UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INFORMÁTICA

TRABAJO FIN DE GRADO

Implementación de diversas estrategias de tratamiento de los saltos en el simulador Simula3MS

Marta María Lorenzo Linuesa

Febrero, 2015

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

Departamento de Sistemas Informáticos

TRABAJO FIN DE GRADO

Implementación de diversas estrategias de tratamiento de los saltos en el simulador Simula3MS

Autor: Marta María Lorenzo Linuesa

Director: Francisco José Alfaro Cortés
Pedro Javier García García

Febrero, 2015

Resumen

Simula3MS es un simulador basado en la arquitectura MIPS en sus versiones monociclo, multiciclo y segmentado. Este simulador está destinado para uso de los alumnos en los laboratorios de prácticas de asignaturas que tratan la arquitectura y organización del computador. Simula3MS permite observar y aplicar los conocimientos adquiridos en clases teóricas, pudiendo detectar errores y ofreciendo más posibilidades que un entorno real, no modificando elementos físicos del computador. Este simulador cuenta con un entorno de trabajo gráfico y de fácil uso que permite depurar programas y observar la evolución de la memoria, así como la ejecución de las instrucciones sobre distintos caminos de datos. La herramienta ofrece distintas implementaciones del cauce, lo que permite observar las diferencias existentes a la hora de ejecutar un mismo código según las características del procesador.

En el trascurso de esta memoria se irán describiendo los cambios realizados en el simulador para incrementar de dos a seis las posibilidades de resolución del salto que ofrece el simulador. La versión utilizada hasta ahora recoge la posibilidad de utilizar las técnicas "Salto retardado" y "Predecir no tomado" para un hueco de retardo, pasando ahora a ofrecerlas mismas técnicas, pero para las implementaciones de dos y tres huecos de retardo. Este documento comienza explicando los objetivos y la metodología de desarrollo a seguir para este TFG. A continuación se ofrece un repaso de la arquitectura MIPS, viendo su repertorio de instrucciones y diferentes tipos de cauce. Seguidamente se explicará en detalle el proceso de desarrollo del TFG, concretamente la implementación de los cambios necesarios en el simulador. Finalmente se describen las pruebas de sistema realizadas y las conclusiones y líneas de trabajo futuro que han quedado abiertas.

Agradecimientos

Este trabajo ha sido realizado gracias al apoyo de las personas que han creído y confiado en mí y el esfuerzo por lograr los objetivos que nos planteamos.

Agradecer a mis tutores Francisco José Alfaro y Pedro Javier García por la paciencia y confianza que depositaron en mí para poder acabar este trabajo.

Agradecer sobre todo a mi familia y amigos, por estar ahí y no perder nunca la fe en mí. En especial a mis padres por el esfuerzo realizado a lo largo de sus vidas para ofrecerme todas las posibilidades de las que he disfrutado y sigo disfrutando.

Marta.

ÍNDICE

ÍNDICE DE FIGURAS

CAPÍTULO 1. INTRODUCCIÓN

En este capítulo se describe la motivación para el desarrollo de este trabajo, así como los objetivos que se han propuesto alcanzar en el mismo. Se incluye también, para finalizar el capítulo, una descripción de cómo se ha estructurado esta memoria.

1.1 CONTEXTO

Los simuladores de procesadores son ampliamente utilizados en la docencia de asignaturas del área de Arquitectura y Tecnología de Computadores [6] [7] debido a que ofrecen un entorno más sencillo de manipular que una máquina real, además de ofrecer otras ventajas: pueden detectar errores, ofrecen más posibilidades que un ordenador real y no modifican elementos físicos del computador.

Simula3MS [9] es un simulador resultado de un proyecto del grupo de Arquitectura de Computadores de la Universidad de A Coruña. El proyecto abarca la implementación de un simulador de una arquitectura de procesador básica (concretamente una adaptación de MIPS [8]) en sus versiones monociclo, multiciclo y segmentado. Actualmente cuenta con tres opciones de simulación diferentes: entrada/salida, técnicas de salto y camino de datos. Esta última opción permite escoger entre diferentes configuraciones del camino de datos: monociclo, multiciclo, segmentado básico, Marcador y algoritmo de Tomasulo.

Como ya se ha indicado Simula3MS está especialmente orientado al estudio de las asignaturas que tratan la arquitectura y organización del computador, ya que permite observar y aplicar los conocimientos adquiridos en las clases de teoría. Cuenta con 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 utilizarlos distintos

cauces en la misma herramienta permite observar las diferencias de ejecución de un mismo código según cuales sean las características del procesador.

1.2 MOTIVACIÓN

Como se ha comentado en el punto anterior, Simula3MSsupone un gran apoyo en la docencia, especialmente en la realización de prácticas. No obstante las prestaciones que ofrece son mejorables.

Analizando las funcionalidades que ofrece actualmente el simulador, observamos que en el tratamiento de los saltos se contempla una implementación con un único hueco de retardo (es decir, el salto se resuelve en la etapa ID del cauce segmentado) con las estrategias "Salto Retardado" y "Predecir no tomado". Sin embargo este simulador no contempla el tratamiento para dos y tres huecos de retardo con las mismas estrategias. Contar con estas otras cuatro opciones para los saltos (2 implementaciones combinadas con las 2 estrategias) sería de gran importancia para la docencia, pudiendo comprobar así los alumnos el modo en el que se tratarían los saltos con dos y tres huecos de retardo y ver las diferencias que se producen entre todos ellos.

1.3 OBJETIVOS

El principal objetivo para este TFG es, utilizando el código fuente de la última versión del simulador Simula3MS, ampliarlo para introducir cuatro nuevas estrategias de salto:

- “Salto retardado” con dos huecos de retardo.
- “Predecir no tomado” con dos huecos de retardo.
- “Salto retardado” con tres huecos de retardo.
- “Predecir no tomado” con tres huecos de retardo.

Para llegar a la consecución de este objetivo principal se plantean las siguientes tareas:

1. Estudio del diseño de la ruta de datos de un procesador y su segmentación. Es necesario conocer la teoría necesaria para entender el funcionamiento de un procesador segmentado.
2. Familiarizarse a nivel usuario con el funcionamiento de Simula3MS para así conocer las características del simulador y las prestaciones que ofrece.

3. Estudio del código fuente del simulador con la herramienta Eclipse para comprender el funcionamiento y organización de las clases y métodos que componen la herramienta.
4. Modificación del código existente y creación de código nuevo para llegar a la consecución del objetivo final de este TFG. Esta tarea es la que supone un mayor esfuerzo ya que se trata de un código muy complejo.
5. Pruebas de validación. Es necesario realizar pruebas de simulación para validar la versión final del simulador, resultante de la tarea anterior.

1.4 ESTRUCTURA DE LA MEMORIA

Para proporcionar al lector una visión global de la estructura de esta memoria del TFG, se expone a continuación un breve resumen de los capítulos de la misma.

- **Capítulo 1. Introducción:** En este capítulo se ofrece al lector una idea general de lo que se encontrará en el documento. Concretamente, se describen las motivaciones y objetivos principales del trabajo desarrollado en este TFG.
- **Capítulo 2. El procesador MIPS y su Modelado en Simula3MS:** En este capítulo se muestran aspectos básicos de la arquitectura MIPS y cómo se modelan dentro del simulador Simula3MS.
- **Capítulo 3. Diseño e implementación de los distintos tipos de salto:** En este capítulo se describen las modificaciones que se han ido realizando sobre el código original para llegar a la consecución del objetivo final.
- **Capítulo 4. Pruebas de sistema:** En este capítulo se muestran y analizan los resultados de las pruebas realizadas para comprobar el comportamiento del simulador Simula3MS después de implementar todas las modificaciones planteadas para este TFG.
- **Capítulo 5. Conclusiones y trabajo futuro:** En el último capítulo del documento se indican las conclusiones del proyecto y se plantean objetivos de cara a un desarrollo futuro.
- **Bibliografía:** En este apartado se recogen las referencias que se hacen en este TFG.

CAPÍTULO 2. EL PROCESADOR MIPS Y SU MODELADO EN SIMULA3MS

En este capítulo veremos un resumen de cómo funciona una arquitectura MIPS, describiendo los formatos de instrucciones que componen su repertorio de instrucciones y los posibles caminos de datos a utilizar: monociclo, multiciclo y segmentado.

A continuación, se describe el modelado de esta arquitectura en el simulador Simula3MS. Se hará un recorrido por las diferentes características y opciones que ofrece el simulador en su versión anterior a la desarrollada en este TFG.

2.1 ARQUITECTURA MIPS

La arquitectura MIPS, mostrada en la figura 2.1, posee 32 registros genéricos ($0-$31) de 32 bits para uso del procesador, siendo el registro $0 de solo lectura y con valor 0. De los restantes registros solo el $31 es implícitamente usado por una instrucción. Concretamente, por la instrucción de invocación de una subrutina (jal) [8] y se utiliza para guardar la dirección de retorno.

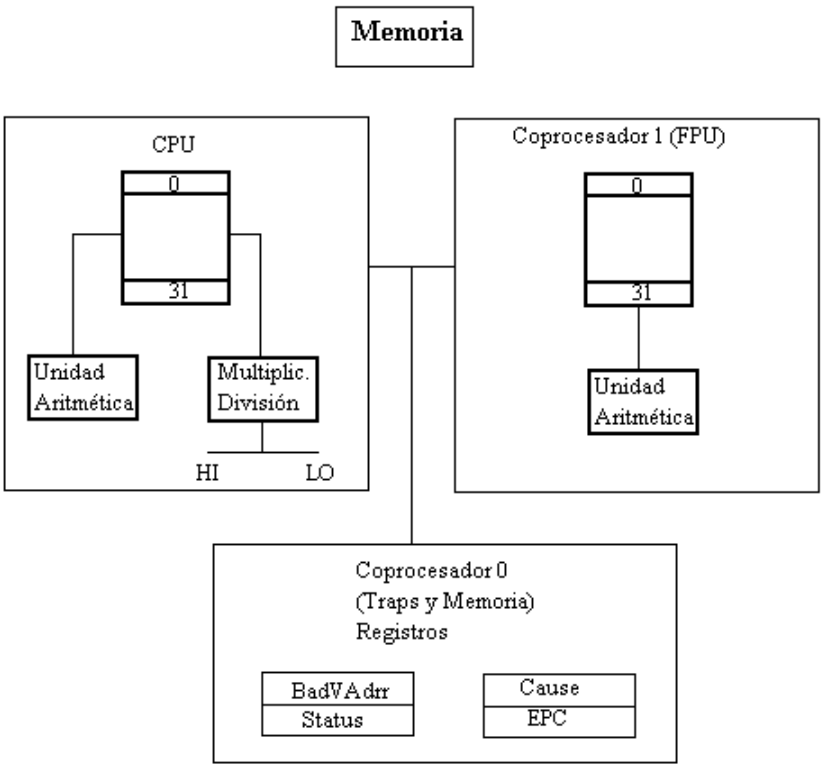

Figura 2. 1. Arquitectura MIPS.

Los procesadores MIPS no disponen de unidad de coma flotante incluida en el microprocesador. Estas funciones se implementan en coprocesadores separados. La arquitectura MIPS tiene en cada coprocesador 32 registros de 32 bits para coma flotante ($f0-$f31), que pueden ser organizados en 16 registros de doble precisión, con 64 bits.

En cuanto al direccionamiento de la memoria, éste se realiza por bytes. Esto quiere decir que las direcciones de memoria de 2 palabras consecutivas estarán separadas en 4 unidades (ya que las palabras son de 4 bytes). Cuando una palabra se carga desde memoria a un registro o se pasa a memoria desde un registro, la dirección de memoria involucrada ha de ser múltiplo de 4. Esto es lo que se denomina restricción de alineamiento.

Los tipos básicos de instrucciones que soporta MIPS son los siguientes:

- Transferencia de datos.
- Aritméticas y lógicas.
- De control de flujo.
- Salto condicional.
- Bifurcación.

Las características más importantes de estas instrucciones en el procesador MIPS son:

- La longitud de todas las instrucciones MIPS es de 32 bits.
- Los operandos de las instrucciones aritméticas son siempre registros. MIPS es, por tanto, una arquitectura de carga/almacenamiento (registro-registro).
- El acceso a memoria se hace a través de las operaciones de carga y almacenamiento (transferencia de datos).
- Para acceder a una palabra en memoria hay que indicar su dirección. MIPS direcciona bytes individuales. No obstante debe tenerse en cuenta que la mayor parte de las instrucciones que acceden a memoria lo hacen de forma alineada, por lo que la dirección a la que se accede debe ser múltiplo de 4.

Como hemos visto antes, todas las instrucciones del repertorio del MIPS tienen un tamaño fijo de 32 bits y están repartidas en tres formatos de instrucción diferentes, como podemos ver en la figura 2.2. Estos tres formatos serían los siguientes:

- Instrucciones entre registros (Tipo R): en este tipo se incluyen las instrucciones aritméticas y lógicas.
- Instrucciones con inmediato (Tipo I): dentro de este tipo de instrucciones están comprendidas las cargas y almacenamientos, todas las operaciones con inmediatos y las instrucciones de salto condicional.
- Instrucciones de salto (Tipo J): este tipo comprende las instrucciones de bifurcación.

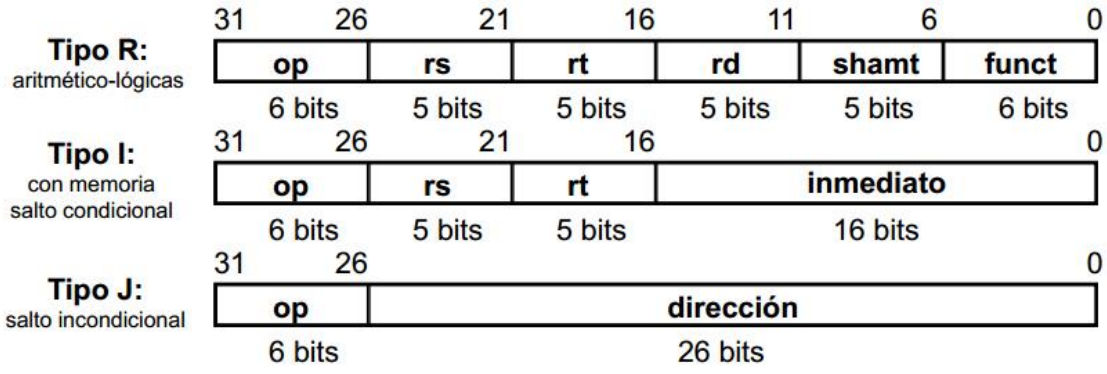

Figura 2. 2. Formatos de los diferentes tipos de instrucciones MIPS.

Como puede verse en la figura 2.2, a los campos de las instrucciones MIPS se les da unos nombres para identificarlos fácilmente. Se definen a continuación:

- **op:** identificador de instrucción, tradicionalmente llamado código de operación.
- **rs:** primer registro operando fuente.
- **rt:** segundo registro operando fuente.
- **rd:** registro operando destino.
- **shamt:** tamaño del desplazamiento.
- **funct:** selecciona la operación aritmética a realizar.
- **inmediato:** operando inmediato o desplazamiento en direccionamiento a registro base.
- **Dirección:** dirección destino del salto.

Aunque tener múltiples formatos complica la circuitería, se puede reducir la complejidad guardándolos de forma similar. Por ejemplo, los tres primeros campos de los formatos tipo R y tipo I son del mismo tamaño y tienen los mismos nombres. Los formatos se distinguen por el valor del primer campo: a cada formato se le asigna un conjunto de valores distintos en el primer campo y por lo tanto la circuitería sabe cómo ha de tratar la última mitad de la instrucción, bien como tres campos (tipo R) o como un campo simple (tipo I), o si la instrucción es tipo J.

2.2 CAMINO DE DATOS.

El procesador [4] es el elemento del computador que se encarga de ejecutar las instrucciones especificadas por el programa y coordina las actividades de las otras unidades del computador. Existen diferentes estrategias en el desarrollo de procesadores.

Con un procesador monociclo se ejecutan todas las instrucciones en un solo ciclo, siendo ésta la estrategia más sencilla. Esto significa que ningún elemento del camino de datos puede utilizarse más de una vez por instrucción, de forma que cualquier recurso que se necesite más de una vez deberá estar replicado. Otra estrategia parecida sería el procesador multiciclo. En este caso las instrucciones diferentes duran diferentes ciclos de reloj. La última estrategia sería un procesador segmentado, en el cual se permite solapar la ejecución de instrucciones.

Veremos a continuación una descripción del funcionamiento de cada uno de los casos anteriores.

2.2.1 PROCESADOR MONOCICLO.

Se trata de una implementación simple que utiliza sólo un ciclo de reloj para ejecutar cada instrucción. Este enfoque no es muy práctico debido a que todas las instrucciones tardarán el mismo intervalo de tiempo, lo que significa que las instrucciones más lentas determinan la duración del ciclo de reloj. El tiempo necesario para la ejecución puede variar sensiblemente de una instrucción a otra. Por ejemplo, una instrucción de salto incondicional necesita mucho menos tiempo que una carga.

El concentrar la totalidad de la ejecución en un ciclo de reloj influye en la circuitería necesaria. Esto significa que ningún elemento del camino de datos puede utilizarse más de una vez por instrucción, de forma que cualquier recurso que se necesite más de una vez deberá estar replicado. En la figura 2.3 puede verse el camino de datos de un procesador monociclo de la arquitectura MIPS.

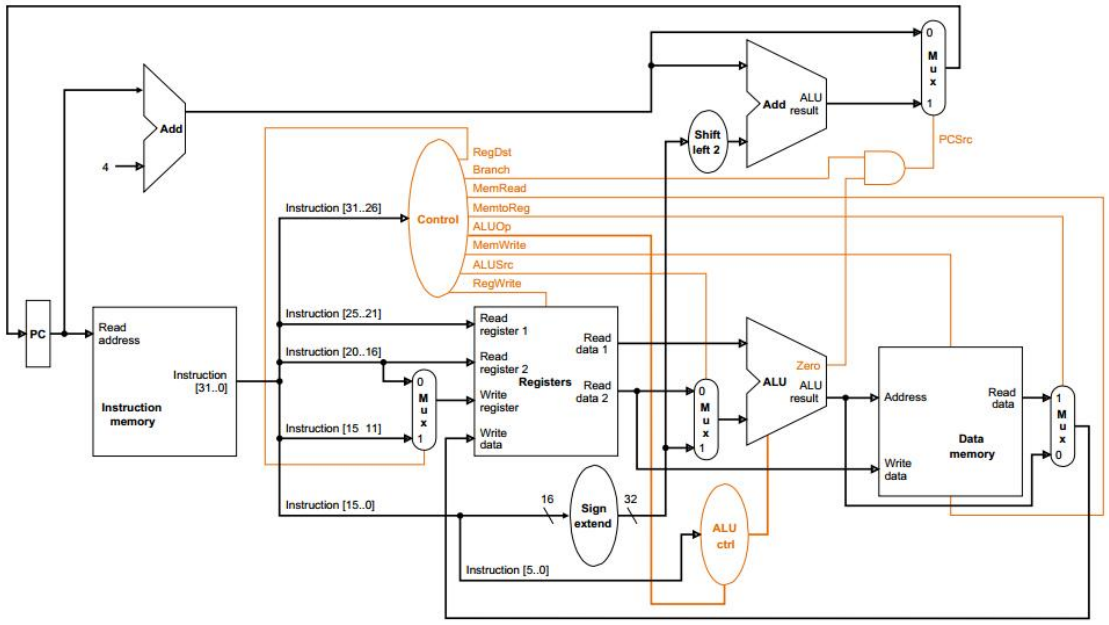

Figura 2. 3. Implementación monociclo de la arquitectura MIPS.

Este camino de datos se caracteriza, entre otras cosas, por tener una memoria de datos y otra de instrucciones, necesarias ambas porque al ejecutarse cada instrucción en un ciclo de reloj puede ser necesario acceder a ambas durante ese ciclo. Así, en la memoria de instrucciones se accede a aquella instrucción almacenada en la posición asociada al valor del contador de programa (PC), y en la memoria de datos se pueden guardar los datos almacenados en los registros relacionados con la instrucción a ejecutar, o cargar a registro el dato deseado para ser usado posteriormente por otras instrucciones.

Uno de los principales componentes de un camino de datos es la unidad de control, que a partir de los seis bits correspondientes al código de operación modifica los valores de las distintas señales de control del camino de datos. Hay algunos elementos como el banco de registros que necesitan una señal de escritura, o como la memoria de datos que requiere una señal de escritura y otra de lectura. Cada multiplexor necesita una línea de control y además, hay una unidad de control de la ALU que dependiendo de los valores de las líneas ALUOp generadas por la unidad de control principal, controla el funcionamiento de la misma. En las figuras 2.4 y 2.5 se pueden ver las distintas señales de control y su efecto.

Estas señales de control se activan en función de las seis líneas de entrada a la unidad de control. Estas líneas se corresponden con los 6 bits más significativos de la codificación de cada instrucción.

Señal	Acción cuando es desactivada(0)	Acción cuando se activa (1)
RegDst	El registro destino para las escrituras viene del campo rt (bits 20-16)	El registro destino para las escrituras viene del campo rd (bits 15-11)
RegWrite	Ninguno	Escribe el dato en "WriteData" en el registro dado por "WriteRegister".
AluSrc	El segundo operando de la ALU viene del banco de registro (salida 2)	El segundo operando de la ALU son los 16 bits menos significativos de la instrucción extendidos en signo
PCSrc	Selecciona PC+4 como nuevo valor del PC	Selecciona la dirección de salto computada como nuevo valor del PC
MemWrite	Ninguna	Escribe en la dirección de memoria "Address" el dato "WriteData"
MemRead	Ninguna	Lee un dato de la dirección de memoria "Address" y lo deja en la salida "ReadData"
MemToReg	El valor a realimentar al campo "WriteData" viene de la salida de la ALU	El valor a realimentar el campo "WriteData" viene de la memoria

Figura 2. 4. Señales de control de una línea para una realización MIPS monociclo.

2.2.2 PROCESADOR MULTICICLO.

En una implementación multiciclo, como la de la figura 2.5, cada instrucción se divide en un conjunto de pasos y cada paso se ejecuta en un ciclo de reloj. El hecho de que cada instrucción se divida en una serie de pasos, que se ejecutan en distintos ciclos de reloj, permite usar un mismo recurso hardware varias veces durante la ejecución de la instrucción, siempre y cuando se use en ciclos de reloj distintos. Evidentemente esto permite ahorrar hardware en la implementación del cauce frente a la versión monociclo estudiada antes. Además, cada instrucción durará sólo el número de ciclos que necesite, por lo que el procesador será mucho más eficiente.

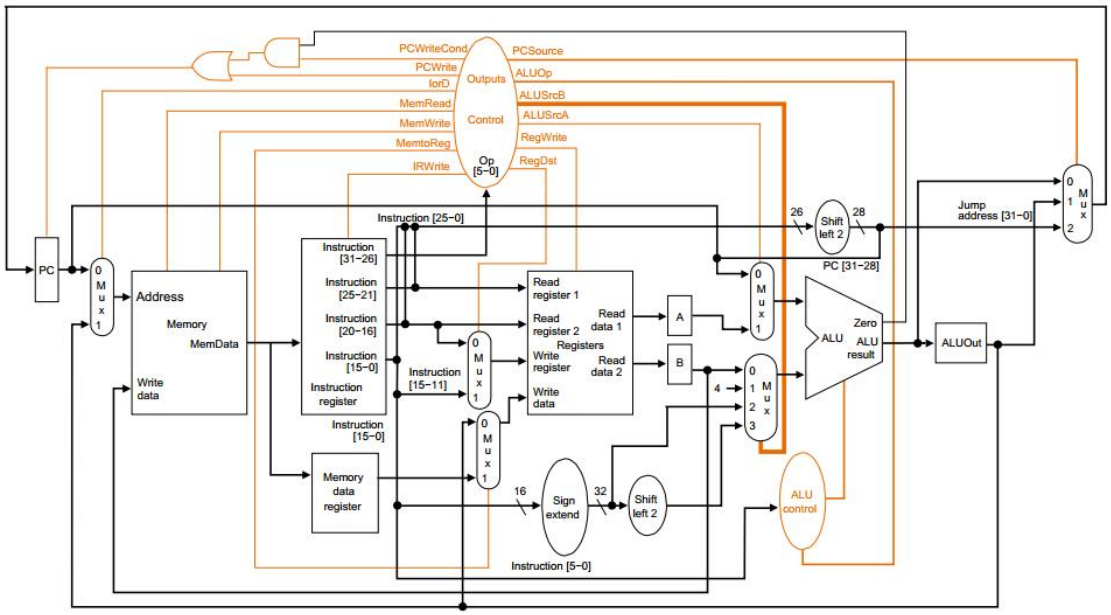

Figura 2. 5. Implementación multiciclo de la arquitectura MIPS.

Si comparamos esta implementación multiciclo (figura 2.4) con la anterior, monociclo (figura 2.3) se observan las siguientes diferencias:

- Sólo hay una memoria compartida para instrucciones y datos. Esto implica que la lectura de instrucciones y el acceso a datos han de realizarse en ciclos distintos (Arquitectura Von Newmann)
- Hay una sola ALU, en lugar de la ALU y los dos sumadores de la implantación monociclo.
- Es necesario añadir una serie de registros temporales entre unidades funcionales para almacenar la salida de éstas, de forma que dicha salida pueda usarse en el siguiente ciclo por la siguiente unidad funcional.

Como el camino de datos multiciclo tarda más de un ciclo en ejecutar una instrucción, se va a necesitar un conjunto de señales de control diferente al utilizado en la realización monociclo. Los elementos accesibles para el programador (el PC, la memoria y los registros), así como el registro IR necesitan señales de control de escritura. Cada uno de los multiplexores de dos vías requiere una única señal de control, mientras que los de cuatro necesitan dos líneas. La ALU sigue teniendo una unidad de control asociada cuyo funcionamiento depende del valor de las líneas ALUOp. En el camino de datos multiciclo hay catorce líneas de control cuyo nombre y efecto viene explicado en las figuras 2.5 y 2.6.

Señal	Acción cuando es desactivada(0)	Acción cuando se activa (1)
RegDst	El registro a escribirse en el archivo de registros se indica en el campo rt	El registro a escribirse en el archivo de registros se indica en el campo rd
RegWrite	Ninguno	Habilita la escritura en el archivo de registros
AluSrcA	El primer operando de la ALU es el PC	El primer operando de la ALU viene del registro A
MemRead	Ninguno	Habilita la lectura de la memoria
MemWrite	Ninguno	Habilita la escritura en la memoria
MemtoReg	El valor a escribirse en el archivo de registros viene del registro temporal ALUOut	El valor a escribirse en el archivo de registros viene del registro temporal MDR
IorD	El PC es usado para suministrar la dirección a la unidad de memoria	ALUOut es usado para suministrar la dirección a la unidad de memoria
IRWrite	Ninguno	La salida de la memoria es escrita en el registro IR
PCWrite	Ninguno	El PC es escrito; su valor lo determina PCSource
PCWriteCond	Ninguno	El PC es escrito si la salida zero de la ALU también está activa

Figura 2. 6. Señales de control de una línea para una realización multiciclo.

2.2.3 PROCESADOR SEGMENTADO.

La segmentación [4] es una técnica de implementación que consiste en solapar la ejecución de múltiples instrucciones. Por tanto, la segmentación incrementa el número de instrucciones que se están ejecutando a la vez y la rapidez con que las instrucciones empiezan y acaban. Conviene aclarar que la segmentación no reduce el tiempo que se tarda en completar una instrucción individual, también llamada latencia de la instrucción. Realmente, la segmentación mejora las prestaciones incrementando la productividad de las instrucciones, en lugar de disminuir el tiempo de ejecución de cada instrucción individual, pero la productividad de las instrucciones es la métrica importante, ya que los programas reales ejecutan miles de millones de instrucciones.

La segmentación explota el paralelismo existente entre instrucciones de un flujo secuencial, añadiendo la ventaja de ser totalmente invisible al programador. Actualmente la segmentación es clave para hacer que los procesadores sean rápidos. En condiciones ideales, el incremento de la velocidad con la segmentación iguala al número de etapas que se pueden solapar.

En el procesador MIPS usado para la docencia, el cauce de ejecución está segmentado en 5 etapas. Esto significa que durante un ciclo de reloj se podrían estar ejecutando simultáneamente hasta cinco instrucciones consecutivas. Estas etapas son:

1. **IF:** Búsqueda de instrucciones.
2. **ID:** Decodificación de instrucciones y lectura del banco de registros.
3. **EX:** Ejecución de la instrucción o cálculo de dirección.
4. **MEM:** Acceso a la memoria de datos.
5. **WB:** Escritura del resultado.

Estas etapas son ejecutadas por separado, realizándose la ejecución de las instrucciones tal como se muestra en la figura 2.7.

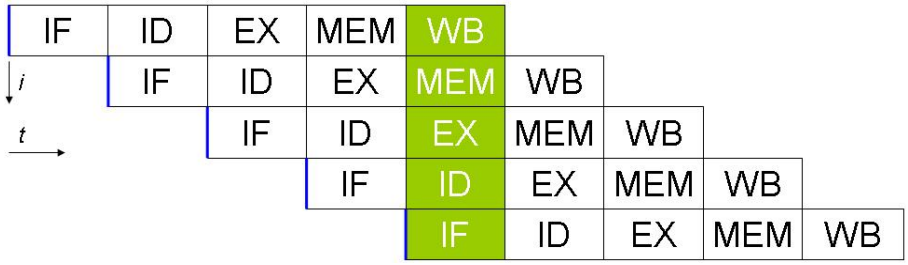

Figura 2. 7. Ejecución de instrucciones en una arquitectura MIPS segmentada.

En la figura 2.8 se muestra el esquema del camino de datos en una arquitectura MIPS segmentada.

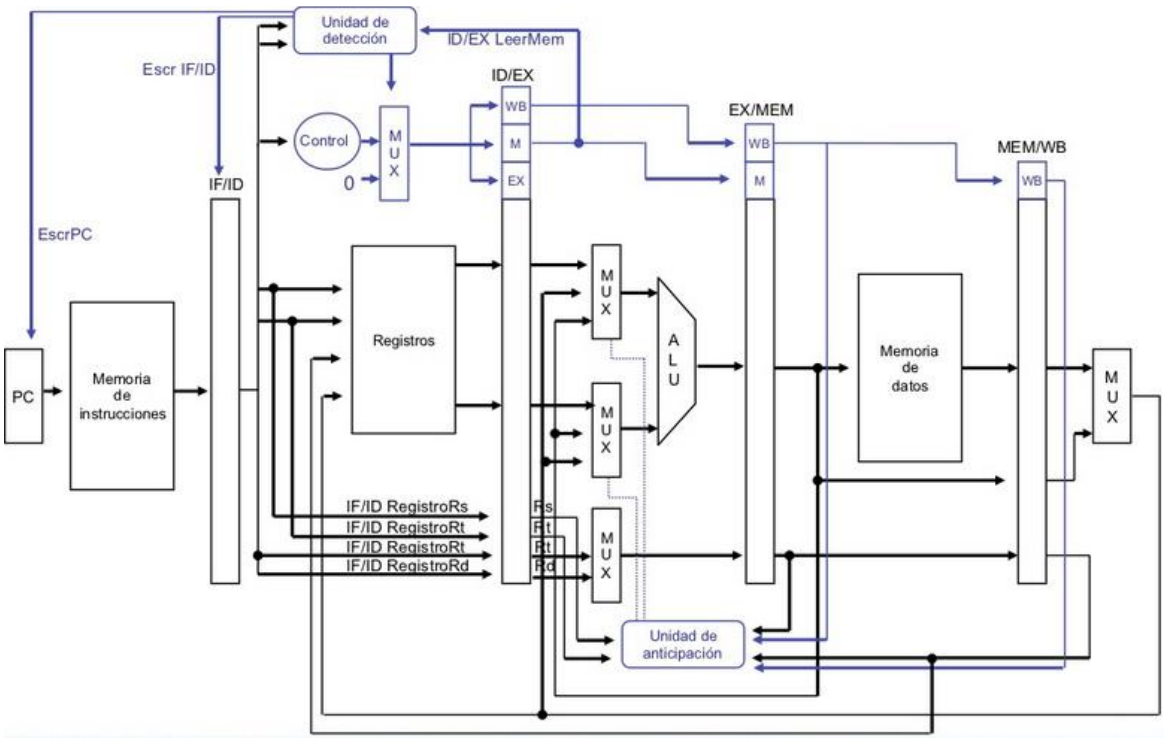

Figura 2. 8. Implementación segmentada de la arquitectura MIPS.

Las instrucciones de datos fluyen por regla general de izquierda a derecha a través de las cinco etapas hasta que completan la ejecución. Entre una etapa y la siguiente del camino de datos se sitúan registros de segmentación, para conservar el valor de los campos necesarios de una instrucción individual durante varias etapas. Dichos registros almacenan los valores que necesitará la instrucción en las etapas siguientes para continuar su ejecución. Los registros de segmentación también guardan información de las líneas de control, pues se necesita activar los valores de control en cada etapa de segmentación. Ya

que cada línea de control se asocia con un componente activo en una sola etapa, se pueden dividir las líneas de control en cinco grupos de acuerdo con las etapas de la segmentación.

Durante la ejecución de un código en un procesador segmentado se pueden producir riesgos, es decir, situaciones que impiden que se ejecute la siguiente instrucción del flujo de instrucciones durante su ciclo de reloj. Los riesgos reducen el rendimiento de la velocidad ideal lograda por la segmentación. Se distinguen tres tipos de riesgos:

- **Riesgos de datos:** Surgen cuando una instrucción depende de los resultados de una instrucción previa que aún está en el cauce. La solución a este tipo de riesgo consiste en añadir al procesador una unidad de detección de riesgos. La unidad de detección de riesgos puede usar distintas técnicas para evitar el mismo. Simula3MSusa bloqueo o burbuja, que consiste en detener por hardware las instrucciones, insertando burbujas, hasta que se resuelva el riesgo.
- **Riesgos estructurales:** Surgen de conflictos en los recursos, cuando el hardware no puede soportar las combinaciones posibles de instrucciones en ejecuciones solapadas simultáneamente. La solución adoptada en el MIPS consiste en la segmentación de unidades funcionales y duplicación de los recursos para permitir todas las posibles combinaciones de instrucciones. Sin embargo, esta solución sólo es necesaria en el caso de operaciones que precisen de etapas que ocupan más de un ciclo de reloj, como por ejemplo las operaciones en punto flotante.
- **Riesgos de control:** Un riesgo de control se produce cuando es necesario llevar a cabo una decisión basada en el resultado de una instrucción mientras se están ejecutando otras instrucciones. Surgen de la segmentación de los saltos y otras instrucciones que cambian el PC. En el camino de datos de Simula3MS la decisión del salto se toma en la etapa ID (se adelanta la comprobación del salto desde la etapa EX para ganar un ciclo de reloj).

2.2.4 IMPLEMENTACIÓN DEL SALTO Y ESTRATEGIAS APLICADAS.

Cuando se ejecuta un salto no se conoce de antemano cuál será la siguiente instrucción que deberá ser ejecutada. Si la condición del salto falla, entonces se debe ejecutar la instrucción inmediata, pero si la condición del salto se cumple se debe actualizar el PC con la dirección de la siguiente instrucción que debe ejecutarse. Esto introduce un riesgo de control, tal y como se ha indicado anteriormente.

Una estrategia para resolver este tipo de problema es asumir que la condición del salto no se cumplirá, y por lo tanto la ejecución continuará con la instrucción que se encuentra inmediatamente después de la instrucción de salto. Si la condición del salto se cumple, entonces se deben desechar del cauce las instrucciones que fueron captadas y la ejecución continúa con la instrucción que se encuentra en la dirección de salto. Si la mitad de las veces los saltos son no tomados, y si descartar instrucciones cuesta poco, esta optimización reduce el coste de los riesgos de control a la mitad.

Una manera de mejorar el rendimiento de los saltos es reducir el coste de los saltos tomados. Si se supone que el siguiente valor del PC en el caso de un salto se selecciona en la etapa MEM, tendríamos que eliminar tres instrucciones del cauce en el caso de un salto tomado. Si se mueve la ejecución del salto a una etapa anterior del cauce se tendrían que eliminar menos instrucciones.

En el simulador Simula3MS está implementada una técnica de salto denominada salto retardado, que consiste en introducir en el cauce la siguiente instrucción al salto y dejar que se ejecute completamente, tanto si el salto se realiza como si no lo hace. También está implementada la técnica denominada salto fijo, que consiste en introducir en el cauce la siguiente instrucción al salto, pero en el momento en que se conoce si el salto no se realiza la elimina del cauce de manera que no acaba su ejecución. Estas dos técnicas resuelven el salto en la etapa ID, por lo tanto con un hueco de retardo. En este trabajo se pretende introducir otras cuatro técnicas nuevas que serían salto retardado y salto fijo (que a partir de ahora llamaremos predecir no tomado) para dos y tres huecos de retardo, es decir, se sabrá si el salto es o no tomado en la etapa EX y MEM respectivamente.

2.3 MODELADO DE LA ARQUITECTURA MIPS EN SIMULA3MS.

Simula3MS [9] es un simulador resultado de un proyecto del grupo de Arquitectura de Computadores de la Universidad de A Coruña. El proyecto abarca la implementación de un simulador de una arquitectura básica de tipo MIPS en sus versiones monociclo, multiciclo y segmentado, que se pretende usar en los laboratorios de prácticas de asignaturas que tratan la organización del computador.

Simula3MS modela un procesador RISC (Reduced Instruction Set Computer) [4] [10], que implementa un subconjunto de instrucciones basadas en el repertorio de instrucciones del procesador MIPS [4]. La elección de un procesador RISC frente a uno CISC (Complex Instruction Set Computer) está basada en la mayor relevancia de este tipo de arquitectura. También cabe destacar que estos procesadores presentan una estructura y un repertorio de instrucciones fácil de comprender. Usando una arquitectura RISC, los estudiantes aprenden los fundamentos básicos del repertorio de instrucciones y

de la programación en lenguaje ensamblador en menos tiempo que utilizando un procesador CISC. Por otra parte, Simula3MS modela un coprocesador de punto flotante en todas sus configuraciones

Simula3MS es un simulador de un procesador configurable, que cuenta con un entorno de trabajo sencillo que permite depurar fácilmente los programas y observar la evolución de la memoria, así como la ejecución de las instrucciones sobre distintos caminos de datos como pueden ser monociclo, multiciclo y segmentado. La posibilidad de utilizar los distintos cauces en la misma herramienta permite observar las diferencias de ejecución de un mismo código según cuales sean las características del procesador. La interacción del usuario con la herramienta se hace por medio de una interfaz gráfica implementada con Java [1] [2] [3].

Un punto importante en Simula3MS consiste en poder observar la evolución de todos estos componentes ciclo a ciclo, pero teniendo también la opción de ejecutar conjuntamente todas las instrucciones y observar únicamente el efecto que produce la ejecución completa del programa cargado.

Simula3MS [3] modela un subconjunto de instrucciones basadas en el repertorio de instrucciones del procesador MIPS R2000/R3000. El simulador incluye un editor que permite analizar sintácticamente las instrucciones antes de pasar a observar la ejecución de las mismas. Desde la misma ventana se permite también elegir tres configuraciones: Entrada/Salida, técnicas de salto y camino de datos. Podemos ver esta ventana en la figura 2.9 y figura 2.10. Una vez analizadas sintácticamente las instrucciones y seleccionada la configuración que se desee se puede seguir la evolución del segmento de datos, así como de los registros y del resto de elementos de la ventana de ejecución.

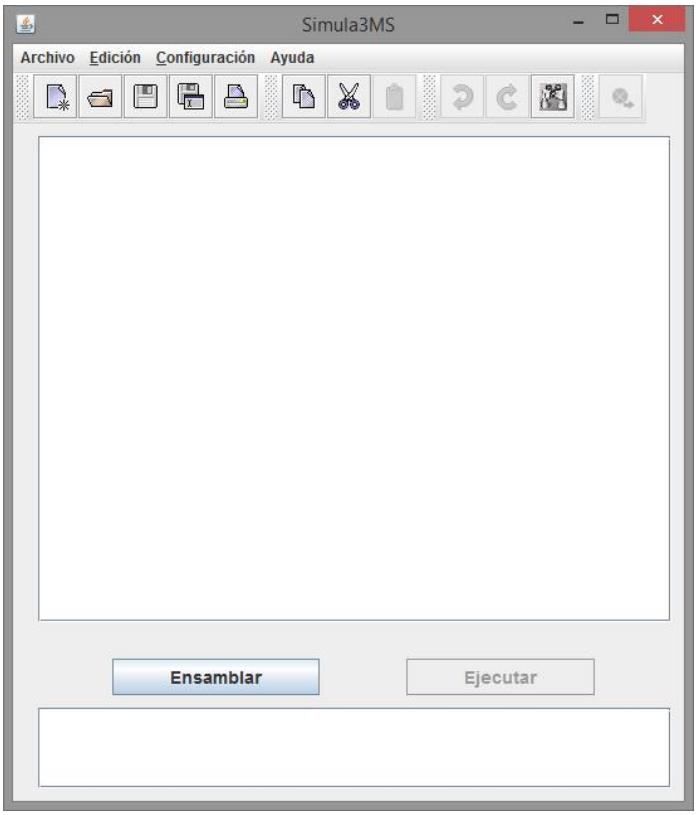

Figura 2. 9. Ventana del editor de Simula3MS.

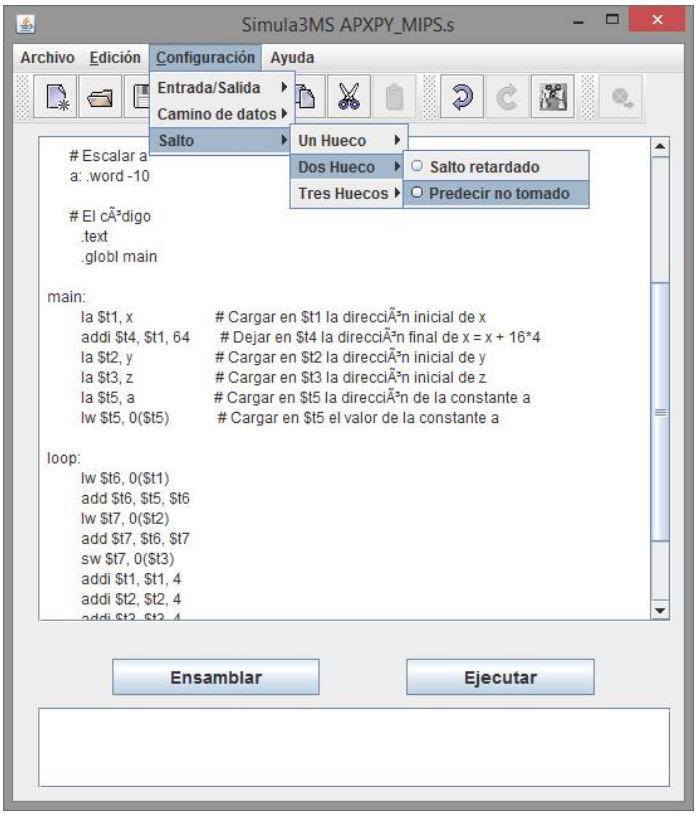

Figura 2. 10. Ventana del editor de Simula3MS con código y opciones desplegadas.

La parte superior de esta ventana, engloba los menús y la barra de botones típicos de un editor, con la salvedad del menú Configuración y el botón de Error siguiente. El menú Configuración permite elegir al usuario entre los diferentes procesadores implementados. El botón Error siguiente se activará en el caso de que una vez analizado el código se produzca más de un error sintáctico. Su función es ayudar al usuario en la corrección del código permitiendo avanzar al siguiente error de forma sencilla.

La zona central es un área de texto destinada a la elaboración del código en lenguaje ensamblador, y la zona inferior será utilizada para destacar los errores que se detecten al ensamblar el código.

Los botones *Ensamblar* y *Ejecutar* aparecen desactivados en un principio. La finalidad de *Ensamblar* es indicar si el código es, o no, sintácticamente correcto. Si lo es, se activará el botón *Ejecutar*, y si no se mostrarán los errores en la parte inferior de la pantalla, indicando brevemente el motivo. Al pulsar *Ejecutar* se pasará a la ventana que muestra el simulador, la ventana de ejecución.

La ventana de ejecución de Simula3MS presenta pequeñas variaciones dependiendo del tipo de camino de datos que se escoja. En la figura 2.11 se muestra esta ventana con un camino de datos segmentado.

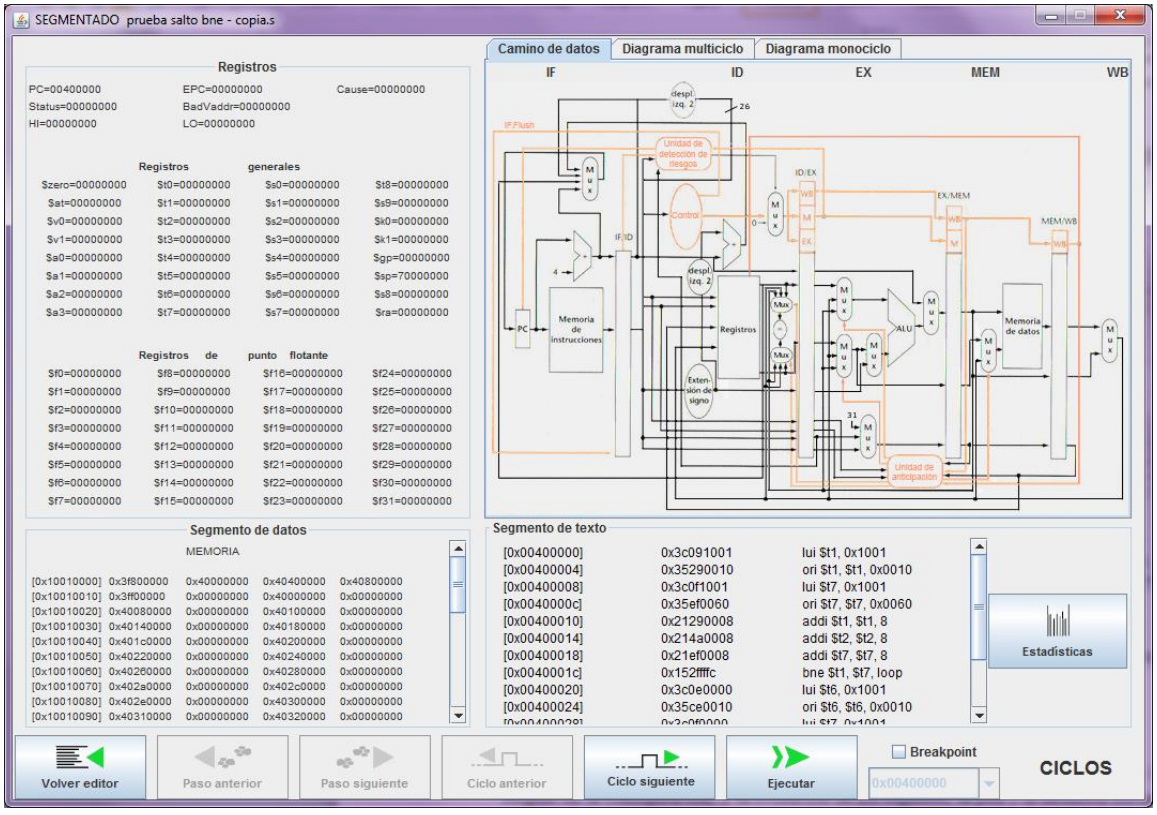

Figura 2. 11. Ventana de ejecución de Simula3MS para un camino de datos segmentado.

Los *Registros* se muestran en la parte superior izquierda y están divididos en tres partes: *Registros especiales* (como son PC, HI, LO...), *Registros generales* y los *Registros de punto flotante*. En la parte superior derecha se puede ver el camino de datos correspondiente a un procesador segmentado sobre el que se representará la realización concreta de cada instrucción. En la implementación monociclo (figura 2.12) y segmentado el color de esta representación dependerá del tipo de la instrucción. Mientras, en el caso de multiciclo (figura 2.13) variará dependiendo de la etapa en la que nos encontremos.

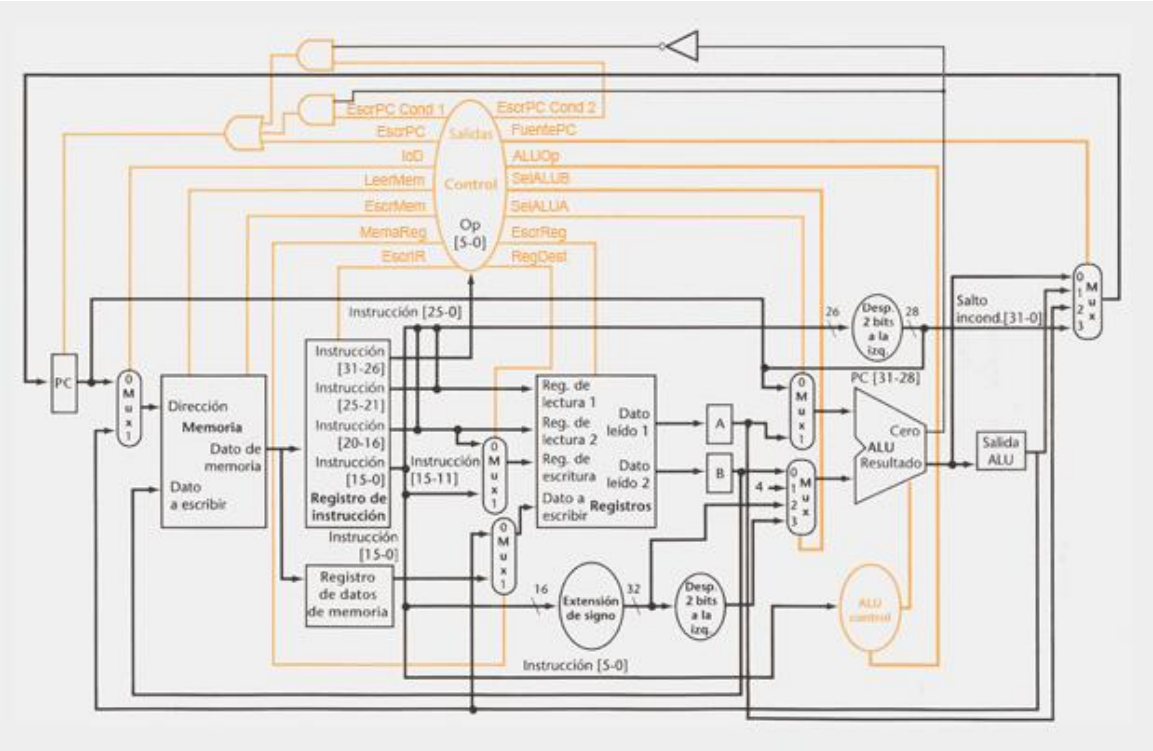

Figura 2. 12. Camino de datos monociclo.

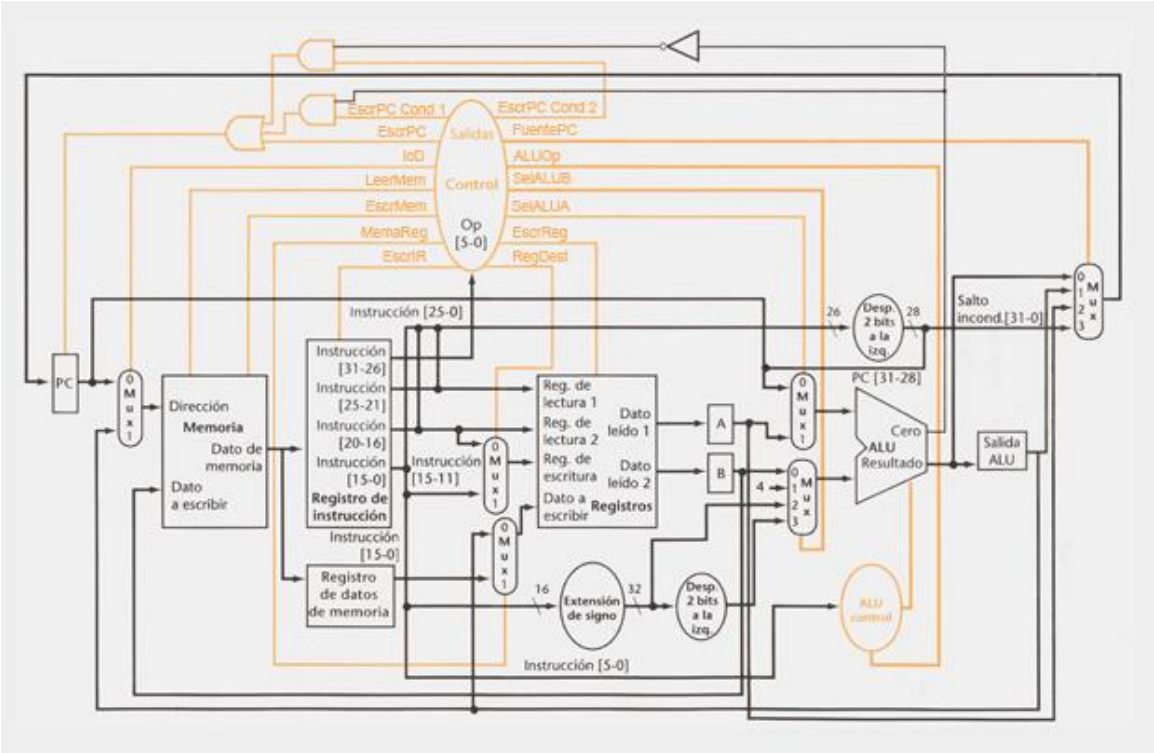

Figura 2. 13. Camino de datos multiciclo.

La realización segmentada incluye además el diagrama multiciclo (figura 2.14) que se utiliza para dar una perspectiva general de diferentes situaciones dentro de la segmentación. Se considera que el tiempo avanza de izquierda a derecha, situación que se indica con el número de ciclo en la parte superior de la imagen, y las instrucciones se desplazan de la parte superior a la inferior del panel.

Figura 2. 14. Ventana de ejecución de Simula3MS para un camino de datos segmentado.

El *Segmento de datos* se puede dividir en dos zonas: memoria y pila. La memoria está dividida en dos partes, la primera columna, cuyo valor está ente corchetes es el valor de comienzo de la primera palabra de la línea. Las otras cuatro columnas son los datos almacenados en la memoria de forma consecutiva. La pila crece hacia direcciones inferiores de memoria. Igual que la memoria esta zona también está dividida en dos partes, la primera columna indica la posición del puntero de pila y las restantes los datos.

El *Segmento de texto* contiene una lista de instrucciones que componen el programa a ejecutar. Está dividido en tres columnas. En la primera columna se indica entre corchetes la dirección de memoria hexadecimal (PC) de la instrucción. Las instrucciones se almacenan a partir de la dirección 0x0040000. La segunda columna es la codificación numérica de la instrucción en hexadecimal, y la tercera columna es la instrucción en lenguaje ensamblador. Aquella instrucción que esté siendo ejecutado en ese momento aparecerá resaltada en color. Si se ha insertado algún punto de ruptura (breakpoint) las instrucciones posteriores aparecerán en gris y en cursiva. Al lado de las instrucciones podemos observar que existe un botón llamado "Estadísticas", el cual, si lo pulsamos, abre otra ventana en la que podemos visualizar las estadísticas de rendimiento y los accesos a memoria, como podemos ver en la figura 2.15.

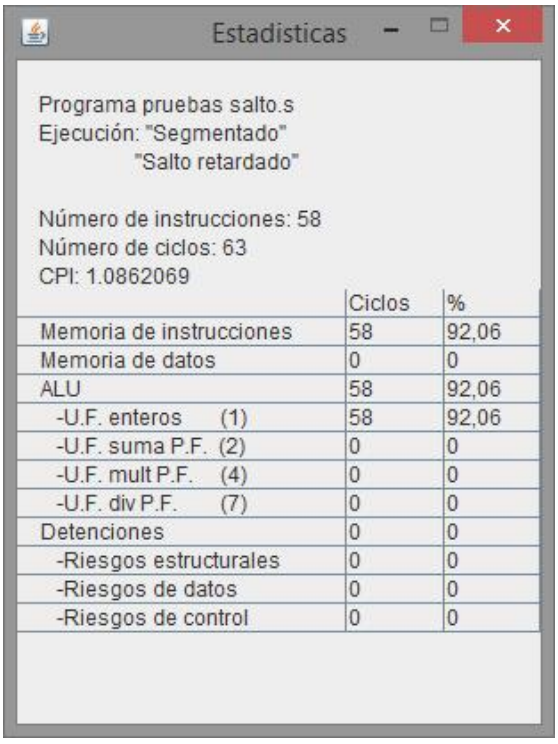

Figura 2. 15. Ventana “Estadísticas” de Simula3MS.

La parte inferior de la ventana contiene los siguientes botones: *Volver al editor*, *Ciclo anterior*, *Ciclo siguiente* y *Ejecutar* que ejecuta la totalidad del código o hasta el

punto de ruptura. Además, en la configuración multiciclo aparecen habilitados los botones de *Paso siguiente* y *Paso anterior*.

Después de estos botones hay un botón de selección (*Breakpoint*) que permite poner un punto de ruptura en el código. Si se selecciona este botón se activará el menú desplegable situado debajo de él para permitir elegir la situación del punto de ruptura entre las direcciones de las instrucciones desde la que se está ejecutando hasta la última del código. Por último, al final de la barra hay un número que indica el número de ciclos que han sido ejecutados hasta ese momento.

Simula3MS ofrece tres posibles configuraciones del camino de datos: monociclo, multiciclo y segmentado [5]. La configuración Monociclo es una implementación simple que utiliza un ciclo de reloj largo para cada instrucción. Esta aproximación no es práctica, pero es apropiada para explicar y comprender los conceptos básicos del camino de datos de un procesador. En la configuración multiciclo cada instrucción se divide en un conjunto de pasos y cada paso se ejecuta en un ciclo de reloj. Por lo tanto, esta configuración permite que diferentes instrucciones necesiten distinto número de ciclos de reloj, siendo más rápida que la configuración monociclo. Además, puede configurarse la latencia de las unidades funcionales de punto flotante (sumador, multiplicador y divisor). Si la seleccionada es la tercera de las configuraciones, segmentado, el usuario puede elegir entre camino de datos segmentado básico o dos técnicas de planificación dinámica, Marcador y algoritmo de Tomasulo. La configuración segmentada básica tiene cinco etapas: búsqueda, lectura de registros y decodificación de la instrucción, ejecución de la operación o cálculo de una dirección, acceso a un operando en memoria de datos y finalmente escritura del resultado en un registro. El simulador implementa una unidad de anticipación para la etapa de ejecución con el objetivo de reducir los riesgos de datos.

Una vez visto lo que nos ofrece Simula3MS, se pueden observar diferentes funcionalidades de las que carece y que serían interesantes de usar en un entorno docente. En este trabajo se aborda el añadido de una de estas funcionalidades, concretamente la introducción de cuatro nuevas estrategias a la hora de resolver los saltos. Las modificaciones en el simulador necesarias para ello se detallan en el siguiente capítulo.

CAPÍTULO 3. DISEÑO E IMPLEMENTACIÓN DE LOS DISTINTOS TIPOS DE SALTO

En este capítulo, en primer lugar veremos los cambios introducidos en las imágenes del camino de datos, necesarias para contemplar las nuevas opciones de salto. Concretamente, se ha modificado la imagen inicial, introduciendo los cambios necesarios para la opción con sólo un hueco de retardo, que ya estaba implementada. Además, se han creado dos nuevas imágenes para las nuevas opciones de dos y tres huecos de retardo. También se han modificado las diferentes imágenes en las que se ilumina el paso de las instrucciones, ajustándose a cada camino de datos.

Por otra parte, una vez estudiado el código del simulador Simula3MS y teniendo claras las modificaciones a introducir para implementar las nuevas opciones de salto, se llega a la conclusión de que no es necesario añadir ninguna clase nueva, aunque sí muchos métodos en clases ya existentes, así como modificar muchos de los ya existentes. Seguidamente en este capítulo se irán describiendo las modificaciones realizadas en las diferentes clases del código original que son necesarias para conseguir el objetivo principal de este TFG. Para ello se irá describiendo, para cada clase, qué hacía originalmente y qué hace ahora, en la nueva versión del simulador, sin llegar a explicar líneas de código en detalle. Este código puede verse en el Anexo A.

3.1 MODIFICACIONES DE LAS IMÁGENES DEL CAMINO DE DATOS.

En primer lugar se ha modificado la imagen del camino de datos segmentado original añadiendo dos multiplexores a la entrada del comparador para tener cubierto el caso de que haya que adelantar algún dato de otra etapa para evitarnos huecos innecesarios, como podemos ver en la figura 3.1. En esta figura se han señalado en rojo las principales modificaciones introducidas. Este adelantamiento no estaba contemplado en la versión original del simulador.

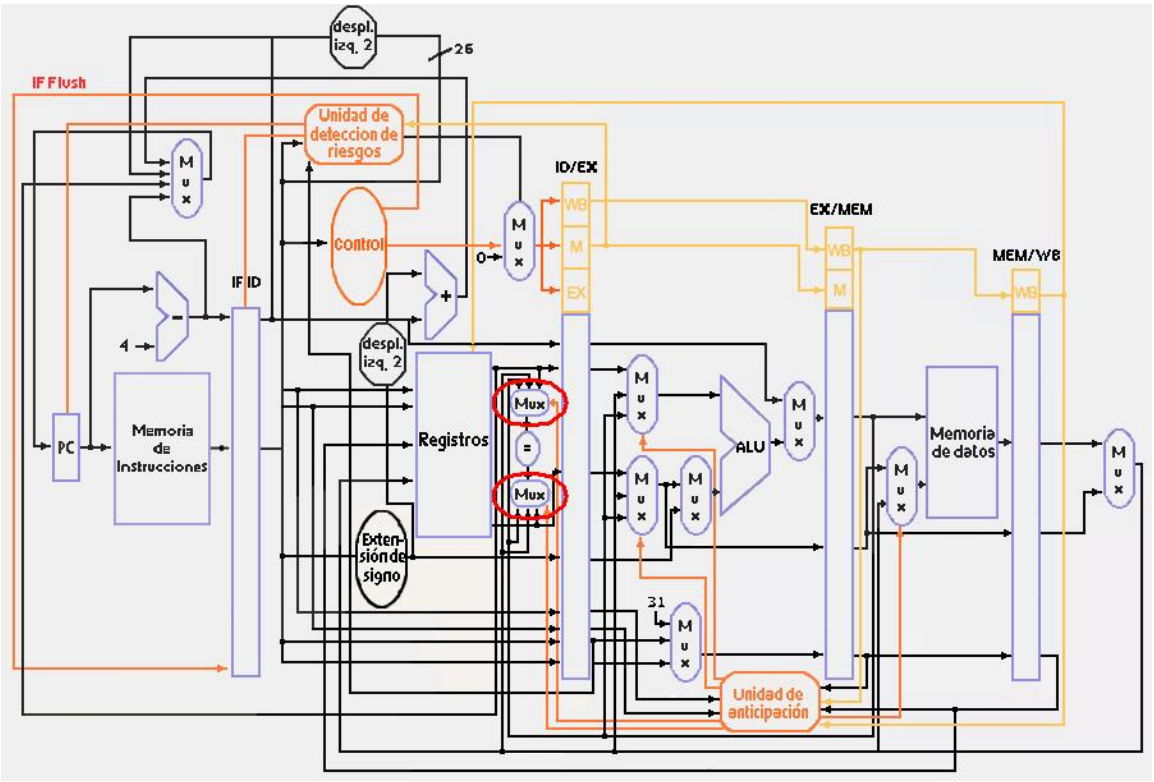

Figura 3.1. Nueva imagen del camino de datos segmentado para un hueco de retardo.

Para los casos de dos y tres huecos de retardo, se han creado imágenes nuevas en las que se ha desplazado la circuitería en la que se calcula la dirección del salto, como podemos ver en las figuras 3.2 y 3.3. Para el caso de dos huecos de retardo (figura 3.2) se ha desplazado a la etapa EX el sumador y el desplazamiento para calcular el resultado del salto, y se hará la comprobación para ver si el salto es o no tomado en la ALU. En el caso de tres huecos de retardo (figura 3.3) el cálculo se hará igual, con la diferencia que el registro PC se actualiza un ciclo más tarde.

Las imágenes que emplea Simula3MS son de tipo [*.Gif*]. Para modificar y crear estas imágenes (y el resto de imágenes del capítulo) se ha utilizado la herramienta Photoshop. Con esta herramienta se han podido generar las nuevas imágenes conservando la calidad de las originales.

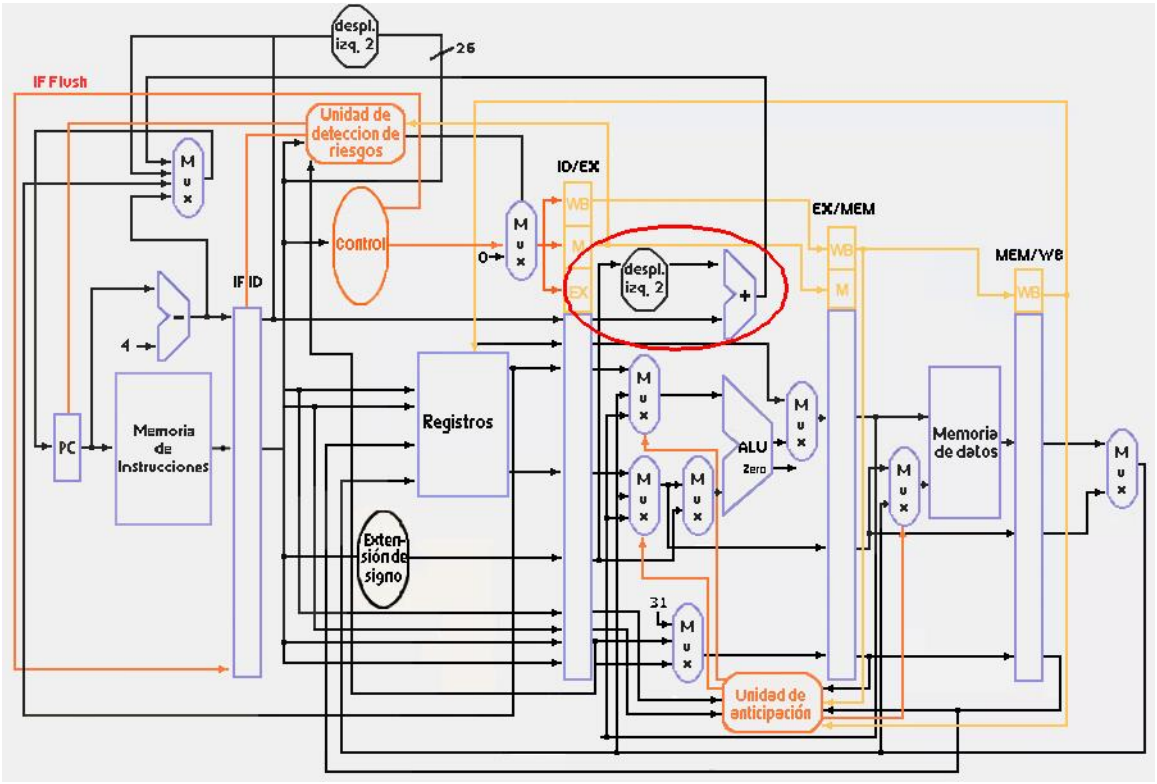

Figura 3.2. Nueva imagen del camino de datos segmentado, para dos huecos de retardo.

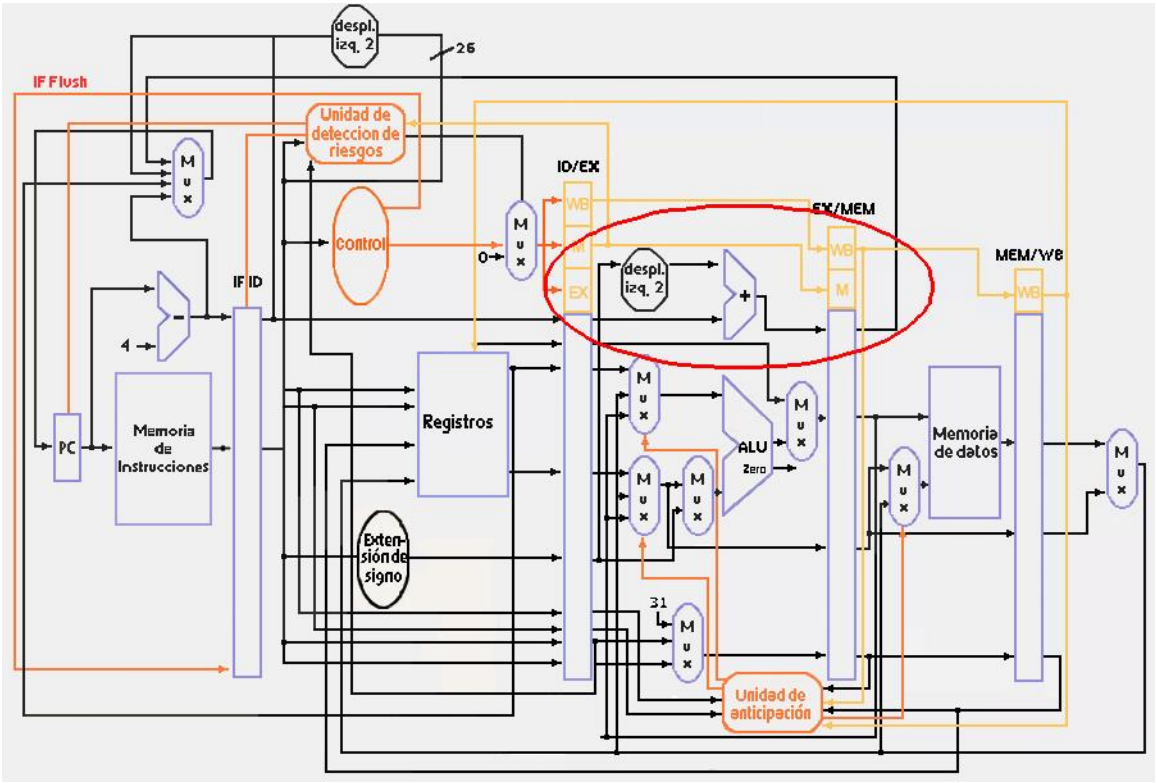

Figura 3.3. Nueva imagen del camino de datos segmentado, para tres huecos de retardo.

Para el caso de un hueco de retardo también se han tenido que modificar todas las imágenes en las que se iluminan las diferentes líneas implicadas en el salto. Para dos y tres huecos de retardo han tenido que crearse nuevas líneas, como podemos observar en la figura 3.4. Para el caso de dos y tres huecos, además de modificar las imágenes para la instrucción de salto, también se han tenido que modificar el resto de imágenes de las demás instrucciones para la etapa EX, donde se han introducido los cambios.

Figura 3.4. Comparativa de la iluminación de líneas implicadas en el salto en los distintos tipos de salto para la misma etapa.

3.2 MAPA DE CLASES.

El código del simulador Simula3MS está estructurado siguiendo un mapa de clases que se muestra en la figura 3.5. En la figura aparecen destacadas en color rojo aquellas clases que se han modificado y que por tanto son significativas para explicar el trabajo desarrollado en este TFG.

En los siguientes puntos se describen las modificaciones introducidas en estas clases.

Figura 3.5. Mapa de clases del código de Simula3MS, con las clases que se han modificado en este proyecto resaltadas en color rojo.

3.3 CLASE TIPOS.

En esta clase, entre otras, existen dos constantes que se utilizan para comprobar en distintos lugares si se ha seleccionado "Salto Retardado" o "Predecir No Tomado" (también llamado salto fijo). Estas constantes son:

- SALTO_RETARDAD0
- SALTO_FIJO

Para poder implementar las nuevas opciones de salto necesitaremos otras cuatro constantes con las que identificar si el salto seleccionado ha sido "Salto Retardado" o "Predecir No Tomado", en cada caso con dos o tres huecos de retardo. Estas constantes son:

- SALTO_RETARDAD0_2HUECOS
- SALTO_FIJO_2HUECOS
- SALTO_RETARDAD0_3HUECOS
- SALTO_FIJO_3HUECOS

Las constantes tendrán los valores que se muestran en la figura 3.6.

```
263     int SALTO_RETARDAD0=0;
264     int SALTO_FIJO=1;
265
266     // Nuevos tipos para dos y tres huecos
267     int SALTO_RETARDAD0_2HUECOS=2;
268     int SALTO_FIJO_2HUECOS=3;
269     int SALTO_RETARDAD0_3HUECOS=4;
270     int SALTO_FIJO_3HUECOS=5;
```

Figura 3.6. Valores de las constantes de la clase Tipos.

Como se observa en la figura 3.6, se han introducido estas nuevas constantes en el constructor de la clase Tipos. Estas constantes servirán a lo largo de todo el programa para comprobar en cada momento con qué estrategia de salto estamos trabajando, y determinar así el desarrollo a seguir.

3.4 CLASE ENSAMBLADOR.

Esta clase es la encargada de mostrar la ventana inicial donde aparece el menú que permite seleccionar el modo de visualizar el simulador y las diferentes opciones disponibles para cada caso. Originalmente, Simula3MS sólo nos permitía seleccionar las opciones “Salto Retardado” o “Salto fijo”, según se muestra en la figura 3.7.

Figura 3.7. Opciones de salto en el simulador Simula3MS original.

En el caso del simulador resultante de este TFG en particular, visualizaremos el simulador en modo segmentado y tendremos 6 opciones diferentes para el salto, según se muestra en la figura 3.8.

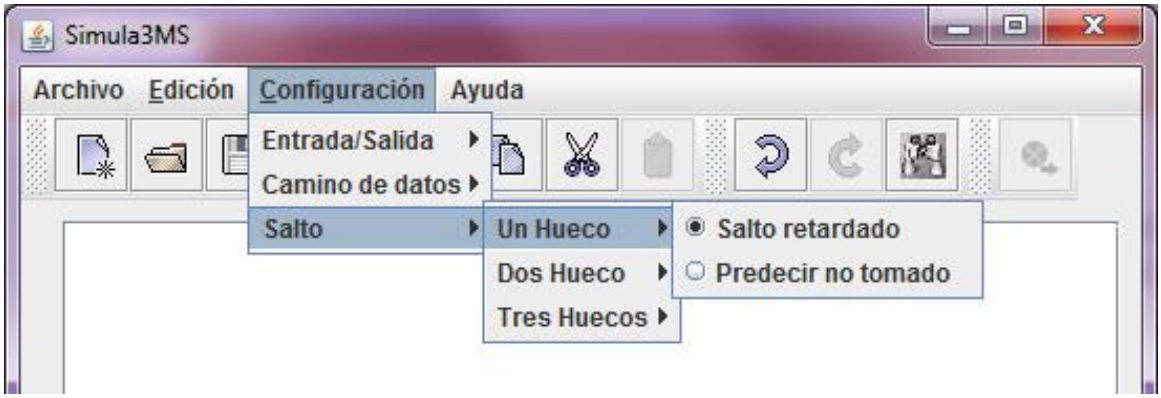

Figura 3.8. Opciones de salto en el simulador actual.

Para llegar a este resultado se han tenido que modificar diferentes métodos de la clase y crear algunos otros. Veamos los cambios que se han ido realizando:

- Método ***itemSaltoRetardadoFlotanteActionPerformed***: Este método se ha modificado para activar el parámetro *itemSaltoRetardadoFlotante* y desactivar el resto. De esta manera estamos indicando que la estrategia de salto elegida es "Salto retardado" para un hueco de retardo.

- Método ***itemSaltoFijoFlotanteActionPerformed***: Este método se ha modificado para activar el parámetro *itemSaltoFijoFlotante* y desactivar el resto. De esta manera estamos indicando que la estrategia de salto elegida es "Predecir no tomado " para un hueco de retardo.
- Método ***itemSaltoRetardadoFlotante2HuecosActionPerformed***: Este método se ha creado para activar el parámetro *itemSaltoRetardadoFlotante2Huecos* y desactivar el resto. De esta manera estamos indicando que la estrategia de salto elegida es "Salto retardado" para dos huecos de retardo.
- Método ***itemSaltoFijoFlotante2HuecosActionPerformed***: Este método se ha modificado para activar el parámetro *itemSaltoFijoFlotante2Huecos* y desactivar el resto. De esta manera estamos indicando que la estrategia de salto elegida es "Predecir no tomado" para dos huecos de retardo.
- Método ***itemSaltoRetardadoFlotante3HuecosActionPerformed***: Este método se ha creado para activar el parámetro *itemSaltoRetardadoFlotante3Huecos* y desactivar el resto. De esta manera estamos indicando que la estrategia de salto elegida es "Salto retardado" para tres huecos de retardo.
- Método ***itemSaltoFijoFlotante3HuecosActionPerformed***: Este método se ha modificado para activar el parámetro *itemSaltoFijoFlotante3Huecos* y desactivar el resto. De esta manera estamos indicando que la estrategia de salto elegida es "Predecir no tomado" para tres huecos de retardo.
- Método ***EnsamblarActionPerformed***: Este método se ha modificado para indicar el tipo de salto escogido. De esta manera indicaremos qué estrategia de salto hemos elegido, "Salto retardado" o "Predecir no tomado" para uno, dos o tres huecos de retardo.
- Método ***EjecutarActionPerformed***: Este método se utiliza para generar la vista del procesador elegida (monociclo, multiciclo o segmentado). En caso de ser segmentado se ha modificado la manera de indicar el tipo de salto seleccionado.

3.5 CLASE PIPELINEFLOTANTE.

Esta clase, en el simulador original, es la encargada de ir avanzando las instrucciones por las distintas etapas del camino de datos. Mediante el método *avanzarPipeline(Instruccion, int, Estadisticas)* se van cambiando de etapa las instrucciones, introduciendo burbujas en el cauce si es necesario. En el caso en que se aplica la estrategia del salto de “Predecir no tomado”, si el salto es tomado, hay que

quitar del cauce la instrucción que se inició de forma especulativa pero que al fallar la predicción no debe finalizarse.

En el caso de un hueco se elimina la instrucción que había en la etapa IF para que no avance en el cauce, ya que esta instrucción no debe acabarse. Al avanzar en el método se introducirá una nueva instrucción en la etapa IF pero no avanzará a la etapa ID ninguna, por lo que se generará un `nop` en la etapa ID.

Para el caso de dos huecos de retardo se eliminan las instrucciones que había en las etapas IF e ID. Por lo tanto estas instrucciones ya no avanzarán en el cauce y se introducirá `nop` en las etapas ID y EX.

Para el caso de tres huecos de retardo se eliminan las instrucciones que había en las etapas IF, ID y EX. Por lo tanto estas instrucciones ya no avanzarán en el cauce y se introducirá `nop` en las etapas ID, EX y MEM.

Avanzando en el método, en el simulador original se comprobaba si el salto era tomado en la instrucción que se encontraba en la etapa ID. Ahora, con las modificaciones introducidas se comprobará con la instrucción que esté en la etapa ID, en la etapa EX o en la etapa MEM, dependiendo de si la estrategia de salto elegida es uno, dos o tres huecos de retardo respectivamente. Esta comprobación servirá para decidir si la instrucción que está en la eta IF se ejecuta o no. En el caso de resultar salto tomado la instrucción no se ejecutará para que el valor del PC sea el de la dirección del salto. En el caso de que sea salto no tomado, si la instrucción que se encuentra en la etapa ID es una instrucción de salto (tanto condicional como incondicional) no se ejecutará tampoco la instrucción que está en IF, puesto que el incremento del PC para esta instrucción también se contemplará en la ejecución del salto, tal y como se verá en el siguiente apartado.

3.6 CLASE BNE Y BEQ.

En estas clases, entre otras cosas, tenemos los métodos *ejecutarIF(), ejecutarID(), ejecutarEX(), ejecutarMEM()* y *ejecutarWB()*. Mediante estos métodos se indica lo que se hace en cada una de estas etapas para las instrucciones. Originalmente, el método *ejecutarIF()* incrementaba el PC para que apuntase a la siguiente instrucción y el método *ejecutarID()* modificaba el PC con la dirección de salto, en el caso de que éste fuese tomado, o la de la siguiente instrucción. El resto de métodos no hacían nada. En la nueva versión del simulador se han quedado igual los métodos *ejecutarIF()* y *ejecutarWB()*, y se han modificado el resto de métodos de manera que dependiendo de si hay uno, dos o tres huecos de retardo, se calcula la dirección de salto en *ejecutarID()*, *ejecutarEX()* o *ejecutarMEM()*, respectivamente.

Se ha modificado el método *ejecutarID()* de manera que si la estrategia elegida es un hueco de retardo y el salto es tomado se calculará la dirección del salto y se guardará en el PC. Por el contrario, si el salto no es tomado o la estrategia elegida es dos o tres huecos de retardo se guardará en el PC la dirección de la siguiente instrucción.

El método *ejecutarEX()* se ha modificado de manera que si la estrategia elegida es dos huecos de retardo y el salto es tomado se calculará la dirección del salto y se guardará en el PC. Por el contrario, si el salto no es tomado o la estrategia elegida es uno o tres huecos de retardo no hará nada.

En el caso del método *ejecutarMEM()* se ha modificado para que si la estrategia elegida es tres huecos de retardo y el salto es tomado se calculará la dirección del salto y se guardará en el PC. Por el contrario, si el salto no es tomado o la estrategia elegida es uno o dos huecos de retardo no hará nada.

3.7 CLASE PANELCAMINODATOS.

Esta clase es la encargada de dibujar el camino de datos en la ventana "Segmentado", que se muestra en la figura 3.14. En esta figura vemos que se introduce en la ventana la imagen correspondiente según el tipo de salto que hayamos seleccionado. Se han introducido cambios también en las distintas imágenes que sirven para resaltar las líneas de cada instrucción en cada momento de su ejecución tal y como se ha explicado en la sección 3.1 de este capítulo. Ha sido necesario introducir modificaciones en el constructor *PanelCaminoDatos(int,int*) y los métodos *actualizarImag()* y *paintComponents(Graphics).*

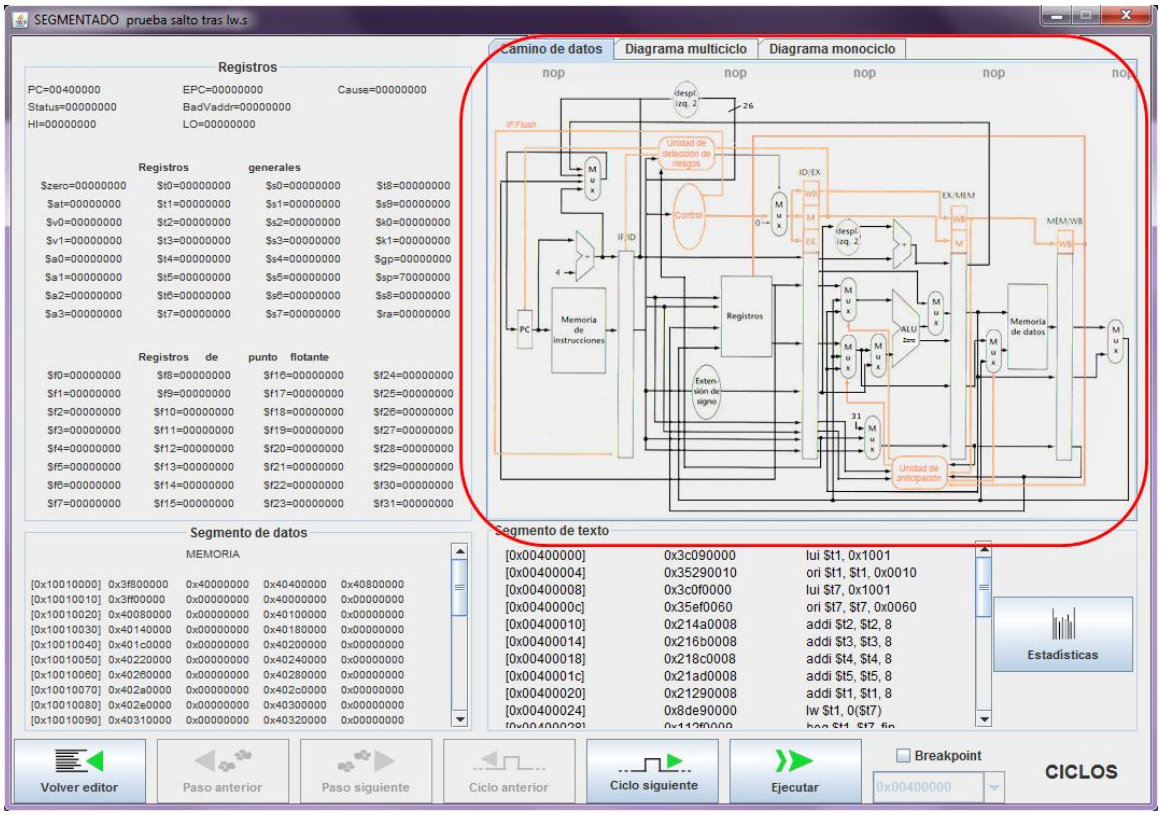

Figura 3.9. Panel de control en el procesador segmentado.

En el constructor, los cambios introducidos tienen que ver con la imagen que aparece en la ventana mostrando el camino de datos (Seleccionada en rojo en la imagen 3.14). Dependiendo de si la estrategia elegida es uno, dos o tres huecos de retardo se mostrará la imagen para segmentado con un hueco de retardo (figura 3.1), para dos huecos de retardo (figura 3.2) o segmentado para tres huecos de retardo (figura 3.3). Estas figuras aparecen en el apartado 3.1 de este capítulo.

En el método *actualizarImag()* se han añadido dos imágenes nuevas para iluminar el adelantamiento que no estaba contemplado en la versión original del simulador, como se mencionó en el apartado 3.1 de este capítulo.

En el método *paintComponents(Graphics)* se han añadido las líneas donde se muestran las imágenes nuevas en el caso de estar activadas.

3.8 CLASE SALTOCONDICIONAL.

En la clase *SaltoCondional.java* se han creado dos variables nuevas llamadas *imagSegmentado2Huecos* e *imagSegmentado3Huecos*, además de la que ya existe

imagSegmentado, en las que se han introducido las nuevas imágenes creadas para los casos de dos y tres huecos de retardo, respectivamente, como vemos en la figura 3.18.

```
private String imagSegmentado[]={"beqIF.gif","beqID.gif","beqEX.gif","beqMEM.gif","beqWB.gif"};
private String imagSegmentado2Huecos[]={"beqIF.gif","beqID2Huecos.gif","beqEX2Huecos.gif","beqMEM.gif","beqWB.gif"};
private String imagSegmentado3Huecos[]={"beqIF.gif","beqID2Huecos.gif","beqEX3Huecos.gif","beqMEM3Huecos.gif","beqWB.gif"};
```

Figura 3.10. Asignación de las diferentes imágenes a las variables que identifican los diferentes tipos de salto.

Estas variables se utilizan en el método *imagenSegmentado* dentro de esta misma clase, mediante el cual se iluminarán las líneas del salto en la etapa que corresponda para el tipo de salto que se trate.

CAPÍTULO 4. PRUEBAS DE SISTEMA

En este capítulo se muestra cómo se comporta el simulador Simula3MS después de implementar todas las modificaciones planteadas para este TFG. Se verá también cómo se comportan los diferentes tipos de saltos posibles y cómo se tratan éstos para cada opción que nos ofrece ahora Simula3MS. También se verán varios casos en los que se observa cómo se resuelven diferentes dependencias entre la instrucción de salto y otras instrucciones.

Comenzaremos viendo cómo se comporta ahora Simula3MS en el caso de una instrucción de salto condicional. Para ello se analizará primero la configuración con un hueco de retardo (el salto se resuelve en la etapa ID), luego con dos huecos de retardo (el salto se resuelve en la etapa EX) y finalmente con tres huecos de retardo (el salto se resuelve en la etapa MEM). En los tres casos se utilizarán las dos estrategias disponibles: salto retardado y predecir no tomado.

A continuación, y para finalizar este capítulo, se analizará en detalle cómo se comporta ahora Simula3MS en dos casos muy particulares, que son cuando el valor de alguno de los registros involucrados en el salto lo calcula una instrucción aritmética o depende de una carga con cierta proximidad al salto (esta proximidad depende de la versión del salto seleccionado).

4.1 INSTRUCCIONES DE SALTO CONDICIONAL BNE O BEQ.

Para este ejemplo se usará el código *prueba salto bne.s***,** que se encuentra disponible en el CD que acompaña a esta memoria. Se trata de un código sencillo para probar cómo funciona ahora Simula3MS cuando existe este tipo de instrucción. Este código puede observarse en la figura 4.1. Hay que señalar que aunque en este ejemplo concreto se utiliza la instrucción `bne`, el comportamiento sería exactamente el mismo si se utilizara la otra instrucción de salto condicional `beq`.

En este código se introduce un dato en el registro $t7, y se va incrementando el valor del registro $t1 hasta que coincida con el valor de $t7, momento en el que se saldrá del bucle y finalizará el programa. Por lo tanto el salto será tomado hasta la última iteración.

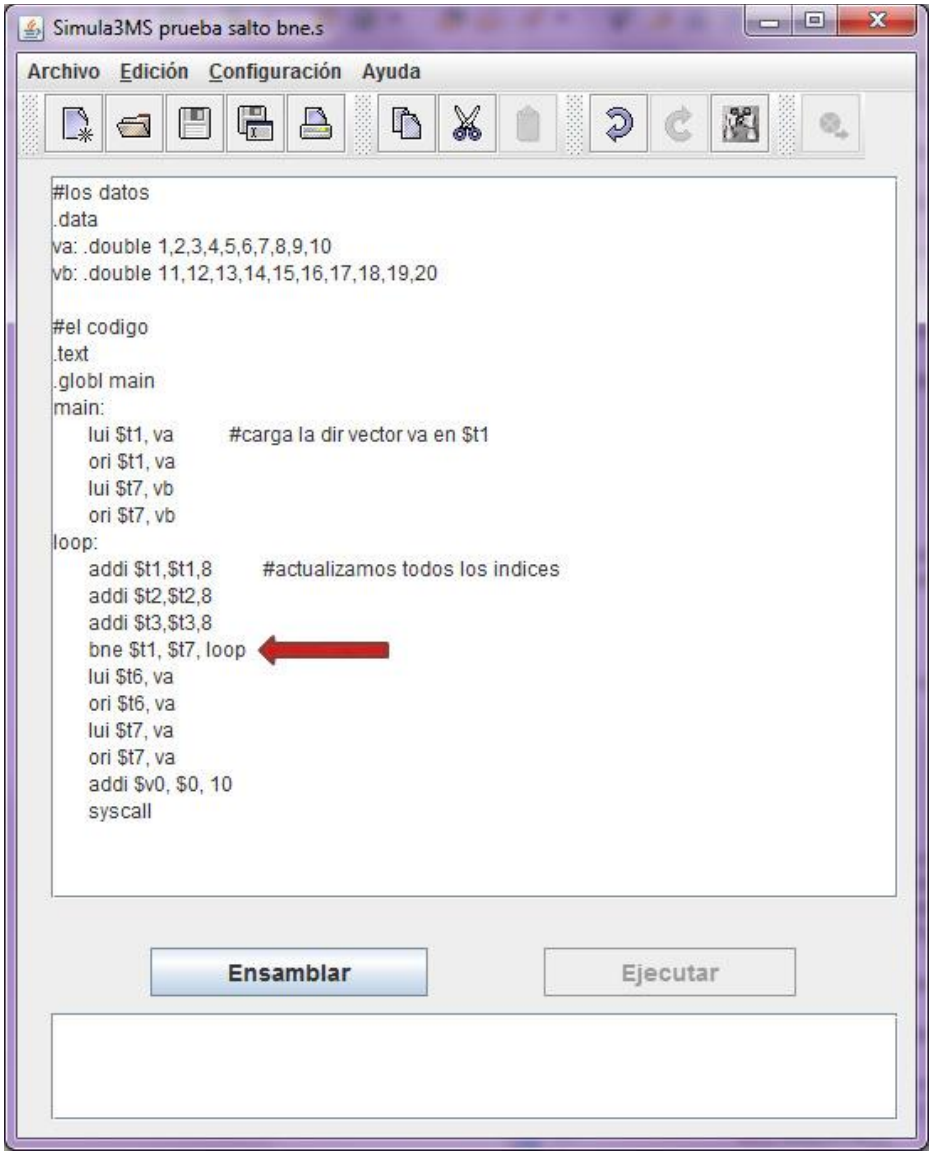

Figura 4.1. Código utilizado para probar el comportamiento del salto condicional.

Una vez cargado el código tendremos que seleccionar el tipo de salto elegido desde el menú configuración, tal como aparece en la figura 4.2.

Figura 4.2. Selección del tipo de salto elegido.

Veamos cómo se va comportando la nueva versión del simulador para los distintos tipos de salto ahora disponibles.

4.1.1 Salto con un hueco de retardo.

Para este caso se sabe si el salto es o no tomado en la etapa ID, y por lo tanto entrará en el camino de datos la siguiente instrucción al salto. Esta instrucción, en el caso de que el salto se tome, no debería ejecutarse. Hay que señalar que éste era el único tipo de salto implementado en el simulador originalmente. Se incluye en esta memoria su análisis para que el lector pueda comparar su comportamiento con los otros dos tipos incorporados en la nueva versión.

Si hemos escogido la opción “Salto retardado”, esta instrucción acabará su ejecución tanto si el salto finalmente se toma como si no lo hace (figura 4.3). En esta figura puede observarse cómo cuando el salto está ya en su etapa EX, y por lo tanto ya está resuelto, la instrucción `lui` que está posicionada justo detrás del salto continúa ejecutándose a pesar de que el salto es tomado.

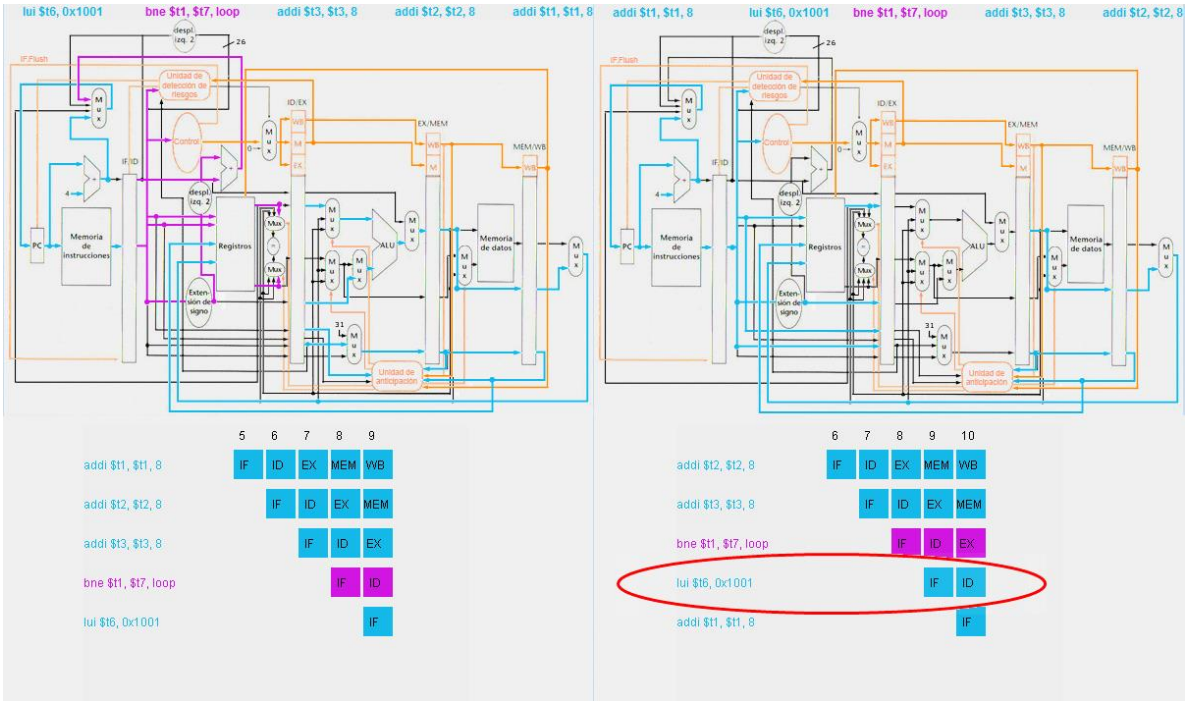

Figura 4.3. Camino de datos para el salto con la opción “Salto retardado” con un hueco de retardo.

Si, por el contrario, la opción elegida es “Predecir No tomado” la instrucción siguiente será borrada del camino de datos, y se introducirá un `nop` en su lugar en el caso de que al resolverse el salto finalmente se tome (figura 4.4).

En esta figura puede observarse cómo cuando el salto está ya en su etapa EX, y por lo tanto ya está resuelto, la instrucción `lui` que había empezado su ejecución justo detrás del salto se elimina del camino de datos y se introduce un `nop` en su lugar.

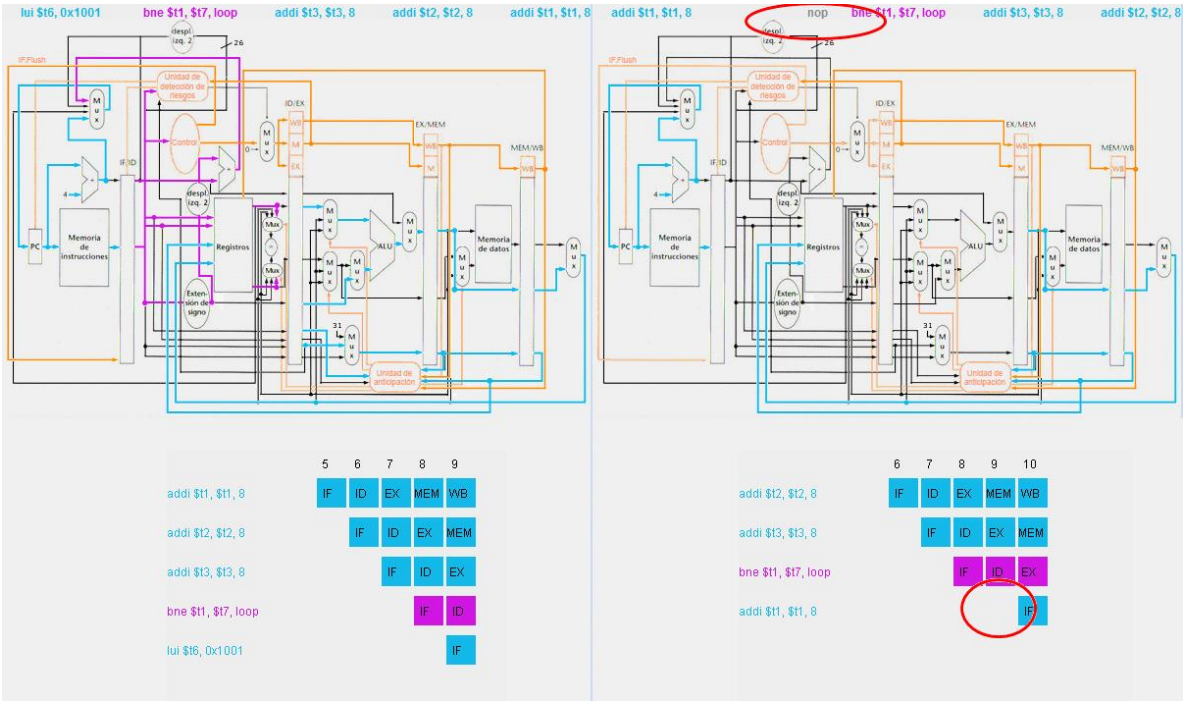

Figura 4.4. Camino de datos para el salto con la opción "Predecir no tomado" con un hueco de retardo.

4.1.2 Salto con dos huecos de retardo.

Para este caso se sabe si el salto es o no tomado en la etapa EX, y por lo tanto entrarán en el camino de datos las dos instrucciones siguientes al salto. En este momento sabremos si el salto debería ser tomado o no. En caso afirmativo la manera de gestionarlo dependerá de la opción de salto escogida. Hay que señalar que la implementación de esta opción se ha añadido al simulador en este TFG.

Si hemos escogido la opción "Salto retardado", la instrucción emitida de forma especulativa acabará su ejecución tanto si el salto finalmente se toma como si no debe tomarse (figura 4.5). En esta figura puede observarse cómo cuando el salto está ya en su etapa MEM, y por lo tanto ya está resuelto, las instrucciones `lui` y `ori` que están posicionadas justo detrás del salto continúan ejecutándose a pesar de que el salto es tomado.

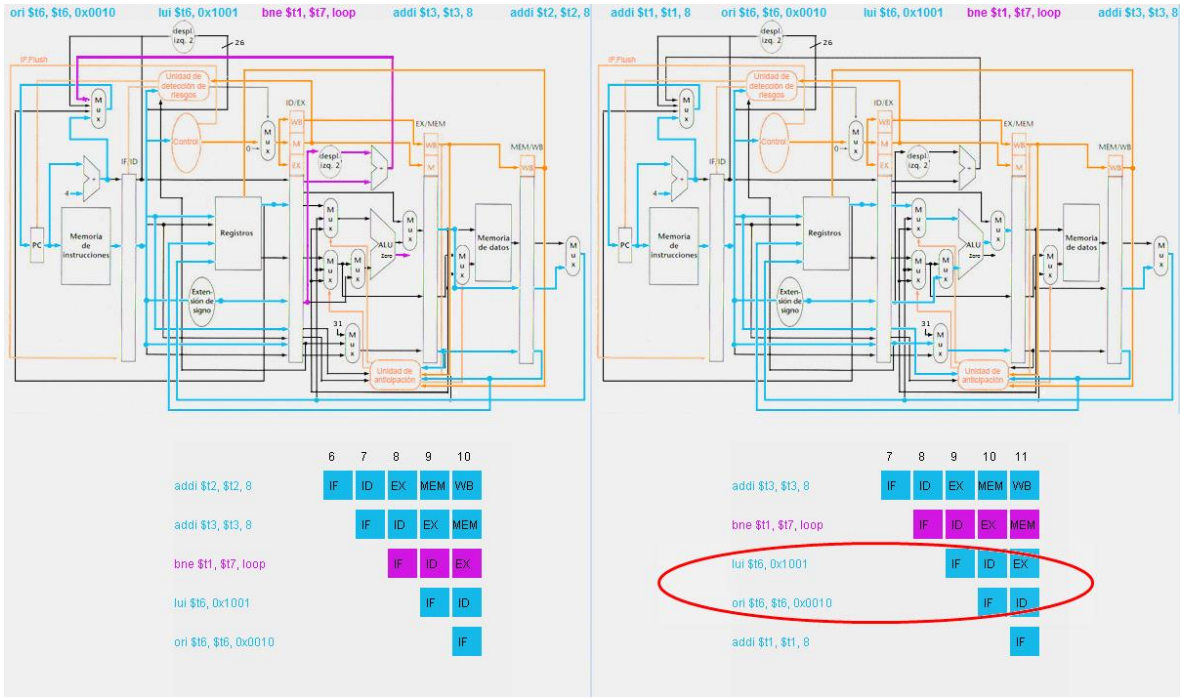

Figura 4.5. Camino de datos para el salto con la opción “Salto retardado” con dos huecos de retardo.

Si la opción elegida es “Predecir no tomado”, cuando el salto se resuelve y la decisión es que debe tomarse el salto, las instrucciones serán borradas del camino de datos, y se introducirán instrucciones `nop` en su lugar (figura 4.6).

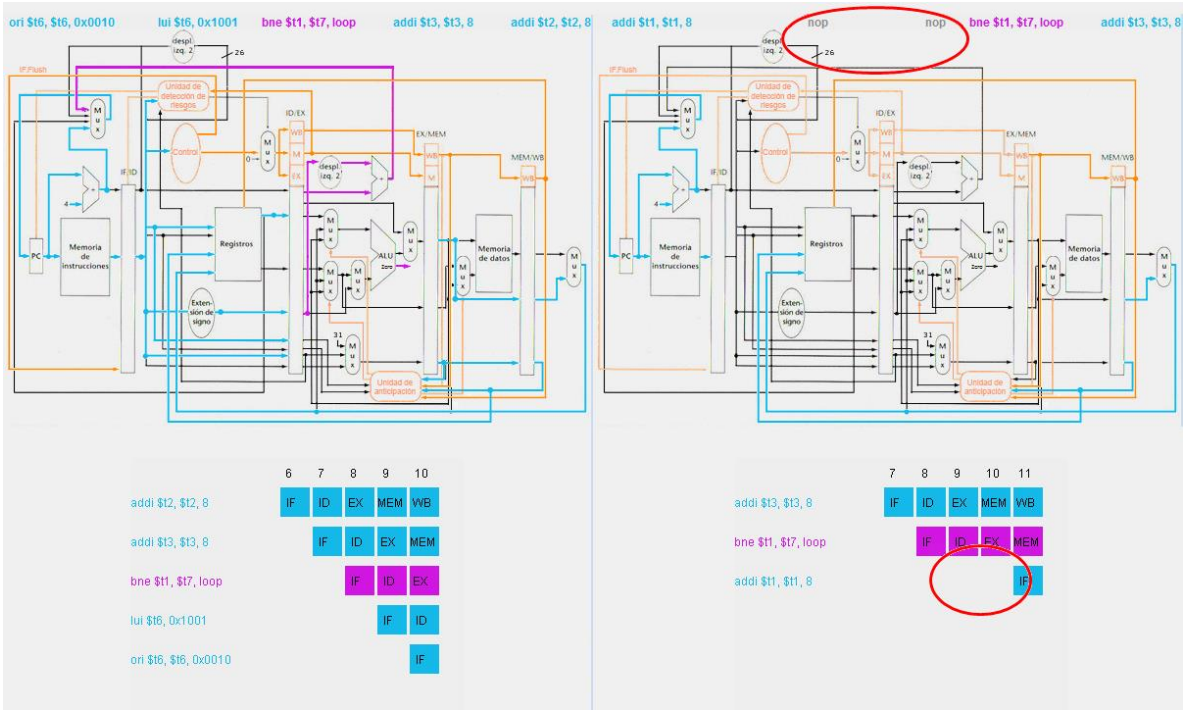

Figura 4.6. Camino de datos para el salto con la opción “Predecir no tomado” con dos huecos de retardo.

4.1.3 Salto con tres huecos de retardo.

Para este caso se sabe si el salto es o no tomado en la etapa MEM, y por lo tanto entrarán en el camino de datos las tres instrucciones siguientes al salto. Estas tres instrucciones no deberían haberse tenido en cuenta en el caso de salto tomado. Hay que señalar que la implementación de esta opción se ha añadido al simulador en este TFG.

Si hemos pulsado la opción “Salto retardado”, las instrucciones añadidas de forma especulativa acabarán su ejecución tanto si el salto finalmente se toma como si no debe tomarse (figura 4.7). En esta figura puede observarse cómo cuando el salto está ya en su etapa WB, y por lo tanto ya está resuelto, las instrucciones `lui`, `ori` y `lui` que están posicionadas justo detrás del salto continúan ejecutándose a pesar de que el salto es tomado.

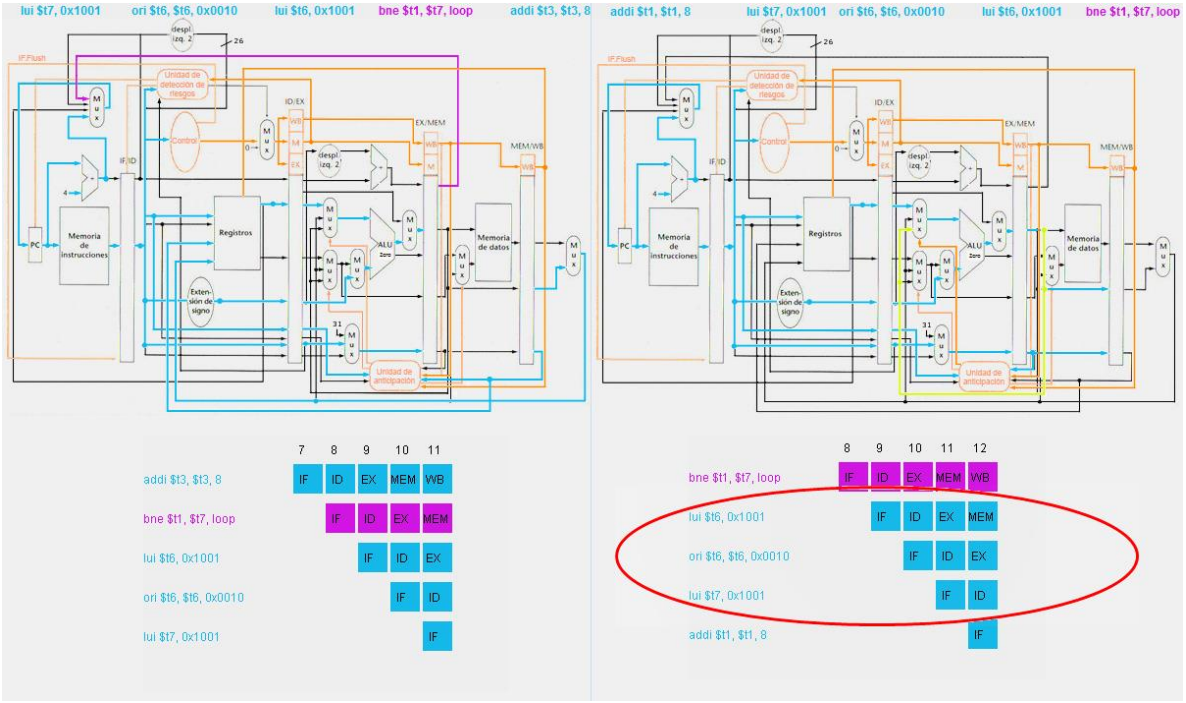

Figura 4.7. Camino de datos para el salto con la opción “Salto retardado “con tres huecos de retardo.

Si, por el contrario, la opción elegida es predecir no tomado, cuando el salto finalmente se resuelve y resulta ser tomado, las instrucciones emitidas de forma especulativa serán borradas del camino de datos, y se introducirán instrucciones `nop` en su lugar (figura 4.8).

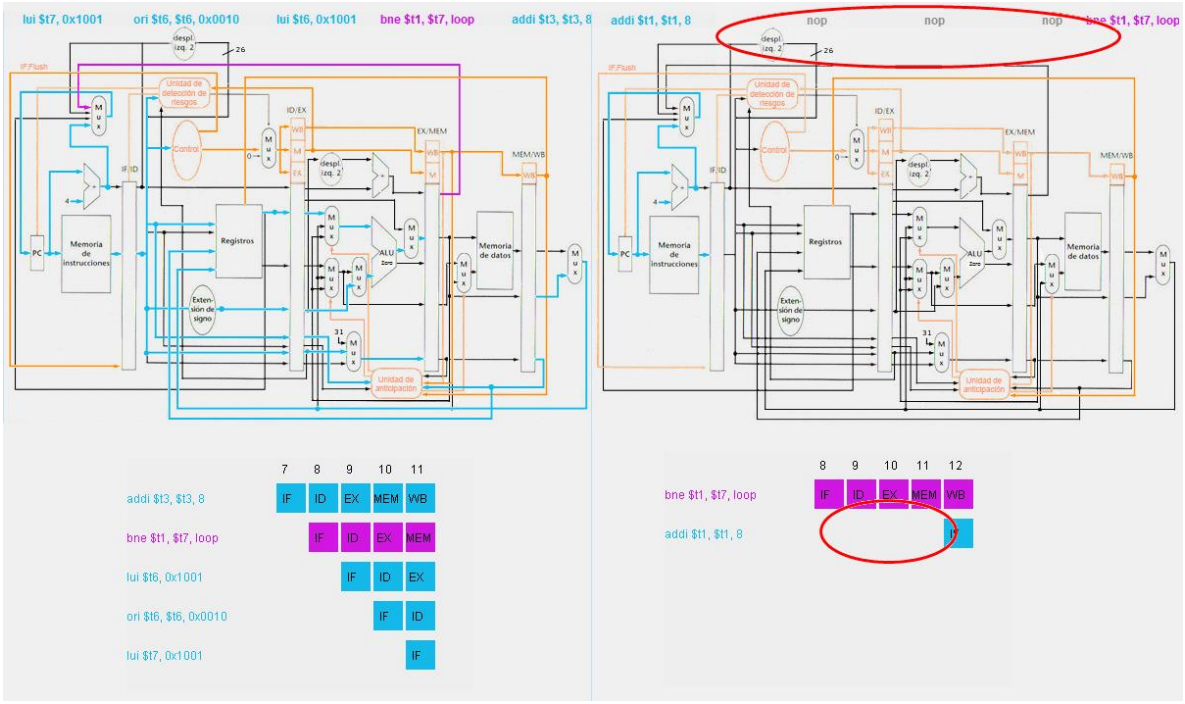

Figura 4.8. Camino de datos para el salto con la opción “Predecir no tomado” con tres huecos de retardo.

Como ha podido verse, en este caso queda comprobado que el simulador se comporta según lo deseado, calculando la dirección del salto donde corresponde de acuerdo con la teoría estudiada.

4.2 SALTO CON DEPENDENCIA DE UNA INSTRUCCIÓN ARITMÉTICA ANTERIOR.

Para este ejemplo se usará el código *prueba salto tras addi.s***,** que se encuentra disponible en el CD que acompaña a esta memoria. Este es un código sencillo para probar cómo funciona ahora Simula3MS cuando la instrucción de salto (`beq` en este caso) tiene una dependencia con una instrucción aritmética anterior (`addi` en este caso). Este código puede observarse en la figura 4.9.

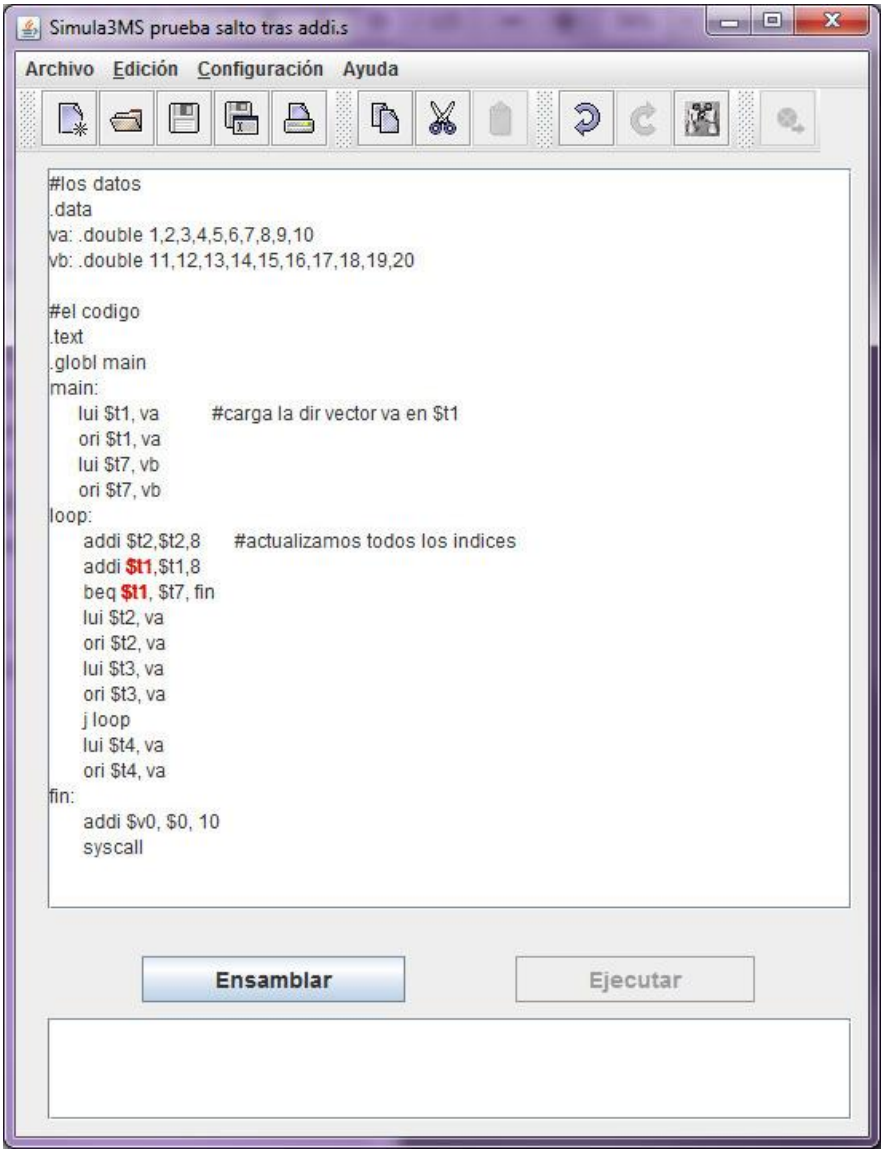

Figura 4.9. Código utilizado para probar el comportamiento del salto con dependencia de una instrucción aritmética anterior.

En este código se introduce un dato en el registro $t7, y se va incrementando el valor de $t1 hasta que coincida con el valor de $t7, momento en el que finalizarán el bucle y el programa. Con este ejemplo se quiere comprobar cómo afectaría a cada tipo de salto la dependencia con una instrucción anterior que se resuelve en la etapa EX, de modo que el dato estará disponible cuando se encuentre en la etapa MEM. En concreto, en este ejemplo puede observarse que la instrucción de salto tiene una dependencia con la instrucción `addi` que está situada justo delante de ella. Por tanto, el salto no podrá resolverse hasta que el valor esté disponible. Dependiendo de si se tiene o no adelantamiento hardware, el comportamiento será distinto. Hay que señalar que para las tres implementaciones del salto disponibles, el comportamiento no variará si la estrategia utilizada es “Salto Retardado” o “Predecir No Tomado”. Esto se debe a que el problema es un riesgo de datos anterior a la resolución del salto, y por tanto es independiente de la estrategia utilizada. Por tanto, y de

cara a mejorar la claridad de esta sección, en cada implementación sólo se va a probar la opción de “Salto retardado”.

Una vez cargado el código tendremos que seleccionar el tipo de salto elegido desde el menú configuración, tal y como vimos en el apartado 4.1 en la figura 4.2.

Veamos cómo se va comportando la nueva versión del simulador para los distintos tipos de salto ahora disponibles.

4.2.1. Salto con un hueco de retardo.

Para este caso se sabe si el salto es o no tomado en la etapa ID. Para poder resolver el salto se necesita conocer el valor de los registros $t1 y $t7. Como el valor de $t1 se calcula en la instrucción justo delante del salto, habrá que esperar un ciclo hasta que el valor de $t1 esté disponible. Esto será cuando la instrucción `addi` esté en la etapa MEM, momento en el que se podrá adelantar el valor al comparador en la etapa ID para poder resolver el salto (figura 4.10).

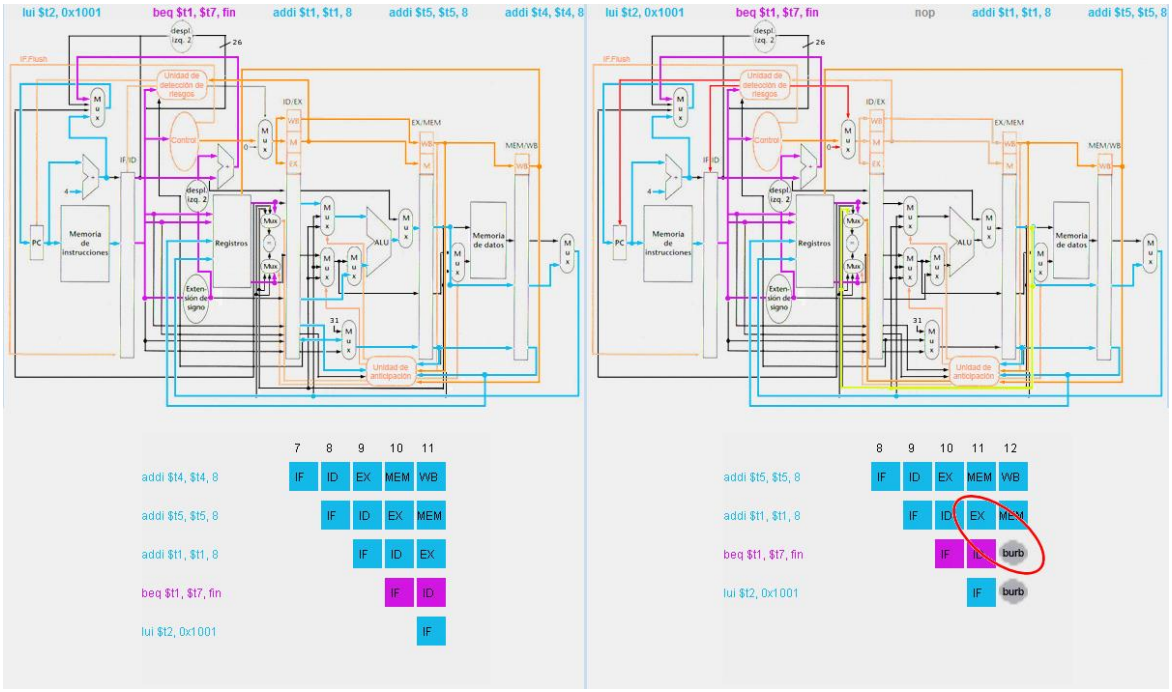

Figura 4.10. Camino de datos para el salto con un hueco de retardo y con dependencia de una instrucción aritmética anterior.

4.2.2. Salto con dos huecos de retardo.

Para este caso se sabe si el salto es o no tomado en la etapa EX. Para poder resolver el salto se necesita conocer el valor de los registros $t1 y $t7. Como el valor de $t1 se calcula en la instrucción justo delante del salto, en este caso no habría ningún ciclo de espera, ya que el dato puede adelantarse desde la etapa MEM de la instrucción `addi` hasta la etapa EX del salto, que entonces ya puede resolverse (figura 4.11).

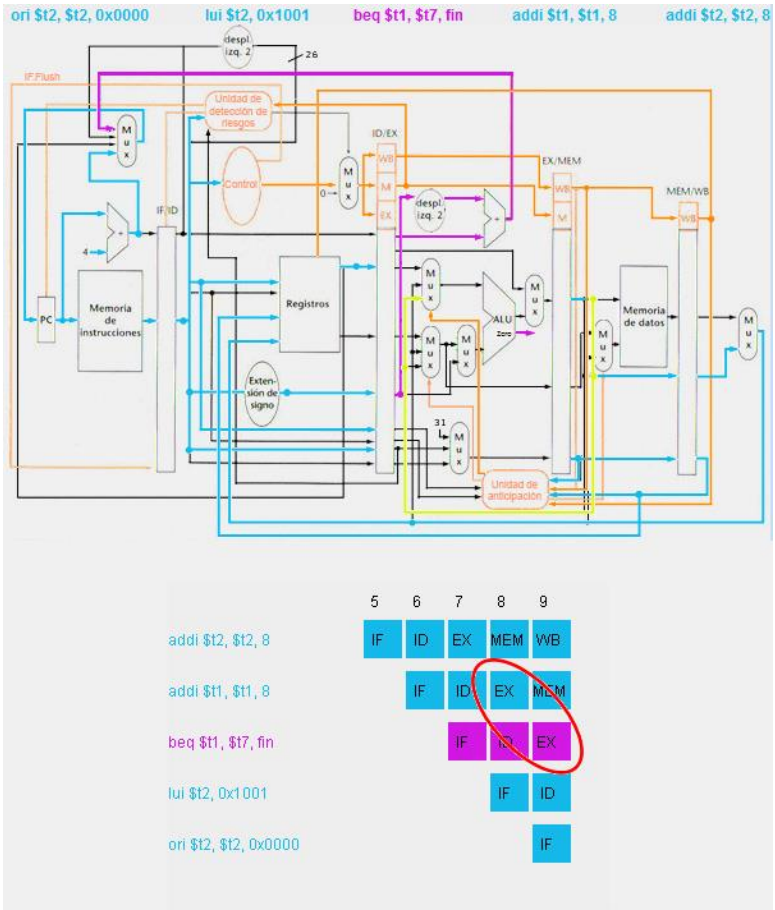

Figura 4.11. Camino de datos para el salto con dos huecos de retardo y con dependencia de una instrucción aritmética anterior.

4.2.3. Salto con tres huecos de retardo.

Para este caso se sabe si el salto es o no tomado en la etapa MEM. Para poder resolver el salto se necesita conocer el valor de los registros $t1 y $t7. Como el valor de $t1 se calcula en la instrucción justo delante del salto, en este caso no habría ningún ciclo de espera, ya que el dato puede adelantarse desde la etapa MEM de la instrucción `addi` hasta la etapa EX del salto, que entonces ya puede resolverse (figura 4.12).

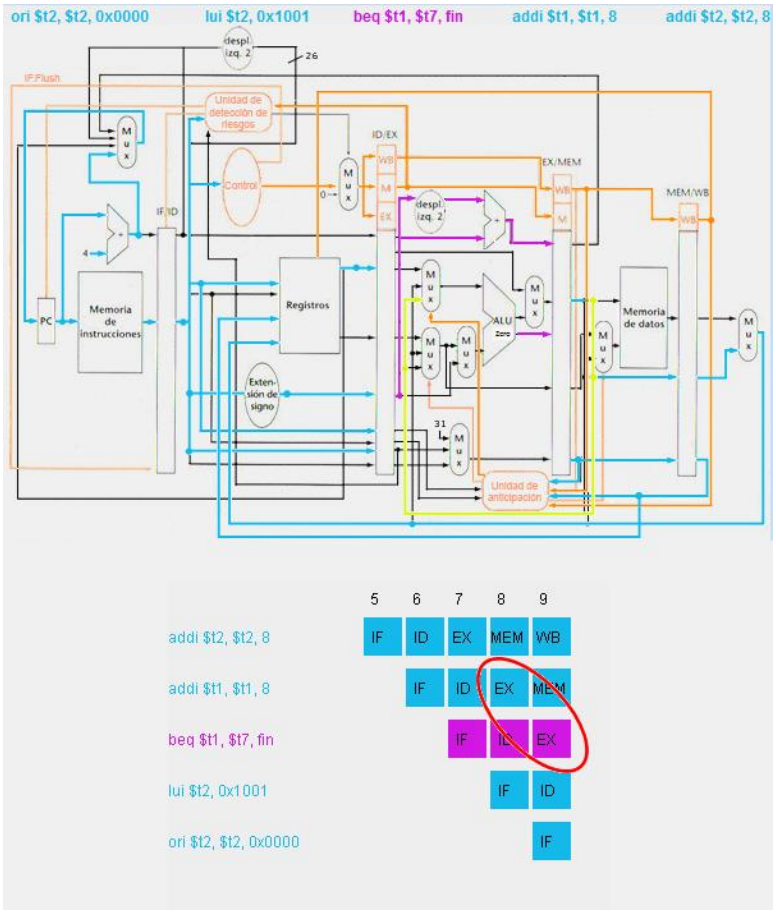

Figura 4. 12. Camino de datos para el salto con tres huecos de retardo y con dependencia de una instrucción aritmética anterior.

Hay que señalar que si la instrucción aritmética estuviera situada a distancia mayor, en vez de estar situada inmediatamente antes del salto, esto no afectaría a las tres implementaciones disponibles. En concreto, si la instrucción aritmética estuviera situada a distancia dos, no habría ningún ciclo de espera entre las dos instrucciones en ninguno de los tres casos. En cualquier caso, el comportamiento sería similar al mostrado en estas secciones y por tanto consideramos que reiterarlo alargaría las explicaciones del trabajo innecesariamente.

Podemos comprobar en este caso que la instrucción de salto funciona según lo deseado, esperando los ciclos necesarios en cada caso para poder calcular la dirección del salto con el dato que necesita de la instrucción aritmética anterior, de acuerdo con la teoría estudiada.

4.3 SALTO CON DEPENDENCIA DE UNA INSTRUCCIÓN DE CARGA ANTERIOR.

Para este ejemplo se usará el código *prueba salto tras lw.s***,** que se encuentra disponible en el CD que acompaña a esta memoria. Este es un código sencillo para probar cómo funciona ahora Simula3MS cuando la instrucción de salto (`beq` en este caso) tiene una dependencia con una instrucción `lw` anterior. Este código puede observarse en la figura 4.13.

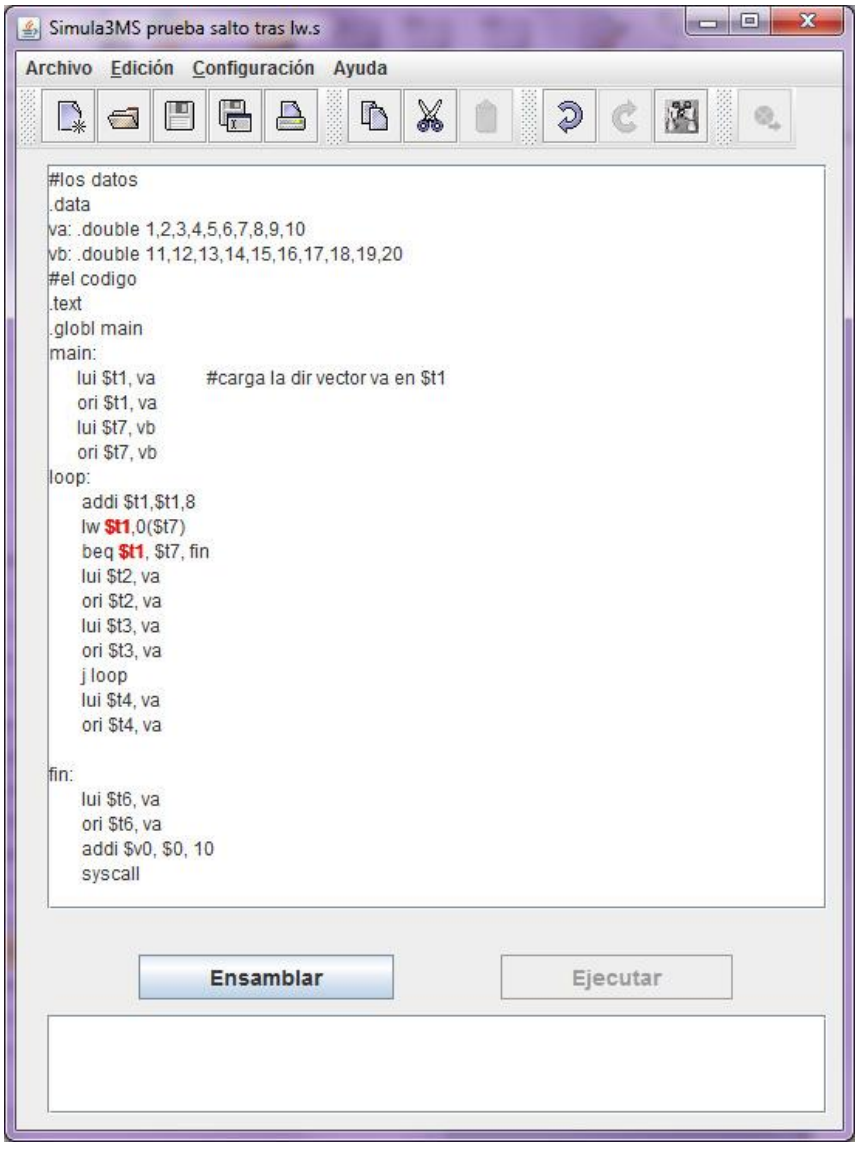

Figura 4.13. Código utilizado para probar el comportamiento del salto con dependencia de una instrucción de carga.

En este código se introduce un dato en el registro $t7, e iremos incrementando el valor de $t1 hasta que coincida con el valor de $t7, momento en el que finalizarán el bucle y el programa.

Con este ejemplo se quiere comprobar cómo afectaría a cada tipo de salto la dependencia con una instrucción de carga anterior que se resuelve en la etapa WB, momento en el que estará disponible el dato a la salida de la memoria de datos y por tanto podrá adelantarse.

Hay que señalar que para las tres implementaciones del salto disponibles, el comportamiento no variará si la estrategia utilizada es "Salto retardado" o "Predecir no tomado". Esto se debe a que el problema es un riesgo de datos anterior a la resolución del salto y por tanto es independiente de estrategia utilizada. Por tanto, y en beneficio de la claridad de esta sección, en cada implementación sólo se muestra el comportamiento de Simula3MS para la opción de "Salto retardado".

Una vez cargado el código tendremos que seleccionar el tipo de salto elegido desde el menú configuración, tal y como vimos en el apartado 4.1 en la figura 4.2.

Veamos cómo se va comportando la nueva versión del simulador para los distintos tipos de salto ahora disponibles.

4.3.1. Salto con un hueco de retardo.

Para este caso se sabe si el salto es o no tomado en la etapa ID. Para poder resolver el salto se necesita conocer el valor de los registros $t1 y $t7. Como el valor de $t1 se calcula en la instrucción justo delante del salto, habrá que esperar dos ciclos hasta que el valor de $t1 esté disponible. Esto será cuando la instrucción `lw` esté en la etapa WB, momento en el que se podrá adelantar el valor al comparador en la etapa ID para poder resolver el salto (figura 4.14).

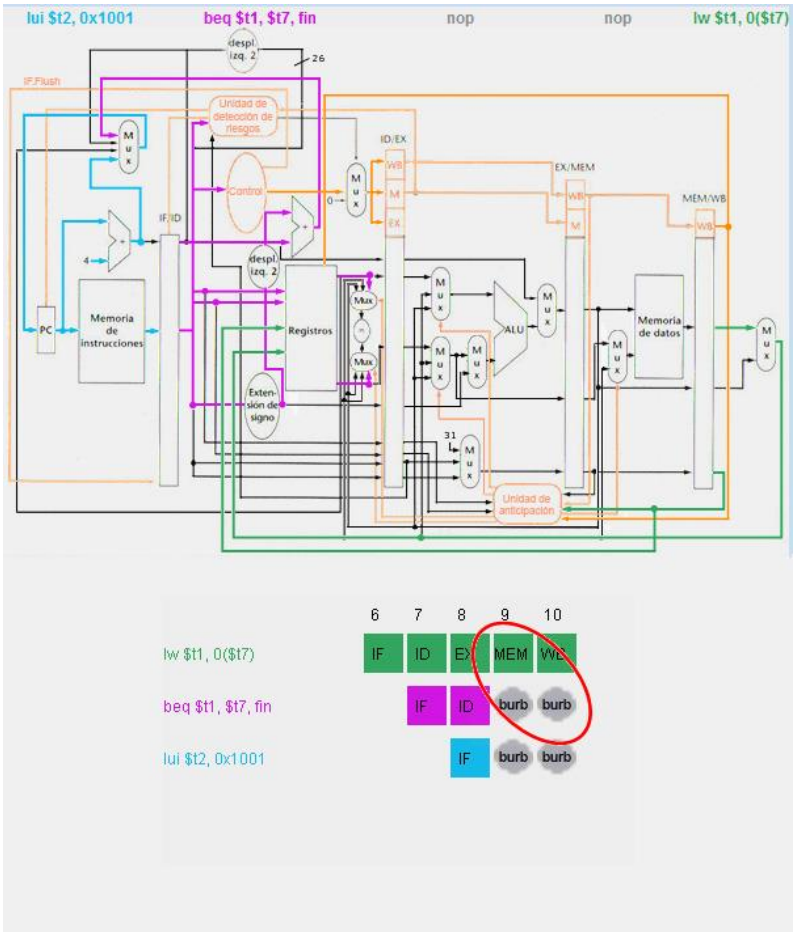

Figura 4.14. Camino de datos para salto con un hueco de retardo y con dependencia de una instrucción de carga anterior.

4.3.2. Salto con dos huecos de retardo.

Para este caso se sabe si el salto es o no tomado en la etapa EX. Para poder resolver el salto se necesita conocer el valor de los registros $t1 y $t7. Como el valor de $t1 se calcula en la instrucción justo delante del salto, en este caso habría un ciclo de espera hasta que el valor de $t1 esté disponible. Esto será cuando la instrucción `lw` esté en la etapa WB, momento en el que se podrá adelantar el valor a la unidad aritmético lógica en la etapa EX para poder resolver el salto (figura 4.15).

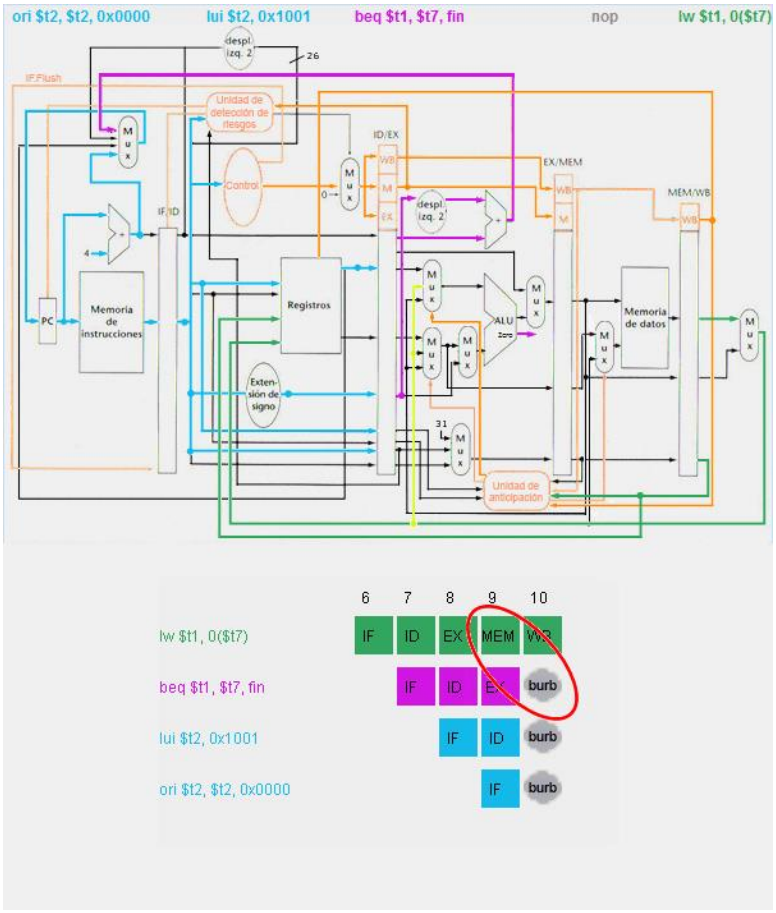

Figura 4.15.Camino de datos para el salto con dos huecos de retardo y con dependencia de una instrucción de carga anterior.

4.3.3. Salto con tres huecos de retardo.

Para este caso se sabe si el salto es o no tomado en la etapa MEM, pero se calcula en la etapa EX. Para poder resolver el salto se necesita conocer el valor de los registros $t1 y $t7.Como el valor de $t1 se calcula en la instrucción justo delante del salto, en este caso habría un ciclo de espera hasta que el valor de $t1 esté disponible. Esto será cuando la instrucción `lw` esté en la etapa WB, momento en el que se podrá adelantar el valor a la unidad aritmético lógica en la etapa EX para poder resolver el salto (figura 4.16).

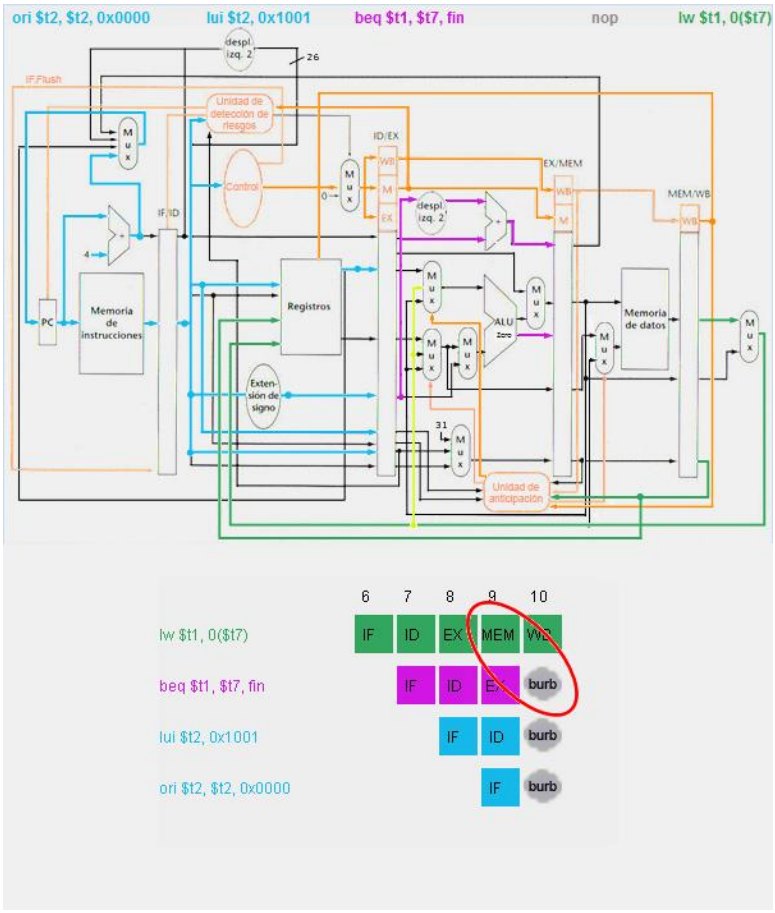

Figura 4.16. Camino de datos para el salto con tres huecos de retardo y con dependencia de una instrucción de carga anterior.

Hay que señalar que si la instrucción de carga estuviera situada a distancia mayor, en vez de estar situada inmediatamente delante del salto, esto no afectaría a las tres implementaciones disponibles. En concreto, si estuviera situada a distancia dos, habría un ciclo de espera entre las dos instrucciones en caso de un hueco de retardo, y ninguno en los otros dos casos. En cualquier caso, el comportamiento sería similar al mostrado en estas secciones y consideramos que no vale la pena reiterarlo.

Podemos comprobar en este caso que la instrucción de salto funciona según lo deseado, esperando los ciclos necesarios para poder calcular la dirección del salto con el dato que necesita de la instrucción de carga anterior, de acuerdo con la teoría estudiada.

4.4 CONCLUSIONES.

Como hemos mostrado en este capítulo, tras las pruebas realizadas para los casos de dos y tres huecos de retardo, la nueva versión de Simula3MS presenta el comportamiento esperado en todos los casos, por lo que consideramos que el código añadido a la versión original cumple con los objetivos planteados para este TFG.

Cabe destacar que los códigos mostrados a lo largo de este capítulo no son los únicos con los que se ha probado la nueva versión del simulador, obteniéndose siempre para cada caso los resultados esperados.

CAPÍTULO 5. CONCLUSIONES Y TRABAJO FUTURO

En este capítulo se comentan las contribuciones principales que ha supuesto este trabajo fin de grado. También se describe el trabajo futuro que se puede realizar para seguir ampliando las funcionalidades del simulador de cara a obtener una herramienta aún más completa.

5.1 CONCLUSIONES

Una vez cumplidos los objetivos planteados en el capítulo 1 del documento se ha obtenido un simulador ampliado con cuatro nuevas estrategias para el tratamiento de los saltos. Como se ha explicado en el capítulo 4, todas las opciones añadidas funcionan según lo esperado, es decir, de acuerdo con la teoría. Esta herramienta ampliada servirá para su uso en la docencia de diversas asignaturas que tratan sobre la arquitectura y organización de computadores.

Ha resultado complicado y ha requerido un gran esfuerzo asimilar toda la documentación requerida para el desarrollo del proyecto, así como analizar el código del simulador Simula3MS (figura 5.1), y sobre todo modificarlo para implementar las nuevas funcionalidades.

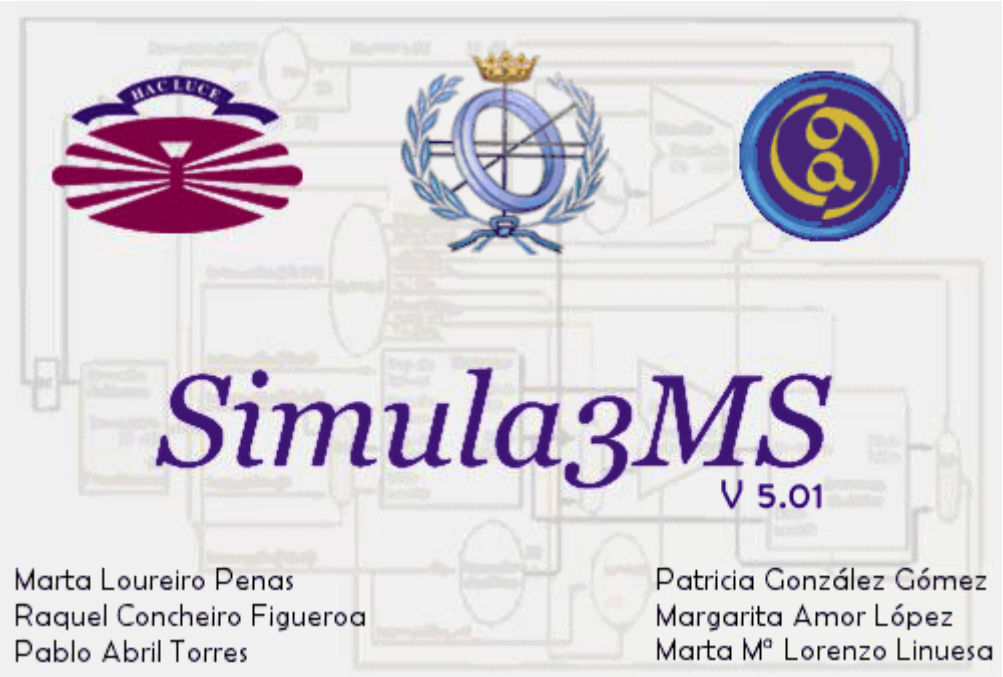

Figura 5.1. Nueva ventana de bienvenida del simulador Simula3MS.

5.2 TRABAJO FUTURO Y POSIBLES AMPLIACIONES

Para finalizar con este último capítulo, comentaremos algunas de las propuestas que se podrían añadir al trabajo actual en trabajos futuros, según lo que hemos descubierto al analizar el funcionamiento del simulador. Concretamente, el simulador carece de una serie de las siguientes funcionalidades que no se han desarrollado en este proyecto por situarse fuera del ámbito de los objetivos planteados para el mismo:

- Permitir elegir la codificación de la visualización de los datos; hexadecimal, binario, decimal, etc.
- Permitir configurar el cauce con o sin adelantamiento.
- Permitir usar técnicas especulativas para la ejecución de instrucciones fuera de orden.

Todas estas funcionalidades, como otras que fuera interesante incluir, deberían ser desarrolladas en otros trabajos fin de grado o por otros grupos de trabajo, y ser recogidos en la misma aplicación para conseguir un simulador más completo, de cuyo uso podrían beneficiarse los alumnos de cara al estudio de la arquitectura MIPS y los diferentes caminos de datos.

BIBLIOGRAFÍA

[1] Bruce Eckel. Piensa en Java. Prentice-Hall, 2000.

[2] Cay S. Horstmann and Gary Cornell. Java 2. Características avanzadas. Prentice Hall, 2003.

[3] Cay S. Horstmann and Gary Cornell. Java 2. Fundamentos. Prentice Hall, 2003.

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

[5] Dosier para la obtención del diploma de estudios avanzados. Marta Loureiro Penas. http://www.dec.usc.es/file/Tercer_ciclo/dossierDEA_MartaLoureiro.pdf

[6] Fermín Sánchez. Características deseables en un procesador pedagógico para la enseñanza básica de la arquitectura de computadores. Jornadas de Enseñanza Universitaria de la Informática (JENUI), 2002.

[7] Miguel A. Vega and Juan A. Gómez Juan M. Sánchez. Innovación docente en la Arquitectura de Computadores mediante el uso de Simuladores. Jornadas de Enseñanza Universitaria de la Informática (JENUI), 2000.

[8] Robert L. Britton. MIPS Assembly Language Programming. Prentice-Hall, 2004.

[9] Simula3MS. http://simula3ms.des.udc.es/

[10] William Stalling. Organización y estructura de computadores. Prentice Hall, 2000.

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 simulador modificado dentro del directorio Código fuente.
- Nueva versión del simulador MIPS Simula3MS modificado en este trabajo y los tres códigos que se utilizan como ejemplos en el desarrollo del documento (*prueba salto bne.s*, *prueba salto tras addi.s*, *prueba salto tras lw.s*) dentro del directorio Simulador.

ANEXO A. CÓDIGO FUENTE

En este apartado se muestra todo el 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.

A.1 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","di
v",

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

"mul.d","div.d","bc1t","bc1f","abs.s","abs.d","neg.s","neg.d","mov.s","mo
v.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;


    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;
}
```

A.2 ENSAMBLADOR.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();

        grupoBotonesAdelantamiento = new javax.swing.ButtonGroup();


        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();

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


        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.gi
f")));
        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.gi
f")));
        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.gi
f")));
        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.gi
f")));
        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.g
if")));
        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.gi
f")));
        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.gi
f")));
        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.g
if")));
        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.even
t.KeyEvent.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.even
t.KeyEvent.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.ev
ent.KeyEvent.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.aw
t.event.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("ItemGuar
darComo");

        MenuArchivo.add(jSeparator2);


ItemImprimir.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.e
vent.KeyEvent.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("ItemImprimi
r");

        MenuArchivo.add(jSeparator3);


ItemSalir.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.even
t.KeyEvent.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.e
vent.KeyEvent.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.ev
ent.KeyEvent.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(jav
a.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("Seleccionar
Todo");

ItemSeleccionarTodo.getAccessibleContext().setAccessibleDescription("Item
SeleccionarTodo");

        MenuEdicion.add(jSeparator4);


ItemCopiar.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.eve
nt.KeyEvent.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.eve
nt.KeyEvent.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.even
t.KeyEvent.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.even
t.KeyEvent.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.even
t.KeyEvent.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("ItemReemp
lazar");

        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);

        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);
```

```
        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);

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("saltoRetard
ado"));
  itemSaltoRetardadoFlotante2Huecos.addActionListener(new
java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
          itemSaltoRetardadoFlotante2HuecosActionPerformed(evt);
      }
  });
  grupoBotonesSalto3Huecos.add(itemSaltoRetardadoFlotante3Huecos);


itemSaltoRetardadoFlotante3Huecos.setText(lenguaje.getString("saltoRetard
ado"));
  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.e
vent.KeyEvent.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("itemAcercaD
e");

        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("BarraMenuC
omp");

        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.ActionEve
nt 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.ActionEve
nt 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

    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);
    }


/**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);
      }
      if(itemInterrupciones.isSelected() &&
(itemMonociclo.isSelected())){
            ensambla.analizaDato("s1: .word 0");
            ensambla.analizaDato("s2: .word 0");
            new VistaMonoESInterrup(this).setVisible(true);
      }
      if (itemMonociclo.isSelected() && (itemDesactivada.isSelected())){
            new VistaMonociclo(this).setVisible(true);
      }

      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_RETARDADO_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_RETARDADO_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(
).getParent().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;

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




}
```

A.3 PIPELINEFLOTANTE.JAVA

```
/**Pipeline.java
 *Clase que almacena las instrucciones que se estan ejecutando en cada
momento en un procesador segmentado
 **/
public class Pipeline implements Tipos{

    private Instruccion instrucciones[] = new Instruccion[5];
    private static Pipeline pipeline = new Pipeline();
    private Codigo codigo = Codigo.getInstacia();
    private UnidadAnticipacion unidadAnticipacion =
UnidadAnticipacion.getInstancia();
    private UnidadDeteccionRiesgos unidadDeteccionRiesgos =
UnidadDeteccionRiesgos.getInstancia();
    private int salto;
    private boolean saltoTomado;
    private int avance = 0;
    private Vector diagramaMult[];
    private Vector instMult;
    private Vector etapa;


/**Constructor de Pipeline
 *@param Sin parametros
 **/
    private Pipeline() {};

/**Metodo statico que devuelve la unica instancia de esta clase
 *@param Sin parametros
 *@return Pipeline pipeline
 **/
    public static Pipeline getInstacia()
    {
        return pipeline;
    }

/**Metodo que inicializa la unica instancia del pipeline
 *@param Sin parametros
 *@return void
 **/
```

```
    public void inicializar(int salto)      //salto_retardado-->0,
salto_fijo-->1
    {
        this.salto = salto;
        instrucciones=new Instruccion[5];
        for(int i=0; i<5; i++)
            instrucciones[i] = new Nop();

        diagramaMult=new Vector[5];
        for(int i=0;i<5;i++)
            diagramaMult[i]=new Vector();

        instMult=new Vector();
        etapa=new Vector();
    }

    public Instruccion[] getInstrucciones()
    {
        return instrucciones;
    }

/**Metodo que devuelve la instruccion de una posicion del pipeline
 *@param int posicion
 *@return Instruccion instruccion
 **/
    public Instruccion getInstruccion(int pos)
    {
        return instrucciones[pos];
    }

/**Metodo que indica la posicion de una instruccion en el pipeline dada
su direccion
 *@param String direccion
 *@return int posicion
 **/
    public int posPipeline(String dir)
    {
        int dirIns=(int)(new Palabra(dir,true)).getDec();
        for (int i=0; i<instrucciones.length;i++)
        {
            if ((codigo.getPcInst(instrucciones[i]))==dirIns)
                return i;
        }
        return -1;
    }

    //con salto fijo, si se toma el salto, se activa IF_Flush
    public boolean activarIFFlush(){
        if((salto==SALTO_FIJO) && (saltoTomado))
            return true;
        return false;
    }

/**Metodo para avanzar el pipeline
 *@param Instruccion instruccion que se anhade al pipeline
 *@return int tipo de avance
 **/
    public int avanzarPipeline(Instruccion inst, boolean saltoTomado)
    {
        //int numEtapa;
        avance = 0;
```

```
        this.saltoTomado=saltoTomado;

        //con salto fijo, si se toma el salto, se elimina la inst del
hueco de retardo
        if(activarIFFlush())
            instrucciones[0]=new Nop();

        if (!(inst.toString().equals("syscall")))
        {
            if (!unidadDeteccionRiesgos.detener()) {
                for (int i=4; i>0; i--) {
                    instrucciones[i] = instrucciones[i-1];
                }
                instrucciones[0] = inst;
                avance = 0;       //avance=0-->avance de todo el pipeline
            }
            else {
                for (int i=4; i>2; i--) {
                    instrucciones[i] = instrucciones[i-1];
                }
                instrucciones[2] = new Nop();
                avance = 1;         //avance=1-->se inserta burbuja en
etapa EX
            }
        }
        else
        {
            if (!estaParado()) {

                if (!unidadDeteccionRiesgos.detener()) {
                    for (int i=4; i>0; i--) {
                        instrucciones[i] = instrucciones[i-1];
                    }
                    instrucciones[0] = new Nop();
                    Pc.getRegistro().incrementar(-4);       //para
detener el pipeline antes de ejecutar SYSCALL
                    Pc.getRegistro().visualizaCambios();
                    avance = 2;      //avance=2-->se inserta burbuja en
etapa IF
                }
                else {
                    for (int i=4; i>2 ;i--) {
                        instrucciones[i] = instrucciones[i-1];
                    }
                    instrucciones[2] = new Nop();
                    avance = 1;         //avance=1-->se inserta burbuja
en etapa EX
                }
            }
            else {
                for(int i=4; i>0 ;i--) {
                    instrucciones[i]=instrucciones[i-1];
                }
                instrucciones[0]=new Nop();
                avance = 3;         //avance=3-->ejcucion de SYSCALL
            }

        }
        construirDiagMult(inst);
        return avance;
    }
```

```
/**Metodo que indica si el pipeline esta parado
 *@param Sin parametros
 *@return boolean
 **/
    public boolean estaParado()
    {
        int i;
        for(i=0;i<4;i++)
            if (!(instrucciones[i].toString().equals("nop")))
                break;
        if (i!=4)
            return false;
        return true;
    }

/**Metodo que añade una etapa para facilitar la representacion del
diagrama multiciclo
 *@param int indice
 *@return String etapa
 **/
    public String anhadirEtapa(int i)
    {
        switch(i)
        {
            case 0:
                return "burb";
            case 1:
                return "  IF";
            case 2:
                return "  ID";
            case 3:
                return " EX";
            case 4:
                return "MEM";
            case 5:
                return " WB";
            default:
                return "";
            /*case 6:
                return "";
            default:
                return "ERROR"; */
        }
    }

    public void construirDiagMult(Instruccion inst)
    {
        int numEtapa=0;

        if(activarIFFlush())
        {
            instMult.set((instMult.size()-1), new Nop());
        }

        switch(avance)
        {
            case 0: //avance=0-->avance de todo el pipeline
                if(instMult.size()==5){
                    instMult.remove(0);
```

```
                    etapa.remove(0);
                    for(int i=0; i<instMult.size(); i++)
                        diagramaMult[i]=diagramaMult[i+1];
                    diagramaMult[instMult.size()]=new Vector();
                }

                for(int i=0; i<instMult.size(); i++)
                {
                    if(!((Instruccion)instMult.elementAt(i) instanceof
Syscall))
                    {
                    numEtapa=((Integer)etapa.elementAt(i)).intValue()+1;
                    if(numEtapa<6){
                    etapa.set(i, new Integer(numEtapa));
                    diagramaMult[i].addElement(anhadirEtapa(numEtapa));
                    }
                    }
                }

                instMult.add(inst);
                etapa.add(new Integer(1));
                diagramaMult[instMult.size()-
1].addElement(anhadirEtapa(1));

                break;

            case 1: //avance=1-->se inserta burbuja en etapa EX
                for(int i=instMult.size()-1; i>(instMult.size()-3); i--)
                {
                    if(!((Instruccion)instMult.elementAt(i) instanceof
Syscall))
                    {

                    diagramaMult[i].addElement(anhadirEtapa(0));
                    }
                }

                for(int i=instMult.size()-3; i>=0; i--)
                {
                    if(!((Instruccion)instMult.elementAt(i) instanceof
Syscall))
                    {
                    numEtapa=((Integer)etapa.elementAt(i)).intValue()+1;
                    if(numEtapa<6){
                    etapa.set(i, new Integer(numEtapa));
                    diagramaMult[i].addElement(anhadirEtapa(numEtapa));
                    }
                    }
                }
                break;

            case 2: //avance=2-->se inserta burbuja en etapa IF
                for(int i=0; i<instMult.size(); i++)
                {
                    if(!((Instruccion)instMult.elementAt(i) instanceof
Syscall))
                    {
                    numEtapa=((Integer)etapa.elementAt(i)).intValue()+1;
                    if(numEtapa<6){
                    etapa.set(i, new Integer(numEtapa));
                    diagramaMult[i].addElement(anhadirEtapa(numEtapa));
```

```
                    }
                    }
                }
                break;

            case 3: //avance=3-->ejcucion de SYSCALL
                if(instMult.size()==5){
                    instMult.remove(0);
                    etapa.remove(0);
                    for(int i=0; i<instMult.size(); i++)
                        diagramaMult[i]=diagramaMult[i+1];
                    diagramaMult[instMult.size()]=new Vector();
                }

                instMult.add(inst);
                etapa.add(new Integer(1));

                for(int i=0; i<instMult.size(); i++)
                {
                    if(!((Instruccion)instMult.elementAt(i) instanceof
Syscall))
                    {
                    numEtapa=((Integer)etapa.elementAt(i)).intValue()+1;
                    if(numEtapa<6){
                    etapa.set(i, new Integer(numEtapa));
                    diagramaMult[i].addElement(anhadirEtapa(numEtapa));
                    }
                    }
                    diagramaMult[instMult.size()-
1].addElement(anhadirEtapa(6));
                }
                break;

            default:
                System.out.println("Error al construir el diagrama
multiciclo");

        }

    }

/**Metodo que devuelve informacion para dibujar el diagrama multiciclo
 *@param Sin parametros
 *@return Vector informacion
 **/
    public Vector[] getDiagramaMult()
    {
        return diagramaMult;
    }

/**Metodo que devuelve informacion para dibujar el diagrama monociclo
 *@param Sin parametros
 *@return Vector informacion
 **/
    public Vector getInstMult()
    {
        return instMult;
    }

     }
```

A.4 BNE.JAVA

```
/**Bne.java
  **/

public class Bne extends SaltoCondicional {

    Registro r[];
    String etiqueta;
    private int posIns;
    int salto;

     public Bne(String param[],int p, int salto)
    {
       esSalto=true;
          posIns=(int)INICIO_PC.getDec()+(p)*4+4;
          this.salto=salto;
        int posi=-1;
        inicia();
        r = new Registro[numParam()];
        string.append("bne ");
        for(int i=0; i<numParam()-1;i++)
        {
            string.append(param[i]);
            if(i!=numParam()-1)
                string.append(", ");
          for(int j=0;j<registros.length;j++)
            {
                try
                {

posi=java.lang.Integer.parseInt(param[i].substring(1,param[i].length()));
                }
                catch(NumberFormatException e)
                {
                    r[i]=relacionaReg(param[i]);
                        break;
                }
                r[i]=relacionaReg(posi);

            }
        }
        etiqueta=param[2];
        string.append(etiqueta);
    }

    public int numero()
    {
        return 11;
    }


    public int ejecutar()
    {

if(!r[0].getPalabra().getHex().equals(r[1].getPalabra().getHex()))
        {
            int i=(int)Etiquetas.getEtiquetas().getDireccion(etiqueta);
            Pc.getRegistro().modificar(i);
            boolean_pc=true;
        }
```

```
        else
            Pc.getRegistro().incrementar(4);
        boolean_pc=true;
        this.cambiosVisibles();
        return -1;
    }

    public int etapa3()
    {
        imagen_etapa=new StringBuffer();

if(!r[0].getPalabra().getHex().equals(r[1].getPalabra().getHex()))
        {
            int i=(int)Etiquetas.getEtiquetas().getDireccion(etiqueta);
            Pc.getRegistro().modificar(i);
            boolean_pc=true;
             imagen_etapa.append(direccion_multi).append("bne");
        }
        else
             imagen_etapa.append(direccion_multi).append("bne_no");

        return -1;
    }

    public String getCodificacion()
    {
       int posSalto=(int)Etiquetas.getEtiquetas().getDireccion(etiqueta);
       int salto=(posSalto-posIns)/4;
        int offset=(salto);
        String codigo;

codigo=setOp(5)+setR(r[0].numReg())+setR(r[1].numReg())+setIn(offset);
        return bin_a_hex(codigo);
    }

    public Registro regModificable()
    {
        return null;
    }

    public Registro getRegRS()
    {
        return r[0];
    }

    public Registro getRegRT()
    {
        return r[1];
    }

    public boolean hayRiesgo()
    {
      Instruccion instEX, instMEM;
      instMEM =
MEM.getInstancia().getEstadoInstrucciones()[0].getInstruccion();
      instEX = EX.getInstancia().getEstadoInstEntera().getInstruccion();


      if(salto == SALTO_RETARDAD0 || salto == SALTO_FIJO){

            if(r[0].estaLibre() && r[1].estaLibre())
```

```
                return false;

            else if(((instEX.regModificable() == r[0])&& (instEX
instanceof Carga))||((instMEM.regModificable() == r[0])&& (instMEM
instanceof Carga)))
                return true;

            else if((instEX.regModificable()==r[0])&&(instEX instanceof
Inmediatas))
                return true;

            else if((instEX.regModificable()==r[1])&&(instEX instanceof
Inmediatas))
                return true;

            return false;
      }
        else if ((salto == SALTO_FIJO_2HUECOS)||(salto ==
SALTO_FIJO_3HUECOS)||(salto == SALTO_RETARDAD0_2HUECOS)||(salto ==
SALTO_RETARDAD0_3HUECOS)){
            return false;
      }
      return false;


    }

    public int ejecutarIF()
    {
        Pc.getRegistro().incrementar(4);
        boolean_pc=true;
        this.cambiosVisibles();
        return -1;
    }

    public int ejecutarID()
    {

      if ((salto == SALTO_FIJO_2HUECOS)||(salto ==
SALTO_FIJO_3HUECOS)||(salto == SALTO_RETARDAD0_2HUECOS)||(salto ==
SALTO_RETARDAD0_3HUECOS)){
            Pc.getRegistro().incrementar(4);
                boolean_pc=true;
      }
      else
      {

            saltoTomado=false;
//          if(hayRiesgo())
//                  return -1;

      if(!r[0].getPalabra().getHex().equals(r[1].getPalabra().getHex()))
            {

                int
i=(int)Etiquetas.getEtiquetas().getDireccion(etiqueta);
                Pc.getRegistro().modificar(i);
                boolean_pc=true;
                saltoTomado=true;
            }
            else
```

```
            {

                  Pc.getRegistro().incrementar(4);
                  boolean_pc=true;
            }
      }
      this.cambiosVisibles();
            return -1;
    }

    public int ejecutarEX()
    {
      if((salto == SALTO_FIJO_2HUECOS)||(salto ==
SALTO_RETARDAD0_2HUECOS))
      {
            saltoTomado=false;
//          if(hayRiesgo())
//                return -1;

      if(!r[0].getPalabra().getHex().equals(r[1].getPalabra().getHex()))
            {

                  int
i=(int)Etiquetas.getEtiquetas().getDireccion(etiqueta);
                  Pc.getRegistro().modificar(i);
                  boolean_pc=true;
                  saltoTomado=true;
            }

      }
      this.cambiosVisibles();
      return -1;
    }

    public int ejecutarMEM()
    {
      if((salto == SALTO_FIJO_3HUECOS)||(salto ==
SALTO_RETARDAD0_3HUECOS))
      {
            saltoTomado=false;
//          if(hayRiesgo())
//                return -1;

      if(!r[0].getPalabra().getHex().equals(r[1].getPalabra().getHex()))
            {

                  int
i=(int)Etiquetas.getEtiquetas().getDireccion(etiqueta);
                  Pc.getRegistro().modificar(i);
                  boolean_pc=true;
                  saltoTomado=true;
            }

      }
      this.cambiosVisibles();
      return -1;
    }

    public int ejecutarWB()
    {
         return -1;
```

```
    }
    public int setSalto(int salto)
    {
        this.salto=salto;
      return -1;
    }

      @Override
      public int ejecutarIDHuecos() {
            // TODO Auto-generated method stub
            return 0;
      }
     }
```

A.5 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_R
T,imagenFlush, 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.ge
tInstancia().getString("rutaSegmentado")+"segmentado.gif"));
                  else
if((this.salto==SALTO_FIJO_2HUECOS)||(this.salto==SALTO_RETARDAD0_2HUECOS
))

      imSegmentado=(Image)ImageIO.read(getClass().getResource(Lenguaje.ge
tInstancia().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.ge
tInstancia().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.ge
tInstancia().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.ge
tInstancia().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.ge
tInstancia().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()
.getInstruccion().imagenSegmentado(0, salto,
adelantamiento).toLowerCase()));
            imagenID=(Image)ImageIO.read(getClass().getResource(

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

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

      "/ensamblador/segmentado/"+MEM.getInstancia().getEstadoInstruccione
```

```
s()[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/seg
mentado/riesgo.gif"));
            else

      if(ID.getInstancia().getEstadoInstruccion().getInstruccion().toStri
ng().equals("nop"))

      imagenRiesgo=(Image)ImageIO.read(getClass().getResource("/ensamblad
or/segmentado/nop.gif"));
                  else

      imagenRiesgo=(Image)ImageIO.read(getClass().getResource("/ensamblad
or/segmentado/no_riesgo.gif"));

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


            if (ant[ANT_RS_MEM])

imagenAntMEM_RS=(Image)ImageIO.read(getClass().getResource(("/ensamblador
/segmentado/antMEM_RS.gif").toLowerCase()));
            else

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

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

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

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

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

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

imagenAntWB_RT=(Image)ImageIO.read(getClass().getResource("/ensamblador/s
egmentado/nop.gif"));
```

```
            if(pipeline.activarIFFlush())

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

imagenFlush=(Image)ImageIO.read(getClass().getResource("/ensamblador/segm
entado/nop.gif"));
            if ((ant[ANT_RS_EX]) && (adelantamiento ==
CON_ADELANTAMIENTO))

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

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

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

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

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

            if (ant[ANT_WB])

imagenAntWB=(Image)ImageIO.read(getClass().getResource(("/ensamblador/seg
mentado/antWB.gif").toLowerCase()));
            else

imagenAntWB=(Image)ImageIO.read(getClass().getResource("/ensamblador/segm
entado/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);

      }


    }
```

A.6 SALTOCONDICIONAL.JAVA

```
/**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
","beqMEM.gif","beqWB.gif"};
 private String
imagSegmentado3Huecos[]={"beqIF.gif","beqID2Huecos.gif","beqEX3Huecos.gif
","beqMEM3Huecos.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",
"beqEX2Huecos.gif","beqMEM sin adelantamiento.gif","beqWB sin
adelantamiento.gif"};
 private String
imagSegmentado3HuecosSinAdelantamiento[]={"beqIF.gif","beqID2Huecos.gif",
"beqEX3Huecos.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();
     }
```