 2.s*, código utilizado para evaluar dependencia aritmética sobre una carga. 42
Figura 4.10. Camino de datos para el *Ejemplo 2.s* con adelantamiento. 43
Figura 4.11. Diagrama multiciclo para el *Ejemplo 2.s* con adelantamiento. 43
Figura 4.12. Camino de datos para el *Ejemplo 2.s* sin adelantamiento. 44
Figura 4.13. Comparación de las estadísticas aplicando adelantamiento (izquierda) y sin aplicarlo (derecha). 45
Figura 4.14. *Ejemplo 3.s*, código utilizado para evaluar una dependencia de carga sobre almacenamiento. 46
Figura 4.15. Camino de datos para el *Ejemplo 3.s* con adelantamiento en el simulador original. 47
Figura 4.16. Diagrama multiciclo para el *Ejemplo 3.s* con adelantamiento en el simulador original. 47
Figura 4.17. Camino de datos para el *Ejemplo 3.s* con adelantamiento en la nueva versión del simulador.....48
Figura 4.18. Diagrama multiciclo para el *Ejemplo 3.s* con adelantamiento en la nueva versión del simulador.....49
Figura 4.19. Camino de datos para el *Ejemplo 3.s* sin adelantamiento. 49
Figura 4.20. Comparación de las estadísticas aplicando adelantamiento (izquierda) y sin aplicarlo (derecha). 50
Figura 4.21. *Ejemplo 4.s*, código utilizado para evaluar una dependencia de carga sobre el segundo operando del almacenamiento. 51
Figura 4.22. Camino de datos para el *Ejemplo 4.s* con adelantamiento. 51
Figura 4.23. Camino de datos para el *Ejemplo 4.s* sin adelantamiento. 52
Figura 4.24. Diagrama multiciclo para el *Ejemplo 4.s* sin adelantamiento. 52
Figura 4.25. Comparación de las estadísticas aplicando adelantamiento (izquierda) y sin aplicarlo (derecha). 53
Figura 4.26. *Ejemplo 5.s*, código utilizado para evaluar una dependencia entre una operación aritmética y una instrucción de almacenamiento. 54
Figura 4.27. Camino de datos para el *Ejemplo 5.s* con adelantamiento en el procesador original. 55

Figura 4.28. Camino de datos para el *Ejemplo 5.s* con adelantamiento en la nueva versión del procesador. 56
Figura 4.29. Camino de datos para el *Ejemplo 5.s* sin adelantamiento. 57
Figura 4.30. Comparación de las estadísticas aplicando (izquierda) y sin aplicar (derecha) adelantamiento. 57

Figura 5.1. Pantalla inicial del simulador modificado. 60

CAPÍTULO 1. INTRODUCCIÓN

En este capítulo veremos una pequeña descripción del trabajo explicando, su motivación, sus objetivos y la metodología seguida y, por último, la estructura de esta memoria.

1.1 MOTIVACIÓN

El simulador Simula3MS modela el funcionamiento básico de una arquitectura MIPS. Este simulador surgió de un proyecto del grupo de Arquitectura de Computadores de la Universidad de A Coruña [8].

El simulador Simula3MS actualmente se usa en las clases de Estructura de Computadores, Organización de Computadores y Arquitectura de Computadores para poder estudiar la implementación de la ruta de datos de un procesador monociclo y multiciclo sin segmentar, y segmentado. Como se ha comentado, este simulador ya implementa la técnica de adelantamiento, donde una unidad funcional adelanta directamente el resultado calculado a otra unidad funcional que pueda necesitar dicho resultado, sin necesidad de esperar a que se escriba en el banco de registros.

Actualmente, el Simula3MS modela el cauce segmentado del procesador con adelantamiento, pero no permite la ejecución de un código sin adelantamiento. Es decir,

no se contempla que en caso de haber una dependencia de datos, las instrucciones deban esperarse para leer sus registros operandos a que otras instrucciones previas las hayan escrito en el banco de registros. Esta posibilidad sería de gran importancia para que los estudiantes de asignaturas en las que se estudian cauces segmentados entendieran perfectamente este tipo de riesgos de datos, tan comunes en cualquier código.

1.2 OBJETIVOS Y METODOLOGÍA

El objetivo principal de este proyecto es el siguiente:

- Implementación en el simulador Simula3MS del camino de datos segmentado sin adelantamiento: implementación en la que, en caso de tener una dependencia de datos, tendremos que esperar a que llegue el dato a los registros visibles a nivel de programación o a la memoria, antes de proceder a su lectura y uso posterior.

La metodología que seguiremos para alcanzar este objetivo se basa en las siguientes tareas:

- Entender bien toda la teoría relacionada con la arquitectura segmentada del procesador MIPS.
- Conocer perfectamente el funcionamiento a nivel usuario del Simula3MS.
- Analizar el código fuente del Simula3MS. Una vez asimilados los conocimientos teóricos generales de simuladores y sabiendo utilizar el Simula3MS, hay que estudiar y comprender el código fuente del simulador para ver qué partes del código tendremos que modificar para llegar al objetivo de nuestro proyecto.
- Ampliar y modificar el código del Simula3MS para que el simulador implemente el camino de datos segmentado sin adelantamiento.
- Realizar las pruebas de sistema necesarias para comprobar que nuestro proyecto está correctamente realizado.

1.3 ESTRUCTURA DE LA MEMORIA

En este apartado vemos la estructura de esta memoria, que se divide en los siguientes capítulos:

Capítulo 1: Introducción. Donde mostramos una pequeña introducción del proyecto, su motivación, los objetivos y metodología para conseguirlos y la estructura del documento.

Capítulo 2: Descripción del entorno. Donde describimos los aspectos básicos de las arquitecturas MIPS y su modelado en el simulador Simula3MS.

Capítulo 3: Diseño e implementación. Donde describimos la versión actual del Simula3MS y todas las modificaciones introducidas para implementar la ejecución sin adelantamiento.

Capítulo 4: Pruebas de Sistema. Donde aplicamos ejemplos en el nuevo simulador para comprobar que funciona correctamente.

Capítulo 5: Conclusiones y propuestas. Donde describiremos las conclusiones y las propuestas futuras.

Bibliografía. En este apartado recogeremos todas las referencias externas que se usen en este proyecto.

Anexos. Donde se incluirán todos los anexos de este proyecto.

CAPÍTULO 2.
DESCRIPCIÓN DEL ENTORNO

En este capítulo vamos a explicar de forma general la arquitectura básica MIPS y su modelado en Simula3MS. Primero, veremos la arquitectura básica MIPS y sus tipos de instrucciones. A continuación, cómo un simulador de un procesador debe permitir observar la evolución de la memoria y de los registros durante la ejecución de las instrucciones. Después, veremos cómo se construye un camino de datos. Dentro de este punto, podremos ver los tres tipos de camino de datos: monociclo, multiciclo y segmentado. Y por último, podremos ver una visión general del simulador Simula3MS. Una vez vistos todos estos puntos podremos entender mejor el simulador Simula3MS.

2.1 ARQUITECTURA BÁSICA MIPS

MIPS (Microprocessor without Interlocked Pipeline Stages, Microprocesador sin bloqueos en las etapas de segmentación) es una arquitectura de procesador diseñada tanto para optimizar la segmentación en unidades de control como para facilitar la generación automática de código máquina por parte de los compiladores [2].

Los microprocesadores MIPS tienen arquitectura RISC, es decir, el diseño de CPU tiene las siguientes características:

- Instrucciones de tamaño fijo y con un reducido número de formatos.
- Sólo las instrucciones de carga y almacenamiento acceden a la memoria de datos.

Como podemos ver en la figura 2.1, el procesador tiene 32 registros de propósito general y dos registros de propósito específico: HI-LO (usado para productos y divisiones) y PC. El ancho de los registros es de 32 bits. Puede tener hasta 4 coprocesadores. En la figura 2.1 podemos ver el coprocesador 0 que es el de control de sistema y tiene las siguientes funciones: controlar el subsistema de la memoria cache, soportar el sistema de memoria virtual, manejar las excepciones, manejar los cambios en el modo de ejecución, proporcionar control de diagnósticos y recuperación de errores. También podemos ver el coprocesador 1 que está reservado para la unidad de coma flotante [4].

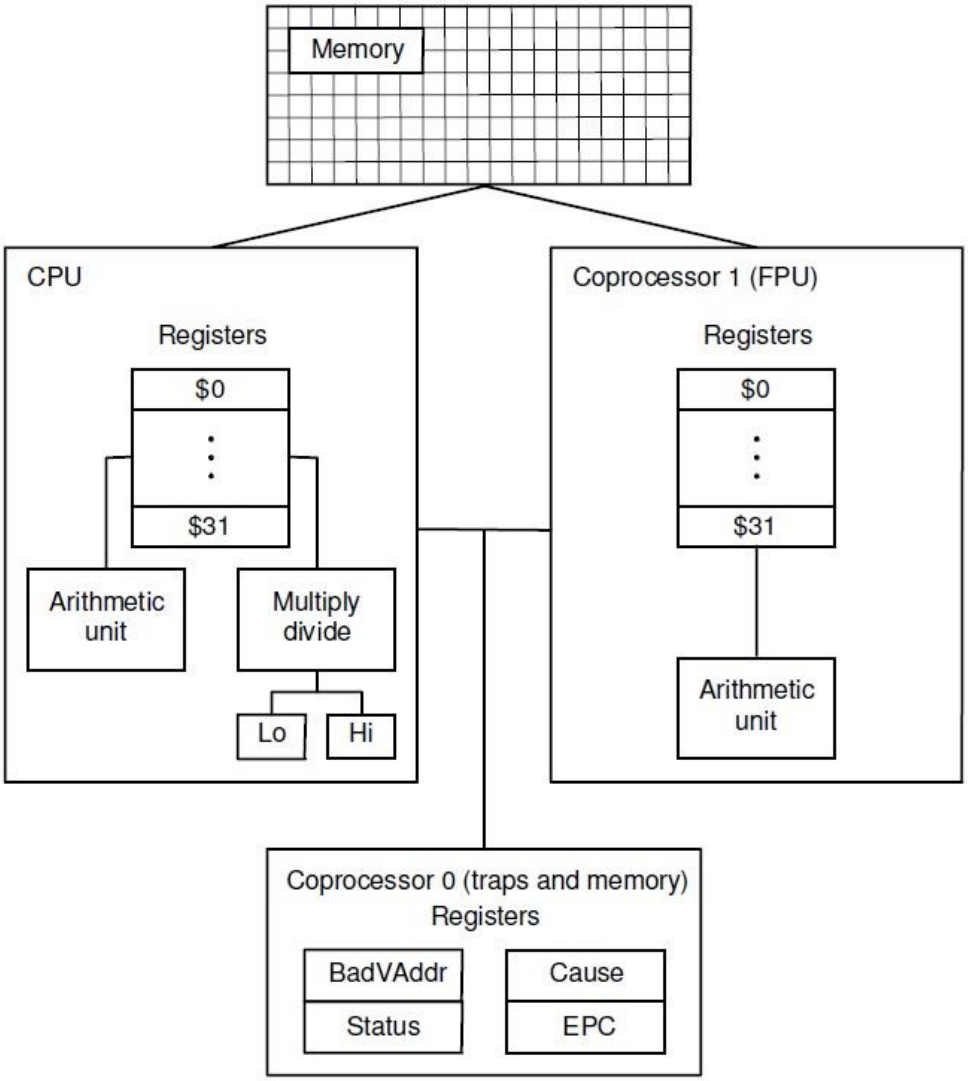

Figura 2.1. Modelo básico de la arquitectura MIPS.

MIPS presenta tres formatos básicos de instrucciones de 32 bits que podemos ver en la figura 2.2. Las instrucciones Tipo R son instrucciones entre registros, es decir, operaciones ALU registro-registro. Disponen de tres operandos registros rs, rt y rd, donde

rs y rt son registros fuente y rd es el registro destino. En el campo *Funct* se encuentra la función a realizar por la ALU. Las instrucciones pertenecientes a este formato son `add, sub, and, or…`

Las instrucciones Tipo I son instrucciones con inmediato, es decir, instrucciones de carga y almacenamiento. Disponen de dos operandos-registros rs y rt, donde rs se utiliza como registro base al cual se le añade el campo de *Desplazamiento (Offset)* de 16 bits para obtener la dirección de memoria y rt se utiliza dependiendo del tipo de instrucción: si la instrucción es de carga, rt es el registro destino del valor cargado de memoria. Y si la instrucción es de almacenamiento, rt es el registro fuente del valor que se debe almacenar en memoria. Las instrucciones pertenecientes a este formato son `lw y sw.`

Las instrucciones Tipo J son instrucciones de salto. Disponen de un campo de 16 bits *Dirección destino (target)* al que se le extiende el signo, se desplaza y se suma al PC para calcular la dirección de salto. La instrucción perteneciente a este formato es `jump`.

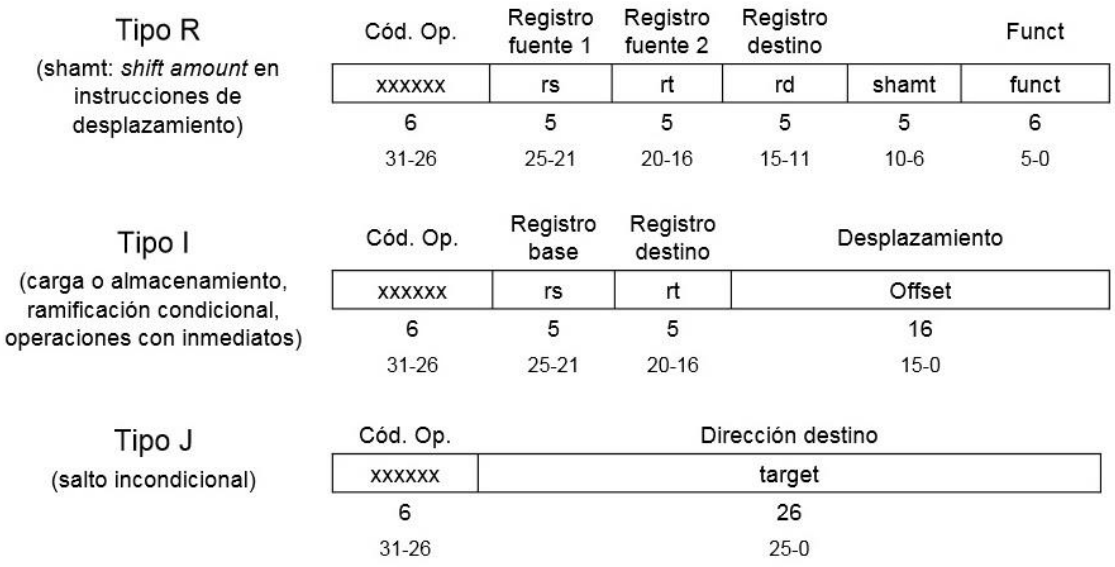

Figura 2.2. Formatos básicos de instrucciones MIPS.

2.2 CONSTRUCCIÓN DE UN CAMINO DE DATOS

El procesador es el encargado de ejecutar las instrucciones especificadas por el programa. Sus funciones básicas son:

- Leer instrucciones de la memoria.
- Interpretar las instrucciones.
- Captar datos.
- Procesar datos.
- Escribir datos.

En la figura 2.3 podemos ver los componentes básicos de un procesador:

- Unidad de control.
- Unidad aritmético-lógica.
- Banco de registros.
- Otros registros internos: PC (contador de programa), IR (registro de instrucciones).

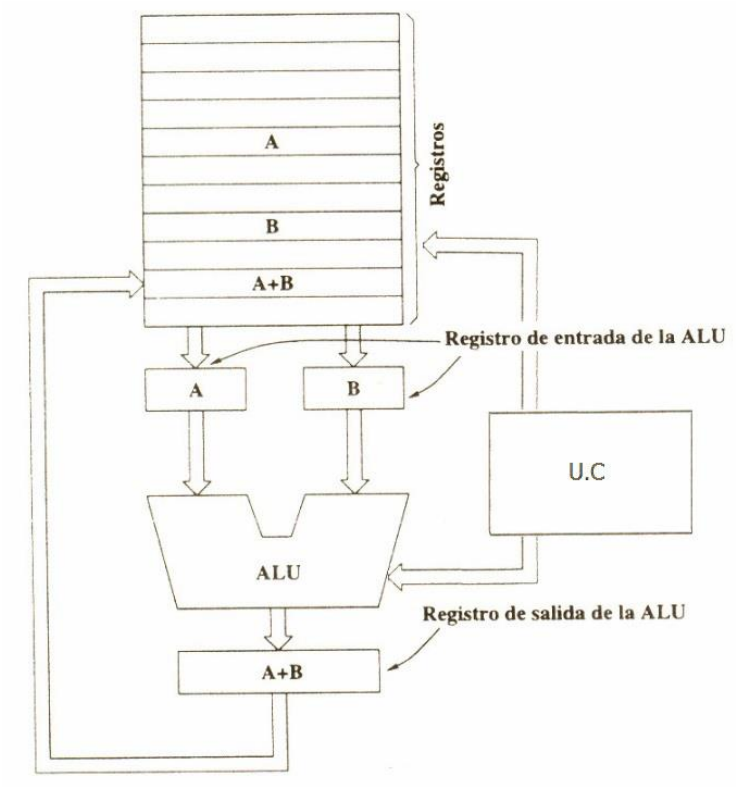

Figura 2.3. Componentes básicos de un procesador.

El procesador puede dividirse en dos entidades funcionales: el camino de datos y la unidad de control. El camino de datos es la sección del computador encargada de manipular y transformar los datos procedentes de la memoria o los registros internos, para obtener los resultados. La unidad de control es la sección del computador encargada de interpretar las instrucciones del programa y gobernar la ejecución de las mismas.

El camino de datos puede implementarse de tres formas distintas:

- Monociclo.
- Multiciclo.
- Segmentado.

Estas versiones se diferencian en la manera de ejecutar las instrucciones en el tiempo. En el procesador monociclo las instrucciones se ejecutan en un sólo ciclo de reloj grande, en el procesador multiciclo las instrucciones se ejecutan en varios ciclos de reloj pequeños y en el procesador segmentado la ejecución de las instrucciones se solapan en el tiempo. A continuación se detallan estas tres versiones.

2.2.1 PROCESADOR MONOCICLO

El camino de datos monociclo es la implementación más sencilla del procesador porque cada instrucción se ejecuta en un solo ciclo de reloj. Esto evita tener que ordenar en el tiempo las operaciones a llevar a cabo.

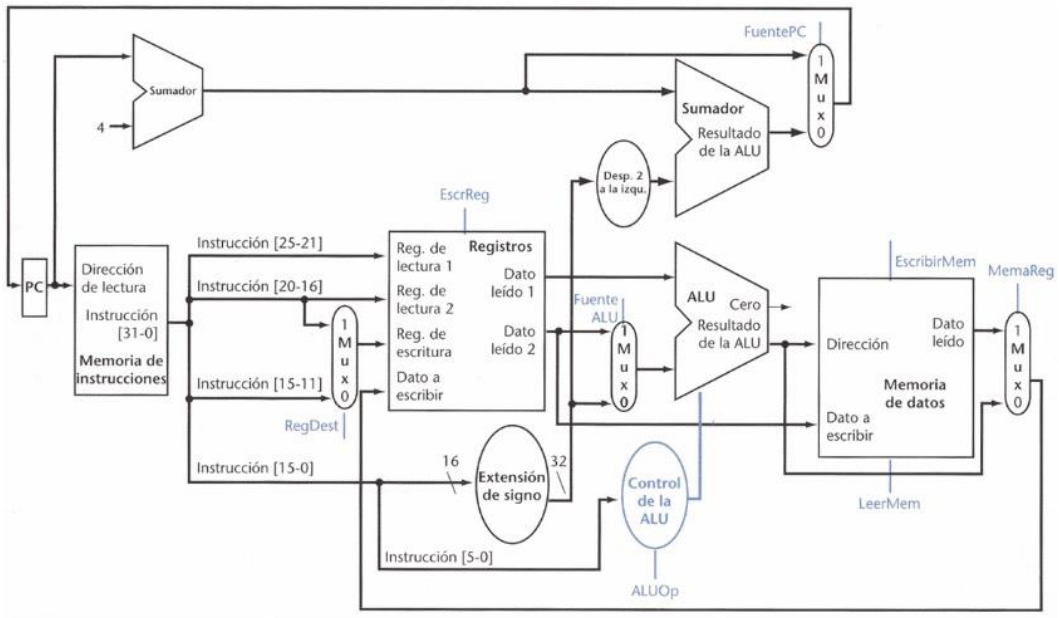

Figura 2.4. Camino de datos monociclo.

Para poder implementar el subconjunto del repertorio MIPS en una implementación monociclo necesitamos los siguientes componentes:

- Memoria de instrucciones.
- Memoria de datos.
- 32 registros de datos: visibles por el programador.
- Contador de programa.
- 2 sumadores: para sumar 4 al PC, y para sumar al PC el valor inmediato de salto.
- ALU: capaz de realizar suma, resta, and, or, comparación de mayoría e indicación de que el resultado es cero (para realizar la comparación de igualdad mediante resta).
- Extensor de signo: para adaptar el operando inmediato de 16 bits al tamaño de palabra (el tamaño de palabra es de 32 bits).
- Desplazador a la izquierda: para implementar la multiplicación por cuatro.

Los inconvenientes del procesador monociclo son:

- El ciclo de reloj está definido por la instrucción más lenta.
- Las unidades funcionales sólo pueden utilizarse una vez por ciclo.
- Estos inconvenientes se verían agravados en una arquitectura más compleja que la MIPS.

Las soluciones alternativas para los inconvenientes del procesador monociclo son las siguientes:

- Reducir el ciclo de reloj y hacer que cada instrucción se ejecute en varios ciclos, lo que sería el procesador multiciclo.
- Solapar la ejecución de varias instrucciones elevando la utilización del hardware y el rendimiento, lo que sería el procesador segmentado.

2.2.2 PROCESADOR MULTICICLO

El camino de datos multiciclo mejora el rendimiento ejecutando cada una de las operaciones de la instrucción en un ciclo de reloj más pequeño. Cada instrucción se ejecutará en un número de ciclos de reloj determinado por las operaciones que haya que llevar a cabo. Por ello, las instrucciones tardan distinto tiempo en ejecutarse. La ejecución

de cada instrucción se divide en varias etapas; cada una de ellas se realiza en un ciclo de reloj. En cada etapa se realizará una operación básica que tardará un ciclo de reloj:

- Un acceso a memoria.
- Un acceso al banco de registros.
- Una operación ALU.

En la figura 2.5 podemos ver el camino de datos multiciclo donde las instrucciones y datos se almacenan en la misma unidad de memoria. Hay una sola ALU en vez de ALU y dos sumadores como ocurría en el camino de datos monociclo. Se añaden uno o más registros tras cada unidad funcional para almacenar los valores de salida hasta que se utilicen en el siguiente ciclo. Los datos que se utilizan en posteriores instrucciones se almacenan en elementos visibles al programador: registro, memoria o PC.

Existen los siguientes registros temporales: registro de instrucción (IR) y registro de datos de memoria (MDR) a la salida de la memoria, registros A y B para guardar valores de los registros operandos, y registro SalidaALU a la salida de la ALU. Salvo el registro IR todos guardan datos entre dos ciclos consecutivos. No necesitan señal de control de escritura.

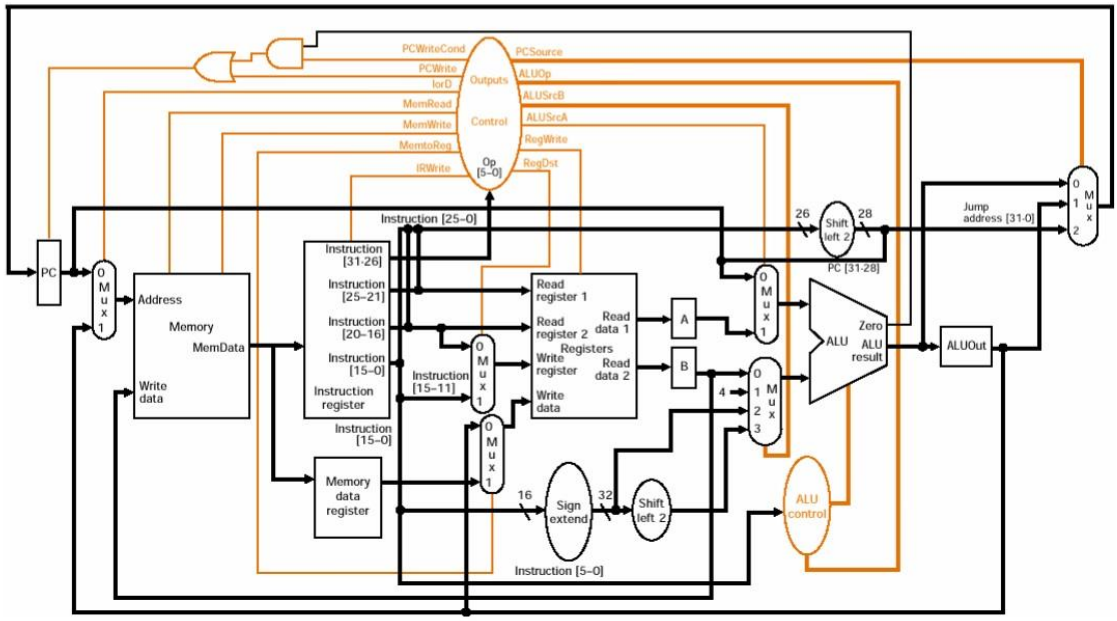

Figura 2.5. Camino de datos multiciclo.

Como podemos ver en la figura 2.5, puesto que se tarda más de un ciclo, las señales de control serán distintas. El PC, la memoria, los registros y el IR necesitan señales de control de escritura. La memoria necesita una señal de lectura. Cada uno de los

multiplexores requiere sus líneas de control. La unidad de control de la ALU monociclo sirve también en el camino multiciclo.

Para ejecutar una instrucción en el camino de datos multiciclo, la instrucción debe atravesar las siguientes etapas:

1. Carga de la instrucción: se lee una instrucción de memoria y se carga en el IR, y además se incrementa el PC.
2. Decodificación de la instrucción: se dedica un ciclo completo a la decodificación porque a veces la circuitería de control necesaria para averiguar cuál es la instrucción leída es compleja.
   Al mismo tiempo pueden realizarse también: operaciones comunes a todas las instrucciones y que puedan adelantar su ejecución, operaciones que puedan adelantar la ejecución de ciertas instrucciones y que no resulten perjudiciales en caso de que la instrucción leída sea distinta a la esperada (lectura de registros en el banco, cálculo de dirección de salto).
3. Ejecución ALU, cálculo de la dirección de memoria o finalización de salto; la ALU realiza una operación:
   - En instrucciones tipo R se ejecuta la operación pedida.
   - En instrucciones `lw` y `sw` se calcula la dirección del operando de memoria.
   - En los saltos condicionales, la ALU se utiliza para efectuar la comparación de igualdad entre los dos registros leídos en la etapa anterior. El salto se toma en función de la señal cero.
   - En los saltos incondicionales, el PC se sustituye por la dirección de salto incondicional.
4. Acceso a memoria o finalización de instrucciones tipo R: se efectúa el acceso a memoria o se escribe el resultado en instrucciones tipo R.
   - En instrucciones tipo R se escribe en el banco de registros el resultado de la operación de la etapa anterior.
   - En instrucciones `lw` se lee el dato de memoria.
   - En instrucciones `sw` se escribe el dato en memoria.
5. Finalización de lectura en memoria: en instrucciones `lw` se escribe el dato leído en el banco de registros.

Las implicaciones que tiene el procesador multiciclo son las siguientes:

- La duración de las instrucciones se ajusta al número de etapas necesarias para su lectura y ejecución (tantos ciclos de reloj como estados le correspondan en la máquina).
- El objetivo es alcanzar unas prestaciones elevadas haciendo sencillo el control.
- El diseño del camino de datos y el control depende en gran medida de la naturaleza y características del repertorio de instrucciones del computador.
- Factores que permiten mantener sencillo el control:
    - Instrucciones sencillas, homogéneas y poco potentes.
    - Pocos modos de direccionamiento.
    - Pocos formatos de instrucción.
    - Codificación uniforme, con códigos de operación de tamaño y posición fijos.

- Factores que complican el control:
    - Existencia de instrucciones potentes que hagan muchas cosas y que duren muchos ciclos de reloj.
    - Muchos modos de direccionamiento.
    - Muchos formatos de instrucción.
    - Codificación no uniforme, con códigos de longitudes y posiciones variables.
    - Tratamiento de excepciones.

2.2.3 PROCESADOR SEGMENTADO

La segmentación o pipeline [1] [3] [5] es la técnica que permite solapar la ejecución de instrucciones. Se utiliza esta técnica para que la ejecución final de una secuencia de código sea más rápida.

Veamos la idea básica de la segmentación mediante el ejemplo de una lavandería que podemos ver en la figura 2.6 [1]. El trabajo en una lavandería se divide en varias fases sucesivas (la ropa se lava, se seca, se plancha y se da al cliente), de esta forma cada fase está trabajando simultáneamente el coladas distintas. De esta forma hacer una colada cuesta el mismo tiempo que antes pero ahora la frecuencia con la que salen las coladas limpias es mucho mayor (tanto como fases tenga su construcción).

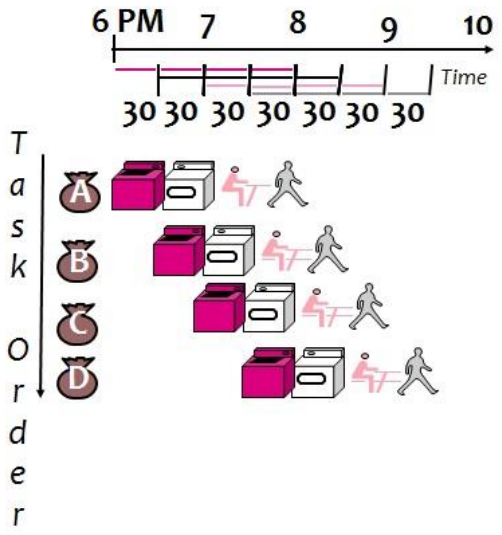

Figura 2.6. Ejemplo segmentación en lavandería.

Para aplicar la técnica de segmentación a la ejecución de las instrucciones, debemos dividir la ejecución de las mismas en varias etapas. En el procesador MIPS el cauce se divide en cinco etapas distintas. Una segmentación de cinco etapas implica que durante un ciclo de reloj se estarán ejecutando hasta cinco instrucciones. Se debe separar el camino de datos en cinco partes, correspondiendo cada una de ellas a una etapa de la ejecución de la instrucción. En la figura 2.7 podemos ver las cinco etapas de ejecución de las instrucciones que a continuación detallaremos:

1. IF: En esta etapa haremos la búsqueda de instrucciones.
2. ID: En esta etapa decodificaremos las instrucciones y leeremos del banco de registros.
3. EX: En esta etapa ejecutaremos la instrucción o calcularemos la dirección.
4. MEM: En esta etapa accederemos a la memoria de datos.
5. WB: En esta etapa haremos la escritura del resultado.

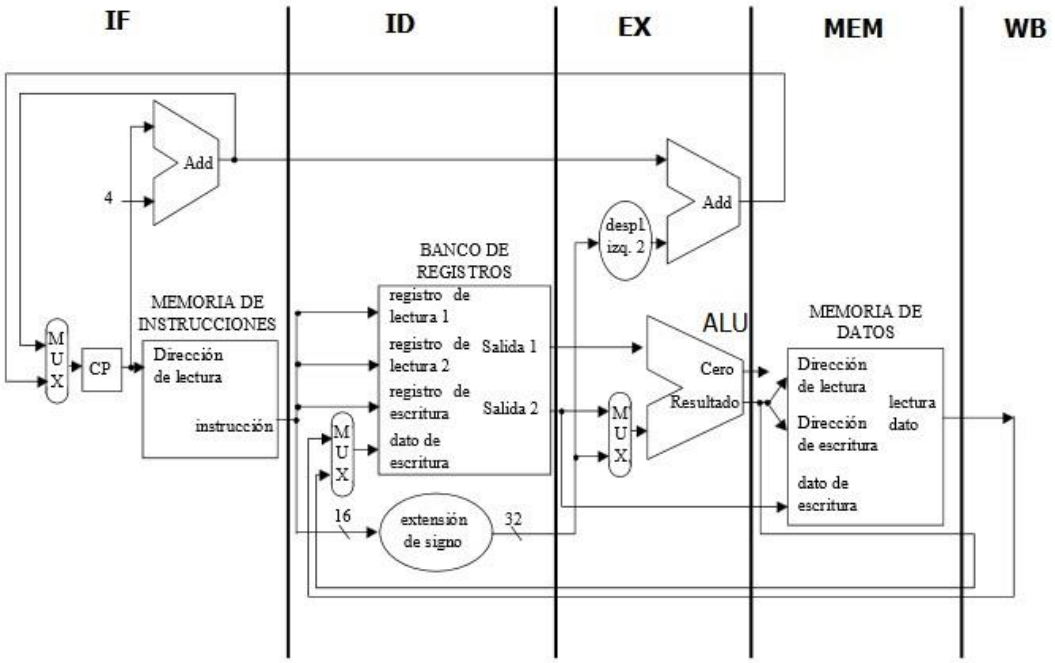

Figura 2.7. Etapas de la ejecución segmentadas de las instrucciones.

En la figura 2.8 podemos ver las diferencias de ejecución entre un procesador secuencial y la ejecución de un procesador segmentado. Cómo podemos ver en la figura 2.8, en un procesador no segmentado las instrucciones se ejecutan secuencialmente, mientras que en un procesador segmentado cuando una instrucción termina por una etapa, esa etapa la ocupa la siguiente instrucción.

Operación de un procesador secuencial

	1	2	3	4	5	6	7	...
Instr.1	IF1	ID1	EX1	MEM1	WB1			...
Instr.2						IF2	ID2	

Operación de un procesador segmentado de 5 etapas

	1	2	3	4	5	6	7
Instr.1	IF1	ID1	EX1	MEM1	WB1	...	...
Instr.2	...	IF2	ID2	EX2	MEM2	WB2	
...	...		IF3	ID3	EX3	MEM3	WB3
...	...	...	...	IF4	ID4	EX4	MEM4
...	...	...	...	...	IF5	EX5	...

Figura 2.8. Diferencias entre la ejecución de un procesador secuencial y un segmentado.

Las ventajas del procesador segmentado son las siguientes: se mejora la productividad y se reduce el tiempo medio de ejecución de las instrucciones.

En la figura 2.9 podemos ver el camino de datos segmentado. La ejecución de instrucciones en un procesador segmentado requiere que en cada uno de los segmentos seamos capaces de almacenar toda la información de la instrucción que se está ejecutando en ese segmento. Por este motivo en los registros de segmentación (IF/ID, ID/EX, EX/MEM, MEM/WB) se almacena toda la información necesaria de la instrucción que se está ejecutando en ese segmento, y cuando pasamos al segmento siguiente, esta información si se va a utilizar en algún segmento posterior, se traslada también, copiándose en los registros que hay entre segmentos.

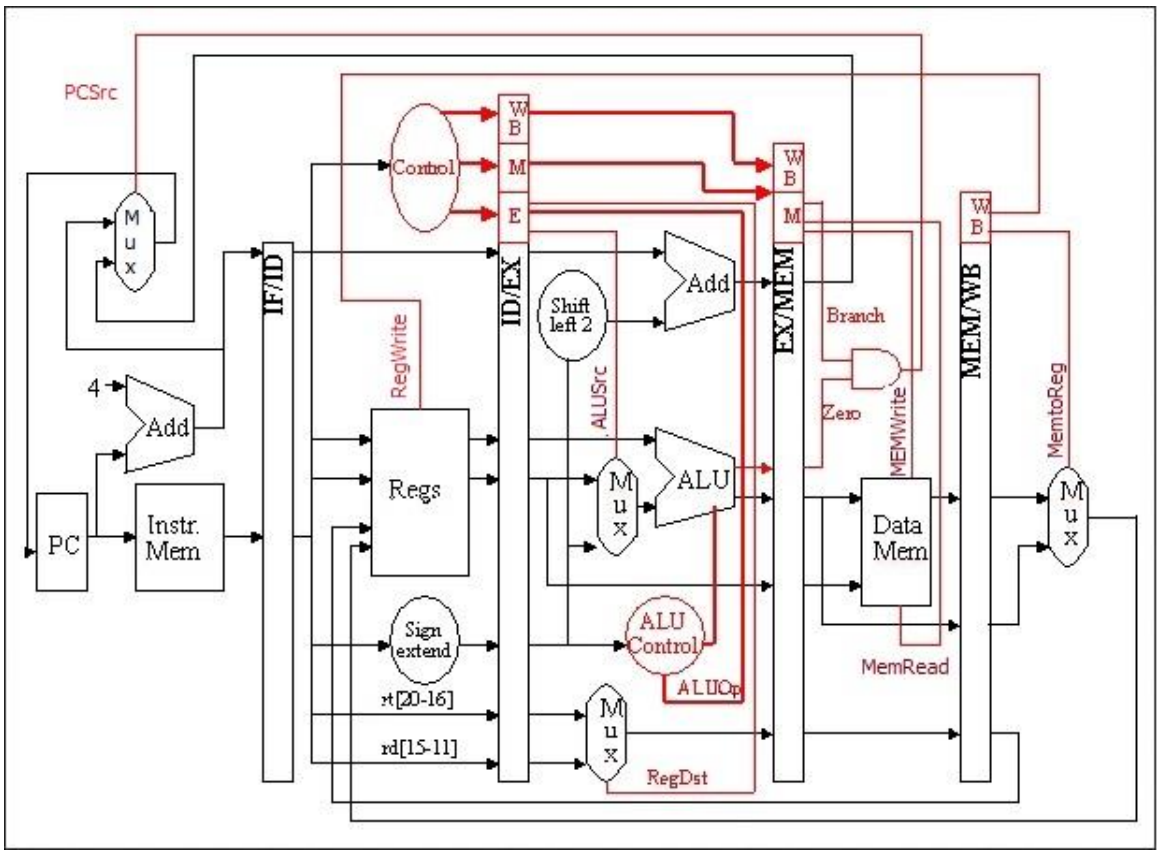

Figura 2.9. Camino de datos segmentado.

Hay situaciones que pueden disminuir el rendimiento del procesador segmentado debido a que provocan la detección del cauce, es decir, impiden que se ejecute la siguiente instrucción del flujo de instrucciones durante su ciclo de reloj. Estas circunstancias se denominan riesgos (hazards). Existen tres tipos de riesgos:

1. Riesgos estructurales: Surgen de los conflictos de los recursos, cuando el hardware no puede soportar todas las combinaciones posibles de instrucciones en ejecuciones solapadas simultáneamente. La solución de estos riesgos sería la

segmentación de unidades funcionales y duplicación de los recursos para permitir todas las posibles combinaciones.

2. Riesgos de control: Surgen cuando hay un cambio no secuencial en el valor del PC producido por un salto u otras instrucciones.
3. Riesgos de dependencia de datos: Surgen cuando una instrucción escribe un resultado en un determinado registro (durante el último segmento) y alguna de las instrucciones siguientes hace uso del valor de este registro antes de que se produzca dicha escritura. Como podemos ver en la figura 2.10 los riesgos de dependencia de datos se dividen en tres tipos, dependiendo del orden de los accesos de lectura y de escritura en las instrucciones. Consideremos dos instrucciones i y j, presentándose i antes que j.

   - RAW (lectura después de escritura Read After Write): la instrucción j intenta leer una fuente antes de que la escriba la instrucción i.
   - WAW (escritura después de escritura Write After Write): la instrucción j intenta escribir un operando antes de que sea escrito por la instrucción i.
   - WAR (escritura después de lectura Write After Read): la instrucción j intenta escribir un operando antes de que sea leído por la instrucción i.

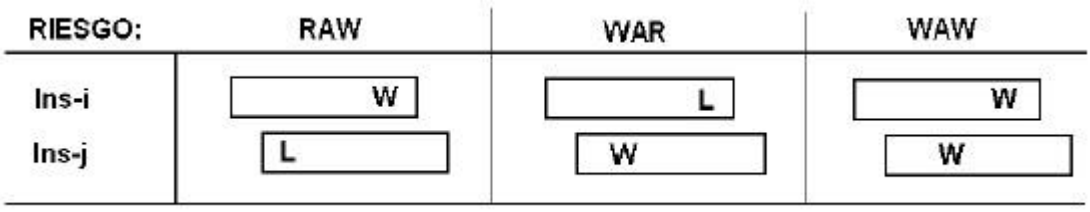

Figura 2.10. Tipos de riesgos de dependencia de datos.

Para poder solucionar los riesgos de dependencias de datos tenemos las siguientes soluciones:

- Bloqueo o burbuja: la forma más sencilla pero menos eficiente de resolver los riesgos por dependencia de datos en hardware es detener las instrucciones en la segmentación (introduciendo burbujas) hasta que se resuelva el riesgo. En la figura 2.11 podemos ver un ejemplo de esta solución.

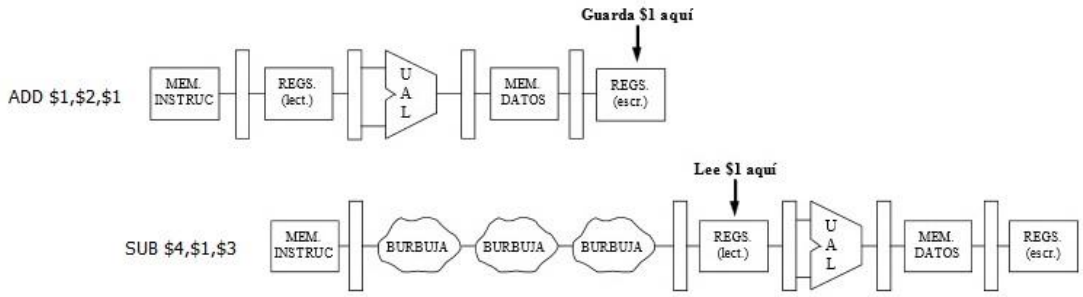

Figura 2.11. Ejemplo introducción de burbujas.

- Anticipación (forwarding): Con esta solución no se necesita esperar a que la instrucción se complete antes de intentar resolver el riesgo de datos. Tan pronto como el resultado se calcule se puede suministrar el resultado como entrada. En la figura 2.12 podemos ver un esquema simplificado de la ruta de datos con Anticipación donde podemos ver cómo se llevan los datos a los registros de entrada de la ALU antes o al tiempo que se llevan al destino. La unidad de control no se espera a que se escriba en el registro destino, sino que se lo pasa directamente a la ALU. En la figura 2.13 podemos ver un esquema simplificado de la anticipación de los resultados leídos de memoria.

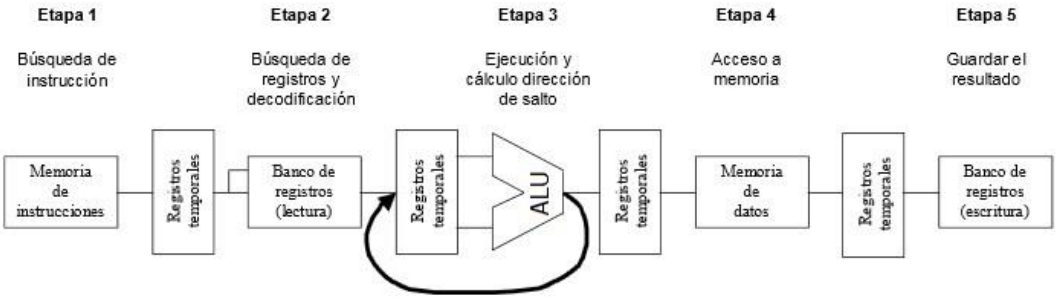

Figura 2.12. Esquema simplificado de la ruta de datos con anticipación.

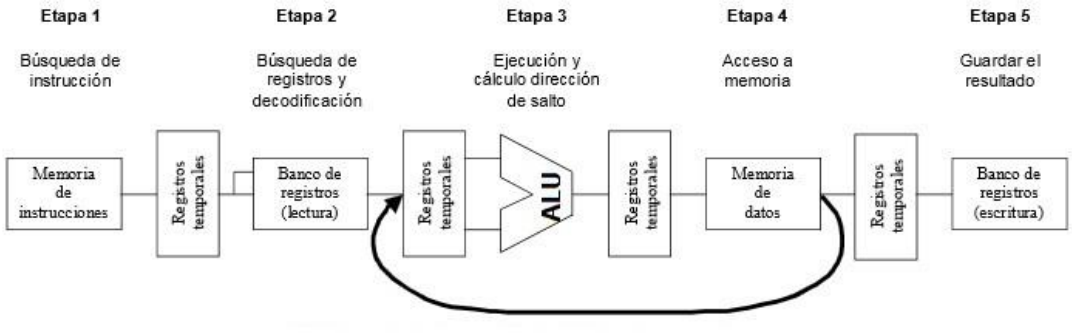

Figura 2.13. Anticipación de los resultados leídos de memoria.

- Reordenación de código: esta solución consiste en cambiar el orden de las instrucciones sin que afecte a la ejecución del programa.

2.3 MODELADO MIPS EN SIMULA3MS

El Simula3MS [7] [8] es un simulador de un procesador RISC que implementa un subconjunto de instrucciones básicas en el repertorio de instrucciones del procesador MIPS. La elección de que sea un procesador RISC es porque tiene una estructura y un repertorio de instrucciones más fáciles que los procesadores CISC, así su aprendizaje es mucho más rápido. Este simulador es configurable y tiene un entorno de trabajo sencillo. La interacción del usuario con el simulador es por medio de una interfaz gráfica implementada con java. En este trabajo hemos usado la herramienta Eclipse SDK [6] para poder modificar la interfaz gráfica.

Cuando abrimos el Simula3MS nos encontramos con la ventana del editor, como podemos ver en la figura 2.14. En esta ventana vamos a abrir el programa que queremos ejecutar (en nuestro caso el Ejemplo3.s, que se encuentra en el CD que acompaña a esta memoria) y tenemos varias opciones que elegir como por ejemplo, el camino de datos, que puede ser monociclo, multiciclo o segmentado, o el salto que puede ser retardado o fijo.

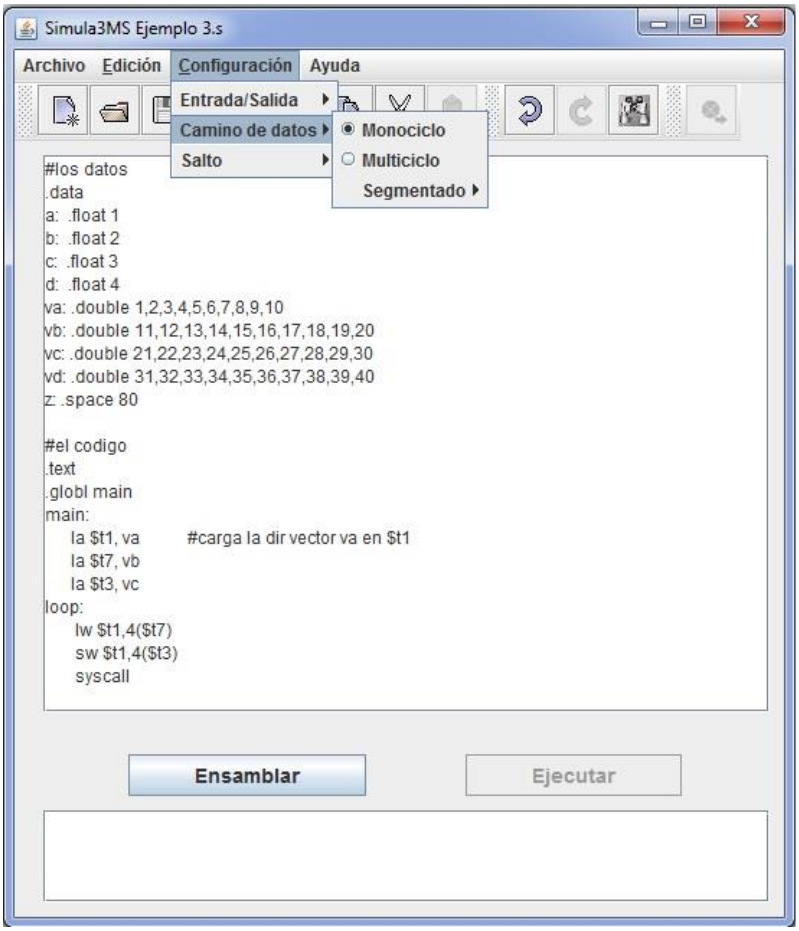

Figura 2.14. Ventana editor del Simula3MS.

En la figura 2.15 podemos ver el resultado de elegir la opción camino de datos monociclo. Como podemos ver, en la parte superior izquierda se muestran los registros, que se dividen en: registros especiales (PC, HI, LO…), registros generales y registros de punto flotante. En la parte superior derecha podemos ver el camino de datos correspondiente a un procesador monociclo, sobre el que se irá representando la ejecución de cada instrucción si vamos dando al botón *Ciclo siguiente*. El color de esta representación dependerá del tipo de instrucción. También podemos ejecutar el programa completo dando al botón *Ejecutar*. En la parte inferior derecha podemos ver el número de ciclos que han sido ejecutados hasta ese momento. El botón *Breakpoint* permite poner un punto de ruptura en el código.

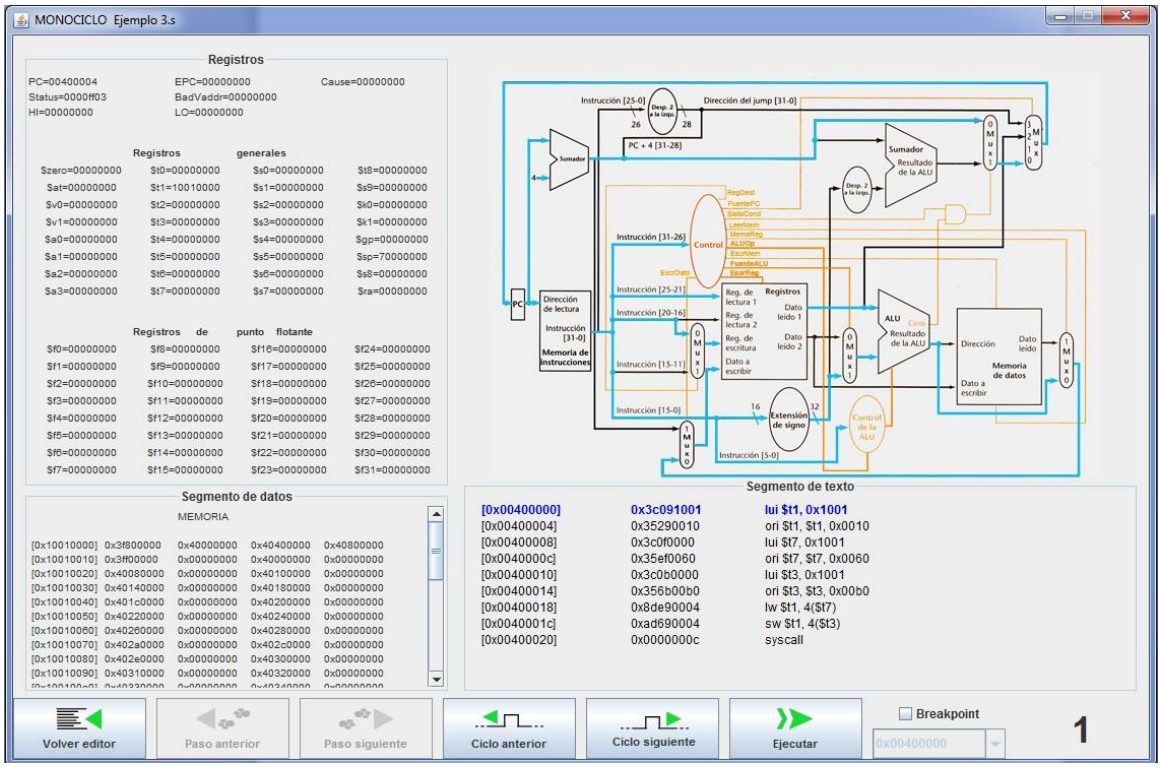

Figura 2.15. Procesador monociclo ejecutando el código *Ejemplo 3.s*.

En la figura 2.16 podemos ver la ejecución del Ejemplo 3.s para el procesador multiciclo. Como podemos ver, lo único que cambia respecto al monociclo es que se activan los botones *Paso anterior* y *Paso siguiente*. En este caso el color de la representación de la instrucción dependerá de la etapa en la que nos encontremos.

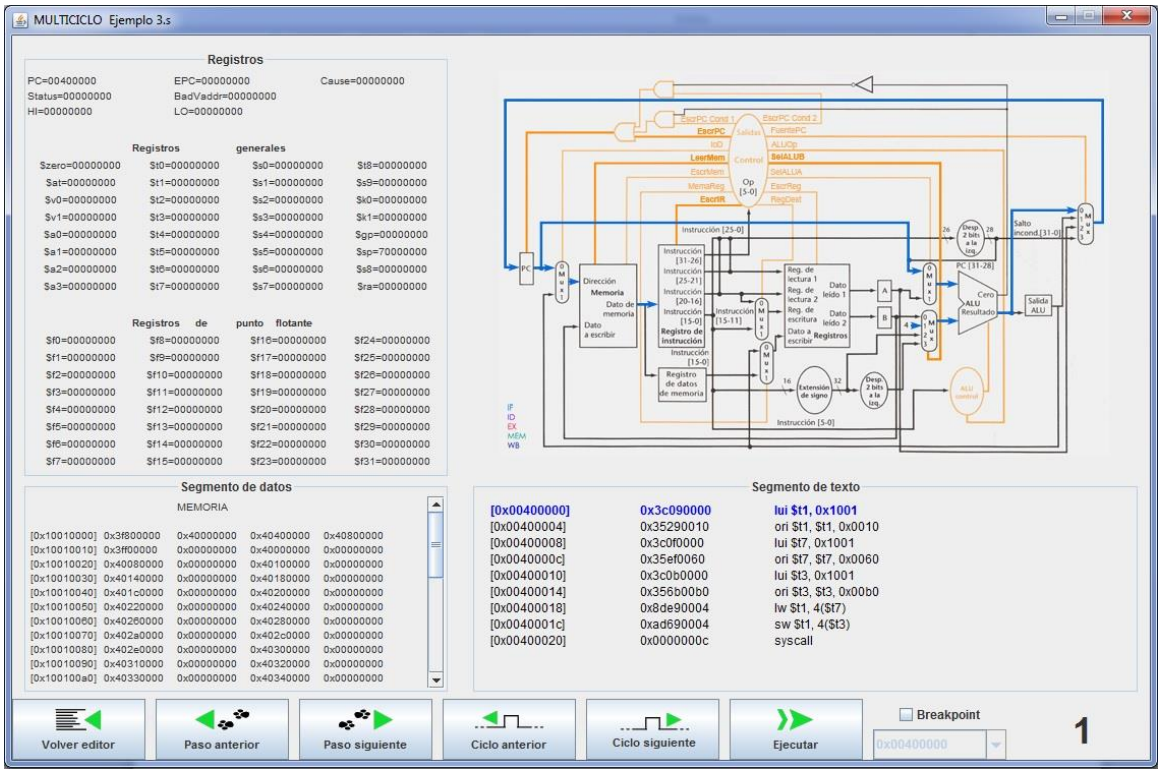

Figura 2.16. Procesador multiciclo ejecutando el código *Ejemplo 3.s*.

En la figura 2.17 podemos ver el resultado de elegir la opción camino de datos segmentado. El segmento de datos se puede dividir en dos partes: memoria y pila. La memoria a su vez está dividida en dos partes: el valor entre corchetes (valor de comienzo de la primera palabra de la línea), y las otras cuatro columnas (datos almacenados en la memoria de forma consecutiva). La pila crece en direcciones inferiores y está dividida en dos partes: la primera columna (posición del puntero de la pila) y las restantes (los datos). El segmento de texto contiene las instrucciones del programa. Se divide en tres columnas: la primera es la dirección hexadecimal de memoria (PC) de la instrucción (las instrucciones se almacenan a partir de la dirección 0x0040000), la segunda es la codificación de la instrucción en hexadecimal y la tercera es la instrucción en lenguaje ensamblador. La instrucción que se está ejecutando está resaltada en azul (el color dependerá del tipo de instrucción). En esta opción de camino de datos segmentado se incluye también un Diagrama multiciclo, como puede verse en la figura 2.18, donde vamos viendo las etapas por las que van pasando las instrucciones, y un Diagrama monociclo, como puede verse en la figura 2.19, donde vamos viendo cómo la instrucción va pasando por las unidades funcionales.

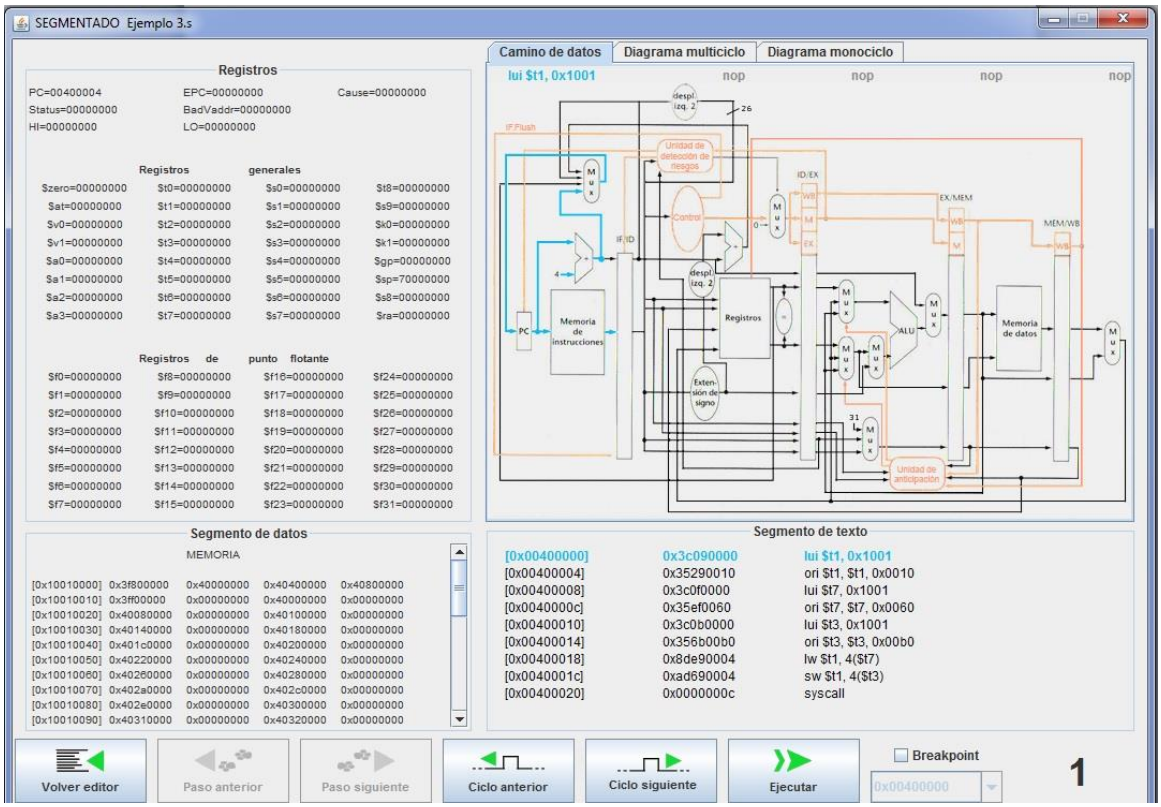

Figura 2.17. Procesador segmentado ejecutando el código *Ejemplo 3.s*.

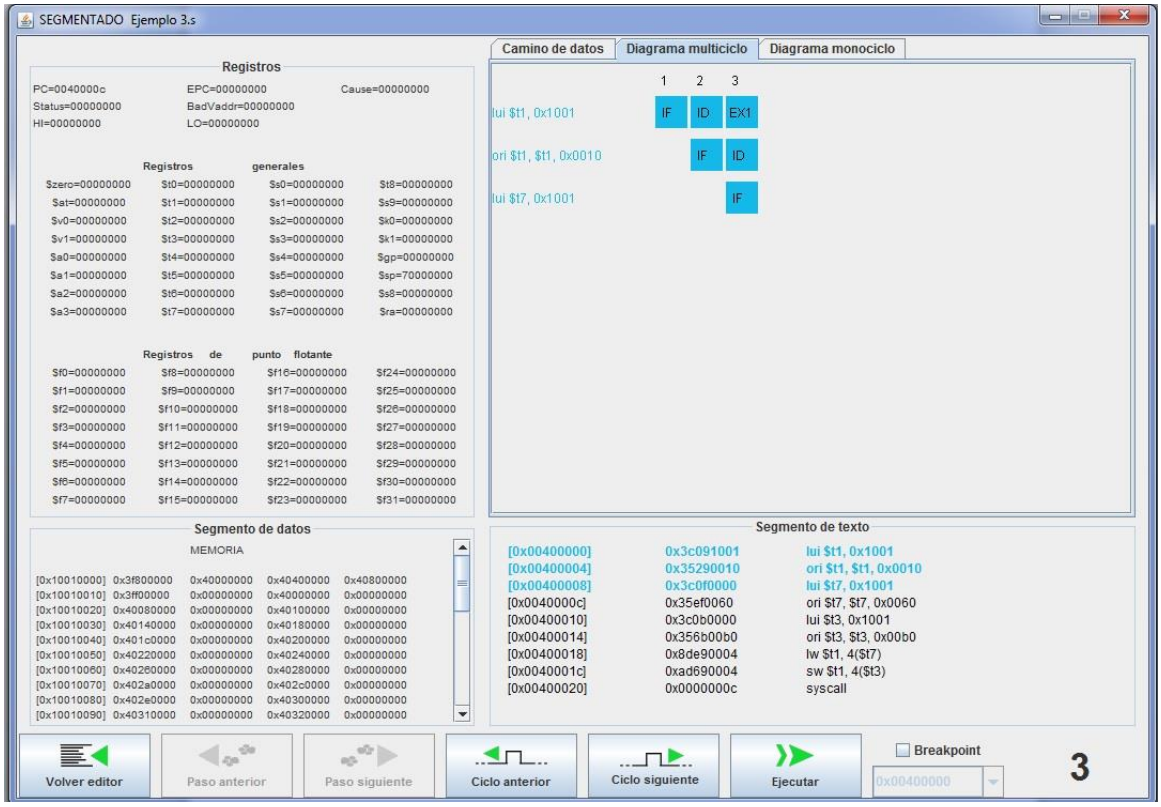

Figura 2.18. Diagrama multiciclo para la ejecución del código *Ejemplo 3.s*.

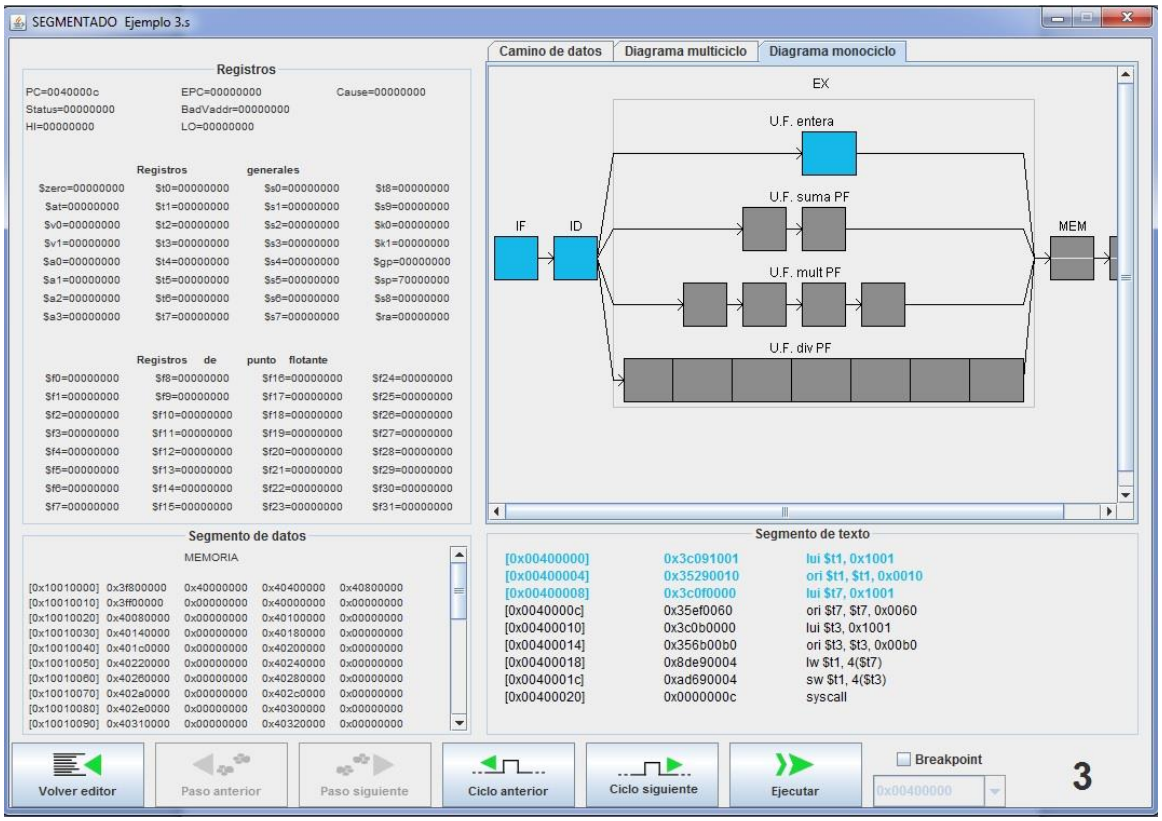

Figura 2.19. Diagrama monociclo para la ejecución del código *Ejemplo 3.s*.

En este capítulo hemos podido ver de una manera general la arquitectura MIPS, el camino de datos monociclo, multiciclo y segmentado y cómo se modela la arquitectura MIPS en el Simula3MS. Tras conocer las nociones básicas aquí explicadas, podremos entender mejor el objetivo de este trabajo, que es la implementación de la opción sin adelantamiento en el Simula3MS.

CAPÍTULO 3. DISEÑO E IMPLEMENTACIÓN DEL CAUCE MIPS SIN ADELANTAMIENTO

En este capítulo vamos a detallar el trabajo que se ha realizado para implementar en Simula3MS el cauce MIPS sin adelantamiento. Se irán explicando todas las clases que se han modificado en este trabajo, ya que no se ha tenido que añadir ninguna clase nueva al trabajo original, pero sí ha sido necesario modificar muchas de las que ya había.

3.1 MAPA DE CLASES

En este apartado veremos un mapa de clases para hacernos una idea de cómo está estructurado el simulador Simula3MS internamente. Como podemos ver en el mapa de clases de la figura 3.1, el simulador se divide en varias carpetas, cada una de las cuales contiene las clases pertenecientes a esa carpeta. Las clases que tenemos señaladas en rojo son aquellas que hemos modificado para poder conseguir llegar al objetivo final del TFG.

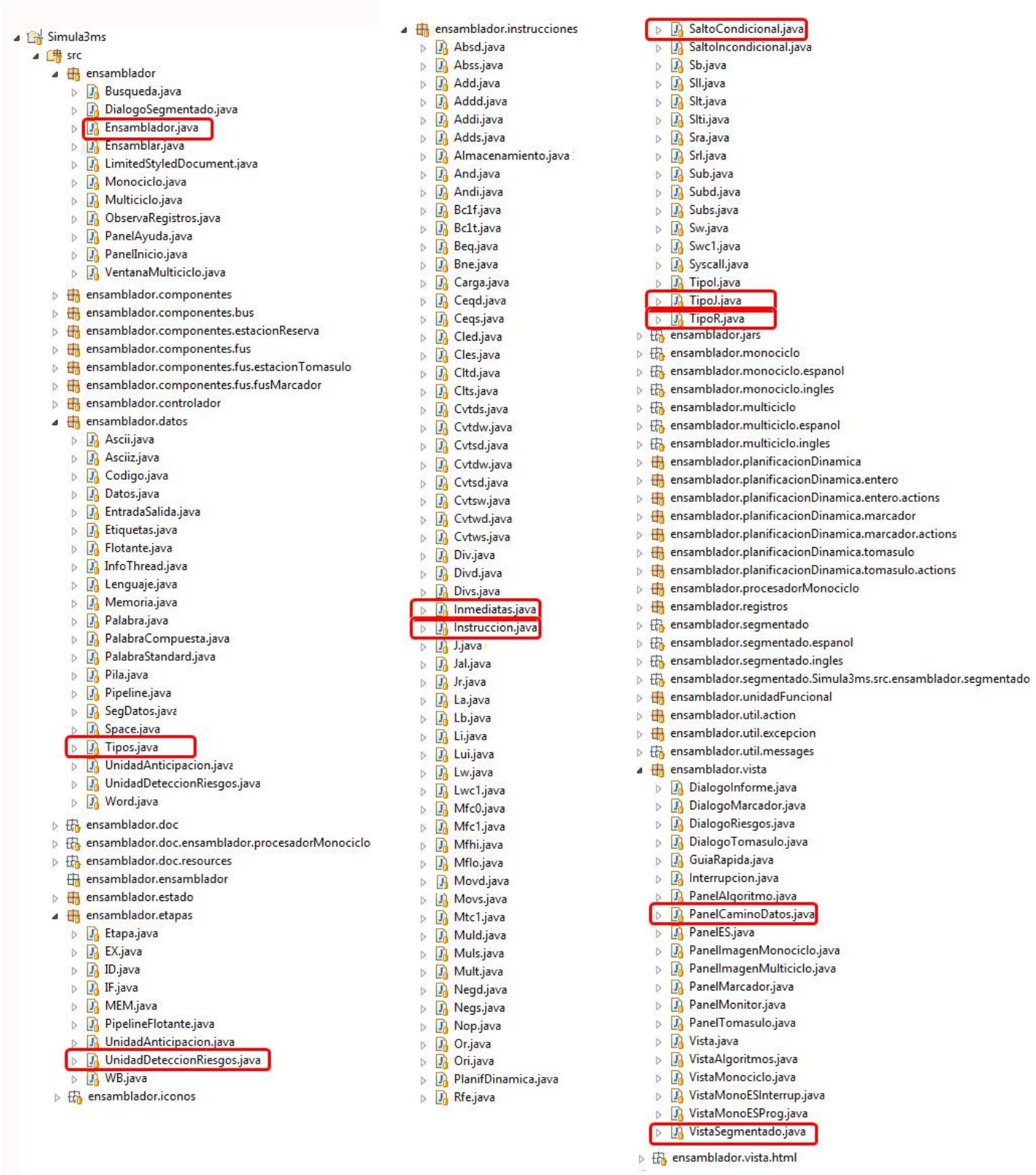

Figura 3.1. Mapa de clases del Simula3MS.

3.2 PLANIFICACIÓN DE LAS MODIFICACIONES

Una vez analizado el código de Simula3S, podemos localizar las clases en las que es necesario introducir modificaciones para implementar el cauce MIPS sin adelantamiento. Dichas modificaciones se detallan en las secciones siguientes, pero se resumen en la siguiente lista:

- En primer lugar se verá la clase *Ensamblador.java,* la cual hemos modificado para que la ventana de inicio tenga las dos opciones posibles: *Con Adelantamiento*, opción ya implementada en el ensamblador original, y *Sin Adelantamiento*, opción que hemos implementado en este trabajo.
- La clase *UnidadDetencionderiesgos.java* es la más importante en este trabajo porque es la encargada de detener el cauce hasta que el dato esté disponible. Por lo tanto, mediante esta clase vamos a gestionar si el cauce debe estar detenido hasta que el dato esté disponible y por lo tanto tendremos nuestro cauce sin adelantamiento, propósito de este trabajo, o por el contrario, dejarlo como está originalmente adelantando el dato siempre que sea posible.
- Para todos los tipos de instrucciones *Inmediatas.java*, *SaltoCondicional.java*, *Tipo I.java*, *Tipo R.java* es necesario modificar la forma en la que se ilumina gráficamente el camino de datos para poder mostrar el camino de datos sin adelantamiento.
- La clase *PanelCaminoDatos.java* se modificará para que pueda dibujar la traza de ejecución según las opciones que hayamos elegido.

Por último, veremos unos casos excepcionales del cauce MIPS con adelantamiento que han tenido que ser modificados para su correcto funcionamiento.

3.3 IMPLEMENTACIÓN DE LA VENTANA DE INICIO

Para poder modificar el código e implementar el cauce MIPS sin adelantamiento, hemos usado la herramienta Eclipse SDK [6], la cual nos permite gestionar el código del Simula3MS.

Como muestra la figura 3.2 la ventana de inicio del simulador original sólo mostraba la opción Camino de Datos→Segmentado→Básico, donde implementa el cauce MIPS con adelantamiento. El objetivo final de este trabajo es implementar el cauce MIPS sin adelantamiento por lo que, en primer lugar, modificaremos la ventana de inicio del simulador para que permita elegir esta nueva opción.

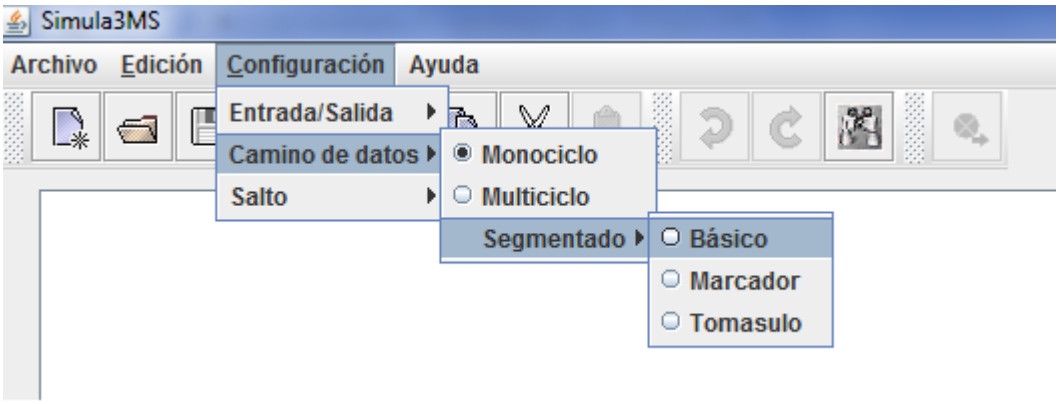

Figura 3.2. Ventana inicio del simulador original.

Para poder modificar esta ventana inicial del simulador modificaremos la clase *Ensamblador.java*. En esta clase, añadiremos a la opción "Básico" dos opciones "Con Adelantamiento" y "Sin Adelantamiento", como puede verse en la figura 3.3.

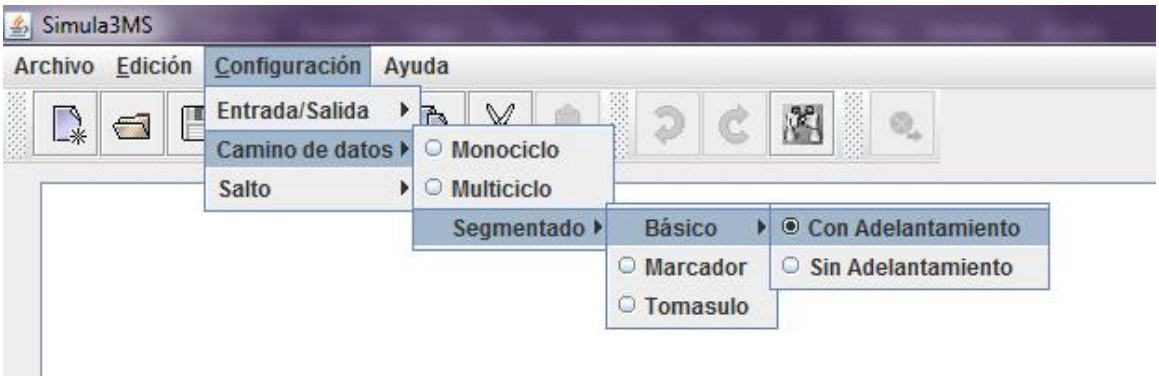

Figura 3.3. Ventana inicio del simulador tras la modificación realizada.

3.4 IMPLEMENTACIÓN INTERNA PARA EL CAUCE SIN ADELANTAMIENTO

En este apartado veremos todas las modificaciones introducidas en el código para poder ejecutar código en un cauce sin adelantamiento. Veremos las clases que hemos tenido que modificar para poder llegar a nuestro objetivo final.

La clase *Tipos.java* es la que recoge todos los valores de todas las posibles opciones que tiene el simulador. En el simulador original, como sólo incluye la opción de adelantamiento, no necesita ningún tipo para controlar esta opción. En nuestro caso, como tenemos dos opciones, "Con Adelantamiento" y "Sin Adelantamiento", sí necesitamos dos nuevos tipos de constantes que nos digan si la opción elegida es "Con Adelantamiento" o,

por el contrario, “Sin Adelantamiento”. Para ello, incluimos en el código dos nuevas constantes:

```
int CON_ADELANTAMIENTO = 0;
int SIN_ADELANTAMIENTO = 1;
```

Con estas constantes lo que haremos es comprobar si la opción elegida es una u otra.

La clase *UnidadDeteccionRiesgos.java* es la clase que va gestionando las instrucciones en cada etapa. En el simulador original, como estaba implementada sólo la opción de adelantamiento, en esta clase cuando una instrucción necesitaba un dato o registro que estaba siendo calculado por otra instrucción previa, se le adelantaba en la etapa adecuada.

La clase *UnidadDeteccionRiesgos.java* se modificará poniendo la opción de que pueda haber adelantamiento o no. Esta modificación consistirá en incluir en el método *public boolean detener (int adelantamiento, int salto)* un parámetro llamado adelantamiento que controlará si hay o no adelantamiento. Ahora, tras la modificación, en esta clase, si el parámetro adelantamiento es igual a uno, estaremos en la opción sin adelantamiento, por lo cual, si una instrucción necesita un dato o un registro que está siendo calculado por otra instrucción previa se esperará hasta que dicha instrucción o registro esté ya escrito en el banco de registros.

3.5 IMPLEMENTACIÓN PARA LA VISUALIZACIÓN DEL CAUCE SIN ADELANTAMIENTO

En este apartado vamos a ver las modificaciones realizadas para que además de poder mostrar gráficamente el camino de datos con adelantamiento, como hace el simulador original Simula3MS, nos pueda mostrar también el camino de datos sin adelantamiento, así como el diagrama multiciclo y monociclo correcto.

En la clase *Instruccion.java* se ha añadido el método *estaLibreRegistro ()* que nos dirá si el registro está libre.

Para las clases *Inmediatas.java*, *SaltoCondicional.java*, *Tipo I.java*, *Tipo R.java* incluiremos las modificaciones necesarias para que el camino de datos sin adelantamiento sea el correcto. En cada clase hemos introducido una variable llamada *adelantamiento* donde si la variable es igual a cero nos dibuja el camino de datos que existía en el simulador original, pero si la variable es igual a uno dibuja el camino de datos nuevo que hemos creado para el cauce sin adelantamiento.

La clase *VistaSegmentado.java* es la clase encargada de visualizar la ruta de datos del procesador segmentado. La modificación realizada en esta clase es para pasar como parámetro si hay o no adelantamiento, para que dependiendo del valor que tenga ese nuevo parámetro se muestre el camino de datos con adelantamiento, como en el simulador original, o el camino de datos sin adelantamiento, tal y como se ha añadido en este trabajo.

La clase *PanelCaminoDatos.java* es la clase encargada de la visualización de la representación monociclo del camino de datos segmentado. La modificación realizada en esta clase es introducir como parámetro si hay o no adelantamiento, para que dependiendo del valor que tenga ese nuevo parámetro se muestre el diagrama multiciclo con adelantamiento, como en el simulador original, o el multiciclo sin adelantamiento tal y como se ha añadido en este trabajo.

3.6 MODIFICACIONES DE LAS IMÁGENES DEL CAMINO DE DATOS

En primer lugar se ha modificado la imagen del camino de datos original con adelantamiento para añadir un multiplexor a la entrada de la memoria de datos. Este multiplexor es necesario para incluir un adelantamiento desde la etapa WB, para así evitarnos esperar ciclos de espera innecesarios, como se ve en la figura 3.4. En esta figura aparecen señaladas en rojo las principales modificaciones introducidas.

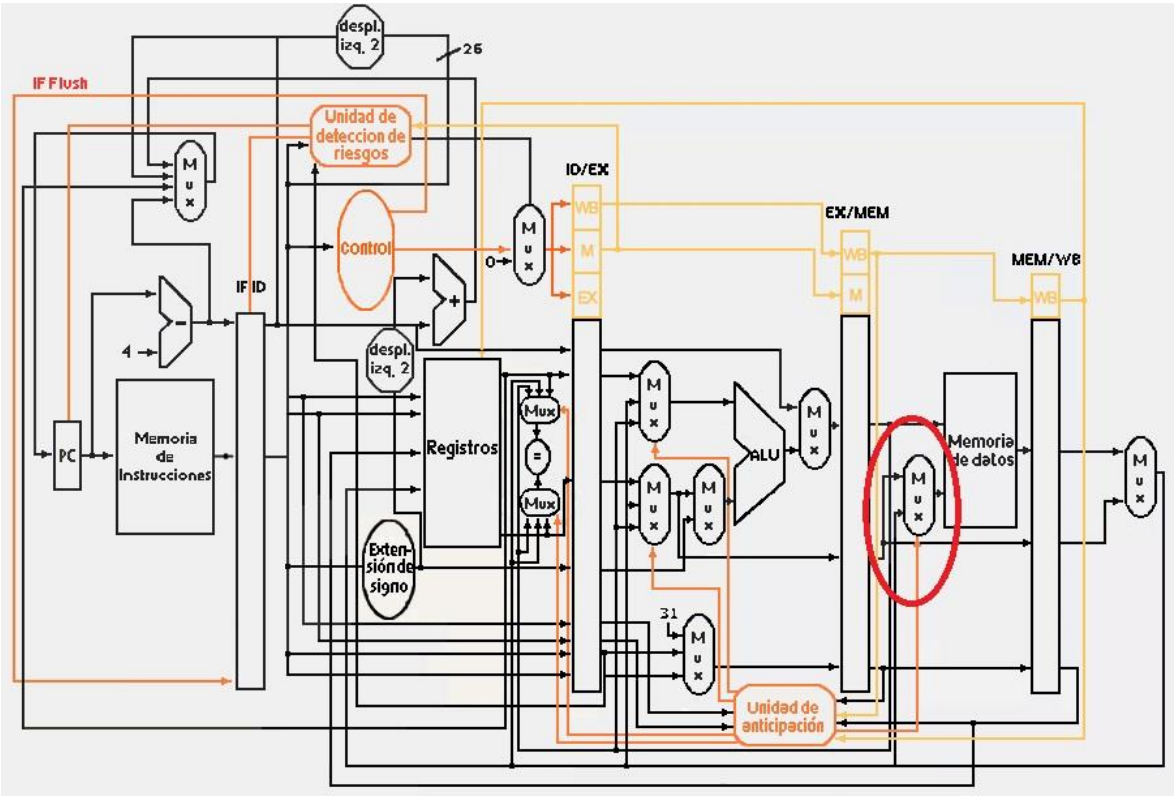

Figura 3.4. Nueva imagen del camino de datos segmentado con adelantamiento.

Para el caso de la nueva implementación del cauce sin adelantamiento se ha creado una imagen nueva, figura 3.5, dónde vemos cómo se ha eliminado la circuitería relativa al adelantamiento. Podemos observar cómo ha desaparecido la unidad de anticipación y todas las líneas que llegan o salen de ella.

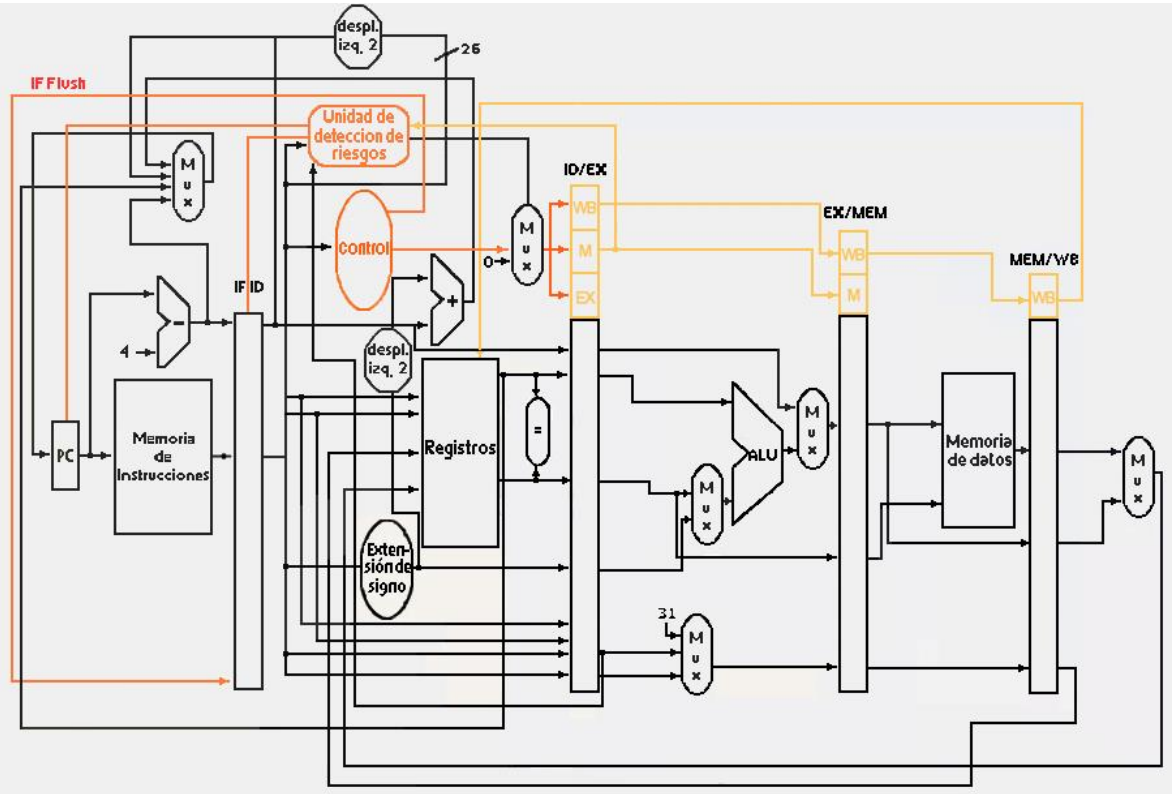

Figura 3.5. Nueva imagen del camino de datos segmentado sin adelantamiento.

Para modificar todas las imágenes de la nueva versión del simulador, cuyo formato es .gif, hemos utilizado la herramienta Photoshop. De este modo hemos podido conservar la calidad de las imágenes del simulador original.

3.7 CASOS ERRÓNEOS DEL CAUCE MIPS CON ADELANTAMIENTO EN EL SIMULA3MS ORIGINAL

Mientras hemos estado comprobando que el simulador se comporta correctamente haciendo el camino de datos sin adelantamiento, también nos hemos dado cuenta que en el camino de datos con adelantamiento había ciertos errores y, en esos casos, el simulador se comportaba de forma errónea a como se tenía que comportar. Para ello hemos tenido que modificar el código para que el simulador se comportara de forma correcta.

El primer caso excepcional que veremos en este apartado tiene que ver con la etapa EX. En el simulador original, en esta etapa siempre ponía EX1 aunque la instrucción fuera entera, lo que vimos que no era lógico ya que para una instrucción entera sólo existe una etapa EX.

Podemos ver la modificación en la figura 3.6. Con las modificaciones que hemos realizado, en una instrucción entera llamaremos a la etapa de ejecución EX y no EX1 como hacía anteriormente, dejando esa etiqueta sólo cuando se trata de una instrucción en coma flotante, que sí tiene varios ciclos de ejecución en la etapa EX.

Para ello, hemos modificado la clase *EX.java* donde hemos rectificado la enumeración haciendo que sólo se enumere la etapa EX en caso de que la instrucción sea en coma flotante.

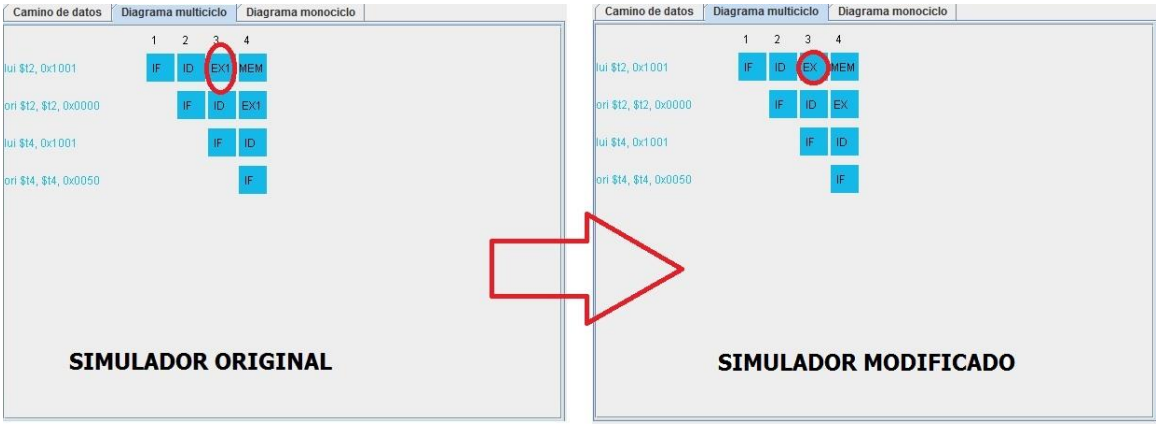

Figura 3.6. Cambios realizados en la etapa EX.

El siguiente caso excepcional que hemos encontrado es con la instrucción `add` seguida de la instrucción `sw` con una dependencia entre ellos del dato a guardar en memoria en el almacenamiento. En este caso, el adelantamiento se hacía en la etapa equivocada, se puede ver este caso en la figura 3.7 para el código Ejemplo 5.s (incluido en el CD que acompaña esta memoria). Para solucionar este error hemos tenido que incluir un multiplexor nuevo, como ya se indicó en el apartado anterior de este capítulo, en la figura de la ruta de datos, además de cambiar el adelantamiento en la etapa adecuada, como podemos ver en la figura 3.8. Hay que señalar que si la dependencia entre esas dos instrucciones es respecto al otro operando del almacenamiento (con el que se calcula la dirección efectiva de acceso a memoria), entonces el adelantamiento sí que hay que hacerlo a la etapa EX, tal cual y como ya se hacía en el simulador original.

En el simulador original la instrucción `add` adelanta el registro que necesita la instrucción `sw` en la etapa EX, cuando lo correcto sería adelantarlo en la etapa MEM, que es quien realmente necesita ese dato.

Para que este caso funcione correctamente hemos tenido que modificar la clase *UnidadAnticipacion.java*, donde hemos añadido el caso de que la instrucción siguiente sea un almacenamiento.

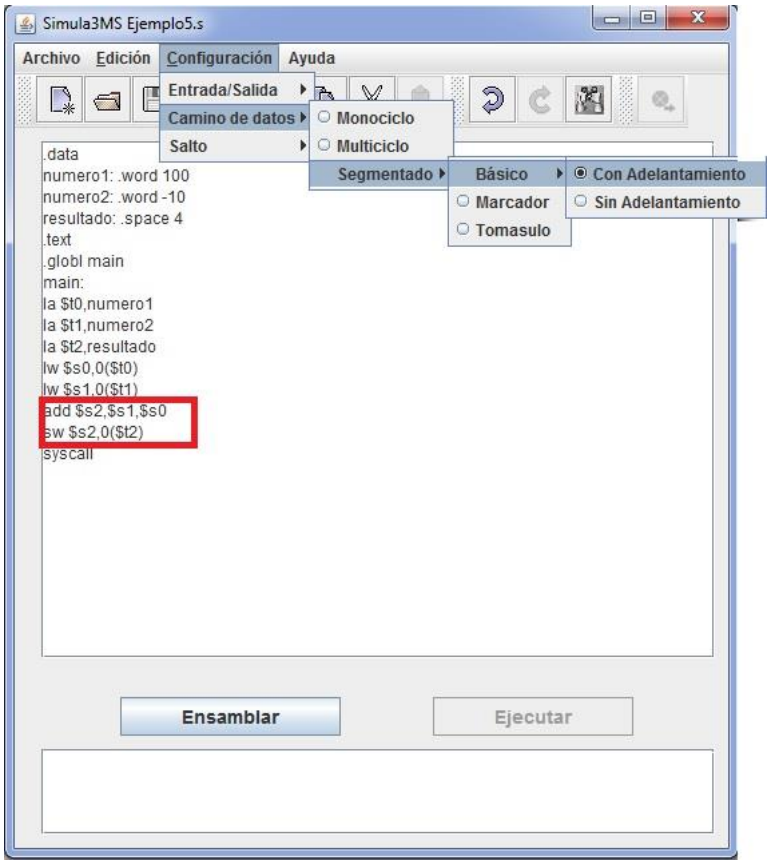

Figura 3.7. Ejemplo 5.s en el simulador.

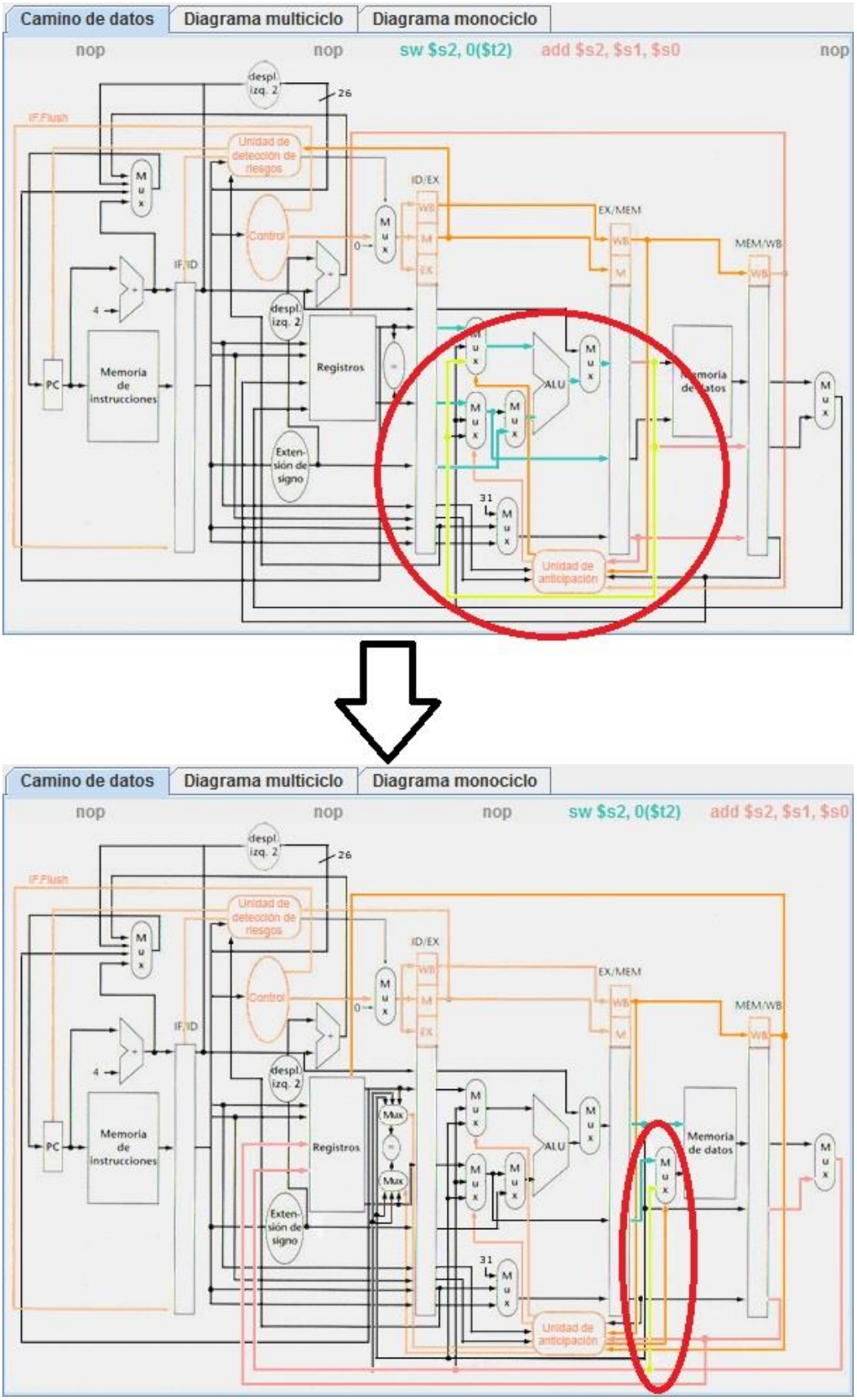

Figura 3.8. Adelantamiento con una instrucción `sw`.

CAPÍTULO 4. PRUEBAS DE SISTEMA

En este apartado veremos los resultados del Simula3MS después de haberlo modificado e incorporar la opción de ejecución SIN ADELANTAMIENTO.

Veremos seis ejemplos con dependencia de datos para comprobar si la ejecución sin adelantamiento se realiza correctamente. Los seis ejemplos serán:

- Dependencia de datos entre dos instrucciones aritméticas enteras.
- Dependencia de datos entre dos instrucciones aritméticas de punto flotante.
- Dependencia de datos entre una instrucción de carga seguida de una instrucción aritmética.
- Dependencia de datos entre una instrucción de carga y una instrucción de almacenamiento en el primer operando del almacenamiento.
- Dependencia de datos entre una instrucción de carga y una instrucción de almacenamiento en el segundo operando del almacenamiento.
- Dependencia de datos entre una instrucción aritmética y una instrucción de almacenamiento.

Estos seis ejemplos cubren todas las posibilidades de dependencias de datos y con ello quedaría comprobada la correcta implementación del cauce sin adelantamiento hardware en la nueva versión del simulador Simula3MS. En las siguientes secciones se muestran los resultados para cada uno de los casos.

4.1 DEPENDENCIA DE DATOS ENTRE DOS INSTRUCCIONES ARITMÉTICAS ENTERAS

En este caso usaremos el código llamado *Ejemplo1.1.s* (que se encuentra disponible en el CD que acompaña a esta memoria) donde estudiaremos el caso de una suma (`add`) y resta (`sub`) enteras, donde cada instrucción depende de los operandos de la anterior. Este código puede observarse en la figura 4.1.

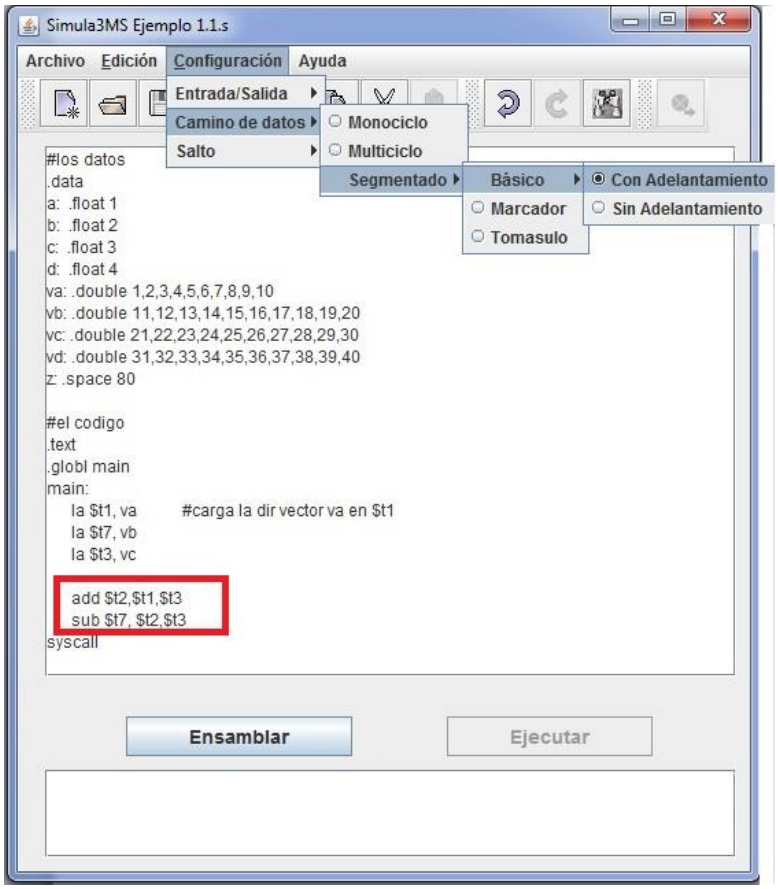

Figura 4.1. *Ejemplo 1.1.s*, código utilizado para evaluar una dependencia aritmética entera.

Ejecutamos el código con adelantamiento, es decir, con la implementación que venía en el simulador original, y nos muestra el resultado de la figura 4.2. Como vemos en esta figura, la instrucción *`add`* adelanta el dato $t2 a la instrucción *`sub`*. En este ejemplo también podemos ver que la instrucción `ori` adelanta el dato $t3 a la instrucción `sub`.

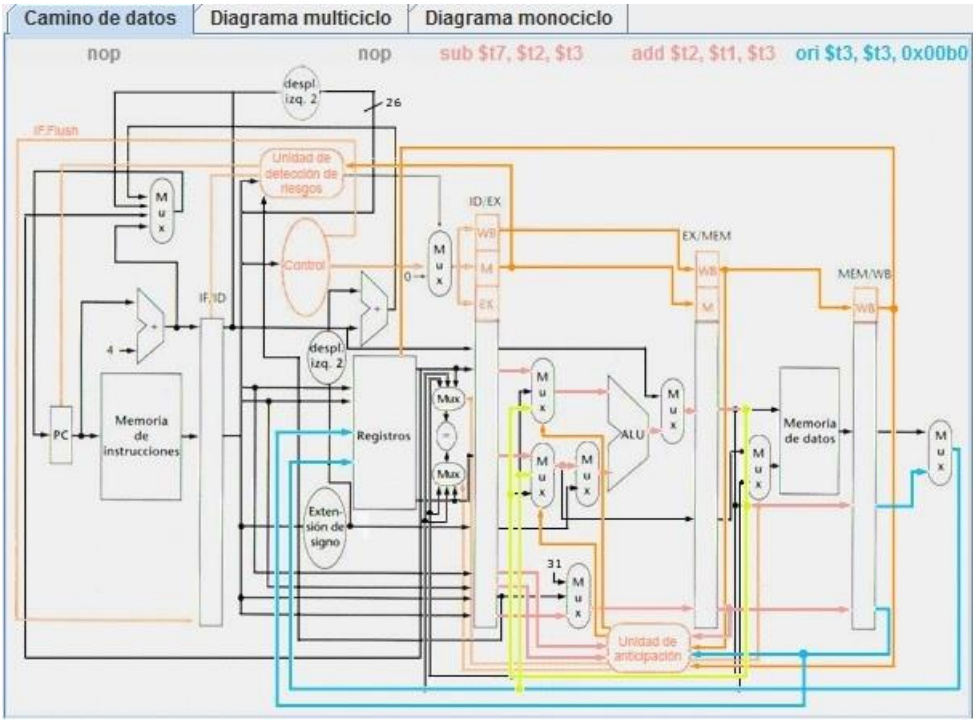

Figura 4.2. *Ejemplo 1.1.s* ejecutando en el camino de datos con adelantamiento de $t2 de `add` a `sub`.

Veamos ahora cómo se comporta el procesador cuando hemos escogido el camino de datos sin adelantamiento. Como vemos en la figura 4.3, cuando se elige la opción sin adelantamiento la instrucción `sub` se espera hasta que la instrucción `add` ha escrito el resultado de la suma en el registro $t2, y a continuación la instrucción `sub` sigue su curso.

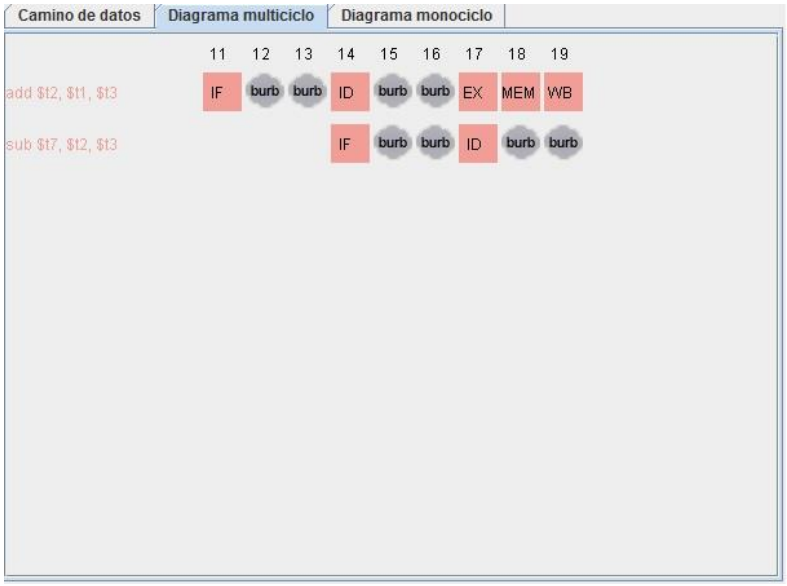

Figura 4.3. Diagrama multiciclo para el *Ejemplo 1.1.s* sin adelantamiento.

Como es obvio, el número de ciclos de la ejecución sin adelantamiento es mayor al número de ciclos de la ejecución con adelantamiento. Además, en la ejecución sin adelantamiento se incrementan las detenciones y por tanto los riesgos de datos, tal y como puede observarse en la figura 4.4.

CON ADELANTAMIENTO **SIN ADELANTAMIENTO**

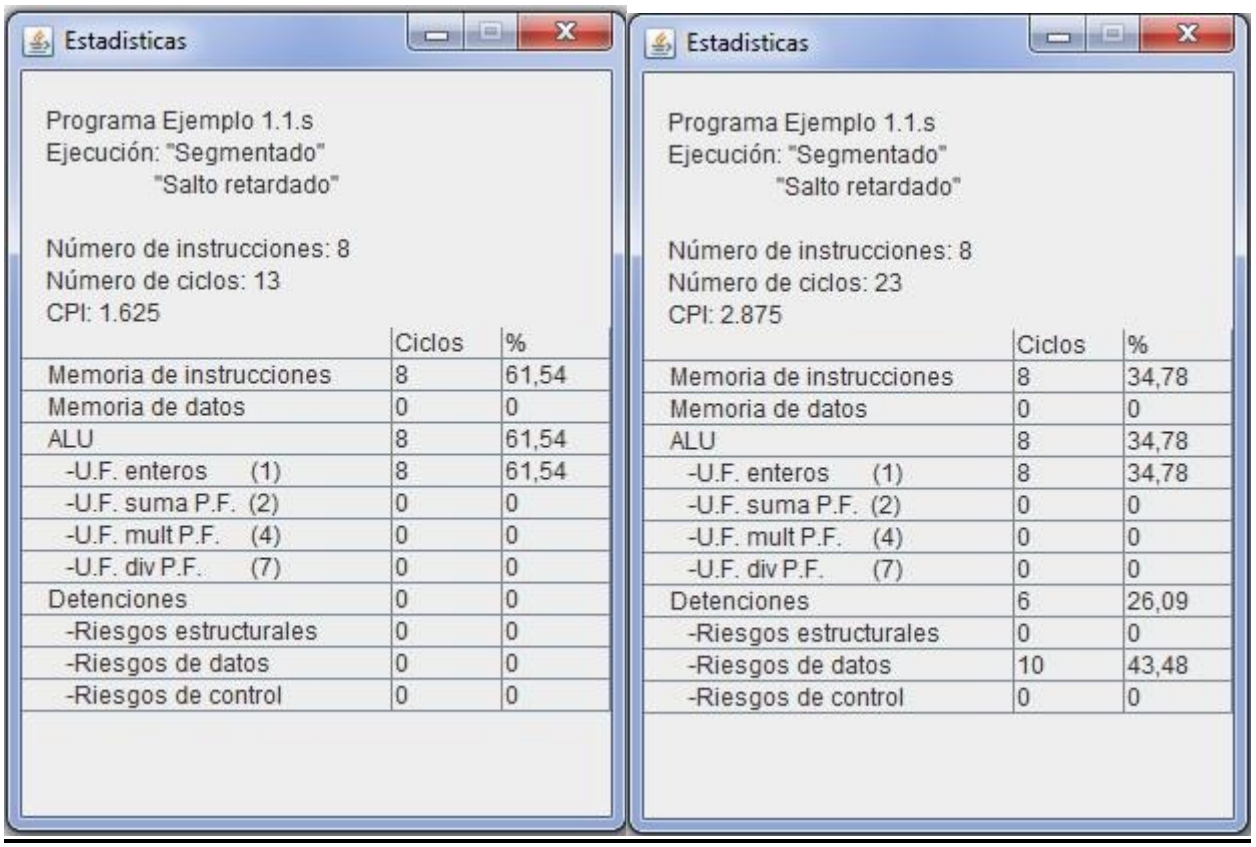

Figura 4.4. Comparación de las estadísticas aplicando adelantamiento (izquierda) y sin aplicarlo (derecha).

4.2 DEPENDENCIA DE DATOS ENTRE DOS INSTRUCCIONES ARITMÉTICAS DE PUNTO FLOTANTE

En este caso usaremos el código llamado *Ejemplo 1.s* (que se encuentra disponible en el CD que acompaña a esta memoria) donde estudiaremos los casos de una multiplicación

(`mul.d`), suma (`add.d`) y resta (`sub.d`) en coma flotante, donde cada instrucción depende de los operandos de la anterior. Este código puede observarse en la figura 4.5.

Ejecutamos el código con adelantamiento y nos muestra el resultado de la figura 4.5.

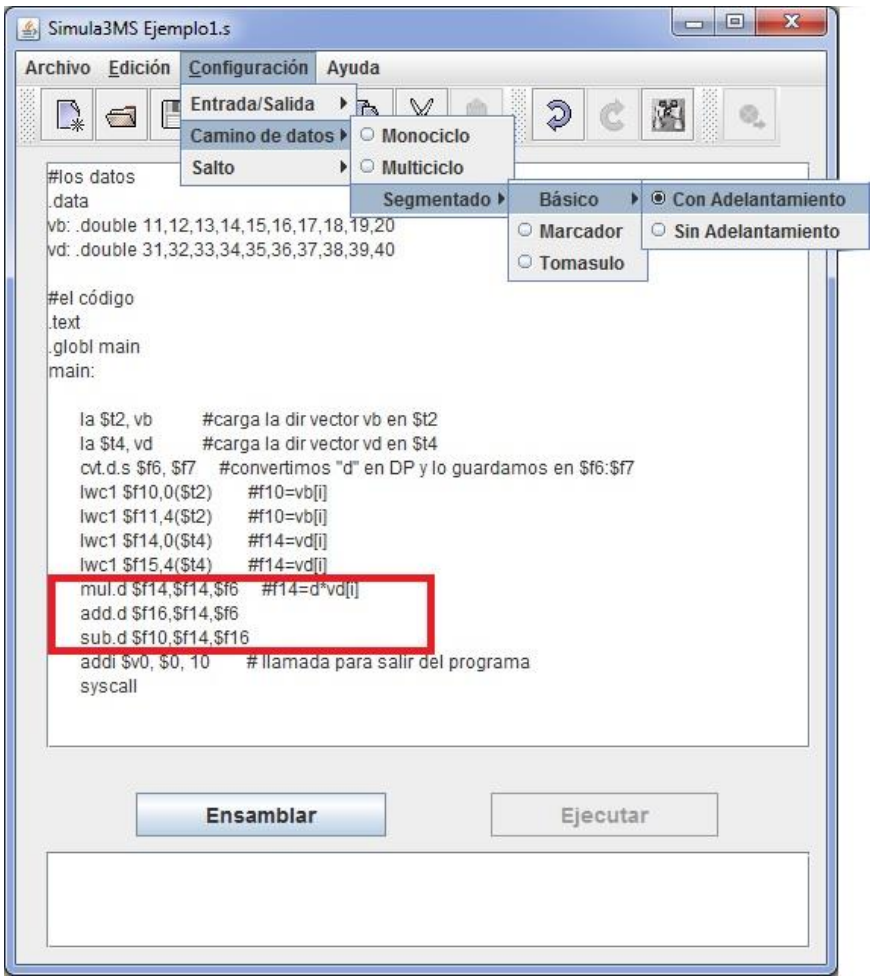

Figura 4.5. *Ejemplo 1.s*, código utilizado para evaluar una dependencia aritmética.

Veamos cómo se adelantan los datos para que las instrucciones puedan seguir avanzando. Como estas instrucciones son en coma flotante y el camino de datos sólo dibuja las instrucciones enteras, vamos a ver el diagrama multiciclo.

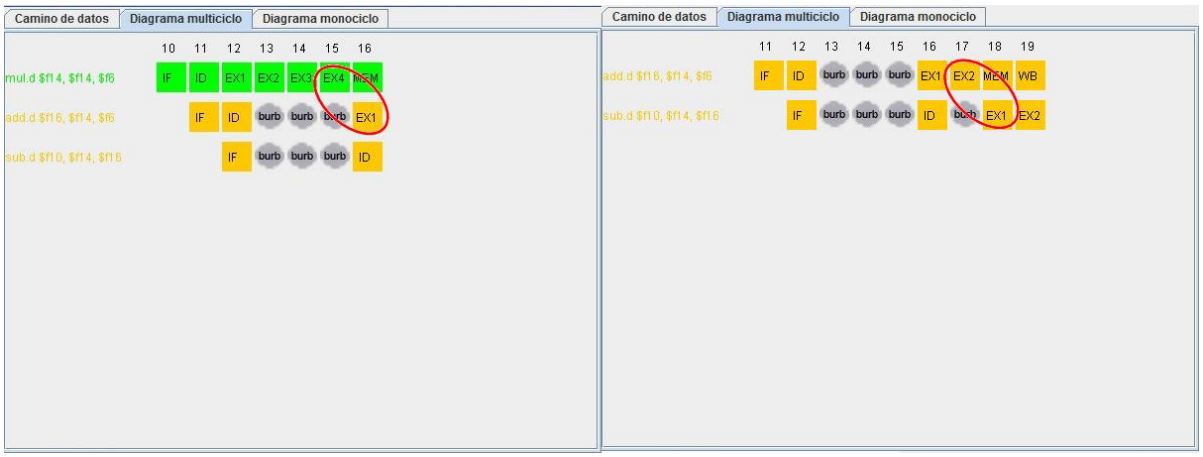

Figura 4.6. Diagrama multiciclo para el *Ejemplo 1.s* con adelantamiento.

En la figura 4.6, en la imagen de la izquierda podemos ver cómo la multiplicación adelanta el dato a la suma desde la etapa MEM a la etapa EX1 de la suma, y en la imagen de la derecha podemos ver que la instrucción `add` adelanta el dato a la resta en la etapa MEM.

Veamos ahora cómo se comporta el procesador cuando hemos escogido el camino de datos sin adelantamiento. En la figura 4.7, en la imagen de la izquierda podemos ver que la multiplicación no adelanta el dato a la suma, por lo que la suma se tiene que esperar a que la multiplicación escriba en el registro $f14. En la imagen de la derecha podemos ver como la instrucción `add` no adelanta el dato a la resta, por lo que la resta tiene que esperarse a que la suma escriba el dato en $f16 en su etapa WB, procediendo entonces la otra instrucción a leer el dato del banco de registros. Hay que recordar que las escrituras y lecturas en el banco de registros se pueden solapar pues son operaciones rápidas que pueden realizase en medio ciclo de reloj. De hecho, en el primer medio ciclo se escribe en el banco de registros procediéndose en el segundo medio ciclo a la lectura en dicho banco.

Figura 4.7. Diagrama multiciclo *Ejemplo 1.s* sin adelantamiento.

Como es obvio, el número de ciclos de la ejecución sin adelantamiento es mayor al número de ciclos de la ejecución con adelantamiento. Además en la ejecución sin adelantamiento se incrementan las detenciones y por tanto los riesgos de datos, tal y como puede observarse en la figura 4.8.

CON ADELANTAMIENTO

Estadisticas

Programa Ejemplo1.s
Ejecución: "Segmentado"
"Salto retardado"

Número de instrucciones: 13
Número de ciclos: 22
CPI: 1.6923077

	Ciclos	%
Memoria de instrucciones	13	59,09
Memoria de datos	4	18,18
ALU	19	86,36
-U.F. enteros (1)	9	40,91
-U.F. suma P.F. (2)	6	27,27
-U.F. mult P.F. (4)	4	18,18
-U.F. div P.F. (7)	0	0
Detenciones	3	13,64
-Riesgos estructurales	0	0
-Riesgos de datos	4	18,18
-Riesgos de control	0	0

SIN ADELANTAMIENTO

Estadisticas

Programa Ejemplo1.s
Ejecución: "Segmentado"
"Salto retardado"

Número de instrucciones: 13
Número de ciclos: 31
CPI: 2.3846154

	Ciclos	%
Memoria de instrucciones	13	41,94
Memoria de datos	4	12,9
ALU	19	61,29
-U.F. enteros (1)	9	29,03
-U.F. suma P.F. (2)	6	19,35
-U.F. mult P.F. (4)	4	12,9
-U.F. div P.F. (7)	0	0
Detenciones	10	32,26
-Riesgos estructurales	0	0
-Riesgos de datos	13	41,94
-Riesgos de control	0	0

Figura 4.8. Comparación de las estadísticas aplicando adelantamiento (izquierda) y sin aplicarlo (derecha).

4.3 DEPENDENCIA DE DATOS ENTRE UNA INSTRUCCIÓN DE CARGA SEGUIDA DE UNA INSTRUCCIÓN ARTIMÉTICA

En este caso usaremos el código llamado *Ejemplo 2.s* (que se encuentra disponible en el CD que acompaña a esta memoria) donde estudiaremos el caso de una instrucción `lw` seguida de la instrucción `add`. Este código puede verse en la figura 4.9.

Veamos la ejecución de este código con adelantamiento. Cómo podemos ver en la figura 4.9, la instrucción `add` necesita los registros $t1 que está usando la instrucción `lw`. Veamos cómo se comporta el camino de datos cuando hay adelantamiento.

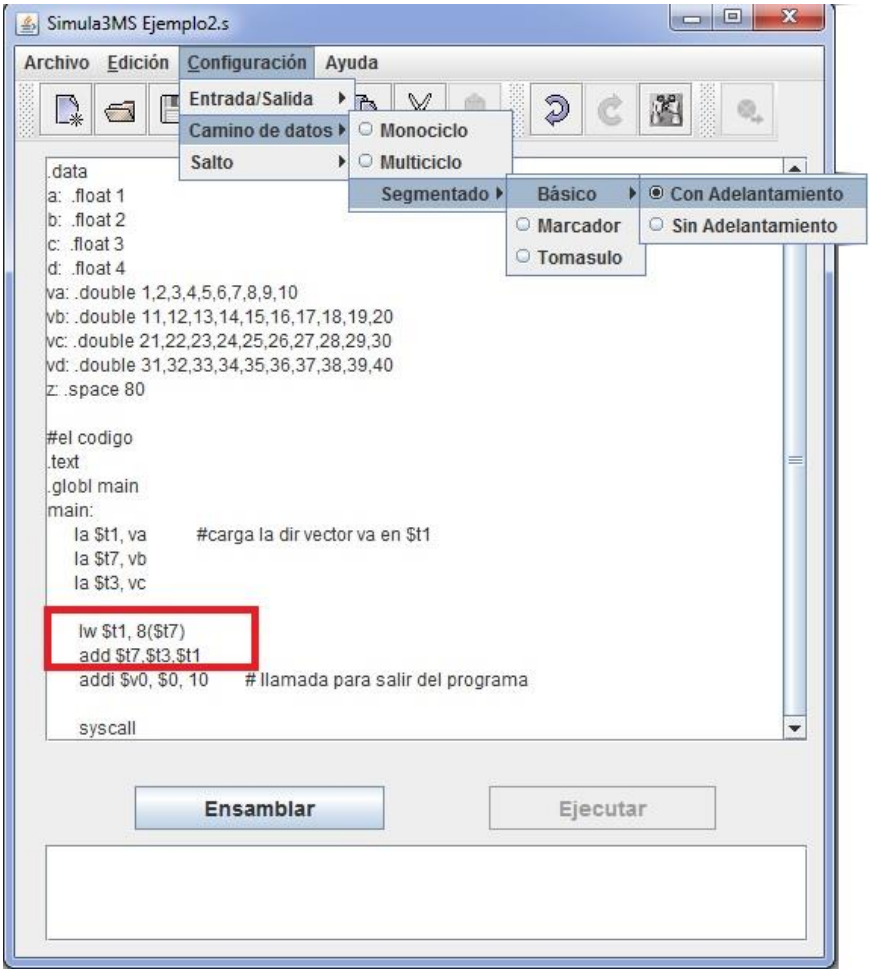

Figura 4.9. *Ejemplo 2.s*, código utilizado para evaluar dependencia aritmética sobre una carga.

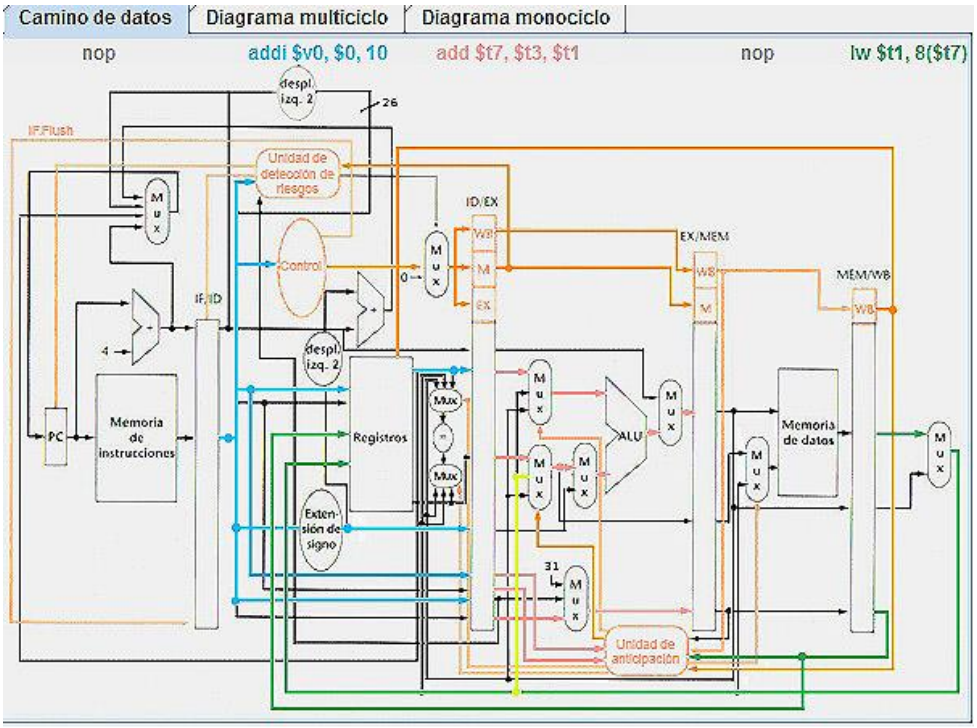

Figura 4.10. Camino de datos para el *Ejemplo 2.s* con adelantamiento.

Como podemos ver en la figura 4.10, la línea amarilla muestra el adelantamiento del registro $t1 a la instrucción `add`. Este valor acaba de leerse de la memoria de datos y aún no está escrito en el banco de registros. De hecho, en este mismo ciclo la instrucción `lw` está realizando esa escritura en el banco de registros, como podemos ver en la figura 4.11.

Figura 4.11. Diagrama multiciclo para el *Ejemplo 2.s* con adelantamiento.

Veamos cómo se comporta el camino de datos sin adelantamiento, que es la opción nueva que se ha incorporado al simulador en este trabajo. Como vemos en la figura 4.12, en el camino de datos sin adelantamiento la instrucción `add` se espera hasta que el dato está guardado en $t1, hasta entonces la instrucción no puede usar el registro $t1 al estar ocupado. Hay que recordar que el primer medio ciclo se realiza la escritura del $t1 en el banco de registros, mientras que es en el segundo medio ciclo cuando se realiza la lectura de ese valor desde el banco de registros.

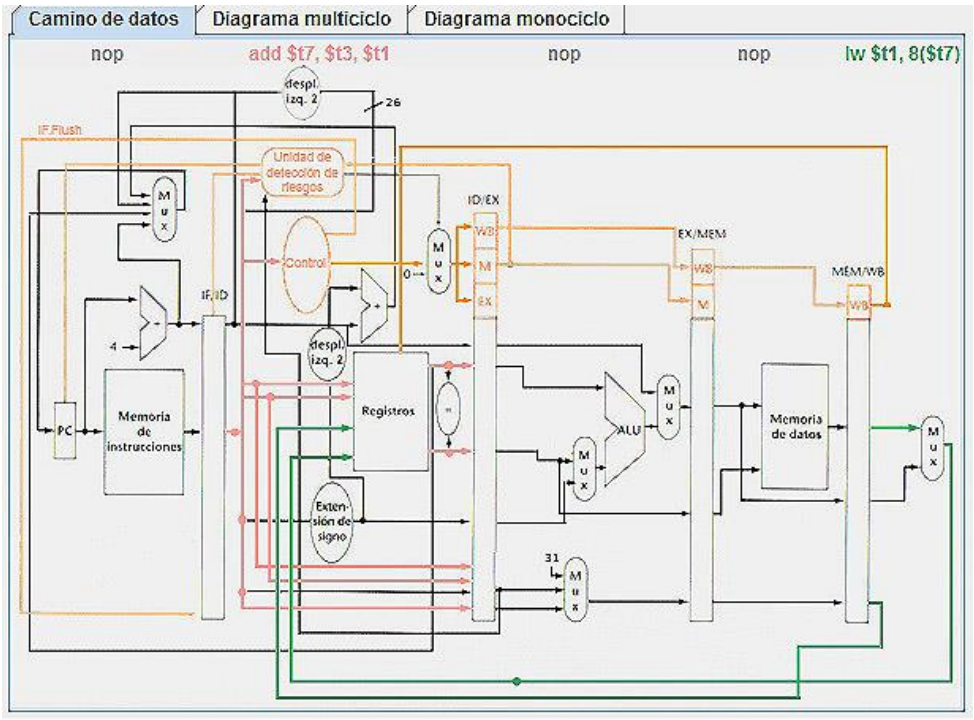

Figura 4.12. Camino de datos para el *Ejemplo 2.s* sin adelantamiento.

En la figura 4.13 pueden verse las diferencias entre hacer la ejecución con adelantamiento y sin adelantamiento.

CON ADELANTAMIENTO

Estadisticas

Programa Ejemplo2.s
Ejecución: "Segmentado"
"Salto retardado"

Número de instrucciones: 9
Número de ciclos: 15
CPI: 1.6666666

	Ciclos	%
Memoria de instrucciones	9	60
Memoria de datos	1	6,67
ALU	9	60
-U.F. enteros (1)	9	60
-U.F. suma P.F. (2)	0	0
-U.F. mult P.F. (4)	0	0
-U.F. div P.F. (7)	0	0
Detenciones	0	0
-Riesgos estructurales	0	0
-Riesgos de datos	1	6,67
-Riesgos de control	0	0

SIN ADELANTAMIENTO

Estadisticas

Programa Ejemplo2.s
Ejecución: "Segmentado"
"Salto retardado"

Número de instrucciones: 9
Número de ciclos: 22
CPI: 2.4444444

	Ciclos	%
Memoria de instrucciones	9	40,91
Memoria de datos	1	4,55
ALU	9	40,91
-U.F. enteros (1)	9	40,91
-U.F. suma P.F. (2)	0	0
-U.F. mult P.F. (4)	0	0
-U.F. div P.F. (7)	0	0
Detenciones	6	27,27
-Riesgos estructurales	0	0
-Riesgos de datos	8	36,36
-Riesgos de control	0	0

Figura 4.13. Comparación de las estadísticas aplicando adelantamiento (izquierda) y sin aplicarlo (derecha).

4.4 DEPENDENCIA DE DATOS ENTRE UNA INSTRUCCIÓN DE CARGA Y UNA INSTRUCCIÓN DE ALMACENAMIENTO EN EL PRIMER OPERANDO DEL ALMACENAMIENTO

El siguiente ejemplo que veremos será el código *Ejemplo 3.s* (que se encuentra disponible en el CD que acompaña a esta memoria). Este código puede verse en la figura 4.14. Puede observarse que el problema está entre la instrucción `lw` que va seguida de la instrucción `sw`. En este caso la dependencia de datos está entre el primer operando de la instrucción `sw` y el primer operando de la instrucción `lw`.

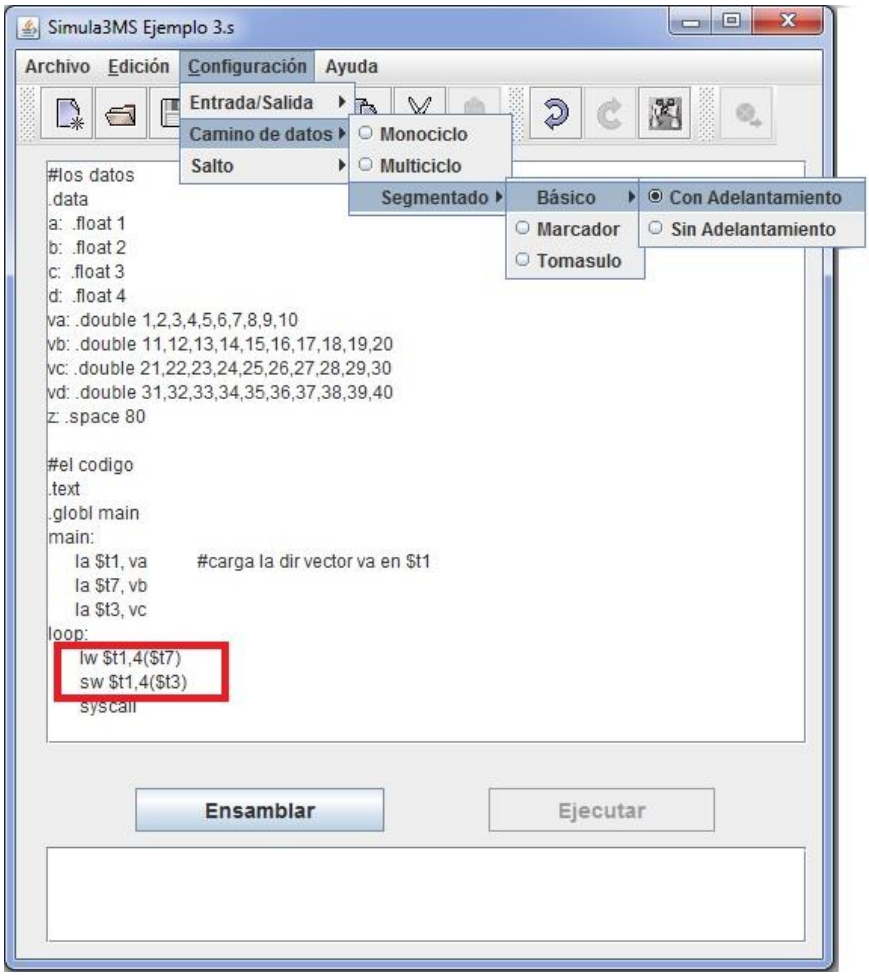

Figura 4.14. *Ejemplo 3.s*, código utilizado para evaluar una dependencia de carga sobre almacenamiento.

Veamos cómo se comporta el camino de datos con adelantamiento. Como vemos en la figura 4.15, la instrucción `lw` adelanta el registro $t1 en la etapa MEM a la instrucción `sw`, teniendo así que esperar un ciclo la instrucción `sw` porque el registro $t1 está ocupado. En la figura 4.16 podremos ver el diagrama multiciclo. Sin embargo este comportamiento no es correcto, pues el adelantamiento debería realizarse a la etapa MEM sin detenciones.

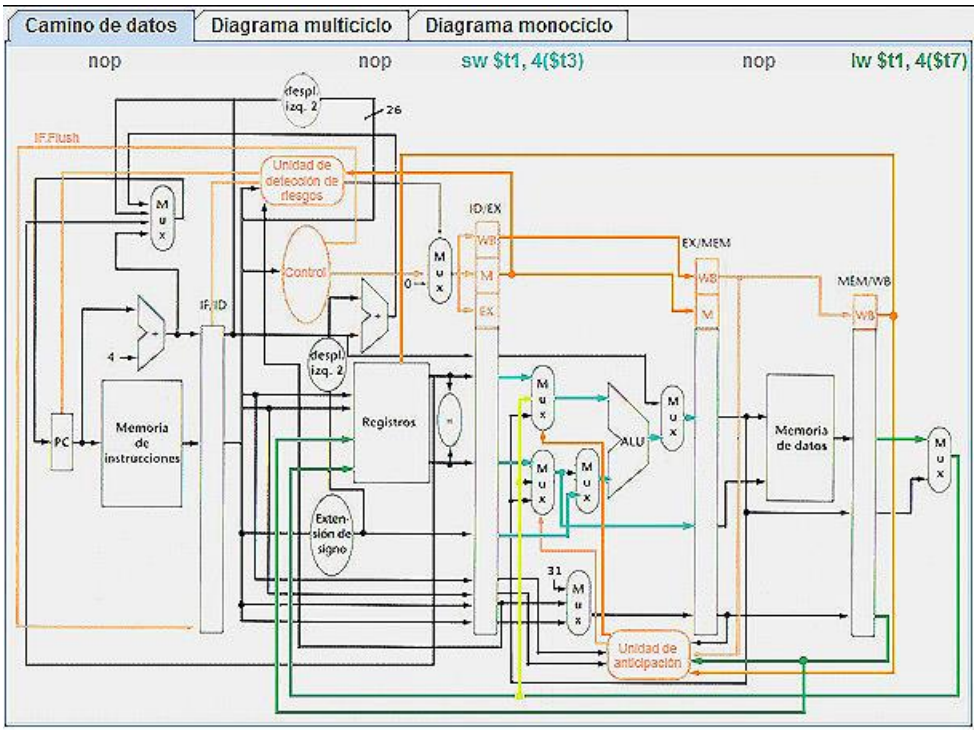

Figura 4.15. Camino de datos para el *Ejemplo 3.s* con adelantamiento en el simulador original.

Figura 4.16. Diagrama multiciclo para el *Ejemplo 3.s* con adelantamiento en el simulador original.

Veamos ahora la ejecución del código con la opción adelantamiento en el simulador modificado en este trabajo, quitando el ciclo de espera y adelantando el dato desde la etapa MEM de `lw` a la etapa EX de `sw`. Como vemos en la figura 4.17, la instrucción `lw` adelanta el dato $t1 en la etapa MEM a la instrucción `sw`, quitando así el ciclo de espera que tenía antes. En la figura 4.18 podremos ver el diagrama multiciclo del simulador actual. Si comparamos esta figura con la figura 4.15, observamos que ahora sí se hace correctamente el adelantamiento justo en el momento en que el dato está disponible.

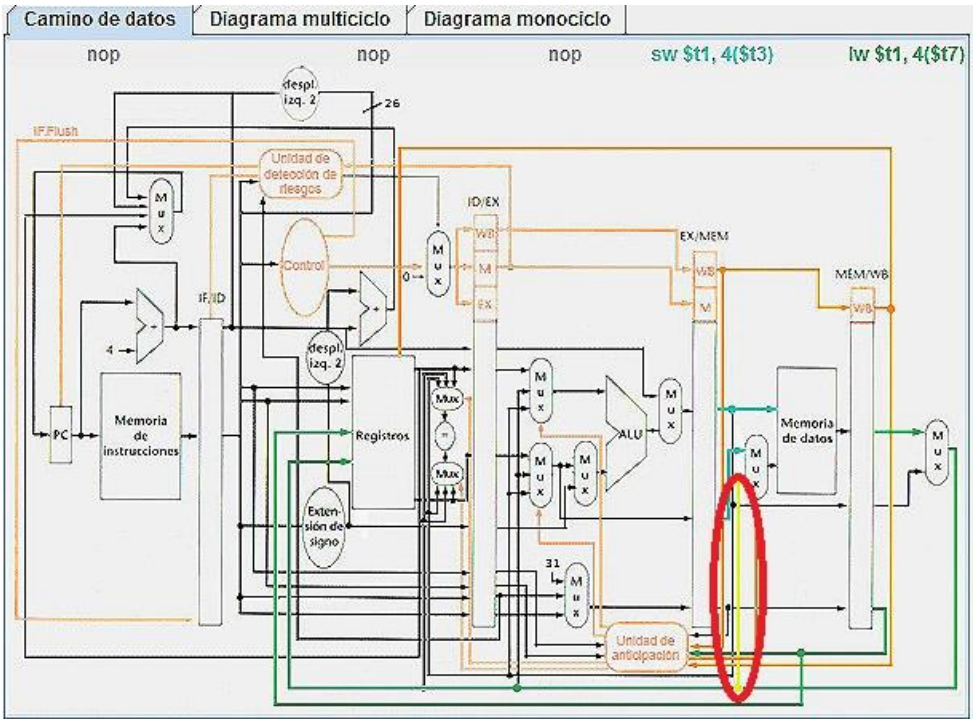

Figura 4.17. Camino de datos para el *Ejemplo 3.s* con adelantamiento en la nueva versión del simulador.

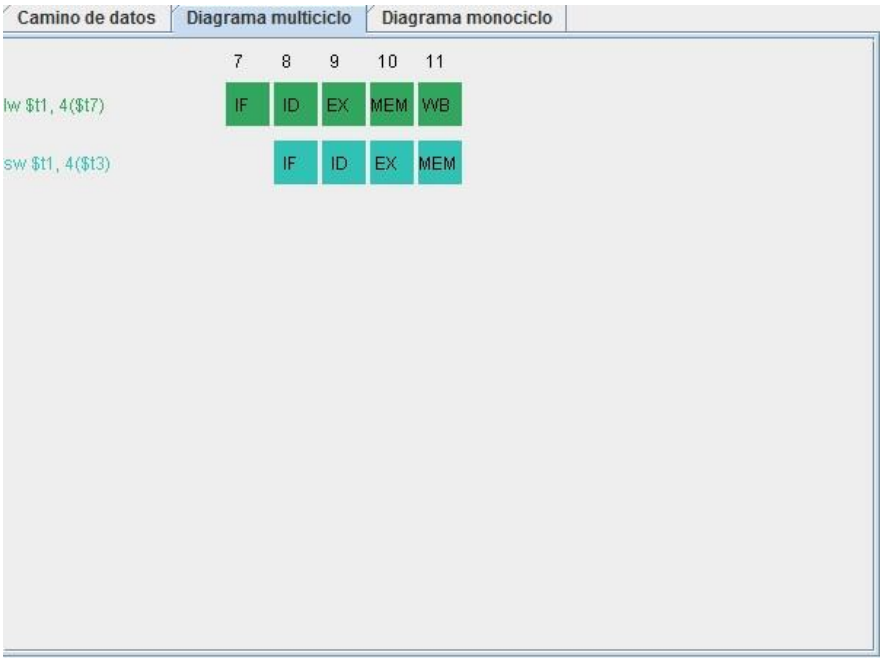

Figura 4.18. Diagrama multiciclo para el *Ejemplo 3.s* con adelantamiento en la nueva versión del simulador.

Veamos ahora la ejecución del código con la opción sin adelantamiento y cómo se comporta el camino de datos sin adelantamiento. Como vemos en la figura 4.19, al elegir la opción sin adelantamiento, la instrucción `sw` debe esperar dos ciclos hasta que la instrucción `lw` carga en el registro $t1 la palabra almacenada en el registro $t4 más el desplazamiento.

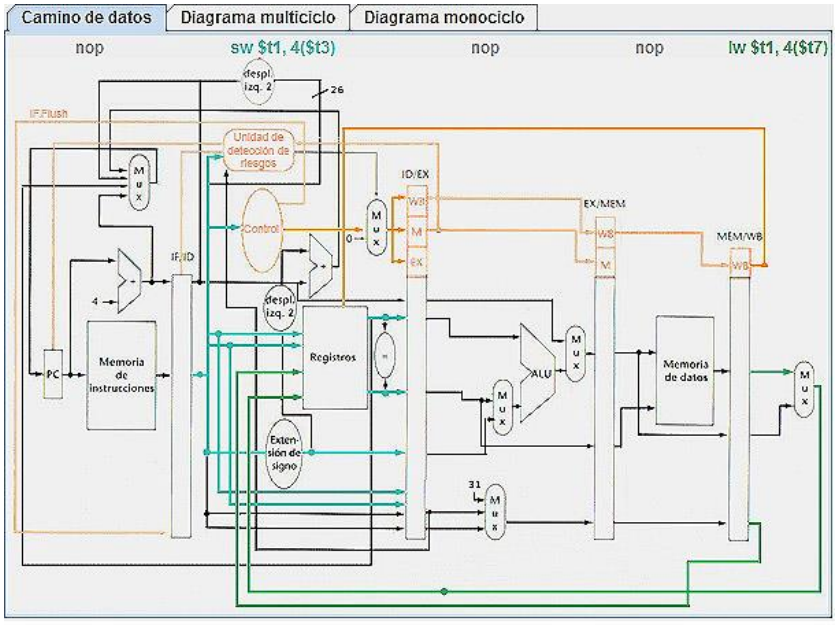

Figura 4.19. Camino de datos para el *Ejemplo 3.s* sin adelantamiento.

En la figura 4.20 pueden verse las diferencias entre hacer la ejecución con adelantamiento y sin adelantamiento.

CON ADELANTAMIENTO **SIN ADELANTAMIENTO**

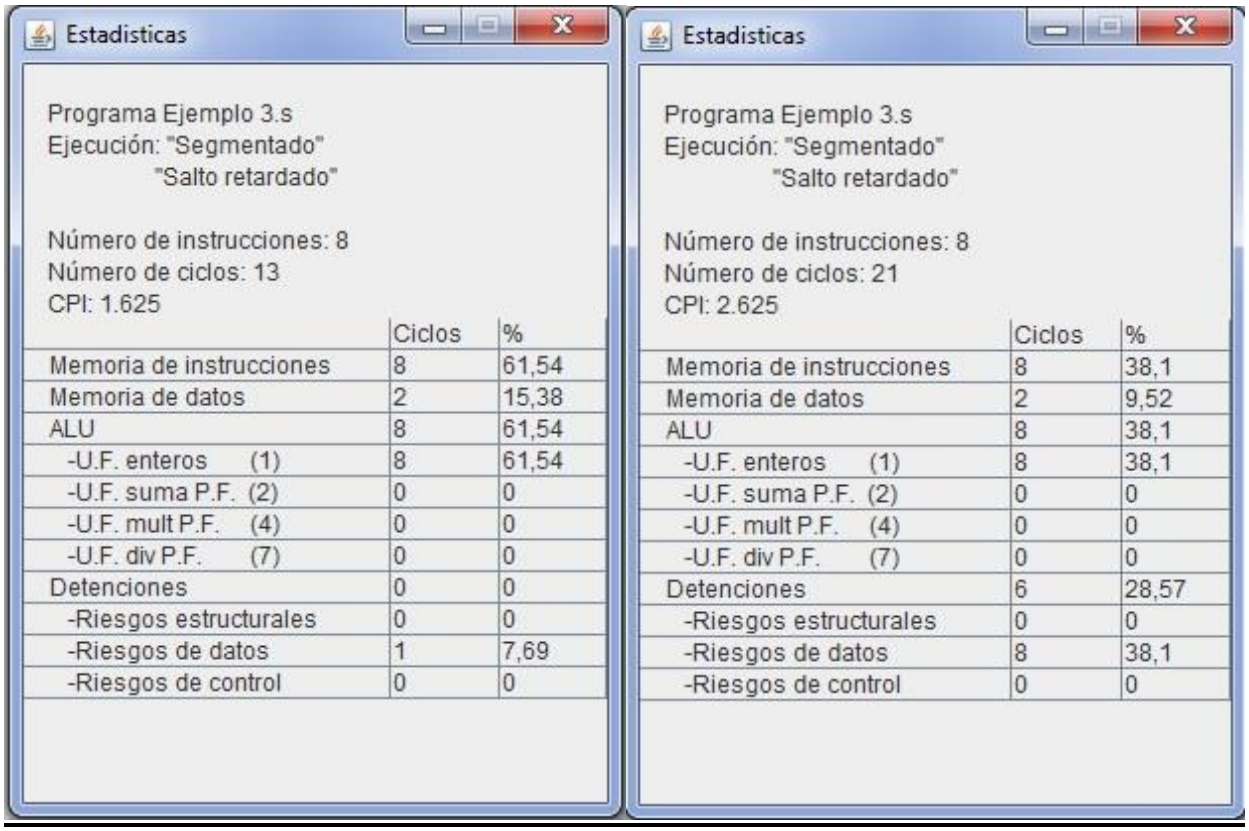

Figura 4.20. Comparación de las estadísticas aplicando adelantamiento (izquierda) y sin aplicarlo (derecha).

4.5 DEPENDENCIA DE DATOS ENTRE UNA INSTRUCCIÓN DE CARGA Y UNA INSTRUCCIÓN DE ALMACENAMIENTO EN EL SEGUNDO OPERANDO DEL ALMACENAMIENTO

En el código *Ejemplo 4.s* (que se encuentra disponible en el CD que acompaña a esta memoria) veremos la instrucción `lw` seguida de la instrucción `sw`, cuando la dependencia es el segundo registro de `sw` y el primer registro de `lw`. Este código puede verse en la figura 4.21.

En la figura 4.22 vemos cómo se comporta el camino de datos con adelantamiento al ejecutar este ejemplo. Podemos ver cómo se activa la línea del segundo multiplexor, la

instrucción `lw` adelanta en la etapa MEM el registro $t1 a la instrucción `sw` para que pueda utilizarlo.

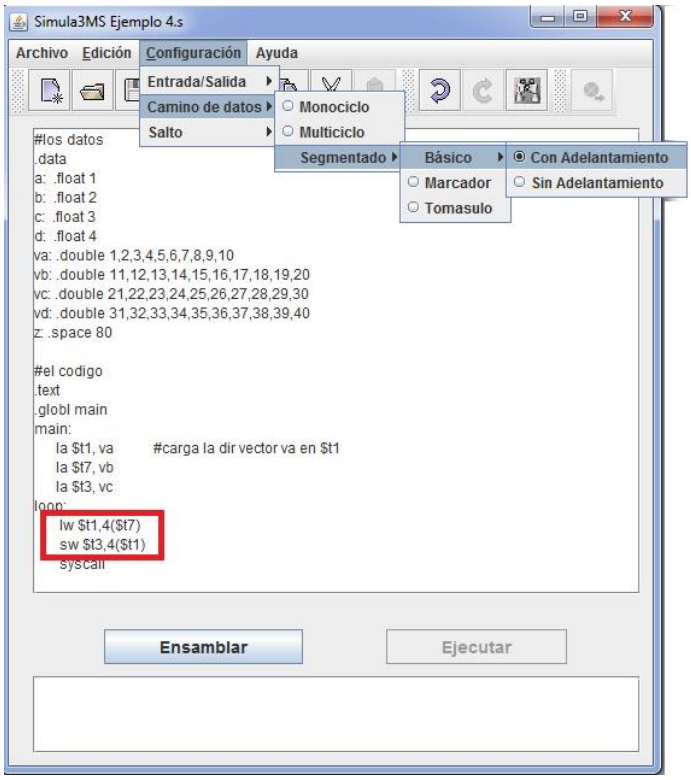

Figura 4.21. *Ejemplo 4.s*, código utilizado para evaluar una dependencia de carga sobre el segundo operando del almacenamiento.

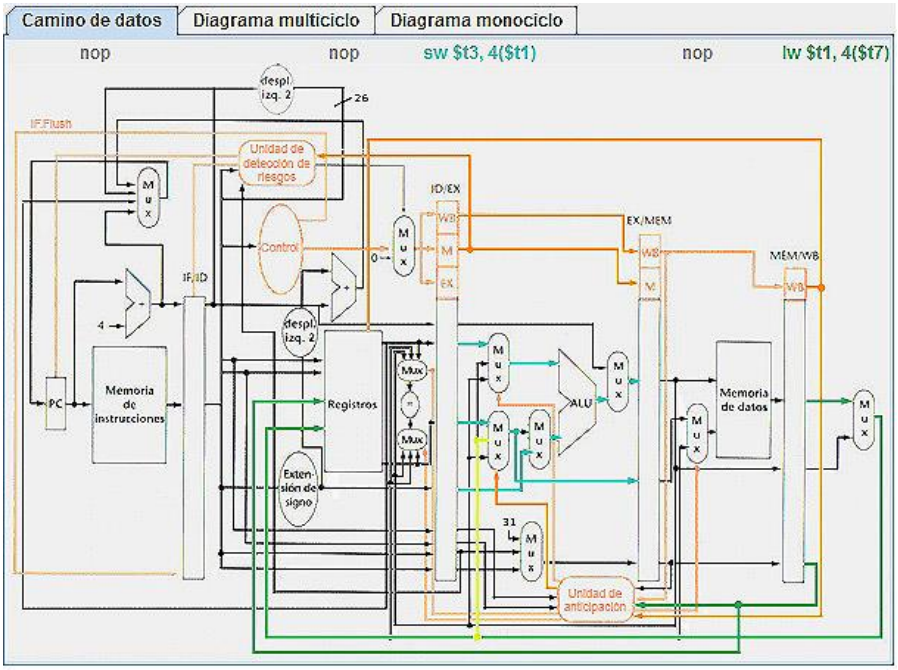

Figura 4.22. Camino de datos para el *Ejemplo 4.s* con adelantamiento.

Veamos cómo se comporta el camino de datos sin adelantamiento. En la figura 4.23, podemos ver que `lw` tiene que almacenar el dato en $t1 para que `sw` pueda usar el registro $t1, teniendo que esperar `sw` dos ciclos. En la figura 4.24 podemos ver la vista multiciclo del *Ejemplo 4.s* sin adelantamiento.

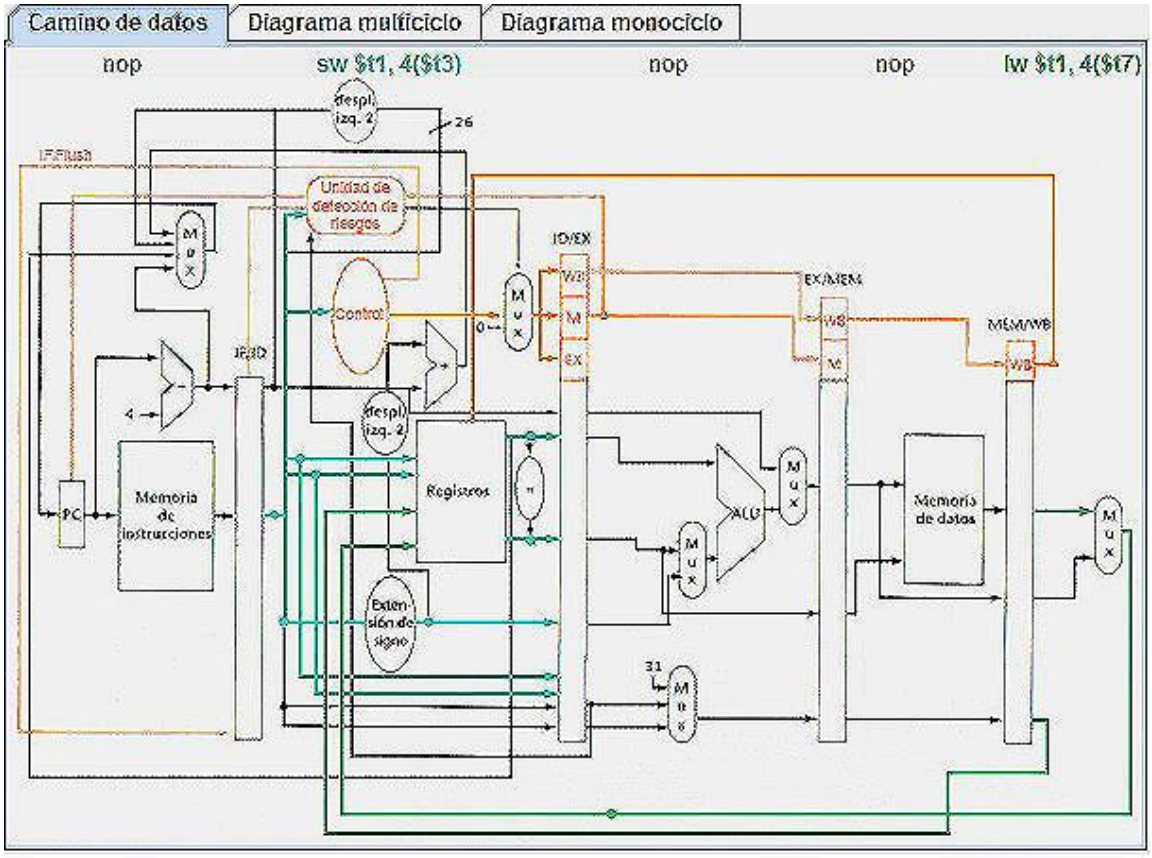

Figura 4.23. Camino de datos para el *Ejemplo 4.s* sin adelantamiento.

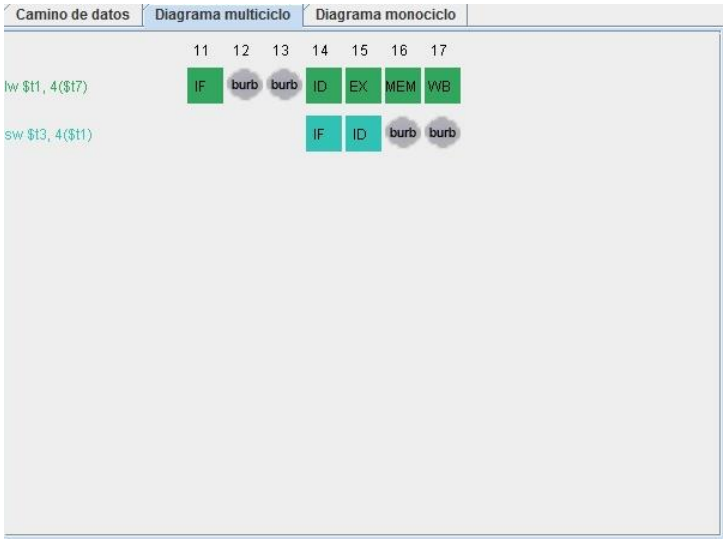

Figura 4.24. Diagrama multiciclo para el *Ejemplo 4.s* sin adelantamiento.

En la figura 4.25 pueden verse las diferencias entre hacer la ejecución con adelantamiento y sin adelantamiento.

CON ADELANTAMIENTO **SIN ADELANTAMIENTO**

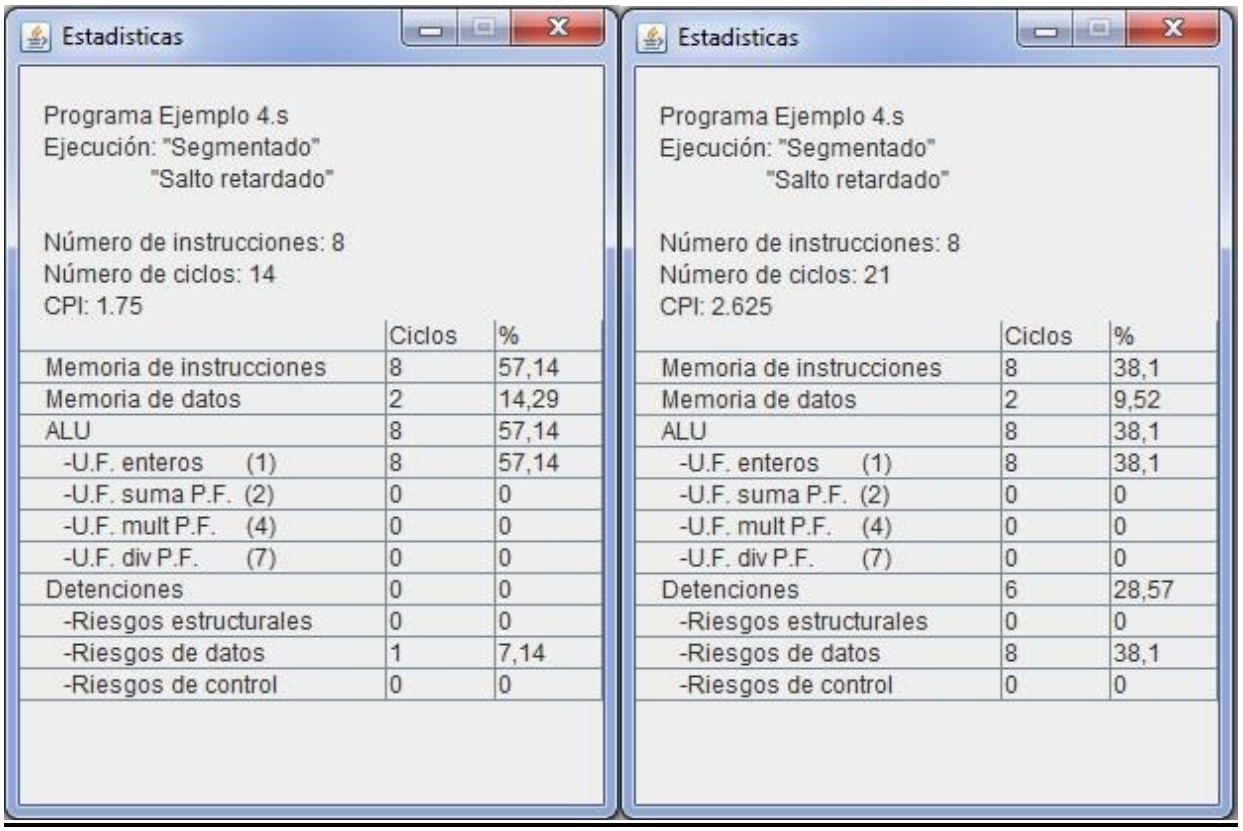

Figura 4.25. Comparación de las estadísticas aplicando adelantamiento (izquierda) y sin aplicarlo (derecha).

4.6 DEPENDENCIA DE DATOS ENTRE UNA INSTRUCCIÓN ARITMÉTICA Y UNA INSTRUCCIÓN DE ALMACENAMIENTO

El código *Ejemplo 5.s* (que se encuentra disponible en el CD que acompaña a esta memoria) representa el caso de dependencia aritmética `add` seguida de una instrucción `sw`, como puede verse en la figura 4.26.

Vamos a ejecutar el código con la opción adelantamiento del procesador original. Como vemos en la figura 4.27, la instrucción `add` adelanta el registro $s2 en la etapa EX a la instrucción `sw`, que está también en la etapa EX. Este adelantamiento no sería correcto

ya que en la etapa EX la instrucción `add` aún no ha guardado la suma de los operandos en el registro $s2.

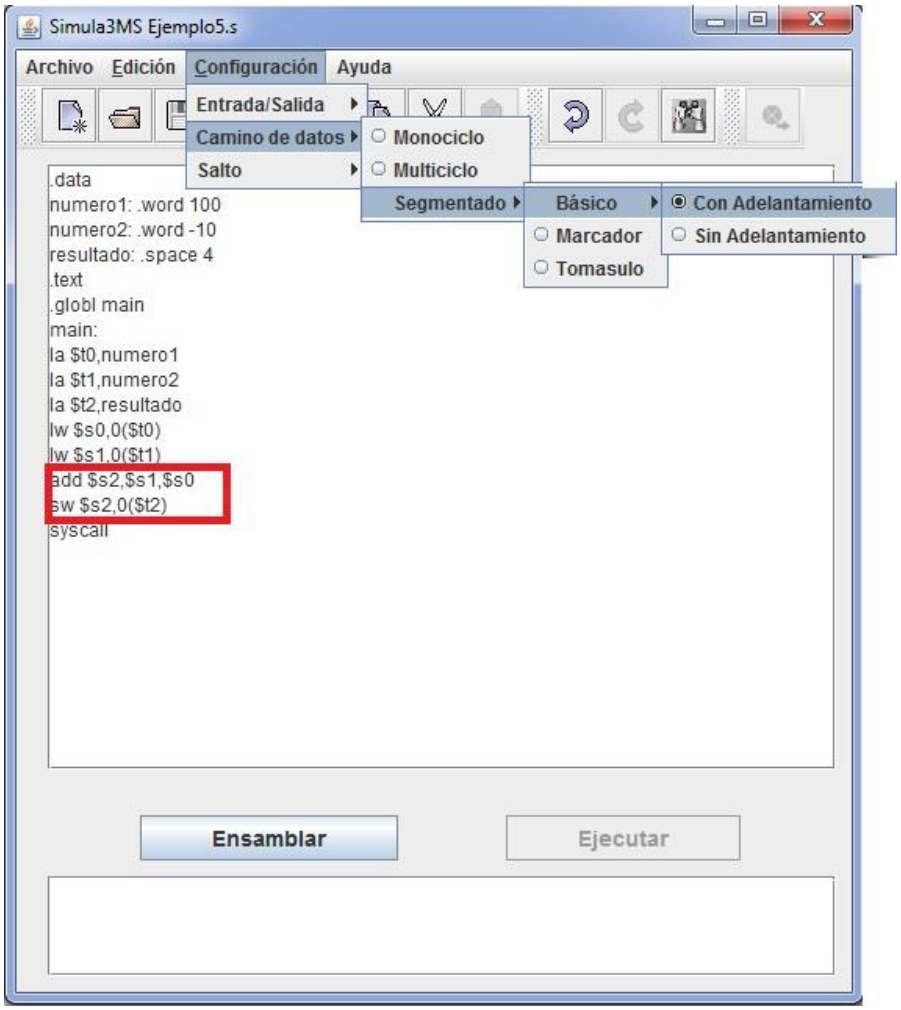

Figura 4.26. *Ejemplo 5.s*, código utilizado para evaluar una dependencia entre una operación aritmética y una instrucción de almacenamiento.

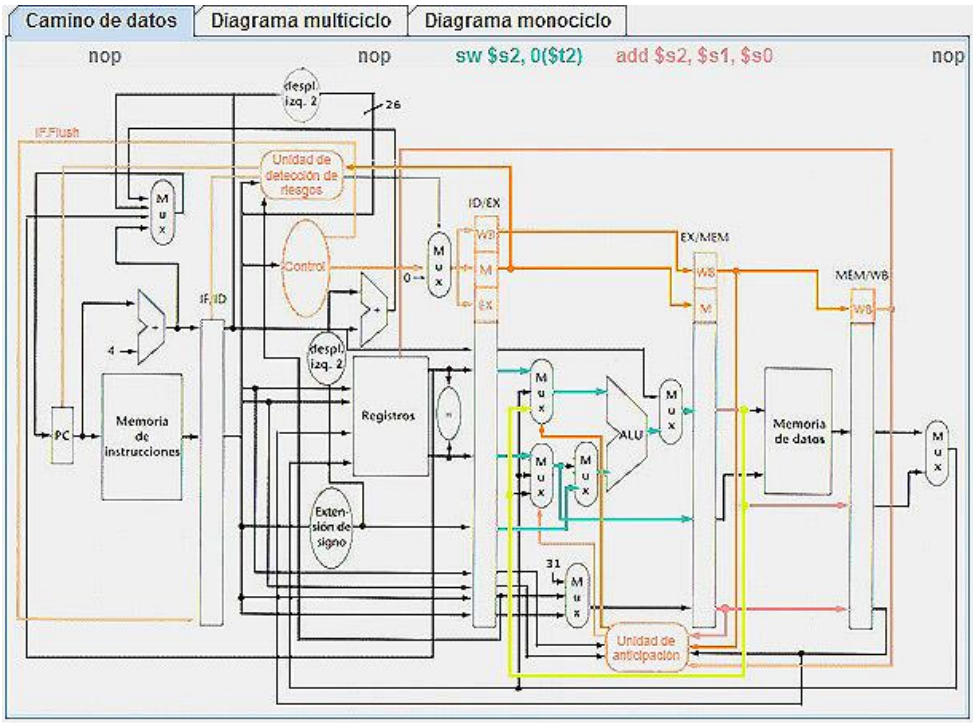

Figura 4.27. Camino de datos para el *Ejemplo 5.s* con adelantamiento en el procesador original.

Veamos ahora la ejecución del código con la opción adelantamiento en el simulador modificado en este trabajo. Ahora adelantaremos el dato en el momento en que se necesite y no antes. Como vemos en la figura 4.28, ahora sí hace bien el adelantamiento adelantando la instrucción `add` el registro $s2 en la etapa MEM a la instrucción `sw` que está también en la etapa MEM. Si comparamos esta figura con la figura 4.27 comprobamos que ahora sí se hace correctamente el adelantamiento en el momento en que el dato esté disponible.

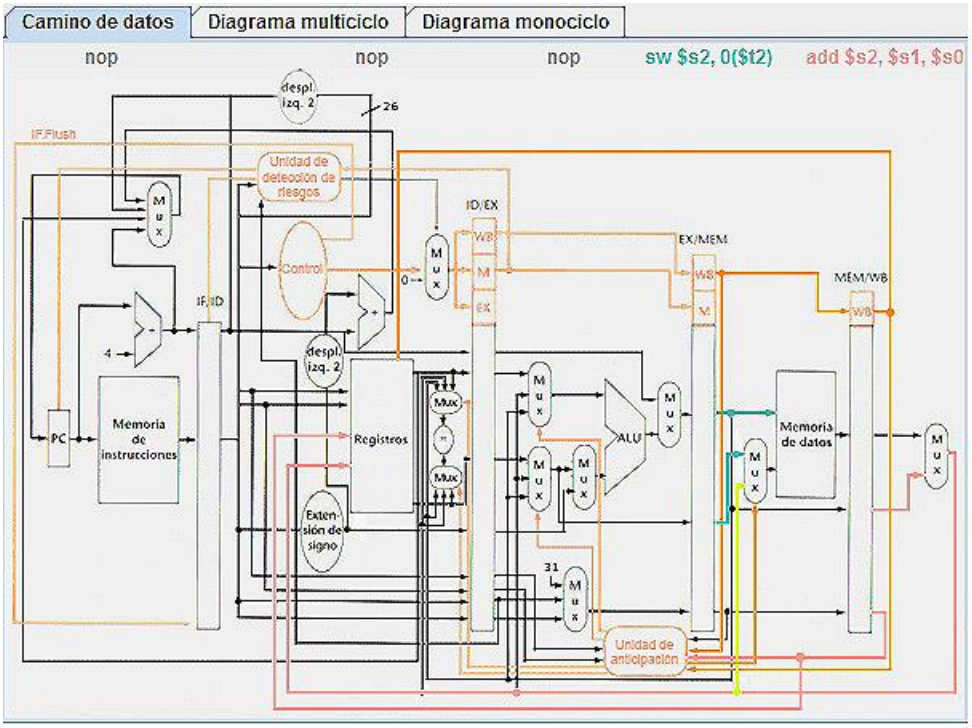

Figura 4.28. Camino de datos para el *Ejemplo 5.s* con adelantamiento en la nueva versión del procesador.

Veamos cómo se comporta el camino de datos sin adelantamiento. Como vemos en la figura 4.29, la instrucción `add` no adelanta el dato, por lo que la instrucción `sw` debe esperarse hasta que `add` ha escrito el resultado en el registro $s2. Una vez escrito el dato, la instrucción `add` deja libre el registro $s2 y la instrucción `sw` sigue con su ejecución.

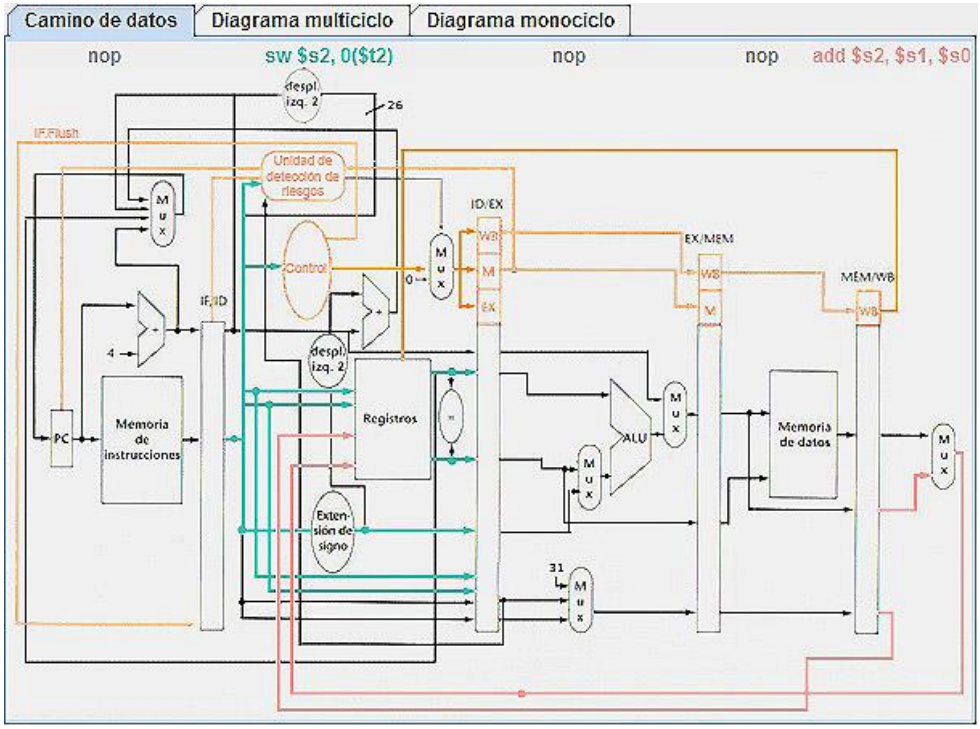

Figura 4.29. Camino de datos para el *Ejemplo 5.s* sin adelantamiento.

En la figura 4.30 podemos ver las diferencias entre hacer la ejecución con adelantamiento y sin adelantamiento.

CON ADELANTAMIENTO **SIN ADELANTAMIENTO**

Estadisticas

Programa Ejemplo5.s
Ejecución: "Segmentado"
"Salto retardado"

Número de instrucciones: 10
Número de ciclos: 16
CPI: 1.6

	Ciclos	%
Memoria de instrucciones	10	62,5
Memoria de datos	3	18,75
ALU	10	62,5
-U.F. enteros (1)	10	62,5
-U.F. suma P.F. (2)	0	0
-U.F. mult P.F. (4)	0	0
-U.F. div P.F. (7)	0	0
Detenciones	0	0
-Riesgos estructurales	0	0
-Riesgos de datos	1	6,25
-Riesgos de control	0	0

Estadisticas

Programa Ejemplo5.s
Ejecución: "Segmentado"
"Salto retardado"

Número de instrucciones: 10
Número de ciclos: 25
CPI: 2.5

	Ciclos	%
Memoria de instrucciones	10	40
Memoria de datos	3	12
ALU	10	40
-U.F. enteros (1)	10	40
-U.F. suma P.F. (2)	0	0
-U.F. mult P.F. (4)	0	0
-U.F. div P.F. (7)	0	0
Detenciones	6	24
-Riesgos estructurales	0	0
-Riesgos de datos	10	40
-Riesgos de control	0	0

Figura 4.30. Comparación de las estadísticas aplicando (izquierda) y sin aplicar (derecha) adelantamiento.

4.7 CONCLUSIONES

En este capítulo se han mostrado y analizado los resultados de las pruebas necesarias para comprobar que las mejoras realizadas en el simulador Simula3MS se comportan como era de esperar. Para ello se han identificado los distintos casos posibles de dependencias relevantes y a continuación se han probado y analizado de forma detallada.

Tras estas pruebas, podemos concluir que el simulador incorpora ahora la posibilidad de no elegir adelantamiento, comportándose de manera adecuada. Además, se han solucionado errores para el caso con adelantamiento.

CAPÍTULO 5. CONCLUSIONES Y TRABAJO FUTURO

En este trabajo hemos detallado cómo hemos modificado el simulador Simula3MS para incluir la nueva funcionalidad del cauce sin adelantamiento. Se han detallado las motivaciones de este trabajo, sus objetivos y la metodología utilizada, y su desarrollo hasta la finalización y pruebas para comprobar el correcto funcionamiento de la nueva versión de Simula3MS. En este capítulo expondremos las conclusiones y el trabajo futuro que puede desarrollarse todavía en el simulador Simula3MS.

5.1 CONCLUSIONES

En la sección 1.3 hemos descrito los objetivos y la metodología seguida para desarrollar este trabajo. Una vez cumplidos todos los objetivos hemos obtenido un simulador mejorado. Este simulador podrá usarse en la docencia de diversas asignaturas como Estructura de Computadores, Organización de Computadores o Arquitectura de Computadores. También se puede seguir trabajando en otras mejoras para que este simulador pueda implementarse con nuevas funcionalidades para conseguir una herramienta lo más completa posible.

Como conclusión a esta sección hay que destacar que para realizar este trabajo se ha necesitado asimilar una gran documentación para poder entender tanto la arquitectura MIPS como el simulador Simula3MS. Aunque el mayor esfuerzo ha sido la modificación y ampliación del código, al tratarse de Java en alto nivel, para poder implementar la nueva funcionalidad del cauce sin adelantamiento, además de corregir los errores encontrados en la implementación original del cauce con adelantamiento. En la figura 5.1 podemos ver la nueva portada de la nueva versión del simulador.

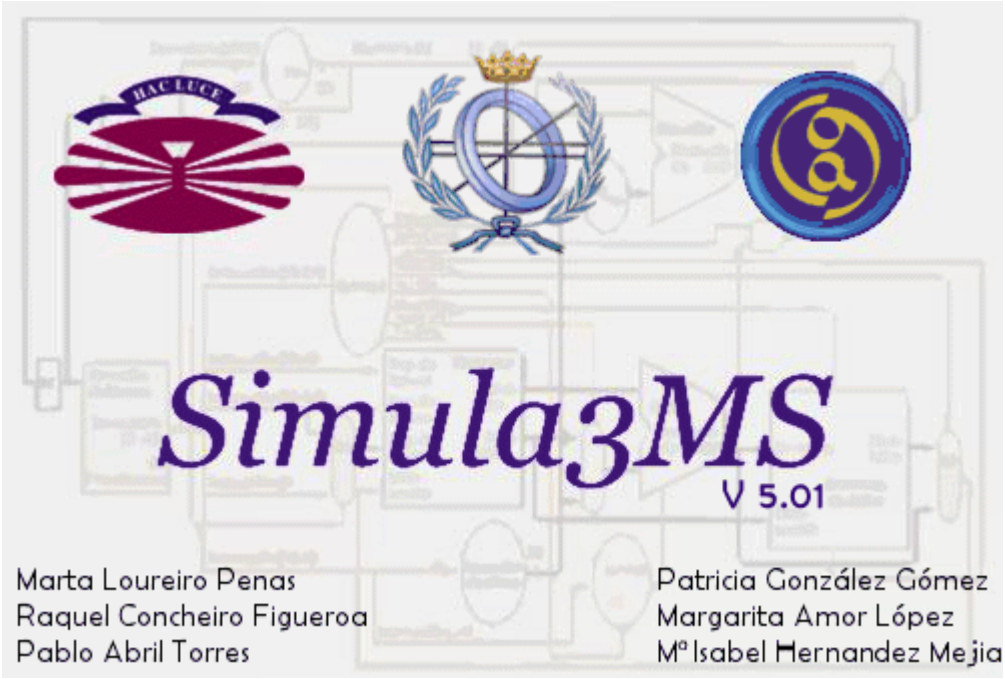

Figura 5.1. Pantalla inicial del simulador modificado.

5.2 TRABAJO FUTURO

Para finalizar con este último capítulo, indicaremos algunas de las propuestas que se podrían añadir, al trabajo actual, en trabajos futuros:

- Nuevas estrategias del tratamiento de saltos.
- Permitir el uso de técnicas especulativas para la ejecución de instrucciones fuera de orden.
- Permitir elegir la codificación de la visualización de los datos: hexadecimal, binario, decimal, etc.

BIBLIOGRAFÍA

LIBROS Y ARTICULOS

[1] David A. Patterson and John L. Hennessy. Estructura y diseño de computadores.
Interficie circuitería/programación. Reverté, 2000

[2] D. SWEETMAN. *See MIPS Run.* Morgan Kaufmann, 2002.

[3] John L. Hennessy and David A. Patterson. Computer Architecure. A Cuantitative
Approach. Morgan-Kaufmann, 2003.

[4] Robert L. Britton. MIPS Assembly Languaje Programming. Prentice-Hall, 2004.

[5] William Stallings. Organización y arquitectura de computadores. Prentice Hall,2000

ENLACES INTERNET

[6] Eclipse http://www.eclipse.org/

[7] http://bioinfo.uib.es/~joemiro/aenui/procJenui/Jen2005/cosimu.pdf

[8] Simula3ms. http://simula3ms.des.udc.es

CONTENIDO DEL CD

En el contenido del CD que acompaña a la memoria podemos encontrar los siguientes recursos:

- Memoria del trabajo en los formatos PDF, DOCX y DOC dentro del directorio Memoria.
- Código fuente del trabajo dentro del directorio Código fuente.
- Códigos de los ejemplos mencionados en la memoria dentro del directorio Ejemplos.
- Nueva versión del simulador Simula3MS dentro del directorio Simulador.

ANEXO A. **CÓDIGO FUENTE**

En este apartado se muestran las principales clases del código fuente que se ha modificado para llegar al objetivo final de este TFG. El código está en el lenguaje de programación Java y utiliza una arquitectura modelo-vista-controlador. Se verán todos los cambios realizados al código original.

1. EMSAMBLADOR.JAVA

```
/**Ensamblador.java
 *Clase del editor y ensamblador de la herramienta
 **/
public class Ensamblador extends javax.swing.JFrame implements Observer{
    String NombreFich=new String("");
    String nombreF=new String("");
    boolean modificar=false;
    String listaErrores=new String();
    StringTokenizer st;
    StringTokenizer st2;
    public UndoManager undoManager;
    public UndoLastAction undoAction;
    public RedoAction redoAction;
    int entero_uno=1, entero_dos=1, entero_tres=1;
    private Vector<Vector> fus;
    private Vector<Integer> tiposFus;
    private Vector<Vector> ers;
    protected   Ensamblar ensambla;

    private Lenguaje lenguaje;
    int salto;
    int adelantamiento;

/**Constructor de Ensamblador
 *@param Sin parametros
```

```
**/
    public Ensamblador() {
      lenguaje = Lenguaje.getInstancia();
       initComponents();
       fus=new Vector<Vector>();
       tiposFus = new Vector<Integer>();
       ers=new Vector<Vector>();
       Editor.requestFocus();
       setSize(550, 640);
       this.setLocation(200, 50);
       Document documento=Editor.getDocument();
       undoManager=new UndoManager();
       undoAction=new UndoLastAction();
       redoAction=new RedoAction();
       documento.addUndoableEditListener(new UndoableEditListener(){
         public void undoableEditHappened(UndoableEditEvent e){
             undoManager.addEdit(e.getEdit());
             undoAction.update();
             redoAction.update();
         }
       });

    }


    private void initComponents() {//GEN-BEGIN:initComponents

        jPanel4 = new javax.swing.JPanel();
        grupoBotones = new javax.swing.ButtonGroup();
        grupoBotonesSalto = new javax.swing.ButtonGroup();
        grupoBotonesSalto1Hueco = new javax.swing.ButtonGroup();
        grupoBotonesSalto2Huecos = new javax.swing.ButtonGroup();
        grupoBotonesSalto3Huecos = new javax.swing.ButtonGroup();
        grupoBotonesES = new javax.swing.ButtonGroup();
        // Cambios por adelantamiento

        grupoBotonesAdelantamiento = new javax.swing.ButtonGroup();

        //fin cambios adelantamiento

        jPanel1 = new javax.swing.JPanel();
        jPanel6 = new javax.swing.JPanel();
        jScrollPane2 = new javax.swing.JScrollPane();
        Editor = new javax.swing.JTextArea();
        jPanel7 = new javax.swing.JPanel();
        jPanel8 = new javax.swing.JPanel();
        jPanel20 = new javax.swing.JPanel();
        Linea = new javax.swing.JTextField();
        jPanel16 = new javax.swing.JPanel();
        jPanel17 = new javax.swing.JPanel();
        jPanel18 = new javax.swing.JPanel();
        Ensamblar = new javax.swing.JButton();
        jPanel19 = new javax.swing.JPanel();
        jPanel21 = new javax.swing.JPanel();
        Ejecutar = new javax.swing.JButton();
        jPanel22 = new javax.swing.JPanel();
        jPanel9 = new javax.swing.JPanel();
        jPanel10 = new javax.swing.JPanel();
        jPanel2 = new javax.swing.JPanel();
```

```
jToolBar1 = new javax.swing.JToolBar();
BotonNuevo = new javax.swing.JButton();
BotonAbrir = new javax.swing.JButton();
BotonGuardar = new javax.swing.JButton();
BotonGuardarComo = new javax.swing.JButton();
BotonImprimir = new javax.swing.JButton();
jToolBar2 = new javax.swing.JToolBar();
BotonCopiar = new javax.swing.JButton();
BotonCortar = new javax.swing.JButton();
BotonPegar = new javax.swing.JButton();
jToolBar3 = new javax.swing.JToolBar();
BotonDeshacer = new javax.swing.JButton();
BotonRehacer = new javax.swing.JButton();
BotonBuscar = new javax.swing.JButton();
jToolBar4 = new javax.swing.JToolBar();
botonError = new javax.swing.JButton();
jPanel3 = new javax.swing.JPanel();
jPanel11 = new javax.swing.JPanel();
jPanel23 = new javax.swing.JPanel();
jScrollPane1 = new javax.swing.JScrollPane();
Errores = new javax.swing.JEditorPane();
jPanel27 = new javax.swing.JPanel();
jPanel24 = new javax.swing.JPanel();
jPanel12 = new javax.swing.JPanel();
jPanel13 = new javax.swing.JPanel();
jPanel14 = new javax.swing.JPanel();
jPanel15 = new javax.swing.JPanel();
jPanel5 = new javax.swing.JPanel();
BarraMenuComp = new javax.swing.JMenuBar();
MenuArchivo = new javax.swing.JMenu();
ItemNuevo = new javax.swing.JMenuItem();
jSeparator1 = new javax.swing.JSeparator();
ItemAbrir = new javax.swing.JMenuItem();
ItemGuardar = new javax.swing.JMenuItem();
ItemGuardarComo = new javax.swing.JMenuItem();
jSeparator2 = new javax.swing.JSeparator();
ItemImprimir = new javax.swing.JMenuItem();
jSeparator3 = new javax.swing.JSeparator();
ItemSalir = new javax.swing.JMenuItem();
MenuEdicion = new javax.swing.JMenu();
ItemDeshacer = new javax.swing.JMenuItem();
ItemRehacer = new javax.swing.JMenuItem();
jSeparator6 = new javax.swing.JSeparator();
ItemSeleccionarTodo = new javax.swing.JMenuItem();
jSeparator4 = new javax.swing.JSeparator();
ItemCopiar = new javax.swing.JMenuItem();
ItemCortar = new javax.swing.JMenuItem();
ItemPegar = new javax.swing.JMenuItem();
jSeparator5 = new javax.swing.JSeparator();
MenuBuscar = new javax.swing.JMenu();
ItemLinea = new javax.swing.JMenuItem();
ItemTexto = new javax.swing.JMenuItem();
ItemReemplazar = new javax.swing.JMenuItem();
MenuConfiguracion = new javax.swing.JMenu();

menuEntradaSalida = new javax.swing.JMenu();
itemMapeada = new javax.swing.JRadioButtonMenuItem();
itemInterrupciones = new javax.swing.JRadioButtonMenuItem();
itemDesactivada = new javax.swing.JRadioButtonMenuItem();
```

```
        menuCaminoDatos = new javax.swing.JMenu();
        itemMonociclo = new javax.swing.JRadioButtonMenuItem();
        itemMulticiclo = new javax.swing.JRadioButtonMenuItem();
        menuSegmentado = new javax.swing.JMenu();
        itemSegmentado = new javax.swing.JRadioButtonMenuItem();
        itemMarcador = new javax.swing.JRadioButtonMenuItem();
        itemTomasulo = new javax.swing.JRadioButtonMenuItem();

        menuSalto = new javax.swing.JMenu();
        menuSalto1Hueco = new javax.swing.JMenu();
        menuSalto2Huecos = new javax.swing.JMenu();
        menuSalto3Huecos = new javax.swing.JMenu();
        itemSalto1Hueco = new javax.swing.JRadioButtonMenuItem();
        itemSalto2Hueco = new javax.swing.JRadioButtonMenuItem();
        itemSalto3Hueco = new javax.swing.JRadioButtonMenuItem();
        itemSaltoRetardadoFlotante = new javax.swing.JRadioButtonMenuItem();
        itemSaltoFijoFlotante = new javax.swing.JRadioButtonMenuItem();
        itemSaltoRetardadoFlotante2Huecos = new
javax.swing.JRadioButtonMenuItem();
        itemSaltoFijoFlotante2Huecos = new javax.swing.JRadioButtonMenuItem();
        itemSaltoRetardadoFlotante3Huecos = new
javax.swing.JRadioButtonMenuItem();
        itemSaltoFijoFlotante3Huecos = new javax.swing.JRadioButtonMenuItem();
        //cambio por adelantamiento

        menuAdelantamiento = new javax.swing.JMenu();
        itemAdelantamiento = new javax.swing.JRadioButtonMenuItem();
        itemNoAdelantamiento = new javax.swing.JRadioButtonMenuItem();

        //fin cambio adelantamiento



        MenuAyuda = new javax.swing.JMenu();
        ItemAcercaDe = new javax.swing.JMenuItem();
        ItemAyuda = new javax.swing.JMenuItem();

setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_CLOSE);
        setTitle("Simula3MS");
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
            }
        });

        jPanel1.setLayout(new java.awt.BorderLayout());

        jPanel6.setLayout(new java.awt.GridLayout(1, 0));

        Editor.setTabSize(10);
        Editor.setWrapStyleWord(true);
        Editor.setFocusCycleRoot(true);
        Editor.setFocusTraversalPolicy(getFocusTraversalPolicy());
        Editor.addCaretListener(new javax.swing.event.CaretListener() {
            public void caretUpdate(javax.swing.event.CaretEvent evt) {
                HaCambiado(evt);
            }
        });
        Editor.addKeyListener(new java.awt.event.KeyAdapter() {
```

```
            public void keyPressed(java.awt.event.KeyEvent evt) {
                nLinea(evt);
            }
            public void keyTyped(java.awt.event.KeyEvent evt) {
                Cambios(evt);
            }
        });

        jScrollPane2.setViewportView(Editor);

        jPanel6.add(jScrollPane2);

        jPanel1.add(jPanel6, java.awt.BorderLayout.CENTER);

        jPanel1.add(jPanel7, java.awt.BorderLayout.NORTH);

        jPanel8.setLayout(new java.awt.GridLayout(0, 1));

        jPanel20.setLayout(new java.awt.GridLayout(1, 0));


        Linea.setEditable(false);
        Linea.setText("\n\n\n\n");
        Linea.setBorder(null);
        jPanel20.add(Linea);

        jPanel8.add(jPanel20);

        jPanel16.setLayout(new javax.swing.BoxLayout(jPanel16,
javax.swing.BoxLayout.X_AXIS));

        jPanel16.add(jPanel17);

        jPanel18.setLayout(new java.awt.GridLayout(1, 0));

        Ensamblar.setFont(new java.awt.Font("Dialog", 1, 14));
        Ensamblar.setText(lenguaje.getString("ensamblar"));

Ensamblar.setDebugGraphicsOptions(javax.swing.DebugGraphics.NONE_OPTION);
        Ensamblar.setEnabled(false);
        Ensamblar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                EnsamblarActionPerformed(evt);
            }
        });

        jPanel18.add(Ensamblar);

        jPanel16.add(jPanel18);

        jPanel16.add(jPanel19);

        jPanel21.setLayout(new java.awt.GridLayout(1, 0));

        Ejecutar.setFont(new java.awt.Font("Dialog", 1, 14));
        Ejecutar.setText(lenguaje.getString("ejecutar"));
        Ejecutar.setEnabled(false);
        Ejecutar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
                EjecutarActionPerformed(evt);
            }
        });

        jPanel21.add(Ejecutar);

        jPanel16.add(jPanel21);

        jPanel16.add(jPanel22);

        jPanel8.add(jPanel16);

        jPanel1.add(jPanel8, java.awt.BorderLayout.SOUTH);

        jPanel1.add(jPanel9, java.awt.BorderLayout.EAST);

        jPanel1.add(jPanel10, java.awt.BorderLayout.WEST);

        getContentPane().add(jPanel1, java.awt.BorderLayout.CENTER);

        jPanel2.setLayout(new javax.swing.BoxLayout(jPanel2,
javax.swing.BoxLayout.X_AXIS));

        jToolBar1.setRollover(true);
        BotonNuevo.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ensamblador/iconos/New2.gif")));
        BotonNuevo.setToolTipText(lenguaje.getString("nuevo"));
        BotonNuevo.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemNuevoActionPerformed(evt);
            }
        });

        jToolBar1.add(BotonNuevo);

        BotonAbrir.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ensamblador/iconos/Open.gif")));
        BotonAbrir.setToolTipText(lenguaje.getString("abrir"));
        BotonAbrir.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemAbrirActionPerformed(evt);
            }
        });

        jToolBar1.add(BotonAbrir);

        BotonGuardar.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ensamblador/iconos/Save.gif")));
        BotonGuardar.setToolTipText(lenguaje.getString("guardar"));
        BotonGuardar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemGuardarActionPerformed(evt);
            }
        });

        jToolBar1.add(BotonGuardar);
```

```
        BotonGuardarComo.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ensamblador/iconos/SaveAs.gif"))
);
        BotonGuardarComo.setToolTipText(lenguaje.getString("guardarComo"));
        BotonGuardarComo.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemGuardarComoActionPerformed(evt);
            }
        });

        jToolBar1.add(BotonGuardarComo);

        BotonImprimir.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ensamblador/iconos/Print3.gif"))
);
        BotonImprimir.setToolTipText(lenguaje.getString("imprimir"));
        BotonImprimir.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                BotonImprimirActionPerformed(evt);
            }
        });

        jToolBar1.add(BotonImprimir);

        jPanel2.add(jToolBar1);

        jToolBar2.setRollover(true);
        BotonCopiar.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ensamblador/iconos/Copy.gif")));
        BotonCopiar.setToolTipText(lenguaje.getString("copiar"));
        BotonCopiar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemCopiarActionPerformed(evt);
            }
        });

        jToolBar2.add(BotonCopiar);

        BotonCortar.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ensamblador/iconos/Cut.gif")));
        BotonCortar.setToolTipText(lenguaje.getString("cortar"));
        BotonCortar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemCortarActionPerformed(evt);
            }
        });

        jToolBar2.add(BotonCortar);

        BotonPegar.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ensamblador/iconos/Paste.gif")))
;
        BotonPegar.setToolTipText(lenguaje.getString("pegar"));
        BotonPegar.setEnabled(false);
        BotonPegar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemPegarActionPerformed(evt);
            }
```

```
        });

        jToolBar2.add(BotonPegar);

        jPanel2.add(jToolBar2);

        jToolBar3.setRollover(true);
        BotonDeshacer.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ensamblador/iconos/Undo.gif")));
        BotonDeshacer.setToolTipText(lenguaje.getString("deshacer"));
        BotonDeshacer.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemDeshacerActionPerformed(evt);
            }
        });

        jToolBar3.add(BotonDeshacer);

        BotonRehacer.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ensamblador/iconos/Redo.gif")));
        BotonRehacer.setToolTipText(lenguaje.getString("rehacer"));
        BotonRehacer.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemRehacerActionPerformed(evt);
            }
        });

        jToolBar3.add(BotonRehacer);

        BotonBuscar.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ensamblador/iconos/search2.gif")
));
        BotonBuscar.setToolTipText(lenguaje.getString("buscar"));
        BotonBuscar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemTextoActionPerformed(evt);
            }
        });

        jToolBar3.add(BotonBuscar);

        jPanel2.add(jToolBar3);

        jToolBar4.setRollover(true);
        botonError.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ensamblador/iconos/error.gif")))
;
        botonError.setToolTipText(lenguaje.getString("errorSiguiente"));
        botonError.setEnabled(false);
        botonError.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                botonErrorActionPerformed(evt);
            }
        });

        jToolBar4.add(botonError);

        jPanel2.add(jToolBar4);
```

```
        getContentPane().add(jPanel2, java.awt.BorderLayout.NORTH);

        jPanel3.setLayout(new java.awt.BorderLayout());

        jPanel11.setLayout(new java.awt.BorderLayout());

        jPanel23.setLayout(new java.awt.GridLayout(1, 0));

        Errores.setEditable(false);
        Errores.setPreferredSize(new java.awt.Dimension(146, 61));
        Errores.setAutoscrolls(false);
        jScrollPane1.setViewportView(Errores);

        jPanel23.add(jScrollPane1);

        jPanel11.add(jPanel23, java.awt.BorderLayout.CENTER);

        jPanel11.add(jPanel27, java.awt.BorderLayout.WEST);

        jPanel11.add(jPanel24, java.awt.BorderLayout.NORTH);

        jPanel3.add(jPanel11, java.awt.BorderLayout.CENTER);

        jPanel12.setLayout(new java.awt.GridLayout(1, 0));

        jPanel3.add(jPanel12, java.awt.BorderLayout.NORTH);

        jPanel3.add(jPanel13, java.awt.BorderLayout.SOUTH);

        jPanel3.add(jPanel14, java.awt.BorderLayout.EAST);

        jPanel3.add(jPanel15, java.awt.BorderLayout.WEST);

        getContentPane().add(jPanel3, java.awt.BorderLayout.SOUTH);

        getContentPane().add(jPanel5, java.awt.BorderLayout.WEST);

        MenuArchivo.setMnemonic('f');
        MenuArchivo.setText(lenguaje.getString("archivo"));
        MenuArchivo.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                MenuArchivoActionPerformed(evt);
            }
        });


ItemNuevo.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyE
vent.VK_N, java.awt.event.InputEvent.CTRL_MASK));
        //ItemNuevo.setMnemonic('n');
        ItemNuevo.setText(lenguaje.getString("nuevo"));
        ItemNuevo.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemNuevoActionPerformed(evt);
            }
        });

        MenuArchivo.add(ItemNuevo);
        ItemNuevo.getAccessibleContext().setAccessibleName("ItemNuevo");
        ItemNuevo.getAccessibleContext().setAccessibleDescription("ItemNuevo");
```

```
        MenuArchivo.add(jSeparator1);


ItemAbrir.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyE
vent.VK_O, java.awt.event.InputEvent.CTRL_MASK));
        //ItemAbrir.setMnemonic('o');
        ItemAbrir.setText(lenguaje.getString("abrir"));
        ItemAbrir.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemAbrirActionPerformed(evt);
            }
        });

        MenuArchivo.add(ItemAbrir);
        ItemAbrir.getAccessibleContext().setAccessibleDescription("ItemAbrir");


ItemGuardar.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.Ke
yEvent.VK_S, java.awt.event.InputEvent.CTRL_MASK));
        //ItemGuardar.setMnemonic('s');
        ItemGuardar.setText(lenguaje.getString("guardar"));
        ItemGuardar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemGuardarActionPerformed(evt);
            }
        });

        MenuArchivo.add(ItemGuardar);

ItemGuardar.getAccessibleContext().setAccessibleDescription("ItemGuardar");


ItemGuardarComo.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.even
t.KeyEvent.VK_S, java.awt.event.InputEvent.CTRL_MASK));
        //ItemGuardarComo.setMnemonic('s');
        ItemGuardarComo.setText(lenguaje.getString("guardarComo"));
        ItemGuardarComo.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemGuardarComoActionPerformed(evt);
            }
        });

        MenuArchivo.add(ItemGuardarComo);

ItemGuardarComo.getAccessibleContext().setAccessibleDescription("ItemGuardarCom
o");

        MenuArchivo.add(jSeparator2);


ItemImprimir.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.K
eyEvent.VK_P, java.awt.event.InputEvent.CTRL_MASK));
        //ItemImprimir.setMnemonic('p');
        ItemImprimir.setText(lenguaje.getString("imprimir"));
        ItemImprimir.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                BotonImprimirActionPerformed(evt);
            }
```

```
        });

        MenuArchivo.add(ItemImprimir);

ItemImprimir.getAccessibleContext().setAccessibleDescription("ItemImprimir");

        MenuArchivo.add(jSeparator3);


ItemSalir.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyE
vent.VK_Q, java.awt.event.InputEvent.CTRL_MASK));
        //ItemSalir.setMnemonic('i');
        ItemSalir.setText(lenguaje.getString("salir"));
        ItemSalir.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemSalirActionPerformed(evt);
            }
        });

        MenuArchivo.add(ItemSalir);
        ItemSalir.getAccessibleContext().setAccessibleDescription("ItemSalir");

        BarraMenuComp.add(MenuArchivo);

MenuArchivo.getAccessibleContext().setAccessibleDescription("MenuArchivo");

        MenuEdicion.setMnemonic('e');
        MenuEdicion.setText(lenguaje.getString("edicion"));
        MenuEdicion.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                MenuEdicionActionPerformed(evt);
            }
        });


ItemDeshacer.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.K
eyEvent.VK_Z, java.awt.event.InputEvent.CTRL_MASK));
        ItemDeshacer.setText(lenguaje.getString("deshacer"));
        ItemDeshacer.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemDeshacerActionPerformed(evt);
            }
        });

        MenuEdicion.add(ItemDeshacer);


ItemRehacer.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.Ke
yEvent.VK_Y, java.awt.event.InputEvent.CTRL_MASK));
        ItemRehacer.setText(lenguaje.getString("rehacer"));
        ItemRehacer.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemRehacerActionPerformed(evt);
            }
        });

        MenuEdicion.add(ItemRehacer);

        MenuEdicion.add(jSeparator6);
```

```
ItemSeleccionarTodo.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.
event.KeyEvent.VK_A, java.awt.event.InputEvent.CTRL_MASK));
        //ItemSeleccionarTodo.setMnemonic('a');
        ItemSeleccionarTodo.setText(lenguaje.getString("seleccionarTodo"));
        ItemSeleccionarTodo.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemSeleccionarTodoActionPerformed(evt);
            }
        });

        MenuEdicion.add(ItemSeleccionarTodo);

ItemSeleccionarTodo.getAccessibleContext().setAccessibleName("SeleccionarTodo")
;

ItemSeleccionarTodo.getAccessibleContext().setAccessibleDescription("ItemSelecc
ionarTodo");

        MenuEdicion.add(jSeparator4);


ItemCopiar.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.Key
Event.VK_C, java.awt.event.InputEvent.CTRL_MASK));
        //ItemCopiar.setMnemonic('c');
        ItemCopiar.setText(lenguaje.getString("copiar"));
        ItemCopiar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemCopiarActionPerformed(evt);
            }
        });

        MenuEdicion.add(ItemCopiar);

ItemCopiar.getAccessibleContext().setAccessibleDescription("ItemCopiar");


ItemCortar.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.Key
Event.VK_X, java.awt.event.InputEvent.CTRL_MASK));
        //ItemCortar.setMnemonic('x');
        ItemCortar.setText(lenguaje.getString("cortar"));
        ItemCortar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemCortarActionPerformed(evt);
            }
        });

        MenuEdicion.add(ItemCortar);

ItemCortar.getAccessibleContext().setAccessibleDescription("ItemCortar");


ItemPegar.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyE
vent.VK_V, java.awt.event.InputEvent.CTRL_MASK));
        //ItemPegar.setMnemonic('v');
        ItemPegar.setText(lenguaje.getString("pegar"));
        ItemPegar.setEnabled(false);
```

```
        ItemPegar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemPegarActionPerformed(evt);
            }
        });

        MenuEdicion.add(ItemPegar);
        ItemPegar.getAccessibleContext().setAccessibleDescription("ItemPegar");

        MenuEdicion.add(jSeparator5);

        //MenuBuscar.setMnemonic('f');
        MenuBuscar.setText(lenguaje.getString("buscar"));
        MenuBuscar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                MenuBuscarActionPerformed(evt);
            }
        });


ItemLinea.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyE
vent.VK_L, java.awt.event.InputEvent.CTRL_MASK));
        //ItemLinea.setMnemonic('l');
        ItemLinea.setText(lenguaje.getString("linea"));
        ItemLinea.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemLineaActionPerformed(evt);
            }
        });

        MenuBuscar.add(ItemLinea);
        ItemLinea.getAccessibleContext().setAccessibleDescription("ItemLinea");


ItemTexto.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyE
vent.VK_T, java.awt.event.InputEvent.CTRL_MASK));
        //ItemTexto.setMnemonic('t');
        ItemTexto.setText(lenguaje.getString("texto"));
        ItemTexto.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ItemTextoActionPerformed(evt);
            }
        });

        MenuBuscar.add(ItemTexto);
        ItemTexto.getAccessibleContext().setAccessibleDescription("ItemTexto");

        MenuEdicion.add(MenuBuscar);
        MenuBuscar.getAccessibleContext().setAccessibleName("Buscar");

MenuBuscar.getAccessibleContext().setAccessibleDescription("MenuBuscar");


ItemReemplazar.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event
.KeyEvent.VK_R, java.awt.event.InputEvent.CTRL_MASK));
        //ItemReemplazar.setMnemonic('r');
        ItemReemplazar.setText(lenguaje.getString("reemplazar"));
        ItemReemplazar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
                ItemReemplazarActionPerformed(evt);
            }
        });

        MenuEdicion.add(ItemReemplazar);

ItemReemplazar.getAccessibleContext().setAccessibleDescription("ItemReemplazar"
);

        BarraMenuComp.add(MenuEdicion);

MenuEdicion.getAccessibleContext().setAccessibleDescription("MenuEdicion");

        MenuConfiguracion.setMnemonic('c');
        MenuConfiguracion.setText(lenguaje.getString("configuracion"));
        MenuConfiguracion.setMinimumSize(new java.awt.Dimension(6, 21));
        MenuConfiguracion.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                MenuConfiguracionActionPerformed(evt);
            }
        });

        menuEntradaSalida.setText(lenguaje.getString("entradaSalida"));

        itemMapeada.setText(lenguaje.getString("encuesta"));
        itemInterrupciones.setText(lenguaje.getString("conInterrupciones"));
        itemDesactivada.setText(lenguaje.getString("desactivada"));
        menuEntradaSalida.add(itemMapeada);
        menuEntradaSalida.add(itemInterrupciones);
        menuEntradaSalida.add(itemDesactivada);
        itemDesactivada.setSelected(true);

        MenuConfiguracion.add(menuEntradaSalida);
        itemMapeada.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                itemESActionPerformed(evt);
            }
        });

        itemInterrupciones.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                itemESActionPerformed(evt);
            }
        });

        grupoBotonesES.add(itemMapeada);
        grupoBotonesES.add(itemInterrupciones);
        grupoBotonesES.add(itemDesactivada);

        menuCaminoDatos.setText(lenguaje.getString("caminoDeDatos"));

        //itemMonociclo.setSelected(true);
        itemMonociclo.setText(lenguaje.getString("monociclo"));
        grupoBotones.add(itemMonociclo);

        menuCaminoDatos.add(itemMonociclo);
```

```
        itemMulticiclo.setText(lenguaje.getString("multiciclo"));
        grupoBotones.add(itemMulticiclo);
        itemMulticiclo.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                itemMulticicloActionPerformed(evt);
            }
        });

        menuCaminoDatos.add(itemMulticiclo);
        // cambios por adelantamiento

        menuAdelantamiento.setText("Básico");
        itemAdelantamiento.setSelected(true);
        itemAdelantamiento.setText("Con Adelantamiento");
        itemAdelantamiento.addActionListener(new
java.awt.event.ActionListener() {
         public void actionPerformed(java.awt.event.ActionEvent evt) {
            itemAdelantamientoActionPerformed(evt);
              }
             });
          grupoBotonesAdelantamiento.add(itemAdelantamiento);
          itemNoAdelantamiento.setText("Sin Adelantamiento");
          itemNoAdelantamiento.addActionListener(new
java.awt.event.ActionListener() {
           public void actionPerformed(java.awt.event.ActionEvent evt) {
              itemNoAdelantamientoActionPerformed(evt);
                }
             });
          grupoBotonesAdelantamiento.add(itemNoAdelantamiento);
          menuAdelantamiento.add(itemAdelantamiento);
          menuAdelantamiento.add(itemNoAdelantamiento);
          menuSegmentado.add(menuAdelantamiento);

          // fin de cambios de adelantamiento


        itemMarcador.setText(lenguaje.getString("marcador"));

        grupoBotones.add(itemMarcador);
        itemMarcador.addActionListener(new java.awt.event.ActionListener() {
          public void actionPerformed(java.awt.event.ActionEvent evt) {
              itemMarcadorActionPerformed(evt);
          }
      });
      menuSegmentado.add(itemMarcador);

      itemTomasulo.setText(lenguaje.getString("tomasulo"));
      grupoBotones.add(itemTomasulo);
      itemTomasulo.addActionListener(new java.awt.event.ActionListener() {
          public void actionPerformed(java.awt.event.ActionEvent evt) {
              itemTomasuloActionPerformed(evt);
          }
      });
      menuSegmentado.add(itemTomasulo);

      menuSegmentado.setText(lenguaje.getString("segmentado"));
      menuCaminoDatos.add(menuSegmentado);
      MenuConfiguracion.add(menuCaminoDatos);
```

```
      //Menu Salto nuevo para introducir 2 y 3 huecos de retardo

      menuSalto.setText(lenguaje.getString("salto"));

      menuSalto1Hueco.setText("Un Hueco");
      menuSalto2Huecos.setText("Dos Hueco");
      menuSalto3Huecos.setText("Tres Huecos");
      itemSaltoRetardadoFlotante.setSelected(true);
      itemSalto1Hueco.setSelected(true);
      //itemSaltoRetardadoFlotante2Huecos.setSelected(true);
      //itemSaltoRetardadoFlotante3Huecos.setSelected(true);
    itemSaltoRetardadoFlotante.setText(lenguaje.getString("saltoRetardado"));
    itemSaltoRetardadoFlotante.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            itemSaltoRetardadoFlotanteActionPerformed(evt);
        }
    });
    grupoBotonesSalto1Hueco.add(itemSaltoRetardadoFlotante);


itemSaltoRetardadoFlotante2Huecos.setText(lenguaje.getString("saltoRetardado"))
;
  itemSaltoRetardadoFlotante2Huecos.addActionListener(new
java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
          itemSaltoRetardadoFlotante2HuecosActionPerformed(evt);
      }
  });
  grupoBotonesSalto3Huecos.add(itemSaltoRetardadoFlotante3Huecos);


itemSaltoRetardadoFlotante3Huecos.setText(lenguaje.getString("saltoRetardado"))
;
  itemSaltoRetardadoFlotante3Huecos.addActionListener(new
java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
          itemSaltoRetardadoFlotante3HuecosActionPerformed(evt);
      }
  });
  grupoBotonesSalto2Huecos.add(itemSaltoRetardadoFlotante2Huecos);
  grupoBotonesSalto3Huecos.add(itemSaltoRetardadoFlotante3Huecos);


    menuSalto1Hueco.add(itemSaltoRetardadoFlotante);
    menuSalto2Huecos.add(itemSaltoRetardadoFlotante2Huecos);
    menuSalto3Huecos.add(itemSaltoRetardadoFlotante3Huecos);

    itemSaltoFijoFlotante.setText("Predecir no tomado");
    itemSaltoFijoFlotante.addActionListener(new java.awt.event.ActionListener()
{
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            itemSaltoFijoFlotanteActionPerformed(evt);
        }
    });

    itemSaltoFijoFlotante2Huecos.setText("Predecir no tomado");
```

```
    itemSaltoFijoFlotante2Huecos.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            itemSaltoFijoFlotante2HuecosActionPerformed(evt);
        }
    });

    itemSaltoFijoFlotante3Huecos.setText("Predecir no tomado");
    itemSaltoFijoFlotante3Huecos.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            itemSaltoFijoFlotante3HuecosActionPerformed(evt);
        }
    });
    grupoBotonesSalto1Hueco.add(itemSaltoFijoFlotante);
    grupoBotonesSalto2Huecos.add(itemSaltoFijoFlotante2Huecos);
    grupoBotonesSalto3Huecos.add(itemSaltoFijoFlotante3Huecos);


    menuSalto1Hueco.add(itemSaltoFijoFlotante);
    menuSalto2Huecos.add(itemSaltoFijoFlotante2Huecos);
    menuSalto3Huecos.add(itemSaltoFijoFlotante3Huecos);
    menuSalto.add(menuSalto1Hueco);
    menuSalto.add(menuSalto2Huecos);
    menuSalto.add(menuSalto3Huecos);


    MenuConfiguracion.add(menuSalto);


        BarraMenuComp.add(MenuConfiguracion);

        MenuAyuda.setMnemonic('h');
        MenuAyuda.setText(lenguaje.getString("ayuda"));
        MenuAyuda.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                MenuAyudaActionPerformed(evt);
            }
        });


ItemAcercaDe.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.K
eyEvent.VK_H, java.awt.event.InputEvent.CTRL_MASK));
        //itemAcercaDe.setMnemonic('a');
        ItemAcercaDe.setText(lenguaje.getString("acercaDe"));
        ItemAcercaDe.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                itemAcercaDeActionPerformed(evt);
            }
        });


        MenuAyuda.add(ItemAcercaDe);

ItemAcercaDe.getAccessibleContext().setAccessibleDescription("itemAcercaDe");

        ItemAyuda.setText(lenguaje.getString("ayuda"));
        ItemAyuda.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
                itemAyudaActionPerformed(evt);
            }
        });


        MenuAyuda.add(ItemAyuda);
        ItemAyuda.getAccessibleContext().setAccessibleDescription("itemAyuda");

        BarraMenuComp.add(MenuAyuda);
        MenuAyuda.getAccessibleContext().setAccessibleDescription("MenuAyuda");

        setJMenuBar(BarraMenuComp);

BarraMenuComp.getAccessibleContext().setAccessibleName("BarraMenuComp");

BarraMenuComp.getAccessibleContext().setAccessibleDescription("BarraMenuComp");

        pack();
    }//GEN-END:initComponents


    private void itemESActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_itemMulticicloActionPerformed
        Editor.requestFocus();
        itemMonociclo.setSelected(true);

    }//GEN-LAST:event_itemMulticicloActionPerformed


    private void itemMulticicloActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_itemMulticicloActionPerformed
        Editor.requestFocus();
        new VentanaMulticiclo(this, Editor, false).show();

    }//GEN-LAST:event_itemMulticicloActionPerformed

    private void itemSegmentadoActionPerformed(java.awt.event.ActionEvent evt)
{
             new DialogoSegmentado(this, true);
             this.itemSegmentado.setSelected(true);

        }

    private void
itemSaltoRetardadoFlotanteActionPerformed(java.awt.event.ActionEvent evt) {
             new DialogoSegmentado(this, true);
             this.itemSegmentado.setSelected(true);
             set2HuecoFalse();
             set3HuecoFalse();
             this.Ejecutar.setEnabled(false);
             this.itemSalto1Hueco.setSelected(true);
             this.itemSalto2Hueco.setSelected(false);
             this.itemSalto3Hueco.setSelected(false);
             this.itemSaltoRetardadoFlotante.setSelected(true);
             this.itemSaltoFijoFlotante.setSelected(false);


        }
```

```
    private void
itemSaltoRetardadoFlotante2HuecosActionPerformed(java.awt.event.ActionEvent
evt) {
                new DialogoSegmentado(this, true);
                this.itemSegmentado.setSelected(true);
                set1HuecoFalse();
                set3HuecoFalse();
                this.Ejecutar.setEnabled(false);
                this.itemSalto2Hueco.setSelected(true);
                this.itemSalto1Hueco.setSelected(false);
                this.itemSalto3Hueco.setSelected(false);
                this.itemSaltoRetardadoFlotante2Huecos.setSelected(true);
                this.itemSaltoFijoFlotante2Huecos.setSelected(false);
                    }

    private void
itemSaltoRetardadoFlotante3HuecosActionPerformed(java.awt.event.ActionEvent
evt) {
                new DialogoSegmentado(this, true);
                this.itemSegmentado.setSelected(true);
                set1HuecoFalse();
                set2HuecoFalse();
                this.Ejecutar.setEnabled(false);
                this.itemSalto3Hueco.setSelected(true);
                this.itemSalto1Hueco.setSelected(false);
                this.itemSalto2Hueco.setSelected(false);
                this.itemSaltoRetardadoFlotante3Huecos.setSelected(true);
                this.itemSaltoFijoFlotante3Huecos.setSelected(false);

        }

    private void
itemSaltoFijoFlotanteActionPerformed(java.awt.event.ActionEvent evt) {
                new DialogoSegmentado(this, true);
                this.itemSegmentado.setSelected(true);
                set2HuecoFalse();
                set3HuecoFalse();
                this.Ejecutar.setEnabled(false);
                this.itemSalto1Hueco.setSelected(true);
                this.itemSalto2Hueco.setSelected(false);
                this.itemSalto3Hueco.setSelected(false);
                this.itemSaltoRetardadoFlotante.setSelected(false);
                this.itemSaltoFijoFlotante.setSelected(true);

        }

    private void
itemSaltoFijoFlotante2HuecosActionPerformed(java.awt.event.ActionEvent evt) {
                new DialogoSegmentado(this, true);
                this.itemSegmentado.setSelected(true);
                set1HuecoFalse();
                set3HuecoFalse();
                this.Ejecutar.setEnabled(false);
                this.itemSalto2Hueco.setSelected(true);
                this.itemSalto1Hueco.setSelected(false);
                this.itemSalto3Hueco.setSelected(false);
                this.itemSaltoRetardadoFlotante2Huecos.setSelected(false);
                this.itemSaltoFijoFlotante2Huecos.setSelected(true);
```

```
        }

    private void
itemSaltoFijoFlotante3HuecosActionPerformed(java.awt.event.ActionEvent evt) {
             new DialogoSegmentado(this, true);
             this.itemSegmentado.setSelected(true);
             set1HuecoFalse();
             set2HuecoFalse();
             this.Ejecutar.setEnabled(false);
             this.itemSalto3Hueco.setSelected(true);
             this.itemSalto1Hueco.setSelected(false);
             this.itemSalto2Hueco.setSelected(false);
             this.itemSaltoRetardadoFlotante3Huecos.setSelected(false);
             this.itemSaltoFijoFlotante3Huecos.setSelected(true);

        }




    private void itemMarcadorActionPerformed(java.awt.event.ActionEvent evt) {
             new DialogoMarcador(this, true);

        }

    private void itemTomasuloActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_itemMulticicloActionPerformed
       new DialogoTomasulo(this, true);

    }

    private void MenuAyudaActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_MenuAyudaActionPerformed

    }//GEN-LAST:event_MenuAyudaActionPerformed


/**Metodo que indica que se ha producido un cambio en el area de texto
 *@param event evento
 *@return void
 **/
    private void HaCambiado(javax.swing.event.CaretEvent evt) {//GEN-
FIRST:event_HaCambiado
        Ensamblar.setEnabled(true);
        Ejecutar.setEnabled(false);
    }//GEN-LAST:event_HaCambiado


    private void MenuConfiguracionActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_MenuConfiguracionActionPerformed
        // Add your handling code here:
    }//GEN-LAST:event_MenuConfiguracionActionPerformed

/**Metodo que cierra la aplicacion
 *@param event evento
 *@return void
 **/
```

```
    private void exitForm(java.awt.event.WindowEvent evt) {//GEN-
FIRST:event_exitForm
        // Add your handling code here:
        ItemSalir.doClick();
    }//GEN-LAST:event_exitForm
    // cambios para adelantamiento

    private void itemAdelantamientoActionPerformed (java.awt.event.ActionEvent
evt)
    {
      new DialogoSegmentado (this,true);
      this.itemSegmentado.setSelected(true);
      this.itemAdelantamiento.setSelected(true);
      this.itemNoAdelantamiento.setSelected(false);


    }
    private void itemNoAdelantamientoActionPerformed
(java.awt.event.ActionEvent evt)
    {
      new DialogoSegmentado (this,true);
      this.itemSegmentado.setSelected(true);
      this.itemNoAdelantamiento.setSelected(true);
      this.itemAdelantamiento.setSelected(false);
      this.itemMonociclo.setSelected(false);
    }

    // fin cambios adelantamiento



/**Metodo para el escuchador del boton ejecutar
 *@param event evento
 *@return void
 **/
    private void EjecutarActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_EjecutarActionPerformed
      // Add your handling code here:
      this.setVisible(false);
      if (itemMapeada.isSelected()&& (itemMonociclo.isSelected())){
            new VistaMonoESProg(this).setVisible(true);
            //System.out.println("monociclo mapeada");
      }
      if(itemInterrupciones.isSelected() && (itemMonociclo.isSelected())){
            ensambla.analizaDato("s1: .word 0");
            ensambla.analizaDato("s2: .word 0");
            new VistaMonoESInterrup(this).setVisible(true);
            //System.out.println("monociclo interrepcion");
      }
      if (itemMonociclo.isSelected() && (itemDesactivada.isSelected())){
            new VistaMonociclo(this).setVisible(true);
            //System.out.println("monociclo");
      }

      if (itemMulticiclo.isSelected())
            new
Multiciclo(this,entero_uno,entero_dos,entero_tres).setVisible(true);
      if(itemSegmentado.isSelected()){
```

```
            if(itemSalto1Hueco.isSelected()){

                  if (itemAdelantamiento.isSelected())
                        adelantamiento = Tipos.CON_ADELANTAMIENTO;
                  else
                        adelantamiento = Tipos.SIN_ADELANTAMIENTO;

                  if (itemSaltoRetardadoFlotante.isSelected()){
                        new VistaSegmentado(this, Tipos.SALTO_RETARDAD0,
this.getFus(), this.getTiposFus(),adelantamiento).setVisible(true);

                  }
                  if (this.itemSaltoFijoFlotante.isSelected()){
                        new VistaSegmentado(this, Tipos.SALTO_FIJO,
this.getFus(), this.getTiposFus(),adelantamiento).setVisible(true);
                  }

            }

            else if(itemSalto2Hueco.isSelected()){

                  if (itemAdelantamiento.isSelected())
                        adelantamiento = Tipos.CON_ADELANTAMIENTO;
                  else
                        adelantamiento = Tipos.SIN_ADELANTAMIENTO;

                  if (itemSaltoRetardadoFlotante2Huecos.isSelected()){
                        new VistaSegmentado(this,
Tipos.SALTO_RETARDAD0_2HUECOS, this.getFus(),
this.getTiposFus(),adelantamiento).setVisible(true);
                  }
                  if (this.itemSaltoFijoFlotante2Huecos.isSelected()){
                        new VistaSegmentado(this, Tipos.SALTO_FIJO_2HUECOS,
this.getFus(), this.getTiposFus(),adelantamiento).setVisible(true);
                  }

            }

            else if(itemSalto3Hueco.isSelected()){

                  if (itemAdelantamiento.isSelected())
                        adelantamiento = Tipos.CON_ADELANTAMIENTO;
                  else
                        adelantamiento = Tipos.SIN_ADELANTAMIENTO;

                  if (itemSaltoRetardadoFlotante3Huecos.isSelected()){
                        new VistaSegmentado(this,
Tipos.SALTO_RETARDAD0_3HUECOS, this.getFus(),
this.getTiposFus(),adelantamiento).setVisible(true);
                  }
                  if (this.itemSaltoFijoFlotante3Huecos.isSelected()){
                        new VistaSegmentado(this, Tipos.SALTO_FIJO_3HUECOS,
this.getFus(), this.getTiposFus(),adelantamiento).setVisible(true);
                  }

            }

      }
      if (itemMarcador.isSelected()){
```

```
            new VistaAlgoritmos(this,
Tipos.MARCADOR,this.getFus()).setVisible(true);
       }
       if (itemTomasulo.isSelected()){

            new VistaAlgoritmos(this, Tipos.TOMASULO,fus).setVisible(true);
       }

    }//GEN-LAST:event_EjecutarActionPerformed


    public void ejecutarAlgoritmo(){

      new VistaAlgoritmos(this, Tipos.MARCADOR,this.getFus()).setVisible(true);

    }

    public Vector<Vector> getErs() {
      return ers;
    }


public void setErs(Vector<Vector> ers) {
       this.ers = ers;
}


public Vector<Vector> getFus() {
       return fus;
}

public void setFus(Vector<Vector> fus) {
       this.fus = fus;
}

public void setTiposFus(Vector<Integer> tiposFus) {
       this.tiposFus = tiposFus;
}

public Vector<Integer> getTiposFus() {
       return tiposFus;
}


/**Metodo para el escuchador del boton error siguiente
 *@param event evento
 *@return void
 **/
    private void botonErrorActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_botonErrorActionPerformed

        String token;
        if(st.countTokens()>0)
        {
              if((token=st.nextToken()).indexOf("\t")>-1)
              {
                st2=new StringTokenizer(token, "\t");
                token=st2.nextToken();
              }
```

```
            BuscarError(token);
           if(st.countTokens()==0)
               botonError.setEnabled(false);
        }
    }//GEN-LAST:event_botonErrorActionPerformed


/**Metodo para el escuchador del boton ensamblar
 *@param event evento
 *@return void
 **/
    private void EnsamblarActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_EnsamblarActionPerformed
        // Add your handling code here:
       if(itemSalto1Hueco.isSelected()){
             itemSaltoRetardadoFlotante2Huecos.setSelected(false);
             itemSaltoFijoFlotante2Huecos.setSelected(false);
             itemSaltoRetardadoFlotante3Huecos.setSelected(false);
             itemSaltoFijoFlotante3Huecos.setSelected(false);
             if(itemSaltoFijoFlotante.isSelected())
                   salto=Tipos.SALTO_FIJO;
             else if (itemSaltoRetardadoFlotante.isSelected())
             salto=Tipos.SALTO_RETARDAD0;
       }
       if(itemSalto2Hueco.isSelected()){
             this.itemSaltoRetardadoFlotante.setSelected(false);
             this.itemSaltoFijoFlotante.setSelected(false);
             this.itemSaltoRetardadoFlotante3Huecos.setSelected(false);
             this.itemSaltoFijoFlotante3Huecos.setSelected(false);

             if(itemSaltoFijoFlotante2Huecos.isSelected())
                   salto=Tipos.SALTO_FIJO_2HUECOS;
             else if(itemSaltoRetardadoFlotante2Huecos.isSelected())
                   salto=Tipos.SALTO_RETARDAD0_2HUECOS;
       }
       if(itemSalto3Hueco.isSelected()){
             itemSaltoRetardadoFlotante2Huecos.setSelected(false);
             itemSaltoFijoFlotante2Huecos.setSelected(false);
             itemSaltoRetardadoFlotante.setSelected(false);
             itemSaltoFijoFlotante.setSelected(false);

             if(itemSaltoFijoFlotante3Huecos.isSelected())
             salto=Tipos.SALTO_FIJO_3HUECOS;
            else if(itemSaltoRetardadoFlotante3Huecos.isSelected())
             salto=Tipos.SALTO_RETARDAD0_3HUECOS;
       }


         ensambla=new Ensamblar(Editor.getText(), salto);
         Errores.setText("");
         listaErrores=ensambla.analiza();
         if(listaErrores!=null)
         {
             Ejecutar.setEnabled(false);
             Errores.setText(listaErrores);
             Errores.setCaretPosition(0);
             Errores.setMargin(new Insets(5,5,5,5));
             botonError.setEnabled(true);
             st=new StringTokenizer(listaErrores, "\n");
```

```
            botonError.doClick();
        }
        else{
            Ejecutar.setEnabled(true);
        }

    }//GEN-LAST:event_EnsamblarActionPerformed

/**Metodo para establecer cambios en el area de texto
 *@param event evento
 *@return void
 **/
    private void Cambios(java.awt.event.KeyEvent evt) {//GEN-
FIRST:event_Cambios
         modificar=true;
        Ensamblar.setEnabled(true);
        Ejecutar.setEnabled(false);
    }//GEN-LAST:event_Cambios

/**Metodo para el escuchador del item rehacer
 *@param event evento
 *@return void
 **/
    private void ItemRehacerActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_ItemRehacerActionPerformed
         redoAction.actionPerformed(evt);
        Editor.requestFocus();
    }//GEN-LAST:event_ItemRehacerActionPerformed

    private void MenuEdicionActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_MenuEdicionActionPerformed
        // Add your handling code here:
    }//GEN-LAST:event_MenuEdicionActionPerformed

/**Metodo para el escuchador del item deshacer
 *@param event evento
 *@return void
 **/
    private void ItemDeshacerActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_ItemDeshacerActionPerformed
         undoAction.actionPerformed(evt);
        Editor.requestFocus();
    }//GEN-LAST:event_ItemDeshacerActionPerformed

/**Metodo para el escuchador del boton imprimir
 *@param event evento
 *@return void
 **/
    private void BotonImprimirActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_BotonImprimirActionPerformed
        // Add your handling code here:

       String impresion = String.valueOf(Editor.getText());
       if (!impresion.equals(""))
       {

              final  String lineBreak = System.getProperty("line.separator");

             impresion=impresion.replaceAll("\n", lineBreak);
```

```
            byte[] bytes;
//          Transformamos el texto a bytes que es lo que soporta la impresora
            bytes=impresion.getBytes();



            PrintRequestAttributeSet pras=new HashPrintRequestAttributeSet();
            DocFlavor flavor = DocFlavor.BYTE_ARRAY.AUTOSENSE;
            PrintService
printService[]=PrintServiceLookup.lookupPrintServices(flavor, pras);
            PrintService
defaultService=PrintServiceLookup.lookupDefaultPrintService();
            PrintService service=ServiceUI.printDialog(null, 200, 200,
printService, defaultService, flavor, pras);
            if(service!=null){
                  DocPrintJob job=service.createPrintJob();
                  DocAttributeSet das=new HashDocAttributeSet();
                  Doc doc=new SimpleDoc(bytes,flavor, das);
                  try {
                        job.print(doc, pras);
                  } catch (PrintException e) {

                        e.printStackTrace();
                  }
            }
      }

      Editor.requestFocus();
      Editor.select(0, 0);

    }//GEN-LAST:event_BotonImprimirActionPerformed

    private void MenuBuscarActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_MenuBuscarActionPerformed

    }//GEN-LAST:event_MenuBuscarActionPerformed

    private void nLinea(java.awt.event.KeyEvent evt) {//GEN-FIRST:event_nLinea

    }//GEN-LAST:event_nLinea

/**Metodo para el escuchador del item acercaDe
 *@param event evento
 *@return void
 **/
    private void itemAcercaDeActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_itemAcercaDeActionPerformed
        new PanelAyuda().setVisible(true);
    }//GEN-LAST:event_itemAcercaDeActionPerformed


    /**Metodo para el escuchador del item ayuda
     *@param event evento
     *@return void
     **/
        private void itemAyudaActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_itemAcercaDeActionPerformed
            new GuiaRapida().setVisible(true);
        }
```

```
/**Metodo para el escuchador del item reemplazar
 *@param event evento
 *@return void
 **/
    private void ItemReemplazarActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_ItemReemplazarActionPerformed
        Editor.requestFocus();
        int reemp=3;
        new Busqueda(this, Editor, false, reemp).setVisible(true);
    }//GEN-LAST:event_ItemReemplazarActionPerformed

/**Metodo para el escuchador del item buscar texto
 *@param event evento
 *@return void
 **/
    private void ItemTextoActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_ItemTextoActionPerformed
        Editor.requestFocus();
        int texto=2;
        new Busqueda(this, Editor, false,texto).setVisible(true);
    }//GEN-LAST:event_ItemTextoActionPerformed

/**Metodo para el escuchador del item busscar linea
 *@param event evento
 *@return void
 **/
    private void ItemLineaActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_ItemLineaActionPerformed
        this.setFocusableWindowState(false);
        Editor.requestFocus();
        int linea=1;
        new Busqueda(this, Editor, false,linea).setVisible(true);
    }//GEN-LAST:event_ItemLineaActionPerformed

/**Metodo para el escuchador del boton pegar
 *@param event evento
 *@return void
 **/
    private void ItemPegarActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_ItemPegarActionPerformed
        Editor.paste();
        Editor.requestFocus();
    }//GEN-LAST:event_ItemPegarActionPerformed

/**Metodo para el escuchador del boton cortar
 *@param event evento
 *@return void
 **/
    private void ItemCortarActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_ItemCortarActionPerformed
        activaPegar();
        Editor.cut();
        Editor.requestFocus();
    }//GEN-LAST:event_ItemCortarActionPerformed

/**Metodo para el escuchador del boton copiar
 *@param event evento
```

```
 *@return void
 **/
    private void ItemCopiarActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_ItemCopiarActionPerformed
        activaPegar();
        Editor.copy();
        Editor.requestFocus();
    }//GEN-LAST:event_ItemCopiarActionPerformed

/**Metodo para el escuchador del item seleccionar todo
 *@param event evento
 *@return void
 **/
    private void ItemSeleccionarTodoActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_ItemSeleccionarTodoActionPerformed
        Editor.selectAll();
    }//GEN-LAST:event_ItemSeleccionarTodoActionPerformed

/**Metodo para el escuchador del etem salir
 *@param event evento
 *@return void
 **/
    private void ItemSalirActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_ItemSalirActionPerformed
        int r=0;
        //if (modificar)
        //{
            r=GuardarCambios();
            if(r!=0)    //r==0 --> cancelar
             System.exit(0);
        //}
    }//GEN-LAST:event_ItemSalirActionPerformed

/**Metodo para el escuchador del item abrir
 *@param event evento
 *@return void
 **/
    private void ItemAbrirActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_ItemAbrirActionPerformed
         int r=0;
        if (modificar)
            r=GuardarCambios();
        if ((r==1) || (!modificar)) {
            FileDialog Dialogo = new FileDialog(this, "Abrir...",
FileDialog.LOAD);
            Dialogo.setVisible(true);
            if (Dialogo.getFile()==null)
                return;
            nombreF=Dialogo.getFile();
            NombreFich = Dialogo.getDirectory() + nombreF;
            java.awt.Frame ventana = (Frame)Dialogo.getParent();
            ventana.setTitle("Simula3MS "+nombreF);
            FileInputStream fis=null;
            String str=null;
            try{
                fis=new FileInputStream(NombreFich);
                int tam=fis.available();
                byte[] bytes=new byte[tam];
                fis.read(bytes);
```

```
                str=new String(bytes);
            } catch (IOException e) {
            } finally{
                try{
                    fis.close();
                }catch (IOException e2){
                }
            }

           if (str!=null)
           {
                Editor.setText(str);
                Ensamblar.setEnabled(true);
                Errores.setText("");
           }
        }
        Editor.requestFocus();

    }//GEN-LAST:event_ItemAbrirActionPerformed

/**Metodo para el escuchador del item nuevo
 *@param event evento
 *@return void
 **/
    private void ItemNuevoActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_ItemNuevoActionPerformed
        int r=0;
        if (modificar)
            r=GuardarCambios();

        if ((r==1) || (!modificar)) {
            NombreFich = "";
            nombreF="";
            javax.swing.JFrame ventana =
(javax.swing.JFrame)Editor.getParent().getParent().getParent().getParent().getP
arent().getParent().getParent().getParent();
            ventana.setTitle("Simula3MS "+nombreF);
            Editor.setText("");
            Errores.setText("");
             Ensamblar.setEnabled(false);
        }
        Editor.requestFocus();

    }//GEN-LAST:event_ItemNuevoActionPerformed

/**Metodo para el escuchador del item guardar como
 *@param event evento
 *@return void
 **/
    private void ItemGuardarComoActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_ItemGuardarComoActionPerformed
        GuardarComo();
        Editor.requestFocus();
    }//GEN-LAST:event_ItemGuardarComoActionPerformed

/**Metodo para el escuchador del item guardar
 *@param event evento
 *@return void
 **/
```

```
    private void ItemGuardarActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_ItemGuardarActionPerformed
        if ("".equals(NombreFich))
            GuardarComo();
        else
            Guardar(NombreFich);
        Editor.requestFocus();
    }//GEN-LAST:event_ItemGuardarActionPerformed

    private void MenuArchivoActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_MenuArchivoActionPerformed

    }//GEN-LAST:event_MenuArchivoActionPerformed

/**Metodo que guarda cambios en un fichero
 *@param String nombre del fichero
 *@return void
 **/
    private void Guardar(String NombreFich) {
        FileOutputStream fos=null;
        String str=Editor.getText();
        try{
            fos=new FileOutputStream(NombreFich);
            fos.write(str.getBytes());
        } catch (IOException e) {
        }
        finally {
            try{
                fos.close();
            } catch (IOException e2){
            }
        }
        modificar=false;
    }

/**Metodo que guarda cambios en un fichero
 *@param Sin parametros
 *@return void
 **/
    private void GuardarComo() {
        FileDialog Dialogo=new FileDialog(this, "Guardar como...",
FileDialog.SAVE);
        Dialogo.setVisible(true);
        if (Dialogo.getFile()==null)
            return;

        nombreF=Dialogo.getFile();
        NombreFich = Dialogo.getDirectory() + nombreF;
        java.awt.Frame ventana = (Frame)Dialogo.getParent();
        ventana.setTitle("Simula3MS "+nombreF);
        Guardar(NombreFich);
    }


/**Metodo que muestra el dialogo de guardar cambios
 *@param Sin parametros
 *@return int respuesta
 **/
    private int GuardarCambios() {
```

```
        JOptionPane panel=new JOptionPane();
        Object[] opciones={"SI", "NO", "CANCELAR"};
        int resp=panel.showOptionDialog(this, "Desea guardar cambios?",
"Guardar cambios", JOptionPane.YES_NO_CANCEL_OPTION,
JOptionPane.QUESTION_MESSAGE, null, opciones, opciones[0]);
        if (resp==JOptionPane.YES_OPTION)
            if ("".equals(NombreFich))
                GuardarComo();
            else
                Guardar(NombreFich);
        if (resp==JOptionPane.CANCEL_OPTION)
        {
             return 0;
        }
        return 1;
    }


/**Metodo que activa el boton pegar
 *@param Sin parametros
 *@return void
 **/
    public void activaPegar(){
        ItemPegar.setEnabled(true);
        BotonPegar.setEnabled(true);
    }


/**Metodo que busca el error siguiente
 *@param Strin error
 *@return int posicion del error
 **/
     public int BuscarError(String textoBuscado)
    {
        String txt, str;
        txt=Editor.getText();
        str=textoBuscado;

        Editor.requestFocus();
        int sel;
        if (Editor.getSelectionStart() != Editor.getSelectionEnd())
        {
            sel =Editor.getSelectionEnd();
        }
        else
            sel=0;
        int pos=txt.indexOf(str, sel);
        if (pos>0)
        {
                Editor.select(pos, pos + str.length());
        }
        return pos;
    }

/**Metodo que actualiza cambios
 *@param Observable o, Object error
 *@return void
 **/
    public void update(Observable o, Object error)
```

```
    {
        String errores=(String)error;
        Errores.setText(errores);
    }

    /**Metodo para obtener el nombre del fichero
     *@param
     *@return String nombreFichero
     **/
    public String getNombreF()
    {
      return this.nombreF;
    }

    public void set1HuecoFalse()
    {

      itemSaltoRetardadoFlotante = new javax.swing.JRadioButtonMenuItem();
        itemSaltoFijoFlotante = new javax.swing.JRadioButtonMenuItem();


    }

    public void set2HuecoFalse()
    {
      itemSaltoRetardadoFlotante2Huecos.setSelected(false);
      itemSaltoFijoFlotante2Huecos.setSelected(false);

    }
    public void set3HuecoFalse()
    {
      itemSaltoRetardadoFlotante3Huecos.setSelected(false);
      itemSaltoFijoFlotante3Huecos.setSelected(false);

    }

/**Metodo de inicio de la aplicacion
 *@param Sin parametros
 *@return void
 **/
    public static void main(String args[]) {

      //Se elige el idioma de la aplicacion

      String leng;

      if (args.length ==1) {
            leng = new String(args[0]);
            }
      else {
            leng = "es";
      }

      Lenguaje.inicializar(leng);

            //Se visualiza el panel de inicio
        PanelInicio p=new PanelInicio();
        p.setVisible(true);
```

```
    try{
        java.lang.Thread.sleep(3000);
    } catch(InterruptedException e){}
    p.setVisible(false);
    new Ensamblador().setVisible(true);

}


// Variables declaration - do not modify//GEN-BEGIN:variables
private javax.swing.JMenuBar BarraMenuComp;
private javax.swing.JButton BotonAbrir;
private javax.swing.JButton BotonBuscar;
private javax.swing.JButton BotonCopiar;
private javax.swing.JButton BotonCortar;
protected javax.swing.JButton BotonDeshacer;
private javax.swing.JButton BotonGuardar;
private javax.swing.JButton BotonGuardarComo;
private javax.swing.JButton BotonImprimir;
private javax.swing.JButton BotonNuevo;
private javax.swing.JButton BotonPegar;
protected javax.swing.JButton BotonRehacer;
protected javax.swing.JTextArea Editor;
private javax.swing.JButton Ejecutar;
private javax.swing.JButton Ensamblar;
private javax.swing.JEditorPane Errores;
private javax.swing.JMenuItem ItemAbrir;
private javax.swing.JMenuItem ItemAyuda;
private javax.swing.JMenuItem ItemAcercaDe;
private javax.swing.JMenuItem ItemCopiar;
private javax.swing.JMenuItem ItemCortar;
protected javax.swing.JMenuItem ItemDeshacer;
private javax.swing.JMenuItem ItemGuardar;
private javax.swing.JMenuItem ItemGuardarComo;
private javax.swing.JMenuItem ItemImprimir;
private javax.swing.JMenuItem ItemLinea;
private javax.swing.JMenuItem ItemNuevo;
private javax.swing.JMenuItem ItemPegar;
private javax.swing.JMenuItem ItemReemplazar;
protected javax.swing.JMenuItem ItemRehacer;
private javax.swing.JMenuItem ItemSalir;
private javax.swing.JMenuItem ItemSeleccionarTodo;
private javax.swing.JMenuItem ItemTexto;
private javax.swing.JTextField Linea;
private javax.swing.JMenu MenuArchivo;
private javax.swing.JMenu MenuAyuda;
private javax.swing.JMenu MenuBuscar;
private javax.swing.JMenu MenuConfiguracion;
private javax.swing.JMenu MenuEdicion;
private javax.swing.JMenu menuSegmentado;
private javax.swing.JButton botonError;
private javax.swing.ButtonGroup grupoBotones;
private javax.swing.ButtonGroup grupoBotonesSalto;
private javax.swing.ButtonGroup grupoBotonesSalto1Hueco;
private javax.swing.ButtonGroup grupoBotonesSalto2Huecos;
private javax.swing.ButtonGroup grupoBotonesSalto3Huecos;

private javax.swing.ButtonGroup grupoBotonesES;
private javax.swing.JRadioButtonMenuItem itemMonociclo;
```

```
    private javax.swing.JRadioButtonMenuItem itemMulticiclo;
    private javax.swing.JRadioButtonMenuItem itemMarcador;
    private javax.swing.JRadioButtonMenuItem itemTomasulo;
    private javax.swing.JRadioButtonMenuItem itemSegmentado;
    private javax.swing.JRadioButtonMenuItem itemMapeada;
    private javax.swing.JRadioButtonMenuItem itemInterrupciones;
    private javax.swing.JRadioButtonMenuItem itemDesactivada;
    // cambios por adelantamiento

    private javax.swing.JMenu menuAdelantamiento;
    public static javax.swing.JRadioButtonMenuItem itemAdelantamiento;
    private javax.swing.JRadioButtonMenuItem itemNoAdelantamiento;
    private javax.swing.ButtonGroup grupoBotonesAdelantamiento;

    // Fin cambios por adelantamiento


    private javax.swing.JMenu menuSegmentadoFlotante;
    private javax.swing.JMenu menuCaminoDatos;
    private javax.swing.JMenu menuEntradaSalida;
    private javax.swing.JMenu menuSalto;
    private javax.swing.JMenu menuSalto1Hueco;
    private javax.swing.JMenu menuSalto2Huecos;
    private javax.swing.JMenu menuSalto3Huecos;
    private javax.swing.JRadioButtonMenuItem itemSalto1Hueco;
    private javax.swing.JRadioButtonMenuItem itemSalto2Hueco;
    private javax.swing.JRadioButtonMenuItem itemSalto3Hueco;
    private javax.swing.JRadioButtonMenuItem itemSaltoRetardadoFlotante;
    private javax.swing.JRadioButtonMenuItem itemSaltoFijoFlotante;
    private javax.swing.JRadioButtonMenuItem itemSaltoRetardadoFlotante2Huecos;
    private javax.swing.JRadioButtonMenuItem itemSaltoFijoFlotante2Huecos;
    private javax.swing.JRadioButtonMenuItem itemSaltoRetardadoFlotante3Huecos;
    private javax.swing.JRadioButtonMenuItem itemSaltoFijoFlotante3Huecos;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel10;
    private javax.swing.JPanel jPanel11;
    private javax.swing.JPanel jPanel12;
    private javax.swing.JPanel jPanel13;
    private javax.swing.JPanel jPanel14;
    private javax.swing.JPanel jPanel15;
    private javax.swing.JPanel jPanel16;
    private javax.swing.JPanel jPanel17;
    private javax.swing.JPanel jPanel18;
    private javax.swing.JPanel jPanel19;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel20;
    private javax.swing.JPanel jPanel21;
    private javax.swing.JPanel jPanel22;
    private javax.swing.JPanel jPanel23;
    private javax.swing.JPanel jPanel24;
    private javax.swing.JPanel jPanel27;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JPanel jPanel5;
    private javax.swing.JPanel jPanel6;
    private javax.swing.JPanel jPanel7;
    private javax.swing.JPanel jPanel8;
    private javax.swing.JPanel jPanel9;
    private javax.swing.JScrollPane jScrollPane1;
```

```
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JSeparator jSeparator2;
private javax.swing.JSeparator jSeparator3;
private javax.swing.JSeparator jSeparator4;
private javax.swing.JSeparator jSeparator5;
private javax.swing.JSeparator jSeparator6;
private javax.swing.JToolBar jToolBar1;
private javax.swing.JToolBar jToolBar2;
private javax.swing.JToolBar jToolBar3;
private javax.swing.JToolBar jToolBar4;
// End of variables declaration//GEN-END:variables

public class RedoAction extends AbstractAction {

    public RedoAction(){
        super("Redo");
        update();
    }

    public void actionPerformed(java.awt.event.ActionEvent e) {
        undoManager.redo();
        undoAction.update();
        update();
    }

    public void update() {
        boolean canRedo=undoManager.canRedo();
        if(canRedo) {
            ItemRehacer.setEnabled(true);
            BotonRehacer.setEnabled(true);
            putValue(Action.NAME, undoManager.getRedoPresentationName());
        }
        else {
            ItemRehacer.setEnabled(false);
            BotonRehacer.setEnabled(false);
            putValue(Action.NAME,"Redo");
        }
    }
}



public class UndoLastAction extends AbstractAction {

    public UndoLastAction() {
        super("Undo");
        update();
    }

    public void actionPerformed(java.awt.event.ActionEvent e) {
        undoManager.undo();
        redoAction.update();
        update();
    }
    public void update() {
        boolean canUndo=undoManager.canUndo();

        if(canUndo) {
```

```
                BotonDeshacer.setEnabled(true);
                ItemDeshacer.setEnabled(true);
                putValue(Action.NAME, undoManager.getUndoPresentationName());
            }
            else {
                BotonDeshacer.setEnabled(false);
                ItemDeshacer.setEnabled(false);
                putValue(Action.NAME, "Undo");
            }
        }
    }




     }
```

2. TIPOS.JAVA

```
public interface Tipos {
    char[] registros={'z', 'a', 'v', 't', 's', 'r', 'f', 'k'};
    int REG_Z=0;
    int REG_A=1;
    int REG_V=2;
    int REG_T=3;
    int REG_S=4;
    int REG_R=5;
    int REG_F=6;
    int REG_K=7;
    int R_BADVADDR=8;
    int R_STATUS=12;
    int R_CAUSE=13;
    int R_EPC=14;
    Palabra INICIO_PILA=new Palabra("6fffffc",true);
    Palabra INICIO_PC=new Palabra("00400000",true);
    Palabra INICIO_MEMORIA=new Palabra("10010000",true);
    Palabra INICIO_ES=new Palabra("ffff0000",true);
    Palabra FIN_ES=new Palabra("ffff000c",true);
    Palabra ReceiverControl=new Palabra("ffff0000", true);
    Palabra ReceiverData=new Palabra("ffff0004", true);
    Palabra TransmitterControl=new Palabra("ffff0008",true);
    Palabra TransmitterData=new Palabra("ffff000c",true);
    Palabra TOPE_POS=new Palabra("7fffffff",true);
    Palabra TOPE_NEG=new Palabra(-2147483648);
    String [] regEspecial={"zero", "at", "k0", "k1", "gp"};

    int TEXT=0;
    int DATA=1;
    int TEXT0X=2;

    String registro[]={"$zero","$at","$v0","$v1","$a0","$a1","$a2","$a3",
    "$t0","$t1","$t2","$t3","$t4","$t5","$t6","$t7","$s0","$s1","$s2","$s3",
    "$s4","$s5","$s6","$s7","$t8","$s9","$k0","$k1","$gp","$sp","$s8","$ra"};

    String regFlotante[]={"$f0","$f1","$f2","$f3","$f4","$f5","$f6","$f7",
```

```
"$f8","$f9","$f10","$f11","$f12","$f13","$f14","$f15","$f16","$f17","$f18",
"$f19","$f20","$f21","$f22","$f23","$f24","$f25","$f26","$f27","$f28",
"$f29","$f30","$f31"};

int REG_ZERO=0;
int REG_AT=1;
int REG_V0=2;
int REG_V1=3;
int REG_A0=4;
int REG_A1=5;
int REG_A2=6;
int REG_A3=7;
int REG_T0=8;
int REG_T1=9;
int REG_T2=10;
int REG_T3=11;
int REG_T4=12;
int REG_T5=13;
int REG_T6=14;
int REG_T7=15;
int REG_S0=16;
int REG_S1=17;
int REG_S2=18;
int REG_S3=19;
int REG_S4=20;
int REG_S5=21;
int REG_S6=22;
int REG_S7=23;
int REG_T8=24;
int REG_S9=25;
int REG_K0=26;
int REG_K1=27;
int REG_GP=28;
int REG_SP=29;
int REG_S8=30;
int REG_RA=31;

int REG_PC=32;
int REG_HI=33;
int REG_LO=34;
int REG_STATUS=35;
int REG_CAUSE=36;
int REG_EPC=37;

int REG_F0=0;
int REG_F1=1;
int REG_F2=2;
int REG_F3=3;
int REG_F4=4;
int REG_F5=5;
int REG_F6=6;
int REG_F7=7;
int REG_F8=8;
int REG_F9=9;
int REG_F10=10;
int REG_F11=11;
int REG_F12=12;
int REG_F13=13;
int REG_F14=14;
```

```
    int REG_F15=15;
    int REG_F16=16;
    int REG_F17=17;
    int REG_F18=18;
    int REG_F19=19;
    int REG_F20=20;
    int REG_F21=21;
    int REG_F22=22;
    int REG_F23=23;
    int REG_F24=24;
    int REG_F25=25;
    int REG_F26=26;
    int REG_F27=27;
    int REG_F28=28;
    int REG_F29=29;
    int REG_F30=30;
    int REG_F31=31;

    String instrucciones[]={"add","addi","sub","lui","slt","slti","and","andi",

"or","ori","beq","bne","j","lw","sw","lb","sb","la","jal","jr","mult","div",

"mfhi","mflo","nop","syscall","add.s","sub.s","mul.s","div.s","add.d","sub.d",

"mul.d","div.d","bc1t","bc1f","abs.s","abs.d","neg.s","neg.d","mov.s","mov.d",
    "c.eq.s","c.eq.d","c.le.s","c.le.d","c.lt.s","c.lt.d","cvt.d.s","cvt.d.w",
    "cvt.s.d","cvt.s.w","cvt.w.d","cvt.w.s","mtc1","mfc1","lwc1","swc1", "li",
    "mfc0", "rfe", "sll", "srl", "sra"};

    int ADD=0;
    int ADDI=1;
    int SUB=2;
    int LUI=3;
    int SLT=4;
    int SLTI=5;
    int AND=6;
    int ANDI=7;
    int OR=8;
    int ORI=9;
    int BEQ=10;
    int BNE=11;
    int INS_J=12;
    int LW=13;
    int SW=14;
    int LB=15;
    int SB=16;
    int LA=17;
    int JAL=18;
    int JR=19;
    int MULT=20;
    int DIV=21;
    int MFHI=22;
    int MFLO=23;
    int NOP=24;
    int SYSCALL=25;
    int ADDS=26;
    int SUBS=27;
    int MULS=28;
    int DIVS=29;
```

```
    int ADDD=30;
    int SUBD=31;
    int MULD=32;
    int DIVD=33;
    int BC1T=34;
    int BC1F=35;
    int ABSS=36;
    int ABSD=37;
    int NEGS=38;
    int NEGD=39;
    int MOVS=40;
    int MOVD=41;
    int CEQS=42;
    int CEQD=43;
    int CLES=44;
    int CLED=45;
    int CLTS=46;
    int CLTD=47;
    int CVTDS=48;
    int CVTDW=49;
    int CVTSD=50;
    int CVTSW=51;
    int CVTWD=52;
    int CVTWS=53;
    int MTC1=54;
    int MFC1=55;
    int LWC1=56;
    int SWC1=57;
    int LI=58;
    int MFC0=59;
    int RFE=60;
    int SLL=61;
    int SRL=62;
    int SRA=63;

    String datos[]={".ascii",".asciiz",".word",".space",".float",".double"};

    int ASCII=0;
    int ASCIIZ=1;
    int WORD=2;
    int SPACE=3;
    int FLOTANTE=4;
    int DOBLE=5;


    String []codHex={"0", "1","2", "3", "4", "5", "6", "7", "8", "9", "a", "b",
"c", "d", "e", "f"};
    String []codBin={"0000", "0001", "0010", "0011", "0100", "0101", "0110",
"0111",
                          "1000", "1001", "1010", "1011", "1100", "1101",
"1110", "1111" };
    String []ascii={"\0"," ", "!", "\"", "#", "$", "%", "&", "`", "(", ")",
"*", "+", ",", "-", ".", "/",
    "0","1","2","3","4","5","6","7","8","9",":",";","<","=",">","?",
    "@","A","B","C","D","E","F","G","H","I","J","K","L","M","N","O",
    "P","Q","R","S","T","U","V","W","X","Y","Z","[","","]","^","-",
    "'","a","b","c","d","e","f","g","h","i","j","k","l","m","n","o",
    "p","q","r","s","t","u","v","w","x","y","z","{","_","}", "~" };
```

```
    String
[]hexa={"00","20","21","22","23","24","25","26","27","28","29","2a","2b","2c","
2d","2e","2f",

"30","31","32","33","34","35","36","37","38","39","3a","3b","3c","3d","3e","3f"
,

"40","41","42","43","44","45","46","47","48","49","4a","4b","4c","4d","4e","4f"
,

"50","51","52","53","54","55","56","57","58","59","5a","5b","5c","5d","5e","5f"
,

"60","61","62","63","64","65","66","67","68","69","6a","6b","6c","6d","6e","6f"
,

"70","71","72","73","74","75","76","77","78","79","7a","7b","7c","7d","7e"};
    int FI=0;
    int FJ=1;
    int FK=2;

    int UF_NO_SEGM=0;
    int UF_SEGM=1;


    int INTEGER_FU=0;
    int FP_ADD_FU=1;
    int FP_MULT_FU=2;
    int FP_DIV_FU=3;
    int TIPOS_FU=4;

    int INTEGER_ER=0;
    int FP_ADD_ER=1;
    int FP_MULT_ER=2;
    int FP_DIV_ER=3;
    int LOAD_ER=4;
    int STORE_ER=5;

    int TIPOS_ER=6;


    int MARCADOR=0;
    int TOMASULO=1;

    int INICIO=-1;
    int EMISION=0;
    int LECTURA=1;
    int EJECUCION=2;
    int ESCRITURA=3;

    int EMISION_TOM=0;
    int EJECUCION_TOM=1;
    int ESCRITURA_TOM=2;

    int ETAPA_IF=0;
    int ETAPA_ID=1;
    int ETAPA_EX=2;
    int ETAPA_MEM=3;
    int ETAPA_WB=4;
```

```
    int SALTO_RETARDAD0=0;
    int SALTO_FIJO=1;

    // Nuevos tipos para dos y tres huecos
    int SALTO_RETARDAD0_2HUECOS=2;
    int SALTO_FIJO_2HUECOS=3;
    int SALTO_RETARDAD0_3HUECOS=4;
    int SALTO_FIJO_3HUECOS=5;

   // Nuevos tipos para con y sin adelantamiento
    int CON_ADELANTAMIENTO = 0;
    int SIN_ADELANTAMIENTO = 1;

    int ANT_RS_MEM=0;
    int ANT_RT_MEM=1;
    int ANT_RS_WB=2;
    int ANT_RT_WB=3;
    int ANT_RS_EX=4;
    int ANT_RT_EX=5;
    int ANT_WB=6;

    int STRING=0;
    int INTEGER=1;
    int FLOAT=2;
    int DOUBLE=3;

    int TRANSMITER_CONTROL=2;
     }
```

3. INMEDIATAS.JAVA.

```
public abstract class Inmediatas extends TipoI {
private String
imagSegmentado[]={"inmediatasIF.gif","inmediatasID.gif","inmediatasEX.gif","inm
ediatasMEM.gif","inmediatasWB.gif"};
private String imagSegmentado2[]={"inmediatasIF.gif","inmediatasID sin
adelantamiento.gif","inmediatasEX.gif","inmediatasMEM.gif","inmediatasWB.gif"};
private String
imagSegmentadoSinAdelantamiento[]={"inmediatasIF.gif","inmediatasID sin
adelantamiento.gif","inmediatasEX sin adelantamiento.gif","inmediatasMEM sin
adelantamiento.gif","inmediatasWB sin adelantamiento.gif"};


    public void inicia()
    {
        String imagenes="inmed.gif";
        super.inicIma(imagenes);
    }

    public static String esValida(String[] param, int numParam)
    {
         String error=null;

        if(param.length==numParam+1)
        {
```

```
            for(int i=0; i<numParam;i++)
            {
                if(esRegistro(param[i]))
                {
                        if(param[i].equals("$zero") || param[i].equals("$0"))
                        {
                            if(i==0)
                                error+=param[i]+"parametros no validos. Reg
esp";
                        }
                }
                else
                {
                    error=param[i]+"parametros no validos. No es Reg";
                     return error;
                }
            }
            try{
             if((param[numParam].startsWith("0x")) &&
(param[numParam].length()>2) && (((param[numParam]).trim()).length()<11))
             {

                 try
                 {

       Long.parseLong(param[numParam].substring(2,param[numParam].length()),
16);

                 }catch (NumberFormatException e)
                 {
                 error=param[numParam]+"parametros no validos";
                 }
             }
             else {

                    Long.parseLong(param[numParam]);

             }
            }catch(NumberFormatException e){
             error=param[numParam]+"parametros no validos";}
        }
        else
             error="parametros no validos. Numero de parametros incorrecto";

        return error;
    }


    public abstract int ejecutar();
    public abstract String getCodificacion();

    public int numEtapas()
    {
        return 4;
    }
    public int etapa3()
    {
        imagen_etapa=new StringBuffer();
        imagen_etapa.append(direccion_multi).append("etapa3_inmed");
```

```
    return -1;
}

public abstract int etapa4();

public int etapa5()
{

    return -1;
}

public String imagenSegmentado(int etapa, int salto, int adelantamiento)
{
    if(salto==SALTO_FIJO ||salto==SALTO_RETARDAD0){
         if(adelantamiento==CON_ADELANTAMIENTO)
                return imagSegmentado[etapa];
         else
                return imagSegmentadoSinAdelantamiento[etapa];
    }
    else{
         if(adelantamiento==CON_ADELANTAMIENTO)
                return imagSegmentado2[etapa];
         else
                return imagSegmentadoSinAdelantamiento[etapa];
    }
}

public java.awt.Color colorSegm()
{
    return new java.awt.Color(20,185,233);
}

public int estilo()
{
    return 3;
}

public abstract int numero();

public abstract boolean hayRiesgo();

public abstract int ejecutarIF();

public abstract int ejecutarID();

public abstract int ejecutarEX();

public abstract int ejecutarMEM();

public abstract int ejecutarWB();

public abstract Registro regModificable();

public abstract Registro getRegRS();

public abstract Registro getRegRT();

public boolean estaLibreRegistro() {
         return true;
```

```
    }
}
```

4. SALTOCONDICIONAL.JAVA

```
public abstract class SaltoCondicional extends TipoI {
 private String
imagSegmentado[]={"beqIF.gif","beqID.gif","beqEX.gif","beqMEM.gif","beqWB.gif"}
;
 private String
imagSegmentado2Huecos[]={"beqIF.gif","beqID2Huecos.gif","beqEX2Huecos.gif","beq
MEM.gif","beqWB.gif"};
 private String
imagSegmentado3Huecos[]={"beqIF.gif","beqID2Huecos.gif","beqEX3Huecos.gif","beq
MEM3Huecos.gif","beqWB.gif"};

 private String imagSegmentadoSinAdelantamiento[]={"beqIF.gif","beqID sin
adelantamiento.gif","beqEX.gif","beqMEM sin adelantamiento.gif","beqWB sin
adelantamiento.gif"};
 private String
imagSegmentado2HuecosSinAdelantamiento[]={"beqIF.gif","beqID2Huecos.gif","beqEX
2Huecos.gif","beqMEM sin adelantamiento.gif","beqWB sin adelantamiento.gif"};
 private String
imagSegmentado3HuecosSinAdelantamiento[]={"beqIF.gif","beqID2Huecos.gif","beqEX
3Huecos.gif","beqMEM3Huecos sin adelantamiento.gif","beqWB sin
adelantamiento.gif"};

 int salto;
 int adelantamiento;

   public void inicia()
   {
       String imagenes="beq.gif";
       super.inicIma(imagenes);
       esSalto=true;
   }

   public static int numParam()
   {
       return 3;
   }


   public static String esValida(String[] param, Etiquetas etiq)
   {
       boolean esValido=false;
       String error=null;   /*0 indica que es valido*/
       Vector etiquetas=etiq.getNombres();
        if(param.length==numParam())
        {
            for(int i=0; i<numParam()-1;i++)
            {
                if(esRegistro(param[i]))
                    esValido=true;
```

```
                else
                    error=param[i]+"parametros no validos";
            }
            if(!esEtiqueta(param[2], etiq))
                error="parametros no validos";
        }
        else
            error="numero de parametros incorrecto"+ param.length;
        return error;
    }

    public abstract int ejecutar();
    public abstract String getCodificacion();

    public int numEtapas()
    {
        return 3;
    }

    public int etapa4()
    {
        return -1;
    }

    public int etapa5()
    {
        return -1;
    }

   public String imagenSegmentado(int etapa, int salto, int adelantamiento)
   {
     this.adelantamiento = adelantamiento;
      this.salto = salto;

      if(adelantamiento==CON_ADELANTAMIENTO){

           if ((salto == SALTO_FIJO)||(salto == SALTO_RETARDAD0))
                 return imagSegmentado[etapa];
           else if ((salto == SALTO_FIJO_2HUECOS)||(salto ==
SALTO_RETARDAD0_2HUECOS))
                 return imagSegmentado2Huecos[etapa];
           else
                 return imagSegmentado3Huecos[etapa];
      }

      else if (adelantamiento==SIN_ADELANTAMIENTO){

           if ((salto == SALTO_FIJO)||(salto == SALTO_RETARDAD0))
                 return imagSegmentadoSinAdelantamiento[etapa];
           else if ((salto == SALTO_FIJO_2HUECOS)||(salto ==
SALTO_RETARDAD0_2HUECOS))
                 return imagSegmentado2HuecosSinAdelantamiento[etapa];
           else
                 return imagSegmentado3HuecosSinAdelantamiento[etapa];
      }
      else return imagSegmentado[etapa];

   }
```

```
    public java.awt.Color colorSegm()
    {
        return new java.awt.Color(209,22,226);
    }

    public int estilo()
    {
        return 6;
    }

    public abstract int numero();

    public abstract boolean hayRiesgo();

    public abstract int ejecutarIF();

    public abstract int ejecutarID();

    public abstract int ejecutarIDHuecos();

    public abstract int ejecutarEX();

    public abstract int ejecutarMEM();

    public abstract int ejecutarWB();

    public abstract Registro regModificable();

    public abstract Registro getRegRS();

    public abstract Registro getRegRT();
     }
```

5. TIPOI.JAVA

```
public abstract class TipoI extends Instruccion {

    protected static int desp;


    public static int numParam()
    {
        return 2;
    }

    public static boolean esDireccion(String direccion)
    {
        String aux;

      Character temp=new Character(direccion.charAt(0));
      Character neg=new Character('-');

        if(Character.isDigit(direccion.charAt(0)) || true)
        {
            if(direccion.endsWith(")"))
            {
```

```
            StringTokenizer st=new StringTokenizer(direccion,"(");
            aux=st.nextToken();
            if(st.hasMoreTokens())
            {
             try{
                 desp=Integer.parseInt(aux);
              }catch(NumberFormatException e){return false;}
            aux=st.nextToken();
            }
            st=new StringTokenizer(aux, ")");
            aux=st.nextToken();
        }
        else
            return false;

    }
    else
    {
        if((direccion.startsWith("(")) && (direccion.endsWith(")")))
            aux=direccion.substring(1,direccion.length()-1);
        else
            return false;
    }

    if (!esRegistro(aux))
        return false;

    return true;
}

public int getDesplazamiento()
{
    return desp;
}

public String obtReg(String param)
{
    if(!param.startsWith("$"))
    {
        int c1=param.indexOf("(");
        int c2=param.indexOf(")");
        param=param.substring(c1+1,c2);
    }
    return param;
}

public String setIn(int cdIn)
{
    String cod=Integer.toBinaryString(cdIn);
    if(cod.length()>16)
    {
        cod=cod.substring(cod.length()-16,cod.length());
    }
    else
    {
        while(cod.length()<16)
            cod="0"+cod;
    }
    return cod;
```

```
    }
    public abstract int ejecutar();

    public abstract int numEtapas();

    public int etapa3()
    {
        imagen_etapa=new StringBuffer();
        imagen_etapa.append(direccion_multi).append("etapa3_acc_mem");
        return -1;
    }

    public abstract int numero();

    public abstract int etapa4();

    public abstract int etapa5();

    public abstract String imagenSegmentado(int etapa, int salto, int
adelantamiento);

    public abstract int estilo();

    public abstract boolean hayRiesgo();

    public abstract int ejecutarIF();

    public abstract int ejecutarID();

    public abstract int ejecutarEX();

    public abstract int ejecutarMEM();

    public abstract int ejecutarWB();

    public abstract Registro regModificable();

    public abstract Registro getRegRS();

    public abstract Registro getRegRT();

    public boolean estaLibreRegistro() {

       return true;
       }

}
```

6. EX.JAVA

```
public class EX extends Etapa {

       private Vector<UFuncional> unidFuncionales;
       private int ciclo;
       private static EX ex = null;
```

```
	private EstadoInstruccion estadoInstruccion;

	private EX() {
		unidFuncionales = new Vector<UFuncional>();
		this.ciclo = -1;
		estadoInstruccion = new EstadoInstruccion(new Nop());
	}

	public static EX getInstancia()
    {
	if(ex == null){
		ex = new EX();
	}
	  return ex;
    }

	public void inicializar() {
		//inicializarUF();
		unidFuncionales = new Vector<UFuncional>();

	}
	//Inicializar para salto 3 Huecos
	public void inicializarUF() {
	for(UFuncional uf:this.unidFuncionales) {
		uf.inicializar();
	}
    }

	public void setUnidFuncionales(Vector<UFuncional> unidFuncionales) {
		this.unidFuncionales = unidFuncionales;
	}

	public Vector<UFuncional> getUnidFuncionales() {
		return unidFuncionales;
	}

	public int getNumeroUF() {
		return unidFuncionales.size();
	}


	public void moverCiclo(MEM mem) {
		for(UFuncional uf:unidFuncionales) {
			if((uf.contieneInstEnFin(mem.getEstadoInstrucciones()[0]))
|| (uf.contieneInstEnFin(mem.getEstadoInstrucciones()[1]))
				|| (uf.contieneInstEnFin(new
EstadoInstruccion(new Nop())))) {
				uf.avanzarCiclo();
			}
		}
	}

	public void recordarCiclo(int ciclo) {
		this.ciclo = ciclo;
	}

	//metodo que devuelve la instruccion entera que pasa a MEM
	public EstadoInstruccion getEstadoInstEntera() {
		return unidFuncionales.elementAt(0).getInstFin();
```

```
	}

	//metodo que devuelve la instruccion flotante que pasa a MEM
	public EstadoInstruccion getEstadoInstFlotante() {
		EstadoInstruccion inst = new EstadoInstruccion(new Nop());
		EstadoInstruccion instAux = new EstadoInstruccion(new Nop());
		int latencia = 0;
		boolean avanza = true;


		for(UFuncional uf:unidFuncionales) {

			if((uf.getTipo() != Tipos.INTEGER_FU) &&
(!uf.getInstFin().getInstruccion().toString().equals("nop"))
					&& (uf.getLatencia()>latencia)) {
				instAux = uf.getInstFin();
				avanza = true;
				if(instAux.getInstruccion().regModificable()!=null) {
				for(UFuncional uf2:unidFuncionales) {
					for(EstadoInstruccion
ei:uf2.getInstrucciones()) {

						if((ei.getInstruccion().regModificable()
== instAux.getInstruccion().regModificable()) &&
								(ei.getCiclo() <
instAux.getCiclo())) {
							avanza = false;

						}
						if(!avanza) {
							break;
						}
					}
				}
				}
				if(avanza) {
					inst = instAux;

					latencia = uf.getLatencia();
				}
			}
		}

		return inst;
	}

	@Override
	public int avanzar(Etapa etapa) {
		EstadoInstruccion inst = ((ID)etapa).getEstadoInstruccion();
		for(UFuncional uf:unidFuncionales) {

			if((inst.getInstruccion().getTipoFU() == uf.getTipo()) &&
(uf.sePuedeInsertar())) {
				uf.anhadirInstruccion(inst, ciclo);
				return -1;
			}
		}
		return 0;
	}
```

```
	@Override
	public int ejecutar() {
		int aux = -1;
		for(UFuncional uf:unidFuncionales) {
			for(EstadoInstruccion es:uf.getInstrucciones()){
				if(es.getInstruccion().estilo()!=8)
					uf.aumetarCiclosUso();
			}
			aux = uf.ejecutar();
			if(aux != -1)
				return aux;
		}
		return aux;
	}


	public void actualizarEtapas() {
		EstadoInstruccion estInst[];
		String ultEtapa;
		int j;
		for(UFuncional uf:unidFuncionales) {
			estInst = uf.getInstrucciones();
			for(int i=0; i<estInst.length; i++) {

				if(estInst[i].getEtapas().size()>0) {
					for(j=0; j<estInst[i].getEtapas().size(); j++){
						// poner EX en entero y EX1 en flotante
para adelantamiento
						if((uf.getTipo() != Tipos.INTEGER_FU)){
							if(estInst[i].getEtapa(j).equals("
EX"+new Integer(i+1).toString())) {
								break;
						}
						}
						if((uf.getTipo() == Tipos.INTEGER_FU)){
							if(estInst[i].getEtapa(j).equals("
EX")) {
								break;
						}
						}

					}
					if(j ==       estInst[i].getEtapas().size()) {
						// poner EX en entero y EX1 en flotante
para adelantamiento
						if((uf.getTipo() != Tipos.INTEGER_FU)){
						estInst[i].anhadirEtapa(" EX"+new
Integer(i+1).toString());
							}
						else
							estInst[i].anhadirEtapa(" EX");
					}
					else {
						System.out.println("burbuja ex");
						estInst[i].anhadirEtapa(" burb");
					}
				}
			}
```

```
            }
        }
      }
```

7. TIPOR.JAVA

```
public abstract class TipoR extends Instruccion{
protected String
imagSegmentado[]={"R_IF.gif","R_ID.gif","R_EX.gif","R_MEM.gif","R_WB.gif"};
protected String imagSegmentado2[]={"R_IF.gif","R_ID sin
adelantamiento.gif","R_EX.gif","R_MEM.gif","R_WB.gif"};
protected String imagSegmentadoSinAdelantamiento[]={"R_IF.gif","R_ID sin
adelantamiento.gif","R_EX sin adelantamiento.gif","R_MEM sin
adelantamiento.gif","R_WB sin adelantamiento.gif"};

   public void inicia()
   {
       String imagenes="R.gif";
       super.inicIma(imagenes);
   }

    public static int numParam()
    {
        return 3;
    }

    public int numEtapas()
    {
        return 4;
    }

    public static String esValida(String[] param, int numParam)
    {
        String error=null;

        if(param.length==numParam)
        {
            for(int i=0; i<numParam;i++)
            {
                if(esRegistro(param[i]))
                {
                        if(param[i].equals("$zero")  || param[i].equals("$0"))
                        {
                            if(i==0)
                                error+=param[i]+"parametros no validos. Reg
esp";
                        }
                }
                else
                    error=param[i]+"parametros no validos. No es Reg";
            }
        }
        else
             error="parametros no validos";

        return error;
```

```
}

public static String esValidaF(String[] param, int numParam)
{
    String error=null;

    if(param.length==numParam)
    {
        for(int i=0; i<numParam;i++)
        {
            if(!esRegistroF(param[i]))
                error=param[i]+"parametros no validos. No es Reg";
        }
    }
    else
         error="parametros no validos";

    return error;
}

public static String esValidaFD(String[] param, int numParam)
{
    String error=null;

    if(param.length==numParam)
    {
        for(int i=0; i<numParam;i++)
        {
            if(!esRegistroFPar(param[i]))
                error=param[i]+"parametros no validos. No es Reg";
        }
    }
    else
         error="parametros no validos";

    return error;
}

public static String esValidaFR(String[] param, int numParam)
{
    String error=null;

    if((param.length==numParam)&&(numParam==2))
    {
            if(!esRegistroF(param[0]))
                error=param[0]+"parametros no validos. No es Reg";
            if(!esRegistro(param[1]))
                error=param[1]+"parametros no validos. No es Reg";
    }
    else
         error="parametros no validos";

    return error;
}



public static String esValidaRF(String[] param, int numParam)
{
```

```
    String error=null;

    if((param.length==numParam)&&(numParam==2))
    {
            if(!esRegistro(param[0]))
                error=param[0]+"parametros no validos. No es Reg";
            if(!esRegistroF(param[1]))
                error=param[1]+"parametros no validos. No es Reg";
    }
    else
         error="parametros no validos";

    return error;
}

public String setFunc(int cdFu)
{
    String cod=Integer.toBinaryString(cdFu);
    while(cod.length()<6)
        cod="0"+cod;
    return cod;
}

  public abstract int ejecutar();

  public int numInst()
  {
      return 4;
  }

  public int etapa3()
  {
      imagen_etapa=new StringBuffer();
      imagen_etapa.append(direccion_multi).append("R_etapa3");
      return -1;
  }

  public abstract int etapa4();
  public abstract Registro regModificable();
  public int etapa5()
  {
      return -1;
  }

public String imagenSegmentado(int etapa, int salto, int adelantamiento)
{
   if(salto==SALTO_FIJO ||salto==SALTO_RETARDAD0){
          if(adelantamiento==CON_ADELANTAMIENTO)
                return imagSegmentado[etapa];
          else
                return imagSegmentadoSinAdelantamiento[etapa];
   }
   else{
          if(adelantamiento==CON_ADELANTAMIENTO)
                return imagSegmentado2[etapa];
          else
                return imagSegmentadoSinAdelantamiento[etapa];
   }
}
```

```
    public java.awt.Color colorSegm()
    {
        return new java.awt.Color(243,157,151);
    }

    public int estilo()
    {
        return 2;
    }

    public String setShamt(int cdIn)
    {
        String cod=Integer.toBinaryString(cdIn);
        if(cod.length()>5)
        {
            cod=cod.substring(cod.length()-5,cod.length());
        }
        else
        {
            while(cod.length()<5)
                cod="0"+cod;
        }
        return cod;
    }
    public abstract int numero();

    public abstract boolean hayRiesgo();

    public abstract int ejecutarIF();

    public abstract int ejecutarID();

    public abstract int ejecutarEX();

    public abstract int ejecutarMEM();

    public abstract int ejecutarWB();

    public abstract Registro getRegRS();

    public abstract Registro getRegRT();
    public boolean estaLibreRegistro() {

      return true;
      }
     }
```

8. PANELCAMINODATOS.JAVA

```
/**PanelImagen.java
 *Clase que se encarga de la visualizacion de la representacion monociclo del
camino de datos segmentado
 **/
public class PanelCaminoDatos extends JPanel implements Tipos
{
```

```
    private Image imSegmentado,imagenIF,imagenID,imagenEX,imagenMEM,imagenWB,

imagenRiesgo,imagenAntMEM_RS,imagenAntMEM_RT,imagenAntWB_RS,imagenAntWB_RT,imag
enFlush, imagenAntEX_RS, imagenAntEX_RT,imagenAntWB;

    PipelineFlotante pipeline;
    private UnidadAnticipacion unidadAnticipacion;
    private UnidadDeteccionRiesgos unidadDeteccionRiesgos;
    private boolean deteccion;
    int salto;
    int adelantamiento;

/**Constructor de PanelImagen
 *@param Sin parametros
 **/
    public PanelCaminoDatos(int salto, int adelantamiento)
    {
        try
        {
             deteccion = false;
             this.salto = salto;
             this.adelantamiento = adelantamiento;
             if(this.adelantamiento == CON_ADELANTAMIENTO){
                   if((this.salto==SALTO_FIJO)||(this.salto==SALTO_RETARDAD0))

      imSegmentado=(Image)ImageIO.read(getClass().getResource(Lenguaje.getInsta
ncia().getString("rutaSegmentado")+"segmentado.gif"));
                   else
if((this.salto==SALTO_FIJO_2HUECOS)||(this.salto==SALTO_RETARDAD0_2HUECOS))

      imSegmentado=(Image)ImageIO.read(getClass().getResource(Lenguaje.getInsta
ncia().getString("rutaSegmentado")+"segmentado 2 Huecos.gif"));
                   else
if((this.salto==SALTO_FIJO_3HUECOS)||(this.salto==SALTO_RETARDAD0_3HUECOS))

      imSegmentado=(Image)ImageIO.read(getClass().getResource(Lenguaje.getInsta
ncia().getString("rutaSegmentado")+"segmentado 3 Huecos.gif"));
             }

             else if(this.adelantamiento == SIN_ADELANTAMIENTO){
                   if((this.salto==SALTO_FIJO)||(this.salto==SALTO_RETARDAD0))

      imSegmentado=(Image)ImageIO.read(getClass().getResource(Lenguaje.getInsta
ncia().getString("rutaSegmentado")+"segmentado sin adelantamiento.gif"));
             else
if((this.salto==SALTO_FIJO_2HUECOS)||(this.salto==SALTO_RETARDAD0_2HUECOS))

      imSegmentado=(Image)ImageIO.read(getClass().getResource(Lenguaje.getInsta
ncia().getString("rutaSegmentado")+"segmentado 2 Huecos sin
adelantamiento.gif"));
             else
if((this.salto==SALTO_FIJO_3HUECOS)||(this.salto==SALTO_RETARDAD0_3HUECOS))

      imSegmentado=(Image)ImageIO.read(getClass().getResource(Lenguaje.getInsta
ncia().getString("rutaSegmentado")+"segmentado 3 Huecos sin
adelantamiento.gif"));

             }
```

```
            pipeline = PipelineFlotante.getInstancia();
            unidadAnticipacion = UnidadAnticipacion.getInstancia();
            unidadDeteccionRiesgos = UnidadDeteccionRiesgos.getInstancia();

        }
        catch(IOException e) {}
    }


    public void actualizarImag()
    {
        try
        {
            imagenIF=(Image)ImageIO.read(getClass().getResource(

      "/ensamblador/segmentado/"+IF.getInstancia().getEstadoInstruccion().getIn
struccion().imagenSegmentado(0, salto, adelantamiento).toLowerCase()));
            imagenID=(Image)ImageIO.read(getClass().getResource(

      "/ensamblador/segmentado/"+ID.getInstancia().getEstadoInstruccion().getIn
struccion().imagenSegmentado(1,salto, adelantamiento).toLowerCase()));
            imagenEX=(Image)ImageIO.read(getClass().getResource(

      "/ensamblador/segmentado/"+EX.getInstancia().getEstadoInstEntera().getIns
truccion().imagenSegmentado(2, salto, adelantamiento).toLowerCase()));
            imagenMEM=(Image)ImageIO.read(getClass().getResource(

      "/ensamblador/segmentado/"+MEM.getInstancia().getEstadoInstrucciones()[0]
.getInstruccion().imagenSegmentado(3,salto, adelantamiento).toLowerCase()));
            imagenWB=(Image)ImageIO.read(getClass().getResource(

      "/ensamblador/segmentado/"+WB.getInstancia().getEstadoInstrucciones()[0].
getInstruccion().imagenSegmentado(4,salto, adelantamiento).toLowerCase()));


            if (unidadDeteccionRiesgos.detener(adelantamiento, salto))
            //if (deteccion)

imagenRiesgo=(Image)ImageIO.read(getClass().getResource("/ensamblador/segmentad
o/riesgo.gif"));
            else

      if(ID.getInstancia().getEstadoInstruccion().getInstruccion().toString().e
quals("nop"))

      imagenRiesgo=(Image)ImageIO.read(getClass().getResource("/ensamblador/seg
mentado/nop.gif"));
             else

      imagenRiesgo=(Image)ImageIO.read(getClass().getResource("/ensamblador/seg
mentado/no_riesgo.gif"));

            boolean ant[] = unidadAnticipacion.anticipar(salto);


            if (ant[ANT_RS_MEM])

imagenAntMEM_RS=(Image)ImageIO.read(getClass().getResource(("/ensamblador/segme
ntado/antMEM_RS.gif").toLowerCase()));
```

```
            else

imagenAntMEM_RS=(Image)ImageIO.read(getClass().getResource("/ensamblador/segmen
tado/nop.gif"));
            if (ant[ANT_RT_MEM])

imagenAntMEM_RT=(Image)ImageIO.read(getClass().getResource(("/ensamblador/segme
ntado/antMEM_RT.gif").toLowerCase()));
            else

imagenAntMEM_RT=(Image)ImageIO.read(getClass().getResource("/ensamblador/segmen
tado/nop.gif"));
            if (ant[ANT_RS_WB])

imagenAntWB_RS=(Image)ImageIO.read(getClass().getResource(("/ensamblador/segmen
tado/antWB_RS.gif").toLowerCase()));
            else

imagenAntWB_RS=(Image)ImageIO.read(getClass().getResource("/ensamblador/segment
ado/nop.gif"));
            if (ant[ANT_RT_WB])

imagenAntWB_RT=(Image)ImageIO.read(getClass().getResource(("/ensamblador/segmen
tado/antWB_RT.gif").toLowerCase()));
            else

imagenAntWB_RT=(Image)ImageIO.read(getClass().getResource("/ensamblador/segment
ado/nop.gif"));

            if(pipeline.activarIFFlush())

imagenFlush=(Image)ImageIO.read(getClass().getResource(("/ensamblador/segmentad
o/IF_Flush.gif").toLowerCase()));
            else

imagenFlush=(Image)ImageIO.read(getClass().getResource("/ensamblador/segmentado
/nop.gif"));

            if ((ant[ANT_RS_EX]) && (adelantamiento == CON_ADELANTAMIENTO))

imagenAntEX_RS=(Image)ImageIO.read(getClass().getResource(("/ensamblador/segmen
tado/antMEM_RS SALTO 1 HUECO.gif").toLowerCase()));
            else

imagenAntEX_RS=(Image)ImageIO.read(getClass().getResource("/ensamblador/segment
ado/nop.gif"));

            if ((ant[ANT_RT_EX]) && (adelantamiento == CON_ADELANTAMIENTO))

imagenAntEX_RT=(Image)ImageIO.read(getClass().getResource(("/ensamblador/segmen
tado/antMEM_RT SALTO 1 HUECO.gif").toLowerCase()));
            else

imagenAntEX_RT=(Image)ImageIO.read(getClass().getResource("/ensamblador/segment
ado/nop.gif"));

            if (ant[ANT_WB])
```

```
imagenAntWB=(Image)ImageIO.read(getClass().getResource(("/ensamblador/segmentad
o/antWB.gif").toLowerCase()));
            else

imagenAntWB=(Image)ImageIO.read(getClass().getResource("/ensamblador/segmentado
/nop.gif"));

        }
        catch(IOException e) {
             System.out.println("Error al crear las imagenes");
        }
      }


/**Metodo para pintar y dibujar en este panel
 *@param Graphics grafico
 *@return void
 **/
    public void paintComponents(Graphics g)
    {
        super.paintComponent(g);
        if (imSegmentado==null)
            return;
        actualizarImag();
        g.drawImage(imSegmentado,0,0,null);
        g.drawImage(imagenIF,0,0,null);
        g.drawImage(imagenID,0,0,null);
        g.drawImage(imagenEX,0,0,null);
        g.drawImage(imagenMEM,0,0,null);
        g.drawImage(imagenWB,0,0,null);
        g.drawImage(imagenRiesgo,0,0,null);
        g.drawImage(imagenAntMEM_RS,0,0, null);
        g.drawImage(imagenAntMEM_RT,0,0, null);
        g.drawImage(imagenAntWB_RS,0,0, null);
        g.drawImage(imagenAntWB_RT,0,0, null);
        g.drawImage(imagenFlush,0,0, null);
        g.drawImage(imagenAntEX_RS,0,0, null);
        g.drawImage(imagenAntEX_RT,0,0, null);
        g.drawImage(imagenAntWB,0,0, null);

      }


       /*public void deteccion(boolean detener){
             deteccion = detener;
       }*/
     }
```

9. UNIDADANTICIPACION.JAVA

```
public class UnidadAnticipacion implements Tipos {

    private static UnidadAnticipacion unidadAnticipacion = null;
    int salto;
```

```
    /** Creates a new instance of UnidadAnticipacion */
    private UnidadAnticipacion() {};

    public static UnidadAnticipacion getInstancia()
    {
      if(unidadAnticipacion == null){
            unidadAnticipacion = new UnidadAnticipacion();
      }
      return unidadAnticipacion;
    }

    public boolean[] anticipar(int salto) {
        boolean ant[] = new boolean[7];
        Instruccion instEX, instMEM, instWB, instID;
        Registro regMEM = null;
        Registro regEX = null;
        Registro regWB = null;
        this.salto = salto;
        for (int i=0; i<ant.length; i++) {
            ant[i] = false;
        }
        instEX = EX.getInstancia().getEstadoInstEntera().getInstruccion();
        instMEM =
MEM.getInstancia().getEstadoInstrucciones()[0].getInstruccion();
        instWB =
WB.getInstancia().getEstadoInstrucciones()[0].getInstruccion();
        instID = ID.getInstancia().getEstadoInstruccion().getInstruccion();


        if (instMEM.regModificable() != null) {
            regMEM = instMEM.regModificable();

            //La segunda parte es para quitar los almacenamientos
            if ((regMEM == instEX.getRegRS()) &&((instEX instanceof Sw) ==
false)&&((instMEM instanceof Lw) == false)){
                ant[ANT_RS_MEM] = true;
            }
            if ((regMEM == instEX.getRegRT()&&((instEX instanceof Sw) ==
false)&&((instMEM instanceof Lw) == false))) {
                ant[ANT_RT_MEM] = true;
            }

        }

        if (instWB.regModificable() != null) {
            regWB = instWB.regModificable();

            // No hacer para saltos
            if (((regMEM != instEX.getRegRS()) && (regWB ==
instEX.getRegRS())&& (((instEX instanceof Beq) == false)&&((instEX instanceof
Bne) == false)))) {
                ant[ANT_RS_WB] = true;
            }
            if (((regMEM != instEX.getRegRT()) && (regWB ==
instEX.getRegRT())&& (((instEX instanceof Beq) == false)&&((instEX instanceof
Bne) == false)))) {
                ant[ANT_RT_WB] = true;
            }
```

```
            //Adelantar de wb a mem para almacenamientos
            if (((regWB == instMEM.getRegRS()) && (instMEM instanceof
Sw))||((regWB == instMEM.getRegRT()) && (instMEM instanceof Sw))) {
                ant[ANT_WB] = true;
            }
            if (((regWB == instEX.getRegRS()) && (instWB instanceof Lw))){
             ant[ANT_RS_WB] = true;
            }
            if (((regWB == instEX.getRegRT()) && (instMEM instanceof Sw))){
             ant[ANT_RT_WB] = true;
            }
        }
        if ((instMEM.regModificable() != null) && (this.salto ==SALTO_RETARDAD0
||this.salto ==SALTO_FIJO)) {
            regMEM = instMEM.regModificable();
            regEX = instEX.regModificable();
           //Adelantamientos en saltos
            if((regEX!=instID.getRegRS() && !(instEX instanceof Inmediatas))){
             if (regMEM == instID.getRegRS()&&((instID instanceof Beq)||(instID
instanceof Bne))&&((instMEM instanceof Lw) == false)) {
                    ant[ANT_RS_EX] = true;
             }
            }
            if (regMEM == instID.getRegRT()&&((instID instanceof Beq)||(instID
instanceof Bne))&&((instMEM instanceof Lw) == false)) {
               ant[ANT_RT_EX] = true;
            }



        }
        return ant;
    }

     }
```