Inter*net* Electronic Journal*

Nanociencia et Moletrónica

Mayo 2007, Vol. 5, N°1, pp. 953-964

# Análisis de riesgos de depencia de datos en arquitecturas de procesadores segmentados

**M. Bustillo-Díaz, A. Rangel-Huerta, A. Toxqui, V. Feodorovich-Melekhin***

Facultad de Ciencias de la Computación, BUAP,
14 sur y av. San claudio, Col. San Manuel C.p 72570, Puebla, Puebla, México

*Universidad Politécnica Estatal de San Peterburgo, Facultad de Cibernética,
Catedra de Ciencias de la Computación,
Av. Politekhnisheskaya 29, San Peterburgo, Russia

recibido: 10.05.07 revisado: 17.05.07 publicado: 31.5.07

Citation of the article;
M. Bustillo-Díaz, A. Rangel-Huerta, A. Toxqui, V. Feodorovich-Melekhin, Análisis de riesgos de depencia de datos en arquitecturas de procesadores segmentados , Internet Electron. J. Nanoc. Moletrón. 2007, Vol. 5, N° 1, pp 953-964

# Análisis de riesgos de depencia de datos en arquitecturas de procesadores segmentados

**M. Bustillo-Díaz, A. Rangel-Huerta, A. Toxqui, V. Feodorovich-Melekhin***

Facultad de Ciencias de la Computación, BUAP,
14 sur y av. San claudio, Col. San Manuel C.p 72570, Puebla, Puebla, México

*Universidad Politécnica Estatal de San Peterburgo, Facultad de Cibernética,
Catedra de Ciencias de la Computación,
Av. Politekhnisheskaya 29, San Peterburgo, Russia

recibido: 10.05.07 revisado: 17.05.07 publicado: 31.5.07

*Internet Electron. J. Nanoc. Moletrón., Vol. 5, N° 1, pp.953-964*

**RESUMEN**

En el presente trabajo se desarrolla un método por hardware para la solución de riesgos de dependencia de datos utilizando técnicas de Candados hardware, adelantamiento de datos, inserción de burbujas, y anticipación de riesgos. Este método se aplica en la arquitectura RS-2000 desarrollado como prototipo en el laboratorio de "Arquitectura de Computadoras" de la FCC que consta de una segmentación de 5 etapas y cumple con las características de la arquitectura RISC.

El método de solución riesgos de dependencia de datos, analiza si la instrucción en la etapa de búsqueda depende del dato de alguna de las instrucciones en las etapas de decodificación, ejecución, o memoria. Cuando la instrucción en etapa de búsqueda pasa a la etapa de decodificación, si se detectó el riesgo, se adelanta el dato desde la etapa donde se encuentra la instrucción que se implica en la dependencia. Para el caso de que una instrucción dependa del dato de una instrucción de carga, se detiene la instrucción dependiente en la etapa de búsqueda para permitir que la instrucción de carga obtenga el dato de memoria y este sea adelantado cuando la instrucción pase a decodificación. En caso de que la instrucción dependiente sea de almacenamiento no se detiene, y se permite que lea el dato no válido, pero al pasar a la etapa de ejecución este sea cambiado por el dato anticipado de la etapa de memoria.

Palabras claves: riesgos, RISC, instrucción, segmentación, etapas

## 1. INTRODUCCIÓN

La arquitectura de computadora del procesador RS2000 de filosofía RISC [1], cuenta con características de 3 formatos de instrucción de tamaño 38 bits, 3 modos de direccionamiento, direccionamiento entre registros, inmediato e indirecto de registros, instrucciones de carga y almacenamiento para acceder a memoria de datos, y 5 etapas de segmentación, búsqueda decodificación, ejecución, memoria, y escritura. Además tiene registros entre las diferentes etapas de la segmentación. Con estas características de la arquitectura, se tratan por vía hardware los riesgos de la segmentación causados por dependencia de datos a partir de la decodificación de instrucciones.

El método Scoreboard [2] resuelve riesgos por dependencia de datos consiste en mantener información como tabla de estado de la instrucción, tabla de estados de las unidades funcionales y tablas de estado de los registros de propósito general. El método permite ejecución de instrucciones fuera de orden, pero en caso de dependencia de datos, la instrucción dependiente es detenida hasta que el registro del que requiere el operando este disponible. Otro método de análisis es el de Tomasulo [2], este provee de un bit de estado y una etiqueta a cada registro de propósito general. Además en la entrada de cada unidad funcional de ejecución se agregan registros que mantienen el operando de la instrucción que es emitida a la unidad de ejecución y un campo de etiqueta que indica la unidad funcional por la que espera un dato (en caso de dependencia de datos). Se tiene buses de intercomunicación entre las unidades funcionales y los registro de propósito general que permiten enviar simultáneamente el dato a las unidades funcionales y a los registros de propósito general. En métodos de solución por software se detecta la dependencia en tiempo de compilación, se realiza la reorganización de las instrucciones y se insertan instrucciones de no operación para resolver los riesgos por dependencia de datos [4,5,7,9]. El método por hardware que se presenta es para realizar una detección anticipada en la etapa de búsqueda y una vez que en la etapa de decodificación se determina si se adelanta el dato de las diferentes etapas o se detiene la instrucción dependiente en la etapa de búsqueda mientras una instrucción de carga obtiene el dato de memoria.

## 2. DESARROLLO

### 2.1 ANÁLISIS DE LA SEGMENTACIÓN.

Analizando el flujo de instrucciones en la segmentación se determina el momento en que se presenta el riesgo en la arquitectura RS2000. En la figura 2.1 se señala la presencia de riesgos en la segmentación.

En la etapa de búsqueda, la primera instrucción que entra en la segmentación para su ejecución es cargada en el registro AUX. En este momento las etapas restantes se encuentran vacías, por lo tanto no existe ningún riesgo. En el siguiente ciclo se obtiene la siguiente instrucción y la instrucción en búsqueda es transmitida a la etapa de decodificación, llegando una nueva instrucción a la etapa de búsqueda que se analiza en combinación con la instrucción en la etapa de decodificación para la detección de la dependencia de datos como se observa en la figura 2.1. La instrucción en la etapa de decodificación, lee el dato de sus registros fuentes de operandos, puesto que es la primera instrucción emitida a la segmentación que no depende del dato generado con ninguna instrucción. Entonces no existe riesgo de dependencia de datos pero se analiza si la

instrucción en etapa de búsqueda depende del dato generado por esta instrucción.
Al avanzar las instrucciones por la segmentación, la instrucción en la etapa de decodificación pasa a la etapa de ejecución, mientras que, la instrucción en la etapa de búsqueda pasa a la etapa de decodificación para que se generen las señales de lectura de sus registros fuente de operandos y entra una nueva instrucción en la etapa de búsqueda que se analiza en combinación con la instrucción en decodificación y ejecución para detectar dependencia de datos como lo señala la figura 2.1. Si la instrucción en la etapa de decodificación causara dependencia de datos con la instrucción en ejecución se estará leyendo un dato no válido, ya que la instrucción en ejecución actualizará el registro hasta la etapa de escritura por tal motivo también se analiza la instrucción en etapa de búsqueda con las instrucciones en las etapas de ejecución y memoria señalado en la figura 2.1.
Dependiendo del formato de la instrucción, en la etapa de búsqueda, el registro fuente de operando involucrado en el riesgo por dependencia de datos, podría ser el registro fuente de operando 1 o el registro fuente de operando 2. La tabla 2.1 muestra las combinaciones de los formatos de instrucción que pueden presentar riesgos en las etapas de búsqueda y de decodificación.

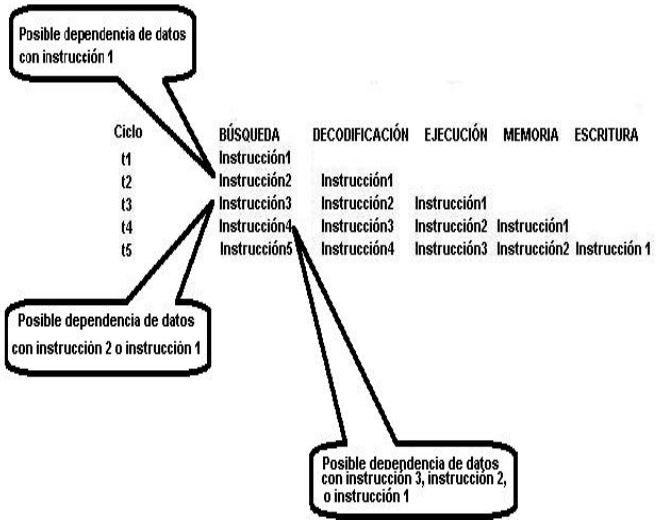


Figura 2.1 Segmentación con riesgos de la segmentación

| BÚSQUEDA | DECODIFICACIÓN |
|---|---|
| Formato 1 | Formato 1 |
| Formato 1 | Formato 2 |
| Formato 1 | Formato 3 |
| Formato 2 | Formato 1 |
| Formato 2 | Formato 2 |
| Formato 2 | Formato 3 |
| Formato 3 | Formato 1 |
| Formato 3 | Formato 2 |
| Formato 3 | Formato 3 |

Tabla 2.1 Combinación de formatos de Instrucción

La figura 2.2 muestra el flujo de instrucción teniendo como base el análisis de los riesgos de la segmentación. Los estados que pueden ser ejecutados son los siguientes:

T0 - inicio, T1 - cargar registro PC, T2 - cargar registro de instrucción auxiliar con la primera instrucción, incrementar el contenido del registro **PC**, decodificar la instrucción en el registro **AUX** para anticipar riesgos, T3 actualizar el registro **PC**, actualizar o poner a ceros el contenido del registro **AUX**, actualizar el registro de instrucción **IR**, decodificar el registro de instrucción **AUX** para anticipar riesgos, si es necesario, se leen los operandos del archivo de registros de propósito general. Determinar si se requiere, adelantamiento de datos, T4 en caso de dependencia de datos los registros **PC** y **AUX** no son actualizados y limpia el contenido del registro IR, se explora el registro **AUX** para determinar riesgos, si es requerido, se leen operandos del archivo de registros, se determina si se requiere adelantamiento de datos[14].

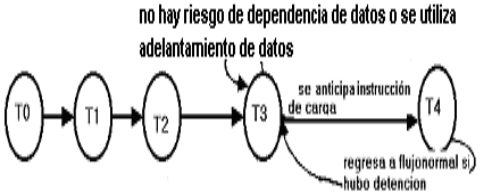


Figura 2.2 Diagrama de estados de flujo de instrucción

## 2.2 DETECCIÓN ANTICIPADA DE DEPENDENCIAS DE DATOS

Para solucionar una dependencia de datos realizamos una detección anticipada de la dependencia. Esto se logra haciendo la exploración de la instrucción en la etapa de búsqueda. La figura 2.3 muestra las señales generadas en esta etapa.

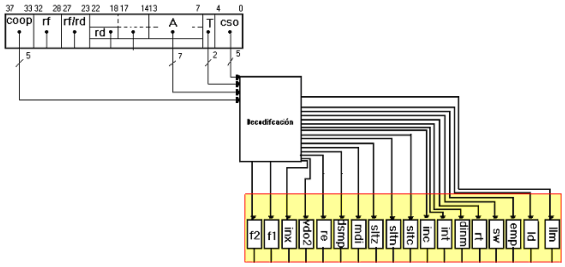


Figura 2.3 Señales de anticipación en la etapa de búsqueda

Las consideraciones que se tienen para tratar los riesgos de dependencia de datos son:
1. Si la instrucción en el registro de instrucción auxiliar indica registros fuente de operandos.
2. Si la instrucción que llega al registro de instrucción auxiliar es de carga.
3. Si entran a la segmentación instrucciones de carga y almacenamiento.

## 2.3 ANALISIS DE DETECCIÓN DE DEPENDENCIA DE DATOS

Para determinar los riesgos por dependencia de datos se analizan los campos de operandos de la instrucción en el registro **AUX** con el campo **rd** del registro **IR**, **DES1, DES2** de las etapas de decodificación, ejecución y memoria respectivamente.
El campo del registro destino **rd** de cada instrucción pasa por la segmentación a través de los registros **DES1, DES2** hasta que la instrucción sale de la segmentación.
Se observan que las condiciones para que aparezca una dependencia de datos son:

1. Si el campo indicador de registro fuente de operando 1 AUX(32-28), coincide con el campo del registro destino de operando IR(27-23) o IR(22-18) dependiendo del formato de instrucción en tal caso activa la señal **"do1"** (ver figura 2.4)**.**
2. Si el campo indicador de registro fuente de operando 2 AUX(27-23), coincide con el campo indicador de registro destino de operando IR(27-23) o IR(22-18) dependiendo del formato de instrucción en tal caso activa la señal **"do2"** (ver figura 2.4)**.**
3. Si el campo AUX(32-28) coincide con el campo **DES1** se activa la señal **"do1x"** o si el campo AUX(27-23) coincide con el campo **DES1** en tal caso se activa **"do2x "**( figura 2.4)**.**
4. Si AUX(32-28) coincide con **DES2** se activa la señal **"do1m"** o si AUX(27-23) coincide con **DES2** se activa **"do2m"**(ver figura 2.4).
5.

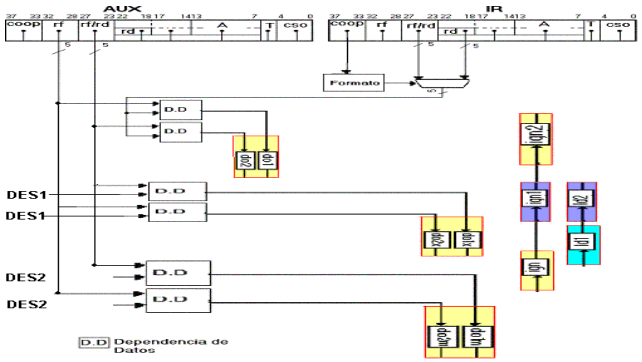


Figura 2.4 Dependencia de datos Búsqueda-Decodificación

La señal **"ign1"** se mantiene en la señal **"ign2"**, para indicar que en las etapas de ejecución y memoria transitan burbujas. La señal **"ld1"** indica que una instrucción de carga se encuentra en la etapa de ejecución, pero como la instrucción de carga obtiene su operando hasta la etapa de memoria esta señal debe mantenerse en **"ld2"** la cual permitirá elegir el dato proveniente de memoria. Estas señales se observan en la figura 2.5.

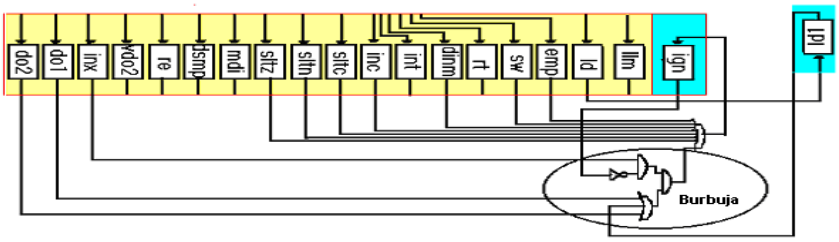


Figura 2.5 Representa inserción de burbujas

Otro tipo de riesgo es almacena-carga, si se presentan las señales **"ld1", "do1"** o **"do2"**o **"sw"**, se permitió que la instrucción de almacena continúe a la etapa de decodificación, pero como el dato se obtiene hasta que la instrucción llegue a la etapa de ejecución se debe anticipar, ya que se presento un riesgo del tipo almacena-carga y la instrucción de almacena debe tomar el dato anticipado de la etapa de memoria. Esto se hace manteniendo las señales que indican la presencia del riesgo de dependencia de datos en **"s_cdr",** si el operando 2 es el involucrado, y **"s_cdt",** si el operando 1 es el involucrado,

como se ve de la siguiente figura 2.6.

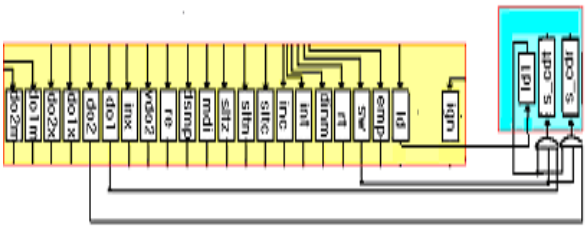


Figura 2.6 Señales de anticipación de riesgo del tipo almacena-carga

## 2.4 SOLUCION A RIESGOS POR DEPENDENCIA DE DATOS

Para la dependencia de datos cuando se anticipa una instrucción de carga es necesario analizar las señales de detección de riesgo. El valor de **“inx”=1** y **“ign”=0** indican que se analizaron fuentes válidos con el campo destino, se valido además **“ld1”** que indica que se involucra una instrucción de carga. Las señales **“do1”** o **“do2”** indican dependencia, por tanto la solución es detener la instrucción dependiente en la etapa de búsqueda e insertar burbuja en la etapa de decodificación. En la figura 2.7 se observa que las señales de anticipación generan las señales del control de flujo de instrucciones.

*Si ((inx='1') ^ (ign='0') ^ (ld1='1')) ===>*
*Si ((do1='1') o (do2='1')) ===>*
*SOLUCIÓN;*
*DETENER ETAPA DE BÚSQUEDA*
*INSERTAR BURBUJA EN DECODIFICACIÓN*
*EN CASO DE NO CUMPLIRSE LAS CONDICIONES ===>*
*SOLUCIÓN;*
*PERMITIR QUE INSTRUCCIÓN EN BÚSQUEDA FLUYA A DECODIFICACIÓN.*
*EXTRAER INSTRUCCIÓN SECUENCIAL.*

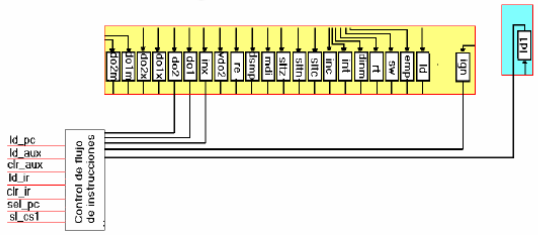


Figura 2.7 señales de control de flujo de instrucciones

Para dependencia de datos, que se resuelven utilizando adelantamientos (no son instrucciones de almacena ni carga), que se detectan en la etapa de búsqueda se resuelven con la decodificación de las señales **“do1”** y **“do2”** indicando el riesgo y por tanto el dato debe adelantarse a la etapa de decodificación. El algoritmo es el siguiente:

*PARA OPERANDO 1:*
*Si (inx =’1 ^ ign=’0’)===>*
*Si (do1=’1’) ===>*
*SOLUCIÓN;*
*LEER DATO PROVENIENTE DE ETAPA DE EJECUCIÓN*
*EN CASO DE NO CUMPLIRSE LAS CONDICIONES ===>*
*SOLUCIÓN;*
*DEPENDIENDO DE LA INSTRUCCIÓN LEER DATO DE ARCHIVO DE REGISTROS O DE PILA DE REGISTROS O INMEDIATO.*
*PARA OPERANDO 2:*
*Si (vdo2=’1’ ^ ign =’0’) ===>*
*Si (do2=’1’ )===>*
*SOLUCIÓN; LEER DATO PROVENIENTE DE EJECUCIÓN*
*EN CASO DE NO CUMPLIRSE LAS CONDICIONES ===>*
*SOLUCIÓN;*
*DEPENDIENDO DE LA INSTRUCCIÓN LEER DATO DE ARCHIVO DE REGISTROS O DE PILA DE REGISTROS O INMEDIATO.*

Cuando se detecta una dependencia de datos de una instrucción en la etapa de búsqueda y causa dependencia con la instrucción en la etapa de ejecución el dato debe de tomarse de la etapa de memoria. En la figura 2.7 se observa como las señales de anticipación se utilizan para controlar la anticipación de datos y se observa en el siguiente algoritmo:

*PARA OPERANDO 1:*
*Si (inx =’1^ ign=’0’^ ign1=’0’) ===>*
*Si (do1X=’1’) ===>*
*SOLUCIÓN;*
*LEER DATO PROVENIENTE DE ETAPA DE MEMORIA.*
*EN CASO DE NO CUMPLIRSE LAS CONDICIONES ===>*
*SOLUCIÓN;*
*DEPENDIENDO DE LA INSTRUCCIÓN LEER DATO DE ARCHIVO DE REGISTROS O DE PILA DE REGISTROS O INMEDIATO.*

*PARA OPERANDO 2:*
*Si se cumple (vdo2=’1’ ^ ign=’0’ ^ ign1=’0’) ===>*
*Si ( do2x=’1’ )===>*
*SOLUCIÓN:*
*LEER DATO PROVENIENTE DE MEMORIA*
*EN CASO DE NO CUMPLIRSE LAS CONDICIONES ===>*
*SOLUCIÓN;*
*DEPENDIENDO DE LA INSTRUCCIÓN LEER DATO DE ARCHIVO DE REGISTROS O DE PILA DE REGISTROS O INMEDIATO.*

Cuando se detecta una dependencia de datos, en donde una instrucción en la etapa de búsqueda causa dependencia con una instrucción en la etapa de memoria, el dato debe tomarse de la etapa de escritura. Esto se observa en el siguiente algoritmo:

*PARA OPERANDO 1:*
*Si (inx='1' ^ ign='0' ^ ign1='0' ^` ign2='0') ===>*
*Si (do1m='1') ===>*
*SOLUCIÓN;*
*LEER EL DATO PROVENIENTE DE LA ETAPA DE ESCRITURA*
*EN CASO DE NO CUMPLIRSE LAS CONDICIONES ===>*
*SOLUCIÓN;*
*DEPENDIENDO DE LA INSTRUCCIÓN LEER DATO DE ARCHIVO DE REGISTROS O DE PILA DE REGISTROS O INMEDIATO.*
*PARA OPERANDO 2:*
*Si (vdo2='1^ ign='0'^ ign1='0' ^ ign2='0') ===>*
*Si (do2M='1') ===>*
*SOLUCIÓN;*
*LEER OPERANDO PROVENIENTE DE ESCRITURA.*
*EN CASO DE NO CUMPLIRSE LAS CONDICIONES ===>*
*SOLUCIÓN;*
*DEPENDIENDO DE LA INSTRUCCIÓN LEER DATO DE ARCHIVO DE REGISTROS O DE PILA DE REGISTROS O INMEDIATO.*

En la figura 2.8 se observa como las señales de anticipación son utilizadas para determinar el control del adelantamiento de datos por medio de las señales **selop1y selop2**.
En esta etapa el dato generado por la unidad matemática es adelantado a la etapa de decodificación si es requerido. En la detección de un riesgo por dependencia de datos donde el tipo de instrucciones involucradas son almacena y carga, la solución es permitir que la instrucción de almacena continúe a la etapa de ejecución y cuando en la etapa de memoria el dato de la instrucción de carga sea recuperado de memoria este sea adelantado a la etapa de ejecución. Las señales **"s_cdt"** y **"s_cdr"** se utilizan para la elección del dato de memoria, este procedimiento se observa como el siguiente algoritmo:

*PARA OPERANDO 1:*
*Si s_cdt='1' ===>*
*AL CAMPO REG12 ASÍGNALE EL DATO PROVENIENTE DE MEMORIA. EN CASO SE NO CUMPLIRSE LA CONDICIÓN ===>*
*SOLUCIÓN;*
*AL CAMPO REG12 ASÍGNALE PROVENIENTE DE REG21.*
*PARA OPERANDO 2:*
*Si (s_cdr='1') ===>*
*SOLUCIÓN;*
*AL CAMPO RD ASÍGNALE EL DATO PROVENIENTE DE MEMORIA.*
*EN CASO SE NO CUMPLIRSE LA CONDICIÓN ===>*
*AL CAMPO RD ASÍGNALE EL DATO PROVENIENTE DE UM.*

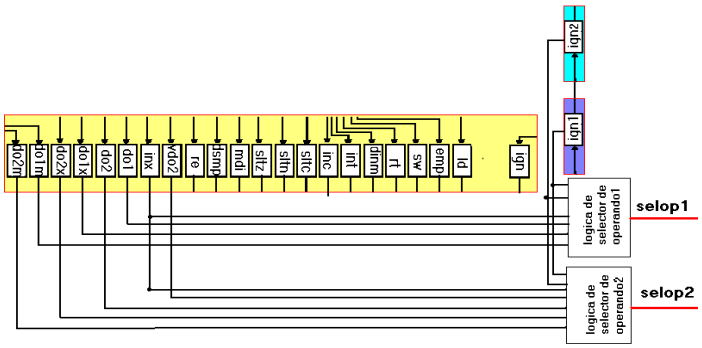


Figura 2.8 Señales de control de anticipación de datos

El dato que es extraído de memoria o que pasa a esta etapa, es emitido a la etapa de decodificación o ejecución de acuerdo al caso de la dependencia. Por esta razón es necesario indicar que en la etapa de memoria se encuentra una instrucción de carga. En tal caso el dato que debe fluir por la segmentación es el dato extraído de memoria, en caso contrario el dato que debe fluir por la segmentación proviene del resultado de la etapa de ejecución.

*Si (ld2='1') ===>*
*SOLUCIÓN;*
*EL DATO QUE DEBE FLUIR ES EL EXTRAÍDO DE MEMORIA*
*EN CASO DE NO CUMPLIRSE LA CONDICIÓN ===>*
*SOLUCIÓN;*
*EL DATO QUE DEBE FLUIR PROVIENE DE REG12.*

## 3. PRUEBAS
## DEPENDENCIA DE DATOS CON DETENCIÓN

La figura 3.1 muestra los resultados de la solución de riesgos de la segmentación cuando se implica una instrucción de carga.

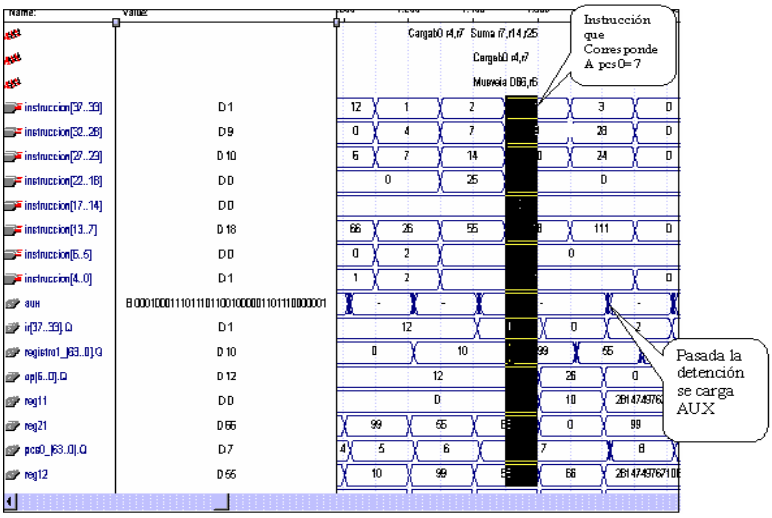


Figura 3.1 Resultados para dependencia de datos

En la parte superior de la tercera columna en la figura 3.1, se muestra la secuencia de ejecución de instrucciones donde se puede ver que en la instrucción de suma, el registro **r7,** fuente de primer operando, causa dependencia de datos con el registro destino de la instrucción de carga. La parte sombreada señala la etapa en la cual la dependencia se detecta. En la segunda columna se puede ver que el registro **AUX** tiene el código de la instrucción **Suma**, y el registro **IR,** el código de la instrucción de carga. En el siguiente ciclo, después de detectar la dependencia que inicia al finalizar el área sombreada en el diagrama de tiempo correspondiente al registro **AUX,** se ve como se mantiene la instrucción **Suma** por un ciclo más. De igual manera en el contador de programas se mantiene la dirección 7 que apunta a la instrucción que sigue a la instrucción detenida en el registro **AUX,** esto para no perder la secuencia de ejecución como se ve en la forma de onda correspondiente al registro **PCS0**. Además se limpia el registro **IR** como se ve en el tren de pulsos del registro **IR** que tiene el valor cero. El registro **AUX** se actualiza con la instrucción que llega de memoria de código, la instrucción detenida fluye a la etapa de decodificación y el contenido del registro **PCS0** se incrementa a 8.

## 4. CONCLUSIONES

La técnica muestra las ventajas de utilizar la implementación por hardware con respecto a la técnica usual de implementación de retardos por software. La propuesta de solución en el tratamiento de riesgos de dependencia de datos solo utiliza detenciones por hardware, tal es el caso cuando se presenta una instrucción de carga en la segmentación.

En otros casos de dependencia de datos, se utiliza adelantamiento con lo cual se evita cualquier detención o inserción de retardos innecesarios de la instrucción aumentando el rendimiento de la segmentación. La solución por hardware de la detención mejora el tráfico y rendimiento en la segmentación, ya que cuando se utiliza detención por hardware, las instrucciones son mantenidas dentro de la segmentación y no se realiza una lectura a la memoria para extraer una instrucción del tipo no operación como se presenta cuando la solución la proporciona el compilador.

Este método se diferencia porque realiza un análisis de anticipación en la etapa de búsqueda y porque la anticipación de datos se aplica a la etapa de decodificación y búsqueda de operandos, con lo cual se evita un ciclo más. Además la solución está ligada a los resultados de la anticipación, que a diferencia de los métodos por hardware, mencionados anteriormente, se utilizan dos módulos independientes uno encargado de realizar la anticipación de datos a la etapa de ejecución y uno de realizar la detención de instrucciones en la etapa de búsqueda, así para riesgos por dependencia de datos, donde se involucra una instrucción de carga, solo se pierde un ciclo con la detención. Para otro tipo de riesgos de dependencia de datos, con el adelantamiento se mantiene el flujo de ejecución uniforme.

Finalmente se programó el diseño del procesador RS-2000 en lenguaje VHDL, para realizar la simulación de los riesgos por dependencia de datos para observar como funciona la solución de riesgos de dependencia de datos.

## BIBLIOGRAFÍA

[1] Javier Monroy Perea, Tesis “Arquitectura de computadora RS2000”.
[2] Mehdi R. Zargham, “Computer Architecture”, Prentice Hall, 1996.
[3] Daniel Tabak, “RISC Systems”, John wyley & sons inc., 1990.
[4] David Patterson, John L. Hennesy, “Organización y Diseño de Computadoras”,España, Mc. Graw Hill, 1995
.

[5] John L. Hennesy, David Patterson, “Arquitectura de Computadoras un Enfoque Cuantitativo”, Mexico, Mc. Graw Hill, 1993.
[6] Vincent P. Heuring, Harry F. Jordan, “Computer Systems Design And Architecture” Addison wesley 1997.
[7] M. Morris Mano, “Arquitectura de Computadoras “, Prentice Hall, 1994.
[8] Andrew S. Tanenbaum, “Organización de Computadoras un Enfoque Estructurado”, México, Prentice Hall, 1992.
[9] Kai Hwang, Faye A. Briggs, “Arquitectura de Computadoras y Procesamiento Paralelo”, Mexico, Mc. Graw Hill, 1988.
Prentice Hall, 1996.
[10] Jurij Silc, Borut Robic, Theo Ungeres, “Processor architecture”, Germany, Springer, 1999.
[11] Willian Stallings, “Organización y Arquitectura de Computadores”, España, Prentice Hall, 1997.
[12] M. Morris Mano, Charles R. Kime “Logic and Computer Design Fundamentals”, USA,
[13] M. Morris Mano, “Lógica Digital y Diseño de Computadoras”, México,
Prentice Hall, 1982.
[14] Bustillo Dìaz M., Cortez I. J., Cortez L., Toxqui Rodrìguez A., Zehe A., Ramirez A., “An improved solution of control risk in segmentation by hardware approach” Weas Transactions on Computers, ISSN: 1109-2750, Issue 6, vol. 3, December 2004,