Predictores de saltos

Guillermo Aguirre

September 9, 2016

1 Consideraciones generales sobre el tema

El aprovechamiento simultáneo de las principales unidades dentro de la CPU, es una excelente aproximación para construir procesadores con un desempeño aceptable. Pero la segmentaciónón (pipelining) ha requerido el desarrollo de un conjunto de técnicas que permitan aprovechar el potencial de la superposición de instrucciones, lo que se conoce como paralelismo a nivel de instrucción (en inglés instruction-level parallelism, ILP). Entre estas técnicas están el desenrollado de iteraciones, el renombramiento de registros y la predicción de saltos. Esta última es particularmente interesante de estudiar porque es responsable de llevar la conducción durante la etapa de recuperación de las instrucciones (fetch) y permite reducir la pérdida de ciclos del procesador debido a riesgos (hazards) de control. Las rupturas de secuencia (no solamente los saltos condicionales sino también los incondicionales, las invocaciones a funciones o procedimientos y los retornos de estos) ocurren muy frecuentemente en los programas y las malas predicciones son muy costosas en términos de ciclos sin el despacho de instrucciones que realicen trabajo útil. Este costo se hace cada vez mayor cuando se aumenta el número de etapas de la segmentación al procurar aumentar el ILP. En este trabajo se aborda el estudio de la predicción de saltos partiendo de los principales conceptos involucrados, la descripción de algunas propuestas clásicas para su utilización y una comparación entre esas propuestas.

2 Introducción de conceptos previos

Detrás de la predicción de salto existe una idea muy simple: anticipar o predecir cual de los dos caminos seguirá un salto: el que corresponde al destino del salto, cuando el salto es *tomado* o el que continúa después de la instrucción de salto, cuando el salto es *no tomado*. Si el salto se predice como tomado además se debe obtener la dirección de la instrucción destino del salto. En cualquiera de los casos la ejecución puede continuar de forma especulativa a lo largo del camino supuesto. Una vez que se resuelve efectivamente la condición se debe hacer una valoración de la predicción realizada y luego reforzarla o corregirla.
El comportamiento de los saltos se puede predecir estáticamente en tiempo de compilación y dinámicamente por medio del hardware en tiempo de ejecución. Los predictores

estáticos son usados ocasionalmente en procesadores donde se espera que el comportamiento del salto sea altamente predecible en tiempo de compilación; la predicción estática también se puede usar para asistir a los predictores dinámicos. Un ejemplo de planificación estática se da en el salto demorado cuando se deben seleccionar las instrucciones más convenientes para ser ubicadas en la ranura (en inglés slot) de demora.
Existen dos clases generales de predictores: locales y globales. Existen algunas propuestas para combinar distintas clases de predictores. De acuerdo a las características que tengan, los predictores necesitan contar con distintas cantidades de recursos, lo que se llama tamaño del predictor. Este parámetro se usa para comparar los distintos predictores.

Predictores estáticos Los primeros predictores fueron estáticos, es decir aquellos que en ejecución predicen todas las ejecuciones de un salto condicional del mismo modo [1]. La estrategia preferida fue predecir los saltos como tomados, pero esta decisión se basó en una equivocación, debido a que las primeras estimaciones tuvieron en cuenta transferencias de control condicionales e incondicionales. Además estas consideraciones generalmente se basaron en programas científicos donde los saltos tomados prevalecen. Estudios posteriores llegaron a cuantificar esta situación, indicando que si bien dos tercios de los saltos resultaban tomados, el 30% de ellos eran incondicionales, incluyendo llamadas a funciones y su retorno. De acuerdo a investigaciones actuales no hay una clara primacía de una determinada resolución del salto por sobre la otra. Debido a eso, se buscaron algunas nuevas alternativas a la estrategia *salto tomado*, como ser:

- tomar ventaja del código de operación y hacer la predicción en base al tipo de instrucción de salto que se trate.
- el hecho que muchos saltos hacia atrás son tomados, ya que corresponden a la finalización de una iteración del tipo *for(...)*, mientras que los saltos hacia adelante sólo se emplea en situaciones muy especiales que no son el caso común.

Con estas variantes tampoco se consiguieron resultados con un gran impacto en el porcentaje de aciertos. En cambio con el uso de perfiles (profiling) si se consiguen algunas mejoras pero, debido al esfuerzo que requiere esta técnica, es solamente aplicable al software que lo justifique.
La ventaja más clara que tiene la predicción estática es en el caso de los procesadores segmentados, ya que implementar la predicción *no tomado* no tiene costo. La desventaja es que va en contra de las estadísticas que indican que el 75% de los saltos son tomados, en consecuencia el precio se paga con los ciclos perdidos debido a la gran cantidad de malas predicciones.

3 Características de los predictores

Desde un punto de vista general, un predictor normalmente está acompañado de una *fuente de eventos*, sobre los cuales es necesario hace algún análisis para realizar la

predicción y de un *mecanismo de retroalimentación* que permita evaluar su desempeño. En el caso específico de los predictores de saltos, tenemos:

- Origen de los eventos, durante la ejecución del programa los saltos se van produciendo de una manera dinámica.
- Selección de eventos, existen diferentes instrucciones que pueden producir ruptura de secuencia: branches, retorno de funciones, etc.
- Identificación de recursos, los distintos esquemas de predicción requieren el acceso a distintas tablas y para determinar la entrada correcta en las tablas se usa: la historia del salto, o el valor del PC, o el comportamiento de otros saltos, o bien una combinación de esos.
- Mecanismo de predicción, que puede ser estático (en cuyo caso no se requiere el paso previo de identificar recursos), pero el caso más habitual es que sea dinámico y su comportamiento esté representado por una autómita finito (contador), donde las transiciones ocurren bajo determinadas condiciones, pero también pueden darse con una cierta probabilidad (procesos Markov) cuando se presenta algún tipo de relación entre las condiciones que deben ser evaluadas por los saltos.
- Retroalimentación y recuperación, el resultado verdadero del salto se conoce algunos ciclos después de realizar la predicción. Si la predicción fue correcta, el mecanismo de retroalimentación debe reforzar la confianza del predictor. Cuando hay un fallo en la predicción, no solo se debe cambiar el estado del predictor, sino también intentar corregir el error.

En este trabajo se describen tres tipos de predictores: *locales*, *globales* y *combinados*, los cuales fueron investigados por distintos grupos y muchas de sus conclusiones sirvieron de sustento para su implentación actual. Algunas de las publicaciones a las que se hace referencia en esta recopilación, son trabajos seminales de comienzos de los noventa que introdujeron ideas, por entonces novedosas, las cuales constituyen la base para trabajos más actuales proponiendo mejoras en algunos casos, en otros complementándolas e incluso cuestionándolas en otros.

3.1 Predictores locales

Esta clase de predictores se caracterizan porque usan solamente información relacionada a la instrucción de salto sobre la cual se hace la predicción. Esta información puede ser el valor de algunos bits del PC actual, o la historia de su comportamiento reciente.

3.1.1 Local con contadores

El elemento principal en estos predictores es un contador ascendente/descendente provisto con una cierta cantidad de bits. Se asocia un contador de N bits con un determinado salto[4]. Cuando el salto va a ser ejecutado, el valor del contador C es usado para hacer

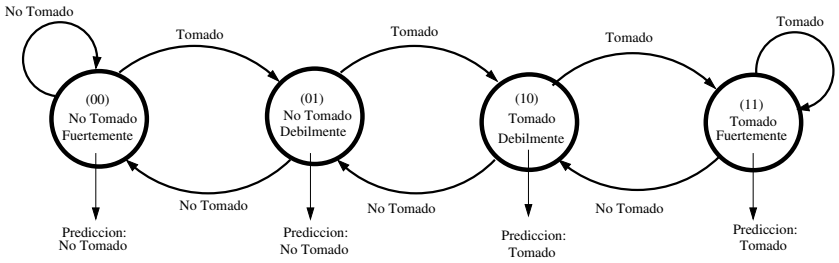

Figure 1: FSM para un contador de dos bits

la predicción. Si C es mayor o igual a un límite predeterminado L, el salto se predice como tomado, en otro caso se predice como no tomado. Un valor típico para L es 2^{N-1}. El contador C es actualizado cada vez que el salto queda resuelto. Si el salto es tomado, el valor de C es incrementado en uno, en otro caso es decrementado en uno. Si C alcanza el valor $2^N - 1$ permanecerá con ese valor mientras el salto sea tomado. Si el valor de C es 0, permanecerá con ese valor mientras el salto sea no tomado. La figura 1 muestra la máquina de estados finita asociada con un predictor de 2 bits.

El principal problema que se presenta en este caso es cuando distintos saltos, debido a limitaciones en el tamaño de la tabla de contadores o por emplear un sistema de indexación deficiente, se ven obligados a compartir contadores. Este problema se conoce como *colisión* (o *aliasing* en inglés). Pese a ello, numerosos experimentos realizados han demostrado que se consiguen buenos resultados con poca demanda de recursos, solamente dos bits por contador.

3.1.2 Local con modelo

Una forma de mejorar el predictor con contadores es reconocer que muchos branches siguen modelos o patrones repetitivos. Quizás el ejemplo más sencillo son los saltos que controlan las repeticiones, como lo muestra el siguiente códio assembly obtenido al traducir el *loop for* de C que aprece en la línea 2:

```
1     #la siguiente línea en C
2     #  for(i=1;i<=4;i++) { }
3     #se traduce como:
4            movl    $1, -4(%ebp)
5            jmp    .L2
6     .L3:   addl    $1, -4(%ebp)
7     .L2:   cmpl    $4, -4(%ebp)
8            jle    .L3      # se evalúa 5 veces: 4 tomados y 1 no tomado
```

El salto de la línea 8 es evaluado 5 veces por cada ejecucción de la repetición y claramente sigue un modelo: en las 4 primeras ejecuciones el salto es *tomado*(1) seguido por un *no tomado*(0). Este modelo es conocido como la *historia* del salto.

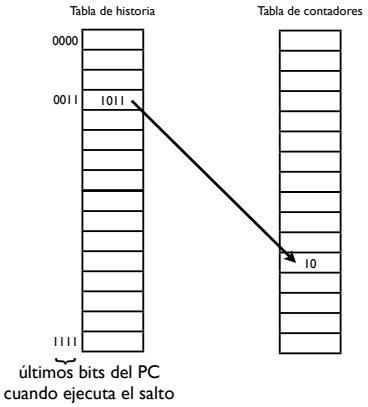

Figure 2: Tablas para el salto con modelo

En este caso, cuando la longitud de la historia alcanza los 4 bits se puede predecir como será el resultado de la próxima ejecución. Con una historia de "1111" el salto se puede predecir correctamente como *no tomado*.
Una forma de crear este tipo de predictores es mediante dos tablas: una donde se mantiene la historia de los saltos, la cual es indexada por los últimos bits del PC al momento de ejecutar el salto, y otra tabla donde se mantienen los contadores (idénticos a los vistos más arriba) que son accedidos según el modelo de historia que tenga el salto. La figura 2 muestra ambas tablas para el caso que tengan 2^4 elementos cada una y los bits menos significativos del PC al momento que se ejecuta el salto son "0011". Considerando los valores que se muestra en la figura, la predicción es que el salto será tomado. Continuando con el ejemplo, es claro que cada ejecución de la iteración hace variar el valor de i desde 1 hasta 5. Inicialmente la entrada "0011" de la tabla de historia tendrá el valor "0000" y luego irá "aprendiendo" un modelo de historia con las sucesivas ejecuciones. La primer parte de esta adaptación se muestra en una tabla que tiene en la primer columna los valores que toma la variable i, en la segunda columna el índice de contador que le corresponde al salto en cuestión y en la tercer columna la operación que se hace sobre el contador.

valor de i	indice del contador	operación
1	0000 (0)	+
2	0001 (1)	+
3	0011 (3)	+
4	0111 (7)	+
5	1111 (15)	-

Las próximas ejecuciones de la iteración ya tendrán una historia que es de utilidad para hacer la predicción y de ahora en más se usará un nuevo conjunto de contadores para hacer las predicciones que siguen, las cuales son todas correctas,

valor de i	indice del contador	operación
1	1110 (14)	+
2	1101 (13)	+
3	1011 (11)	+
4	0111 (7)	+
5	1111(15)	-

Hay que destacar que en total se usan 8 contadores distintos, pero los tres primeros (0000, 0001 y 0011) se usan solamente en la primera ejecución de la iteración. El resto de las veces usarán los contadores 1110, 1101, 10111, 0111, 1111 y esa secuencia se vuelve a repetir. Cuando se repite muchas veces la iteración hace que los distintos contadores lleguen a los valores correctos, debido a las operaciones de incremento y decremento que se hacen sobre ellos. Luego de algunas repeticiones, claramente las cuatro primeras ejecuciones del salto se predecirán como *tomado* y la quinta como *no tomado*.
Es necesario aclarar que no se ha considerado el hecho que pueden existir otros saltos que compartan algunas de las entradas en la tabla de historia, es decir todos aquellos que al ejecutar tienen un valor de PC que termina en "0011", por lo que entre sí se alteran el modelo del salto. También puede ocurrir que existan otros saltos que hayan generado la misma historia y por lo tanto compartan los contadores, en este caso también puede haber interferencias entre los comportamientos de los saltos.

3.2 Predictores globales

Los predictores locales sólo consideran el comportamiento de cada salto particular, en cambio los predictores globales tienen en cuenta otros saltos que han sido ejecutados previamente. Existen dos razones que justifican el uso de predictores globales:

- Cuando se evalúan condiciones distintas que están relacionadas entre sí. Puede darse la siguiente situación: una condición de un salto es $i < 0$ y otro salto consulta por $i >= 0$, si el valor de i no cambia entre ellos y se conoce que uno fue tomado, se puede predecir que el otro no será tomado.

- Cuando la historia global incluye a la historia local. Esta situación ocurre, por ejemplo, cuando hay iteraciones anidadas. Si el salto que controla la iteración interna sigue un modelo local M_l, el cual permite predecir exactamente su comportamiento y si M_l es un substring dentro del modelo global M_g, quiere decir que la predicción realizada con M_g será al menos tan buena como la hecha con M_l .

Para implementar este esquema se requiere mantener el comportamiento real de los saltos que se van ejecutando. Algunas de las variantes para mantener ese comportamiento son, por ejemplo, usar un registro de desplazamiento compartido por todos los saltos, o tablas que mantienen los comportamientos característicos que se establecen.

3.2.1 Global simple

La forma más sencilla de implementar un predictor global es manteniendo un registro de desplazamiento (RD) de n bits, el cual mantiene el comportamiento de los últimos n

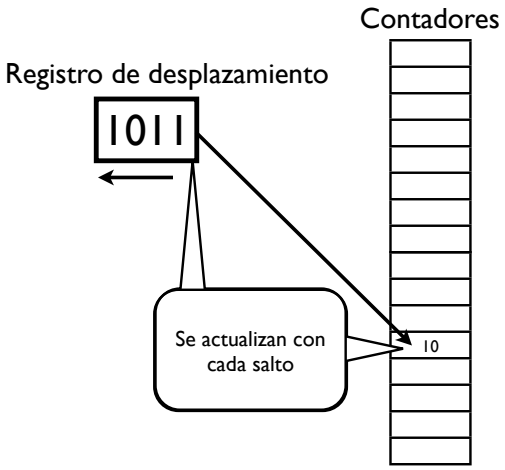

Figure 3: Predictor Global Simple

saltos condicional. Con la ejecución de cada salto se desplazan a izquierda los bits del RD colocando como bit menos significativo un uno si el salto fue tomado o un cero en otro caso. En la figura 3 se muestran los elementos necesarios.

3.2.2 Global según PC

En [4] se propone hacer la predicción de un salto teniendo en cuenta el *camino* recorrido hasta llegar al salto. Este camino es determinado por el comportamiento que han tenido algunos saltos ejecutados previamente. Para hacer la predicción de un salto individual se utilizan varios *contadores* de N bits, similares a los vistos en la sección 3.1.1, son necesarios uno para cada camino posible. usa una combinación de dos componentes, la dirección en que se encuentra el salto (PC) y el registro global de saltos (sección 3.2.2). En este esquema, se deben concatenar algunos bits de cada componente [1]. Aquí existen algunas alternativas posibles variando el número de bits que se usen de cada componente. De esa manera se aprovecha la correlación que puede existir entre algunos saltos y hacer la predicción para un nuevo salto. Si consideramos tres saltos b_1, b_2, b_3 donde los caminos posibles para llegar a b_3 son determinados por b_1 y b_2, quiere decir que hay 4 caminos o *sub-historias* para arribar a b_3. La resolución de b_1 y b_2, como *tomado* o *no tomado*, son los *pasos* considerados para la predicción. En el caso de b_3 son 2 pasos, pero podría ser cualquier cantidad M de pasos. El contador que corresponda ser empleado se utiliza igual que los contadores convencionales, si el bit más significativo del contador es uno, la predicción es tomado, si es cero será no tomado. La actualización también es igual, incrementando cuando el salto es tomado y decremento cuando no se toma.
Este esquema de predicción de saltos basado en correlación se caracteriza mediante el par (M,N), donde M es el número de pasos involucrados en la predicción y N es la

[1] Dadas dos cadenas $S_1 = s_{1,1} \dots s_{1,i}$ y $S_2 = s_{2,1} \dots s_{2,j}$, la operación de concatenación $S_1 : S_2$ genera una nueva cadena $S_3 = s_{1,1} \dots s_{1,i} s_{2,1} \dots s_{2,j}$ de $i + j$ caracteres.

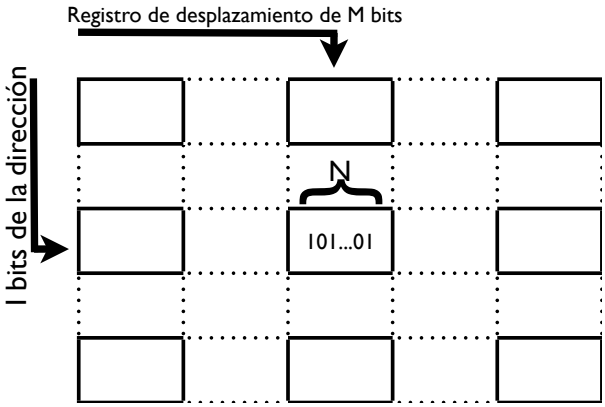

Figure 4: Tabla de predictores de salto

cantidad de bits requeridos por los contadores empleados en cada camino. El sustento de esta técnica es que, en algunas ocasiones, se puede hacer una mejor predicción del salto considerando cada sub-historia de manera individual.
Para implementar la predicción basada en correlación, se emplea una tabla, BPT, que contiene los bits de predicción. La dirección del salto determina su correspondiente entrada en la tabla y con un registro de desplazamiento se indica el conjunto de N bits usados como contador. La cantidad de bits requeridas por el registro de desplazamiento es determinado por el número M de pasos tenidos en cuenta. En general una BPT para I saltos requiere $N \times 2^{I+M}$ bits, organizados en I entradas, cada una con 2^M conjuntos de N bits de predicción. La selección del contador correcto se hace mediante el registro de desplazamiento, ya que cada uno de sus bits se coloca en uno o cero dependiendo de la dirección seguida (tomado o no tomado) en cada salto que forma el camino. La figura 4 muestra una visión genérica de una tabla de predicción de saltos. La BPT se almacena en una memoria capaz de direccionar un total de 2^{I+M} predictores de N bits. Se han propuesto algunas variantes para su implementación, variando la cantidad de bits tomados desde la dirección del salto (I), desder el registro de desplazamiento(M) y la cantidad de bits del predictor (N).

3.2.3 Global con índice según hasing

La historia global H_g no siempre logra identificar completamente al salto que se debe predecir, posiblemente debido a que existe una alta redundancia al generar el índice del contador correspondiente. Para solucionar esta redundancia una propuesta interesante es combinar los bits de H_g con los bits de la dirección del salto. Ver figura 5. Es decir, considerando que la dirección del salto tiene la virtud de identificar univocamente al salto, parece interesante incorporar algo de esa virtud en la historia H_g. Se espera que la operación XOR entre los bits de PC y H_g generen más información que cualquiera de las dos componentes por separado.
En la siguiente tabla se puede ver como el hash genera mayor disperción en el índice

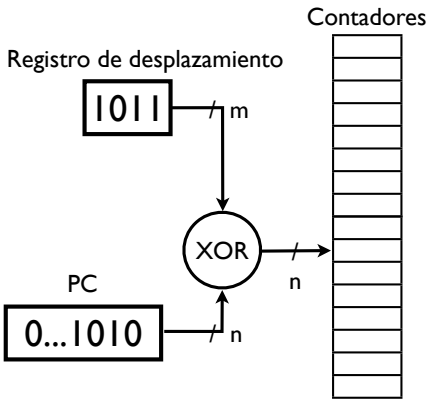

Figure 5: Predictores con hashing

generado. Las dos primera filas corresponden a un único salto que tiene asociado dos historias global (dos H_g's), las otras dos filas corresponden a otro salto que también tiene asociado dos historias global. Observando las dos columnas finales de la tabla quedan claras las diferencias entre la técnica global considerada más arriba (sección 3.2.2) y esta que aplica hash. Usando predicción global con PC no es posible distinguir los cuatro casos posibles, en cambio la predicción global con hash si lo consigue. Eso se debe a que en el primer caso se concatenan cuatro bits del PC y cuatro bits del H_g, en cambio con hash se toman 8 bits de cada componente.

PC	H_g	PC(4):H_g(4)	PC(8) xor H_g(8)
00000000	00000001	00000001	00000001
00000000	00000000	00000000	00000000
11111111	00000000	11110000	11111111
11111111	10000000	11110000	01111111

3.3 Predictores combinados

Es posible encontrar dos motivos para la potencial falla en los predictores vistos hasta ahora. La primera es debido a que los predictores tiene una capacidad limitada de memoria, por lo cual no se mantiene información sobre todos los saltos o pueden ocurrir situaciones donde diferentes saltos deben compartir algunos elementos en las tablas, lo que genera interferencias en el comportamiento. La otra causa es debido a la tensión que se genera entre la predicción hecha en base a la correlación entre distintos saltos (predicción global) y la predicción hecha a partir del modelo de comportamiento de un salto en particular (predicción local). También es necesario tener en cuenta que la predicción depende en parte de las caraterísticas que tengan los programas. En los programas que usan aritmética entera funciona mejor un predictor global, entonces resulta conveniente que el predictor se adecue a las características del programa para tener los mejores resultados.

3.3.1 Combinado general

Teniendo en cuenta que existen diferentes esquemas para hacer la predicción de saltos y que cada esquema tiene sus ventajas, parece razonable contar con predictores que combinen dos técnicas diferentes para realizar el trabajo. Si las alternativas para hacer la predicción son P_a y P_b, cuando llegue el momento, es necesario determinar cual de las dos aplicar. Para elegir el predictor adecuado se requiere mantener un par de bits, CP_a y CP_b para cada salto. Un bit por cada predictor considerado. Estos bits se actualizan de una manera similar a la asignación condicional que se usa en el lenguaje C: si la condición evaluada es verdadera se asigna el primer valor, si es falsa el segundo. Sería similar a lo siguiente:

- CP_a:=(predicción de P_a correcta)? 1 : 0
- CP_b:=(predicción de P_b correcta)? 1 : 0

También se requiere de un contador adicional que es idéntico al considerado en la sección 3.1.1, usa dos bits y se satura al llegar a los valores extremos (no se incrementa si alcanza el valor 3 y no se decrementa si tiene el valor 0). La siguiente tabla muestra como se calcula el valor que hay que sumar al contador para realizar su actualización (Columna CP_a-CP_b). La selección se hace en función del bit más significativo del contador: si es

CP_a	CP_b	CP_a-CP_b	Actualización
0	0	0	Sin cambios
0	1	-1	Decrementar el contador
1	0	1	Incrementar el contador
1	1	0	Sin cambios

Table 1: Valores para actualizar el contador

1 se usa la predicción de P_a y si es 0, la predicción de P_b. En la figura 6 se observa que el bit más significativo del contador controla un multiplexor que deja pasar la mejor de las dos predicciones.

A continuación se describe el funcionamiento del contador que usa el juez para tomar su decisión. Los bits de comportamiento de los predictores CP_a y CP_a se actualizan luego de cada salto, con un 1 si acertó y con un 0 si erró. También se actualiza el contador que usa el juez para tomar su dictamen; el autómata de la figura 7 resume el funcionamiento del contador. Como se muestra en la tabla 1, el valor para sumar al contador puede ser 0,1,-1. Estos valores se obtienen como resultado de la operación $CP_a - CP_b$. Los cambios de estados siguen las siguientes reglas:

- Cuando la condición $\overline{CP_a \ XOR \ CP_b}$ se hace verdadera, los comportamientos son iguales y no se actualiza el contador.
- Cuando la condición $CP_a \ AND \ \overline{CP_b}$ es verdadera, el contador se incrementa.

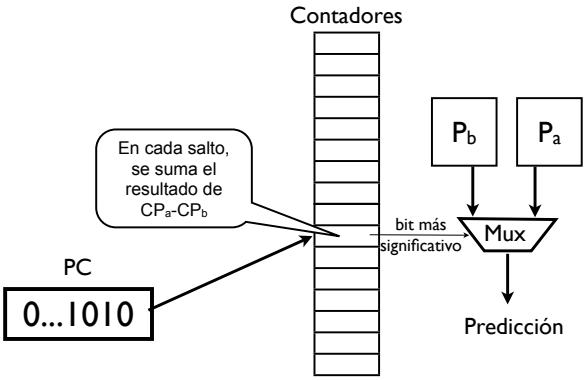

Figure 6: Combinación de predictores

- Cuando la condición $\overline{CP_a}\ AND\ CP_b$ es verdadera, el contador se decrementa.

Hay que destacar que cuando el contador vale 0, solamente cambia si se debe incrementar, algo similar ocurre cuando el valor del contador es 3, solamente puede ser decrementado. Si consideramos al autómata de la figura 7 como una máquina de Moore, la salida del circuito es el bit que aparece en el centro de los nodos y es precisamente la decisión del juez: un 0 da ganador al predictor local y un 1 da ganador al predictor global.

3.3.2 El predictor del ALPHA 21264

Esta arquitectura superescalar, que llega a despachar hasta 4 instrucciones por ciclo, emplea una predicción de saltos que combina la técnica *modelo local* (sección 3.1.2) y la *global simple* (sección 3.2.1) [2]. La predicción se basa en la existencia de una correlación combinada tanto local como global de los branches. Algunas de las razones que hacen necesario que el alpha 21264 posea un buen esquema de predicción de saltos, son:

- El hecho que esta arquitectura cuenta con un pipe profundo, implica que los riesgos de control tengan una alta penalidad de al menos 7 ciclos.

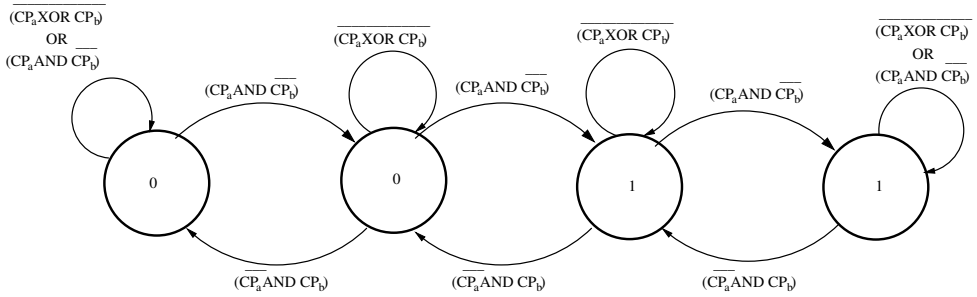

Figure 7: FSM contador del juez

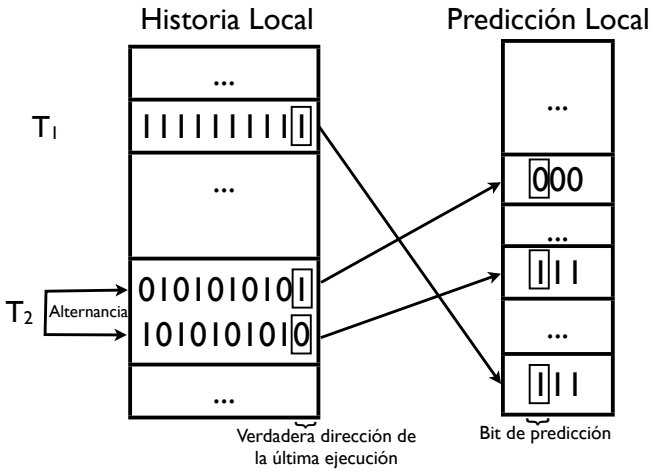

Figure 8: Predicción local de dos niveles

- Las mejoras introducidas para acelerar la ejecución de las instrucciones aumenta la demanda de instrucciones.
- Esta implementación dispone de hardware para la ejecución especulativa de hebras y este hardware puede ser empleado en la predicción de saltos.

Hay que tener en cuenta que la ventana de instrucciones de este procesador puede tener hasta 80 instrucciones en consideración, lo que brinda muchas posibilidades para explotar el paralelismo a nivel de instrucciones. Por esa razón implementa un esquema de predicción de salto sofisticado en el que compiten las aproximaciones local y global. La selección entre las dos alternativas se hace de manera dinámica y se alcanza un porcentaje muy alto de efectividad en la predicción.
La correlación local requiere el uso de dos tablas (ver sección 3.1.2): la *historia* con los modelos y la *predicción* con los contadores.
Como ejemplos se pueden considerar dos tipos de saltos:

- Los que se comportan siempre del mismo modo, tipo T_1
- Aquellos que alternan la dirección tomada, tipo T_2

En la figura 8 se muestran ambas situaciones. A los saltos tipo T_1 (siempre tomados), le corresponde un modelo de historia con todos unos (111...11). Con suficiente entrenamiento, la entrada más alta en la tabla de predicción local aprende el comportamiento correcto para saltos T_1. Por su parte, los saltos tipo T_2 que varían en su comportamiento, alternan entre dos modelos o patrones y cada patrón tendrá su correspondiente contador en la tabla de predicción.

En cuanto a la correspondencia global, ésta tiene en cuenta el comportamiento que han tenido los saltos que ejecutaron recientemente y no la propia historia del salto a

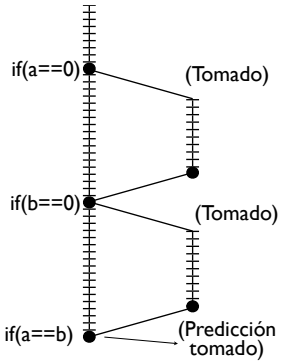

Figure 9: Correlación Global

predecir (ver sección 3.2.1). Aquí se construye un índice a partir de la verdadera dirección tomada por los últimos doce saltos ejecutados, mediante un registro de desplazamiento (RD). El RD se usa como índice para acceso a una tabla con 2^{12} entradas con contadores de dos bits con los que se hace la predicción.
La competencia entre el predictor local y el global requiere de un árbitro, encargado de escoger la mejor predicción, similar a los considerado en la sección anterior.
En la figura 9 se muestra como ejemplo una situación en la cual es conveniente aprovechar la correlación global para hacer la predicción. Cuando ambos saltos, $if(a == 0)$ y $if(b == 0)$, son tomados se puede anticipar que a y b son iguales y por lo tanto el salto $(a == b)$ será tomado. Esto quiere decir que si la historia pasada sigue un patrón XXXXXX11 (los dos últimos bits en uno), los contadores de predicción del alpha 21264 correspondientes a los índices XXXXXX11 llegarán a los valores necesarios para hacer la anticipación correcta, sólo se requiere realizar la cantidad suficiente de ejecuciones. Si bien en este caso una historia con una profundidad de dos bits es suficiente para anticipar el comportamiento del salto, hay casos más elaborados en los cuales se requiere mayor profundidad (hasta 12 bits, para el Alpha 21264). El ejemplo de la figura 9 refleja la utilidad de contar con la posibilidad de elegir entre distintas técnicas de predicción. El selector puede optar por la predicción global para $(a == b)$ cuando los saltos $if(a == 0)$ y $if(b == 0)$ hayan sido tomados y sino es así, quedarse con la opción local.

3.4 Comparación de predictores

En esta sección se hace una comparación de la eficacia que demuestran los predictores hasta ahora considerados. Las gráficas han sido tomadas de [3], en ellas se muestra como varía la efectividad de los predictores en función de la cantidad de memoria asignada. Es por eso que tienen en el eje de las X's el total de memoria disponible para las tablas y otros datos que usan los predictores, esto puede variar desde 32 bytes hasta 64 Kb. En el eje de las Y's se mide el porcentaje alcanzado en la exactitud de la predicción.
Para comenzar se analiza la performance de la técnica *local con contador* o *bimodal*(sección 3.1.1). En la figura 10 se puede observar como progresa su comportamiento cuando

dispone de más memoria, hasta que se satura al superar el 93%. A partir de un tamaño

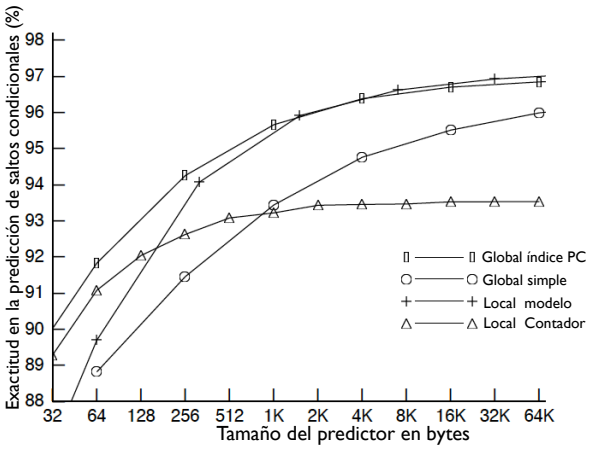

Figure 10: predictores globales vs predictores locales

de 1Kb ya no mejora, aun cuando disponga de más espacio. En la misma figura está la traza que corresponde a la otra variante local, aquella que usa un *modelo* (sección 3.1.2). Cuando los recursos son muy escasos, el uso de contadores supera a los modelos, pero más allá de los 128 bytes la situación se revierte de una manera notable.
El *global simple* (sección 3.2.1), que usa un único registro con la historia de los últimos saltos ejecutados, tiene un desempeño intermedio entre las dos técnicas locales cuando dispone al menos de 1Kb. La justificación para su buen desempeño ha sido analizada anteriormente. La cuarta curva de la figura 10 corresponde a otra predicción global denominada *global según PC* (sección 3.2.2) que mejora a las locales, aquí se han tomando los mejores valores obtenidos por esta técnica para los distintos tamaños del predictor. Según la cantidad de bits que se usen desde el PC y desde la historia, en realidad existen muchas curvas para esta técnica, la curva identificada como *Global índice PC* en la figura 10 es el mejor comportamiento para los distintos tamaños de memoria. Ahí se ve que este caso supera al predictor *local con modelo* cuando se dispone de poca memoria, hasta los 2Kb.
La variante global que entremezcla bits del PC y del registro de historia es la que presenta el mejor desempeño de todos, como se muestra en la figura 11. En el caso de los predictores muy pequeños, con menos de 128 bytes, el *Global índice hash* no supera al *Global índice PC* debido a que hay una gran sobrecarga en el uso de los contadores y el hecho de agregar información global sólo hace que empeore el desempeño.
En cuanto a los predictores combinados una pareja muy interesante es Contadores/Global Hash. Con este par se espera aprovechar la información global en las oportunidades que sea conveniente hacerlo, y sino se puede usar el comportamiento más habitual recurriendo a los contadores.
En la figura 12 se muestran dos técnicas combinadas, una global y una local. Es necesario aclarar que *Global Hash* corresponde a una implementación en la que se han tomado la

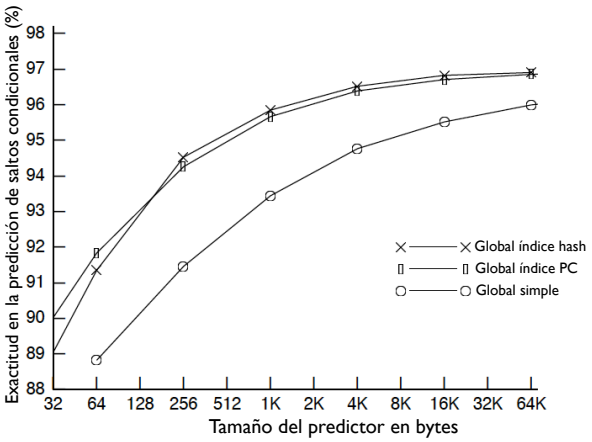

Figure 11: predictores globales

misma cantidad de bits desde el PC y desde la historia. De este modo se busca un equilibrio entre el peso dado a la información global y el peso dado a la dirección en que se encuentra el salto, sabiendo que llegado el caso se puede recurrir a la técnica local con contadores. Según lo que se reporta en [3], para los programas del SPEC'89 la fórmula *Contadores/Global Hash* se comporta mejor que sus componentes consideradas por separado. También se ha observado sobre ese lote de programas que en la gran mayoría de ellos el predictor local es el que se aplica mayoritariamente, eso quiere decir que el juez generalmente ha encontrado más conveniente aplicar predicción local antes que la global. En la figura 12 se muestra el desempeño de dos predictores combinados, uno local y uno global. Los dos combinados superan a los que usan una sola técnica y para hacer una comparación más directa entre los esquemas globales *Hash* y *Global PC*, en la combinación *Ncontadores/HashN+1* se asignan el doble de contadores a *HashN+1* que a *Ncontadores*. La performance de *Ncontadores/HashN+1* es significativamente mejor que *Global PC*, por ejemplo el porcentaje de predicciones correctas alcanzado por *Ncontadores/HashN+1* con 1kb recién se consigue con 16Kb cuando el predictor es *Global PC*.

4 Conclusiones

Para mantener el desempeño de un procesador segmentado en un nivel adecuado, es fundamental disponer de un buen predictor de saltos. Esto se hace aún más evidente en los procesadores superescalares, en los cuales la tendencia es hacer más profunda la segmentación y mayor el número de instrucciones despachadas por ciclo, lo que se conoce como ancho del pipe. En estos casos, los ciclos que se pierden por una mala predicción se deben a que esas instrucciones mal despachadas deberán ser descartadas. Esta es la principal razón por la cual se realizan numerosos trabajos de investigación en este área, con muy buenos resultados, lo que han permitido desarrollar predictores con un nivel

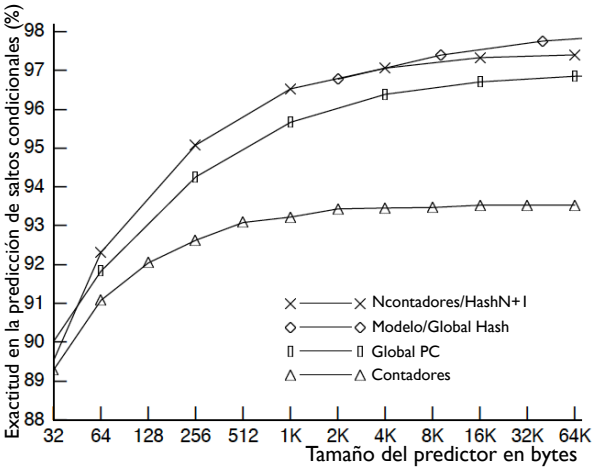

Figure 12: predictores combinados

de exactitud por encima del 95% en los programas enteros y del 97% en los programas científicos.
Hay un elemento que es común a la mayoría de los esquemas de predicción: el contador de dos bits con saturación en los valores extremos. Este constituye el bloque básido para predecir el comportamiento de la próxima instancia del salto, pero como se ha visto, tiene sus limitaciones debido principalmente al problema de colisiones producidas por restricciones en los tamaños de las tablas. Debido a este problema la mayoría de la nuevas propuestas emplean información local y también información global. La información global se obtiene a partir del comportamiento que han mostrado los saltos previos. Ambos tipos de información generalmente se mantienen mediante dos niveles de tablas, una que registra la historia y otra con los contadores. Finalmente los más sofisticados son aquellos en los cuales compiten dos técnicas distintas y se elige dinámicamente la más adecuada según las circunstancias; también existen propuestas para aprovechar el hardware con capacidad de ejecución especulativa para hacer la predicción de saltos.

References

[1] Jean-Loup Baer. *Microprocessor architecture: from simple pipelines to chip multiprocessors*. Cambridge University Press, pub-CAMBRIDGE:adr, 2010.

[2] Richard E Kessler. The alpha 21264 microprocessor. *Micro, IEEE*, 19(2):24–36, 1999.

[3] Scott McFarling. Combining branch predictors. Technical report, Technical Report TN-36, Digital Western Research Laboratory, 1993.

[4] Shien-Tai Pan, Kimming So, and Joseph T Rahmeh. Improving the accuracy of dynamic branch prediction using branch correlation. In *ACM SIGPLAN Notices*, volume 27, pages 76–84. ACM, 1992.