Prevención de riesgos en la segmentación en arquitecturas de computadoras risc.

Bustillo Díaz Mario, Cortez José Italo, Pérez Aguirre Carlos, Celaya Borges Carlos, Toxqui Tomás Arturo
Benemérita Universidad Autónoma de Puebla
31 poniente 1304
Tel: (01) (222) 229-55-00 ext 6405
Puebla, Puebla
e-mail: jitalo@fismat1.fcfm.buap.mx
e-mail: bustillo@cs.buap.mx
e-mail: aguica@starmedia.com
e-mail: ccelaya@siu.buap.mx

## RESUMEN

En el diseño de arquitecturas, todos los expertos se han preocupado en cuanto a obtener el máximo rendimiento del sistema y de esta forma obtener una alta productividad.
De esta forma al plantear una arquitectura con tecnología Risc y con las técnicas de segmentación implementada, se busca obtener un mejor rendimiento de todos los recursos del sistema, es decir, que todas los recursos trabajen a su máxima capacidad.
En este artículo proponemos una metodología que soluciona los problemas relacionados con los riesgos de segmentación, utilizando la unidad de adelantamiento la cual se puede implementar por hardware, permitiendo un mayor rendimiento de los procesadores en donde se implemente la tecnología de segmentación.

## INTRODUCCIÓN

La evolución de las arquitecturas de computadoras ha mostrado una tendencia hacia una complejidad creciente, hasta que en 1975 los investigadores cuestionaron si esta complejidad era la manera ideal de alcanzar un balance óptimo de coste y rendimiento y productividad[8]. Las investigaciones en este campo dieron como resultado el surgimiento de computadoras RISC, en oposición a las CISC.

La tecnología RISC tiene como objetivo: que las instrucciones de un programa sean ejecutadas lo más rápidamente posible. Esta finalidad se consigue gracias a la simplificación de las instrucciones incluidas en el procesador, y a la ejecución solapada (realización de varias instrucciones al mismo tiempo), empleando técnicas de segmentación. Los procesadores RISC racionalizan y optimizan el trabajo interno de los sistemas, aunque para ello requieran más memoria y una mejor tecnología de compilación[3,6,8]

¿Pero qué es la segmentación?. La Segmentación es una técnica que implementa seudo paralelismo, en la cual múltiples instrucciones se solapan durante su ejecución. En la actualidad la segmentación es clave para hacer procesadores rápidos.

Las maquinas RISC refinaron la noción de segmentación planificada por compilador a principios de los años ochenta. Los conceptos de salto retardado y cargas retardadas comunes en microprogramación se extendieron a la arquitectura de alto nivel. Estas instrucciones definen riesgos a distancia: cargas retardadas, significa que el valor del nuevo registro no esta disponible para la próxima instrucción y saltos retardados, significa que el salto se realiza después de la siguiente instrucción, no antes[9].

Como todas las etapas de instrucción están conectadas entre sí, todas deben estar listas para proceder al mismo tiempo; por tanto, la velocidad a la cual las instrucciones salen del cauce no puede exceder de la velocidad a la que entran. El tiempo necesario para que una instrucción avance un paso a lo largo del cauce es un ciclo de reloj. La duración del ciclo del reloj está determinada por el tiempo necesario para la etapa más lenta de la segmentación, porque todas las etapas deben proceder a la misma velocidad. El objetivo de los diseñadores (para la segmentación de las instrucciones), es equilibrar la duración de cada

etapa; de otra forma, habrá tiempo de inactividad durante una etapa.

## 2. DESARROLLO

Las etapas de desarrollo del sistema son las siguientes:

### EL SISTEMA

Dentro de los objetivos del sistema esta por supuesto, obtener un alto rendimiento y una alta productividad, también se busca evitar que los problemas clásicos de la segmentación sean solucionados mediante software, buscar una solución mediante hardware, y que la programación sea menos tediosa. Por otro lado, se busca la posibilidad de realizar operaciones relativamente grandes y obtener todo el resultado, como por ejemplo en una multiplicación donde el resultado puede crecer mucho más rápido que en otra operación. Para lograrlo se plantea la existencia de una unidad especial llamado *detector*, cuya principal tarea es la de solucionar los problemas inherentes a los riesgos de la segmentación como son: la dependencia de datos y los saltos, adelantándose a estos hechos y tener un buen funcionamiento y sobretodo que el programador no tenga que estar pensando en los riesgos.

Se propone un grupo de registros de propósito general con un ancho cada uno de 128 bits (16 bytes), para poder soportar las operaciones. En cuanto al problema de compartir recursos, lo cual se presentan en el diseño de una maquina segmentada, la solución se orienta a lograr una mejor organización de los recursos para evitar lo anterior.

### ESQUEMAS HARDWARE PARA RESOLVER LOS RIESGOS POR DEPENDENCIA DE DATOS

El esquema de hardware es relativamente simple para resolver los problemas por dependencia de datos, lo cual explicamos a continuación.

La figura 1.1 nos muestra la organización de la segmentación, que consta de 5 etapas: IF Búsqueda de la instrucción, ID Decodificación y busqueda de operandos, EX Ejecución, MEM Referencia a memoria, WB Escritura de registros[1,7,9].

En el limite de cada etapa se colocan los registros de la segmentación:

IF/ID: Obtiene la instrucción a ejecutarse de la memoria de código.

ID/EX: Obtiene los operandos a ser emitidos para ejecución de la instrucción.

EX/MEM: Obtiene el resultados de la unidad de ejecución, ademas de mantener el indicador de registro destino.

MEM/WB: Mantiene informacion proveniente del registro de la segmentación EX

- Detectar el riesgo
- Retardar las instrucciones afectadas con el riesgo o anticipar el dato que causa la dependencia.
- En algunas combinaciones de instrucciones que causan dependencia, el dato no se puede anticipar por tanto las detenciones son indispensables.
- Continuar con la ejecución de las instrucciones una vez que no haya dependencia de datos.

De la figura 1.1 podemos ver que un riesgo por dependencia de datos se presenta si la instrucción en la etapa de búsqueda de operandos necesita leer un registro cuyo dato apenas esta siendo generado por la unidad de ejecución o que ya se encuentra en el registro de segmentación MEM/WB, o en el peor de los casos cuando es un dato de una instrucción de carga y apenas será obtenido al final de la etapa de referencia a memoria. Puesto que, se mantiene la secuencia de ejecución de las instrucciones se aseguran resultados validos.

Una forma de resolver estos riesgos por dependencia de datos es detener el flujo de las instrucciones afectadas por la dependencia. Esta detención se debe realizar en las etapas de decodificación y búsqueda de operados, en la primera es con el fin de evitar perder la instrucción retardada en esa etapa y que pueda leer un dato no valido y pasar a ejecución; la segunda con el fin de evitar perder la instrucción que ya ha sido buscada[2]. Otra forma de solución a los riegos de datos es el retardo de instrucciones en cooperación con la anticipación de datos.

Se dice que un dato es anticipado si en el momento que se necesita (que se lee de los registros y no esta disponible) puede tomarse directamente de los registros de la segmentación en las diferentes etapas. aquí se permite que las instrucciones dependientes avancen en la segmentación para después anticipar el dato al momento que será utilizado por la unidad de

ejecución y solo ejecutar las detenciones de las instrucciones cuando una anticipación no se pueda realizar. ya que el dato solo esta disponible hasta la etapa de escritura y una etapa antes el dato será ocupado, como ejemplo cuando el dato es el resultado de una instrucción de carga.

:

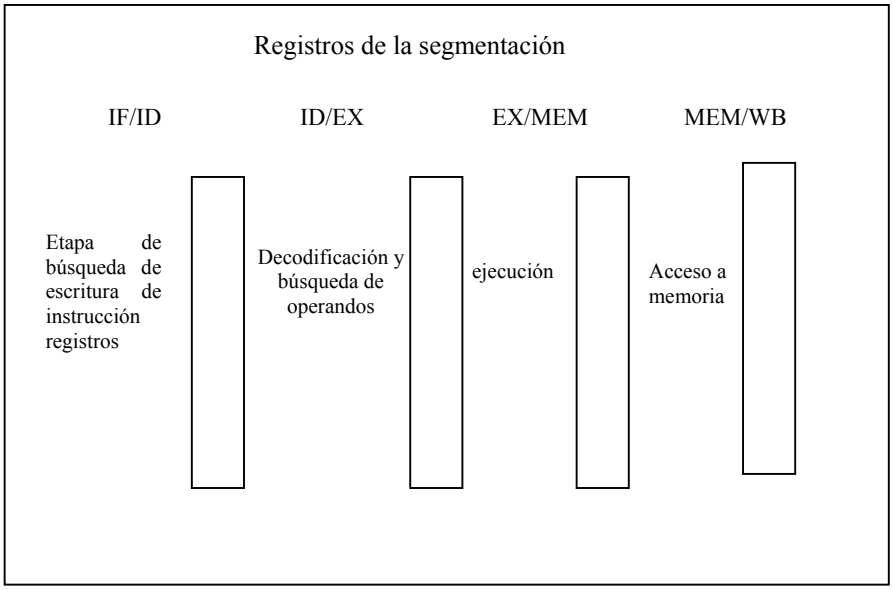


Fig. 1.1 Organización de la Segmentación

Los eventos principales para resolver estos riesgos son:
En la figura 1.2 se presenta la unidad de detección de riesgos, la detección del riesgo es hecha en la etapa de decodificación. Para detectar el riesgo analizamos el campo de la señal de escritura de registro a registro destino, lo que corresponde a los registros de segmentación en las etapas de ejecución, memoria y escritura. Estas señales se dirigirán a la unidad de detección de riesgos.
El retardo de las instrucciones si así se requiere se realiza limpiando los campos de control del registro de segmentación ID/EX (in_mux=0), esta operación permite insertar una burbuja en la unidad de ejecución, y para la etapa de decodificación se desactiva la señal de carga de PC y del registro de segmentación IF/ID, con esto se evita que otra instrucción se busque. Las señales que desactivan al PC y al IF/ID son las que salen de la unidad de anticipación de eventos de la figura anterior.

La otra forma de solucionar los riesgos por dependencia de datos es permitir que siga el flujo de instrucciones aún si existe el riesgo por dependencia de datos, para lo cual debe de haber una unidad que se encargue de detectar los registros que están en dependencia y además solucione estos riesgos a esta unidad le llamamos *detector* o *unidad de adelantamiento de datos*. Esta unidad se encuentra en la etapa de ejecución ya que al principio de esta etapa se analiza la dependencia. O sea, los campos del registro destino correspondiente, de los registros de segmentación, en la etapa de memoria, escritura y los campos de operando del registro de segmentación son enviados a la unidad de adelantamiento de datos, con el fin de detectar una dependencia y que la unidad de adelantamiento genere la señal de control que permita elegir entre las entradas para la ejecución; provenientes del registro ID/EX o de las entradas provenientes del resultado del registro de segmentación EX/MEM

(este resultado se emite en paralelo a MEM/WB y UA).

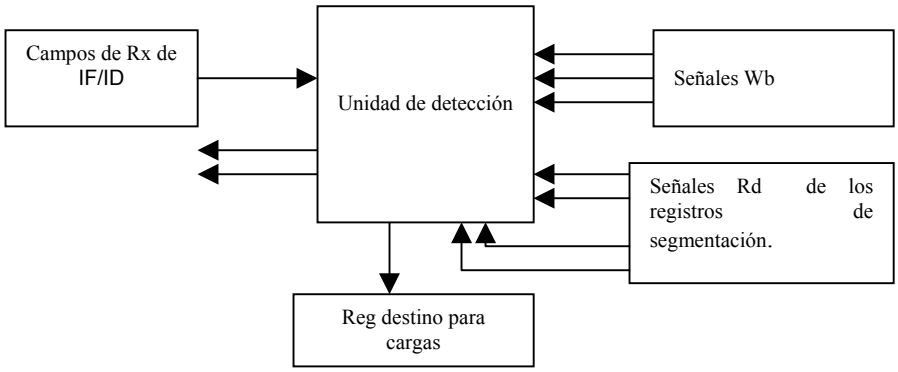


Fig. 1.2 Organización de la unidad de detección de riesgos.

El resultado proveniente del registro de segmentación MEM/WB se emite en paralelo con los resultados de los otros registros de segmentación a la UA en la misma etapa de ejecución como se muestra en la figura1.3. Con este esquema evitamos los retardos causados por las detenciones de las instrucciones (burbujas HW).

dependencias que no pueden ser tratadas con adelantamiento de datos, como las *dependencias*

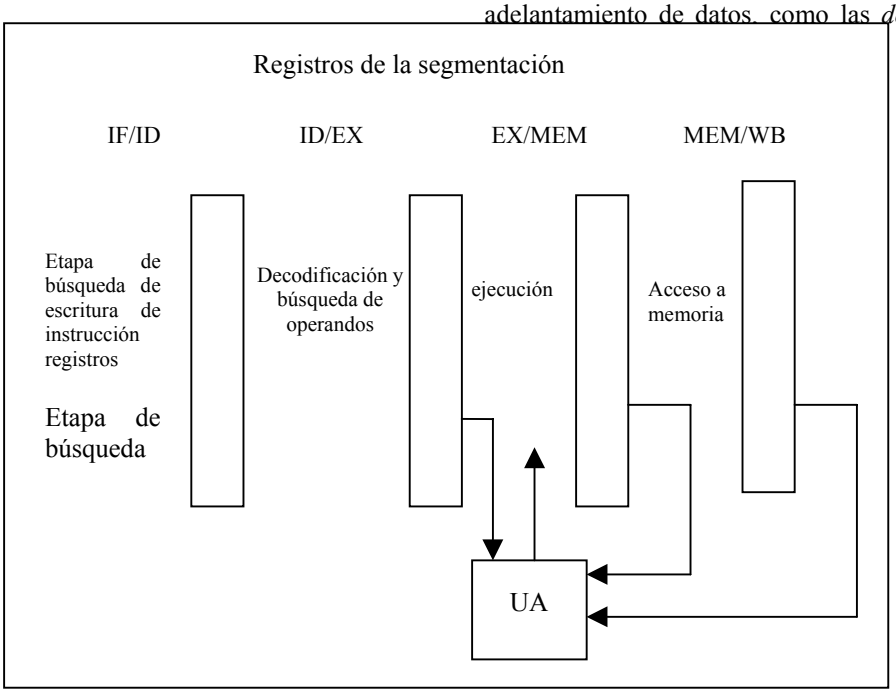


Fig. 1.3 organización de la segmentación con detención de instrucciones y adelantamiento de datos.

Como podemos ver en la figura 1.4, nuestra unidad de detección de riesgos se libera de ciertas dependencias y ahora se concentrara en combinaciones de instrucciones que generan

Un ejemplo: cuando el registro destino de una instrucción de carga se leerá por la instrucción inmediata.

**PROPUESTA**

Ahora podemos ver a nuestra unidad de adelantamiento de datos en la siguiente figura.

Las condiciones de anticipación de los resultados para generar señales de control de los MUX de la ALU se presentan continuación.

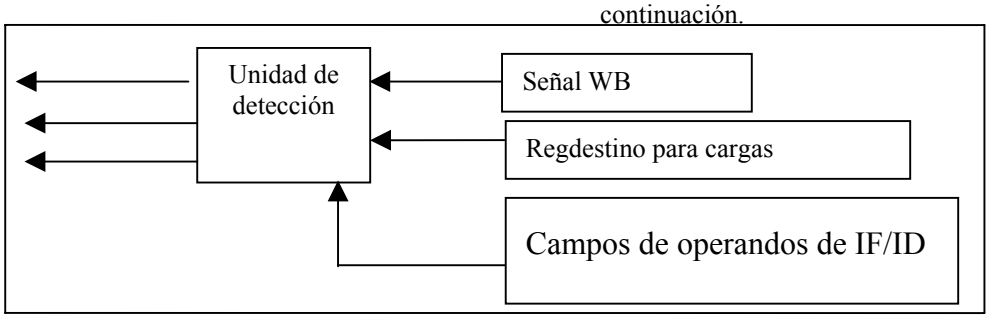


Figura 1.4 Que representa la unidad de detección cuando se utiliza adelantamiento de datos.

WB indica que el rd de esta etapa será escrito.
Salidas:

- Una señal de salida es para controlar el PC.
- Otra para detener la carga de IF/ID.
- Controlar el MUX que elegirá la entrada cero (para limpiar los controles de D/EX).

A continuación presentamos las condiciones de riesgo para nuestra unidad de detección:

Sí ID/EX. regwrite
Y
* ID/EX.regdest = 0
Y → RIESGO
** ID/EX.registro_a_escribir regdest = IF/ID.leer registro1
X=O
*** ID/EX. Registro_a_escribir regdest = IF/ID.leer registro2

* Indica que el rd corresponde a una instrucción de carga.
** Indica que hay dependencia entre el operando uno y el destino de carga.
*** Indica que hay dependencia entre él operando dos y el destino de carga.
Esto permite observar que la unidad de detección se limita a ciertas combinaciones de instrucciones que causan dependencias que no pueden ser resueltas por adelantamiento de datos y como consecuencia se debe introducir detenciones (burbujas hardware)

**TEST1.- RIESGO EX.**

Sí EX/MEM. Regwrite Y aluselA = 01

EX/MEM.escribir registro = ID/EX.leerregistro1

Sí EX/MEM.regwrite Y **aluselB = 01**

EX/MEM.escribirregistro = ID/EX.leeregistro2

**TEST2.- RIESGO MEM**.

Sí MEM/WB.regwrite Y aluselA = 10

MEM/WB.escribirregistro = ID/EX.leerregistro1

Sí MEM/WB.regwrite Y aluselB = 10

MEM/WB.escribir registro = ID/EX.leer registro2

De lo anterior tenemos que el significado de las líneas de control de los MUX es el siguiente:
aluselA = 00 los operandos son tomados del registro de segmentación ID/EX.
aluselA = 01 el primer operando de la alu es adelantado del resultado anterior (contenido EX/MEM).

aluselA = 10 el primer operando de la alu es adelantado del resultado anterior(contenido MEM/WB).
De igual manera para el segundo operando de la alu (MUXB).
aluselB = 00
aluselB =01
aluselB =10.

**Nota 1. -** no olvidar que en EX/MEM el dato es emitido al registro MEM/WB en paralelo con los MUX, y en MEM/WB el dato es emitido al archivo de registros en paralelo con los MUX.

**Nota2.-** el archivo de registros suministra el resultado correcto, sí la instrucción en la ETAPA ID lee el mismo registro que la instrucción en la ETAPA WB lo cual esta escribiendo. Esto se dice porque en el MISMO MOMENTO el registro en el cual se guardará el dato también debe ser leído ese mismo dato.

solución de riesgos por hardware antes mencionado veremos que no es necesario realizar detenciones(burbujas hardware) para este tipo de dependencias de datos y por tanto mejorar la salida de la segmentación.

i1. SUB R2 , R1, R3.

i2. AND R4, R2,R5

i3. OR R8,R2,R6

i4. ADD R9,R4.R2

i5. SLT R1,R6,R7

Presentamos las siguientes instrucciones como un ejemplo de lo que pasa en la segmentación cuando existen dependencias de datos utilizando el esquema de

| t0 | t1 | t2 | | | | |
|---|---|---|---|---|---|---|
| ***IF / ID*** | ***ID / EX*** | ***EX / MEM*** | ***MEM/W*** | | | |
| SUB R2,R1,R3 | Wb fijar ,DR1,DR3 R2, R2, R3 | (Wb,result,Rd) ALU(DR1,DR3) UA_i(Wb,R2,R1,R3) UA_s(smux=00) | (Wb,result,R | almacenar el resultado en R2. | | |
| | ***IF / ID*** | ***ID/EX*** | ***EX/ MEM*** | ***MEM/WB*** | ***Se actualiza el registro*** | |
| | AND R4,R2,R5 | (Wb,DR2,DR5,R4,R2, R5) | (Wb,result,Rd4) **UAi**(rd2-mem,Wb,rd-wr, Wb,R4,R2,R5) **UAs**(smuxa=01) detecta y toma el dato de EX/MEM | (Wb,result,R4) | R4 | |
| | ***IF / ID*** | | ***ID /EX*** | ***EX/MEM*** | ***MEM/WB*** | ***Se actualiza el registro*** |
| | OR R8,R2,R6 | | (Wb,DR2,DR6,R8,R2, R6) | (Wb,resutl,Rd8) **UAi**(rd4-mem,wb,rd2-wr ,Wb,R8,R2,R6) **UAs**(smuxa=10) detecta y toma el dato de MEM/WB que se emite a MUX y el archivo de reg. | (Wb, result,Rd8) | R8 |
| | | | ***IF /ID*** | ***ID/EX*** | ***EX/MEM*** | ***MEM/WB*** |
| | | | ADD R9,R4,R2 | (WB,DR4,DR2,R9,R4, R2) en este momento R2 esta siendo actualizado por la primera instrucción de tal forma que hay escritura y lectura simultanea. | (Wb,result,Rd9) **UAi**(rd8-mem,wb,rd9-wr wb,R9,R4,R2) **UAs**(smuxa=10) detecta y toma el dato de MEM/WB el cual se se emite al MUX y el archivo de registros | (Wb,result,Rd9) |
| | | | | ***IF/ID*** | ***ID/EX*** | ***EX/MEM*** |
| | | | | SLT R1,R6,R7 | (Wb,RD6,RD7,R1,R6 R7) | Se tiene la condicion y la direccion de salto. en la **UA** no se detectan dependencias. |
| | | | | | IF/ID | |

UAi.- INDICA LAS ENTRADAS A LA UNIDAD DE ADELANTAMINETO DE DATOS

UAs.- IINDICA LAS SALIDAS DE LA UNIDAD DE ADELANTAMIENTO DE DATOS.

COMO PODEMOS VER EN ESTA SECUENCIA NO SE UITLIZA LA UNIDAD DE DETECCION DE RIESGOS PARA RESOLVE LAS DEPENDENCIAS DE DATOS QUE SE PRESENTAN.

Existe dependencia entre i1 e i2; i1, i3 e i4; i2 e i4; i3 e i5.
Como veremos en la tabla 1.1 la unidad de adelantamiento nos permitirá detectar y resolver las dependencias de datos sin realizar detenciones de las instrucciones afectadas por la dependencia.
Como nos muestra la tabla 1.1 la instrucción i1 fluye por la segmentación de forma normal, pero para i2 cuando pasa a la etapa de ejecución al inicio del ciclo T3 la unidad de adelantamiento detecta que el primer registro operando de la instrucción i2 es el registro destino de la instrucción i1, pero que el dato requerido se encuentra en el registro de segmentación EX/MEM (esto se puede observar de mejor manera con ayuda de la figura 1.3) entonces en este mismo ciclo la unidad de adelantamiento genera la señal de control que permitirá dirigir el dato que causo el riesgo, del registro de segmentación EX/MEM a la unidad de ejecución. En T4 la unidad de adelantamiento tiene actividad para la instrucción i3, puesto que la actualización del registro destino de la instrucción i1 esta involucrado como operando de la instrucción i3 que pasa a ejecución, y en este ciclo el registro destino será actualizado, nuevamente la unidad de adelantamiento genera la señal de control que permitirá que el dato sea tomado del registro de segmentación MEM/WB y emitido a la pila de registros simultáneamente. En T5 nuevamente se tiene un riesgo de las mismas características al anterior entre la instrucción i2 e i4. Como podemos ver en este tipo de riesgos, detenciones de instrucciones no se han realizado.

## ESQUEMAS HARDWARE PARA RESOLVER LOS RIESGOS DE CONTROL

Otro tipo de problemas que se presentan con la segmentación son los ocasionados por las instrucciones de tipo brinco. La técnica que se presenta es una adición a la estructura de la técnica de adelantamiento y detección de riesgos por dependencia de datos.
Las Forma de tratar estos riesgos en esquemas de hardware puede ser realizando detenciones de las instrucciones siguientes al salto y la otra es suponer que el salto no será realizado[9].
En la primera, las detenciones se realizan de forma similar a las realizadas por dependencias de datos. Cuando se detecta y resulta que se trata de una instrucción de brinco condicional, las instrucciones siguientes se retardan hasta que el brinco se resuelve. Este puede tener la desventaja de que si un brinco no se lleva a cabo, las instrucciones se han retardado sin necesidad.
En el segundo esquema permitimos continuar con la ejecución de las instrucciones que siguen a la instrucción de brinco condicional, si la condición se cumple, las instrucciones que se encuentran en la segmentación después de la instrucción de brinco son invalidadas. Para esto necesitamos una forma de invalidar las instrucciones en la segmentación, la forma de hacer estos es agregar señales de control que permitan limpiar los registros de la segmentación en las etapas de ejecución, decodificación y búsqueda, la necesidad de estas señales para tales etapas, es que en el inicio del ciclo4 de una instrucción de brinco condicional sé esta actualizando el PC con la dirección de brinco y en este ciclo las instrucciones inmediatas en secuencia inician su etapa de ejecución, decodificación y búsqueda respectivamente, la estructura de la segmentación con estas señales adicionales se muestra en la figura 1.6 y su descripción es la siguiente:
IF.flush.- pone a cero el registro de segmentación IF/ID o el campo RD del mismo registro.
Propósito: Lo esencial es que limpie el campo de registro destino en el registro de segmentación IF/ID, de tal forma que la instrucción extraída de memoria fluya por la segmentación sin ningún efecto.
ID.flush.- limpiar los controles de ID/EX.
Propósito: en el momento en que el brinco es realizado la instrucción en la etapa de decodificación y búsqueda de operandos pase a la siguiente etapa con sus señales de control a Cero(WB, EX, MEM).

EX.flush.- limpie el campo de control del registro de la segmentación EX/MEM. Propósito: En el momento que el brinco es efectuado la instrucción que se encuentra en ejecución; pase a la siguiente etapa con la señal Wb y M puestas a cero, con esto invalidando la Escritura del resultado(la operación realizada en la etapa de ejecución no tiene efecto).

CONSIDERE LA SIGUIENTE SECUENCIA:

```
SUB R10,R4,R8
BEQ R1,R3,R28
AND R12,R2,R5
OR   R14,R2,R6
ADD R1,R2,R1
SLT R15,R6,R7
     .
     .
     .
LW
```

TECNICA DE FIJAR DECISIÓN DE BRINCO

CUANDO SE ENCUENTRA UNA INSTRUCCIÓN DE BRINCO CONDICIONAL SE SUPONE QUE EL BRINCO NO SERA REALIZADO Y SE PERMITE FLUIR A LAS SIGUIENTES INSTRUCCIONES . EN ESTE ESQUEMA EL PC TOMA DIRECCIONES SECUENCIALES O DIRECCIONES CALCULADAS EN LA ETAPA DE EJECUCION.

| | T0 | T1 | T2 | T3 | T4 | T5 |
|---|---|---|---|---|---|---|
| | IF/ID SUB | ID/EX DR4.DR8,R10,R4,R8 | EX/MEM (result,R8,Wb) | MEM/WB (Wb,result,R8) | Se actualiza el contenido de R8 | |
| | | IF/ID BEQ | ID/EX DR1,DR3, | EX/MEM (obtiene la condición de brinco) (se calcula la dirección de brinco) | MEM/WB se elige la dirección de brinco | IF/ID ***LW*** |
| | | | IF/ID AND | ID/EX DR2,DR5,R12,R2,R5 | EX/MEM se realiza la operación pero al final de este ciclo se activa EX.flush y limpia los controles . | MEM/WB BURBUJA |
| | | | | IF/ID OR | ID/EX se obtiene DR14,DR6 al final del ciclo también se activa ID.flush y limpia los controles . | EX/MEM BURBUJA |
| | | | | | IF/ID ADD al final de este ciclo la instrucción será cargada en este registro de segmentación, pero | ID/MEM BURBUJA |

Tabla 1.2 segmentación con una instrucción de control

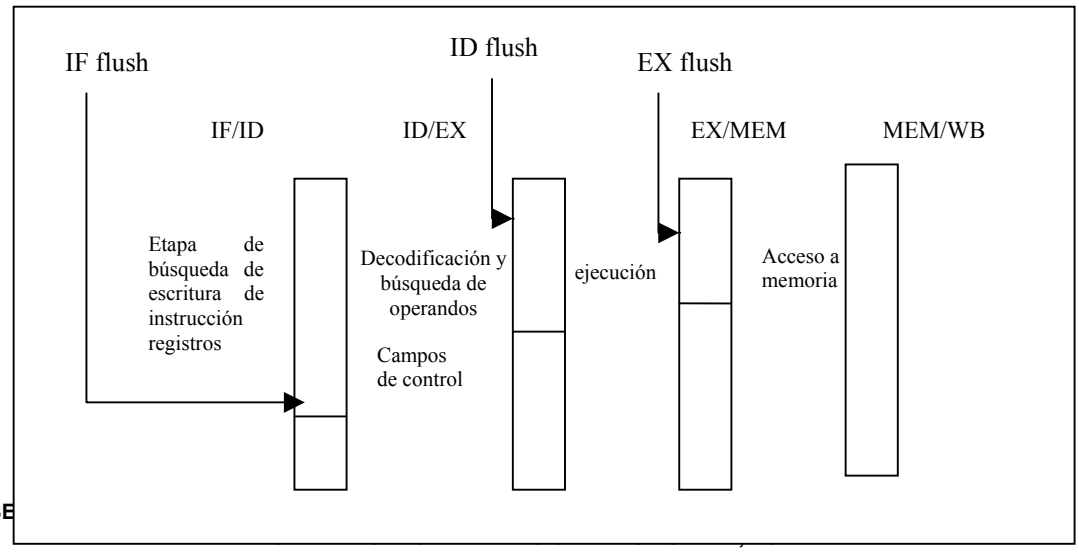


Fig. 1.6 organización de la segmentación

Después que buscamos la primera instrucción a partir del brinco, las instrucciones que se les permitió fluir por la segmentación, inmediatas al brinco, que no modifiquen los registros. Esto es, la instrucción que se encuentra en ejecución pase a MEM con la señal de escritura de registro desactivada, la instrucción en decodificación pase a la siguiente etapa con la señal de escritura de registro, MEM operación desactivadas y por ultimo la instrucción que fue buscada al momento del brinco pase a la siguiente etapa con sus campos cero o Rd=0, con la cual se tienen burbujas en estas etapas. En la tabla 1.2 y con ayuda de la figura anterior se presenta un ejemplo de lo que sucede en la segmentación utilizando este esquema para los riesgos de control.

La ejecución de las instrucciones fluye de forma normal hasta finalizar el ciclo T3 donde se conoce la condición de brinco y al inicio del ciclo T4 se emite la dirección objetivo de brinco al PC, puesto que se permitió que instrucciones siguientes a la de brinco condicional fluyeran por la segmentación, ahora deben ser invalidadas, entonces en este mismo T4 las señales EX.flush, ID.flush, IF.flush son activadas con lo cual se limpia el campo de control EX/MEM, el campo de control del registro de segmentación ID/EX y el campo de registro destino en el registro de segmentación IF/ID respectivamente. Con esto las operaciones que se realizan en las diferentes etapas no tienen efecto y se comportan como burbujas como se puede ver en el ciclo T5.

## CONCLUSIONES

La segmentación es un medio que sirve para aumentar la productividad creando paralelismo en la ejecución de las instrucciones de ensamblador dentro del microprocesador, implica que los programas se ejecutaran en menos tiempos y por lo tanto se elevara el rendimiento y la productividad.

Es importante resolver los riesgos que se dan al utilizar las técnicas de segmentación, de lo contrario el paralelismo ganado al solapar la ejecución de las instrucciones de ensamblador servirá de poco, al contrario será un elemento de retardo en determinados casos, de acuerdo a la metodología utilizada en la solución.

Nuestra propuesta es dar la solución desde el punto de implementación de hardware, teniendo en cuenta que esto significara un mayor uso de dispositivos que redundará en el costo del microprocesador, sin embargo resuelve eficientemente los riesgos de segmentación y no afectar al paralelismo logrado con la implementación de las técnicas de segmentación.

## BIBLIOGRAFÍA Y REFERENCIAS


**1.Implementation and Evaluation of the Complex Streamed Instruction Set.** Ben Juurlink , Dmitri Tcheressiz , Stamatis Vassiliadis, Harry Wijshoff 16 th acm international confernce on supercomputing 2002.

2. **Boolean Formula-based Branch Prediction for Future Technologies.** Daniel Jimenez, Heather Hanson and Calvin Lin 16 th acm international confernce on supercomputing 2000.

3 **Recovery mechanism for latency misprediction.** Enric Morancho, Jose Maria Llaberia and Angel Olive 16 th acm international confernce on supercomputing 1999.

4. **Modeling Superscalar Processors via Statistical Simulation**. Sebastien Nussbaum and James Smith 16 th acm international confernce on supercomputing 2000.

5. **Basic Block Distribution Analysis to Find Periodic Behavior and Simulation Points in Applications.** Tim Sherwood, Erez Perelman 16 th acm international confernce on supercomputing 2000.

6.**Combining brach predictors.** MacfarlingScott, WRL Technical report TN-36, 1993

7.**Organización y diseño de computadoras La interface hardware/software** Patterson D A, Hennessy J L, Mcgraw Hill Interamericana 1995.
8.**Computer architecture: A quantitative approach** Patterson D A, Hennessy J L. Morgan Kaufman Publisher 1998.
9.**Computer architecture tutorial** Prabhu G M,