Tema 11. Riesgos de control y predicción de saltos

Organización de Computadores

LUIS ENRIQUE MORENO LORENTE
RAÚL PÉRULA MARTÍNEZ
ALBERTO BRUNETE GONZALEZ
DOMINGO MIGUEL GUINEA GARCIA ALEGRE
CESAR AUGUSTO ARISMENDI GUTIERREZ
JOSÉ CARLOS CASTILLO MONTOYA

Departamento de Ingeniería de Sistemas y Automática

1. REVISIÓN: EVALUACIÓN DE LA PREDICCIÓN DE SALTO

- Dos estrategias
  - Estática: Suponer que los saltos hacia atrás se efectúan y los saltos hacia adelante no
  - Dinámica: Predicción basada en perfiles: se guarda el comportamiento del salto, y se hace una predicción basada en ejecuciones previas
- Las instrucciones entre saltos mal predichos constituyen una buena métrica de los errores de predicción

REVISIÓN: SUMARIO DE CONCEPTOS PREVIOS

- Los riesgos limitan las prestaciones
  - Estructural: requieren más recursos HW
  - Datos: necesitan forwarding, planificación de código (compilador)
  - Control: evaluación temprana & PC, saltos retardados, predicción
- El incremento de la longitud del cauce aumenta el impacto de los riesgos; la segmentación ayuda con el ancho de banda de instrucciones, pero no con la latencia
- Interrupciones, juegos de instrucciones, operaciones FP complican la segmentación
- Compiladores reducen el coste de los riesgos de datos y de control
  - Load delay slots
  - Branch delay slots
  - Predicción de saltos
- Hoy: Cauces más largos => mejor predicción de saltos, mayor paralelismo de instrucción?

REVISIÓN DE CONCEPTOS

- La ejecución de instrucciones está segmentada
  - Múltiples instrucciones son ejecutadas en paralelo
- Los riesgos en el cauce afectan a las prestaciones de la CPU
  - Riesgos estructurales: la CPU no puede ejecutar la secuencia específica
  - Riesgos de datos: las dependencias de datos entre instrucciones en el cauce.
  - Riesgos de control: las instrucciones branch pueden cambiar el PC
- Soluciones para los riesgos de datos
  - Crear secuencias no-dependientes
  - Esperar hasta que instrucciones previas se completen (se crean burbujas)
  - Anticipar los nuevos valores de registros a las próximas instrucciones
  - Partir el banco de registros: escribir primero, leer después
  - Optimizar el código (scheduling)

RIESGO PROVOCADO POR UN BRANCH

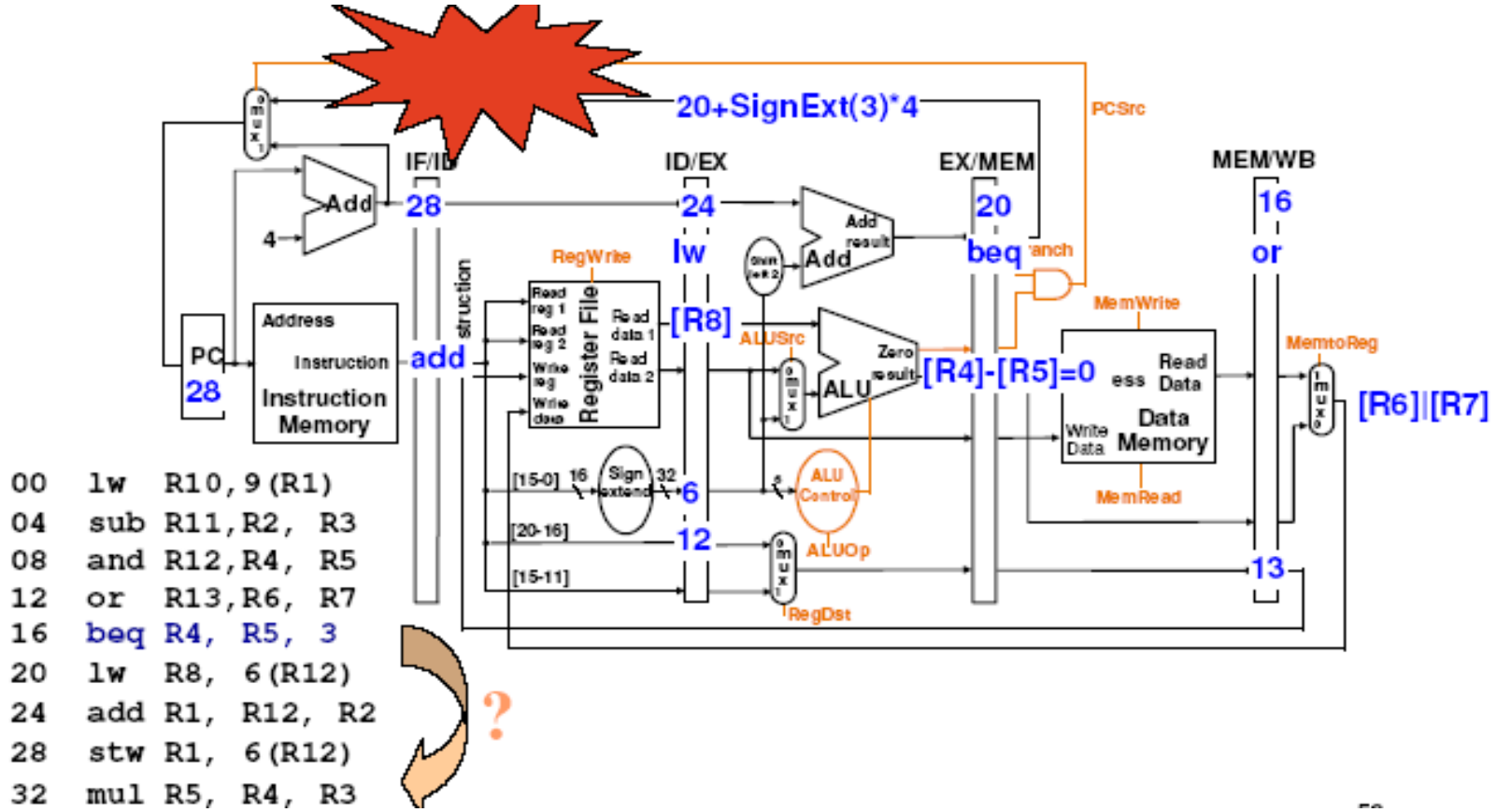


Imagen de Hennessy-Patterson, 1997

RIESGOS DE CONTROL

- Cuando una instrucción branch es buscada, no se sabe dónde continuará
  - Dirección de salto (puede efectuarse o no) nos es desconocido
  - Objetivo del salto (en caso de que se salte) la dirección efectiva de salto nos resulta desconocida
- Solución más simple
  - Detener el cauce hasta que se ejecute la instrucción branch

SALTOS Y PRESTACIONES

- MPI: miss-per-instruction

$$MPI = \frac{\text{número de predicciones incorrectas}}{\text{número total de instrucciones}}$$

- MPI expresa bien las prestaciones. Por ejemplo:
  - MPI=1% => (1 de cada 100 instr , aprox 1 de cada 20 saltos)
  - IPC=2 (IPC número promedio de instr ejecutadas por ciclo de reloj)
  - Penalización por vaciado del cauce de 10 ciclos
- Obtenemos que:
  - MPI=1% => 1 vaciado cada 100 instr
  - Un vaciado cada 50 ciclos (dado que IPC=2)
  - 10 ciclos de penalización por vaciado cada 50 ciclos
  - 20% en las prestaciones

TIPOS DE SALTOS

- Condicionales / incondicionales
  - Condicional:
    - Se necesita predecir dirección y objetivo del salto
    - La dirección real sólo se conoce después de ejecutar la instrucción.
    - Una predicción incorrecta origina un vaciado total del cauce
  - Incondicional:
    - Sólo se necesita predecir el objetivo del salto
- Directos / indirectos
  - Directos:
    - El objetivo del salto está especificado dentro de la instrucción (como un valor inmediato)
    - La dirección objetivo real se conoce después de descodificar
    - Una predicción incorrecta origina un vaciado parcial del cauce
  - Indirectos:
    - La dirección objetivo tiene que ser calculada
    - La dirección objetivo real se conoce después de la etapa de ejecución
    - Una predicción incorrecta origina un vaciado total del cauce

TIPOS DE SALTOS: ALGUNOS DATOS

- Condicionales directos: 70% de todos los saltos
  - Son muy frecuentes
  - La penalización por predicción incorrecta de la dirección es alta (vaciado completo)
  - Es muy importante predecir correctamente la dirección del salto condicional
    - Depende del flag de estado: overflow, signo, cero, paridad, acarreo, auxiliar
    - Dirección: el objetivo se conoce en la etapa de descodificación
    - Relativo al puntero de instrucción de la instrucción de salto

- Incondicionales directos
  - Se efectúan siempre
  - La dirección objetivo se conoce en la etapa de descodificación
  - Es menos frecuente y la penalización por redirección errónea es baja (vaciado parcial)
  - Tiene menor importancia una predicción incorrecta

TIPOS DE SALTOS: ALGUNOS DATOS

- Condicionales indirectos:
  - Muchas máquinas no los tienen
- Incondicionales indirectos
  - La dirección objetivo es el valor de un registro => se conoce en la de ejecución
    - Los returns son saltos incondicionales indirectos
    - Las calls pueden ser también incondicionales indirectos
  - Son poco frecuentes y la penalización por predicción errónea es alta (vaciado completo)
  - A veces es importante el predecirlos correctamente

MEJORA DE LA GESTIÓN DE LOS SALTOS

Determinar si el branch es tomado o no lo antes posible

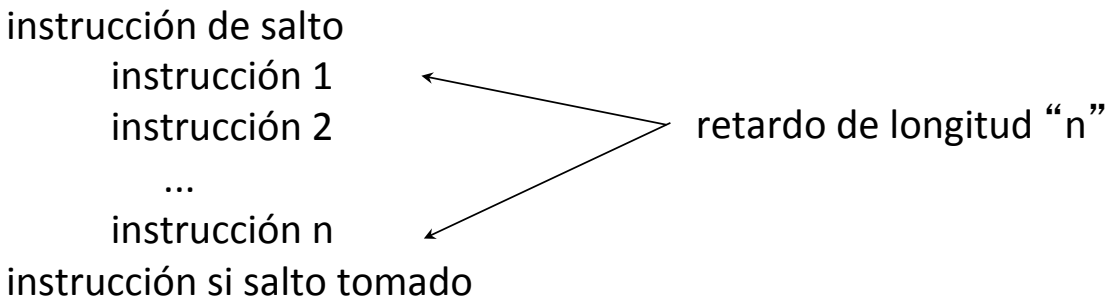
Calcular la dirección del salto anticipadamente

RIESGOS DE CONTROL: ALTERNATIVAS

- Detener hasta que la dirección de salto este clara
- Predecir que el salto no se va a efectuar
  - Ejecutar las instrucciones siguientes en la secuencia
    - PC+4 ya está calculado, se utiliza para obtener la siguiente instrucción
  - Eliminar las instrucciones introducidas en el cauce en el caso de que el salto haya que efectuarlo
  - 47% de los saltos en el proc MIPS no se efectúan (en promedio)
- Predecir que el salto se va a efectuar
  - 53% de los saltos en el proc MIPS se efectúan (en promedio)
  - Pero en el caso de este proc. La dirección objetivo del salto no tiene que calcularse.
    - MIPS sólo tiene una penalización de 1 ciclo por salto
    - Otras máquinas: el objetivo del salto es conocido antes de que finalice la instrucción

RIESGOS DE CONTROL: ALTERNATIVAS (CONT)

- Saltos retardados (delayed branch)
  - Se rellena con instrucciones no dependientes situadas antes de la instrucción de salto la burbuja que genera el salto.
  - Efectividad de esta técnica (la realiza el compilador) para una ventana de 1 ciclo (branch delay slot = 1)
    - El compilador es capaz de rellenar alrededor de un 60% de estas ventanas de retardo
    - Alrededor del 80% de las instrucciones ejecutadas en estas ventanas de retardo son instrucciones útiles en los cálculos
    - Aproximadamente el 50% (60%x80%) de los slots son adecuadamente rellenados



IMPACTO DE LAS DETENCIONES

$$\text{Pipeline Speedup} = \frac{\text{Pipeline depth}}{1 + \text{Branch frecuency} \times \text{Branch penalty}}$$

Optimización	Branch penal.	CPI	speedup v. nosegment	speedup v. stall	tipo
Detención	3	1.42	3.52	1.0	
Predic. taken	3 o 0	1.27	3.93	1.11	
Predic. not taken	3 o 1	1.24	4.04	1.14	
Delayed branch	0.5	1.07	4.6	1.31	

PREDICCIÓN DEL SALTO

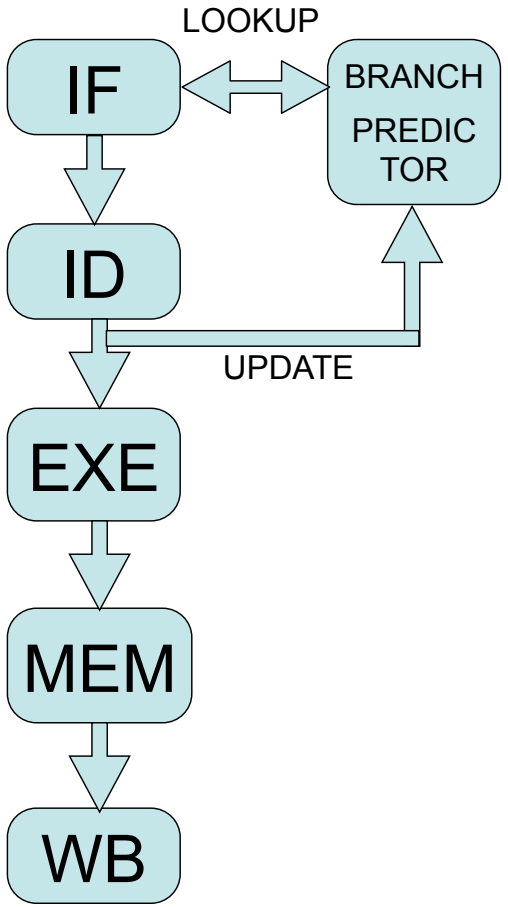
- Para disminuir el efecto sobre las prestaciones de los riesgos de control se utiliza la predicción
  - En la etapa IF (instr fetch) es necesario predecir la dirección y el objetivo del salto
- La predicción puede ser:
  - Estática:
    - Siempre se predice si el salto es efectuado o no
  - Dinámica:
    - La predicción se basa en la historia previa del salto

PREDICCIÓN DINÁMICA DE SALTOS

- Motivo: los saltos tienden a ser consistentes
  - Por ejemplo: bucles, condiciones de error, tratamiento de casos límite
- Idea: predecir dinámicamente si un salto va a ser efectuado o no de acuerdo con la historia previa del salto
  - Predicción:
    - Una vez que el salto es descodificado, la búsqueda continua por uno de sus caminos posibles
  - Predicción incorrecta:
    - Si la ejecución del salto muestra que el camino predicho es incorrecto, es necesario seguir por el otro
      - Se vacían todas las instrucciones del camino incorrecto
      - Los errores de predicción tienen un impacto muy importante en las prestaciones, este impacto depende de:
        - Profundidad de la segmentación y de la frecuencia de las instrucciones de salto

PREDICCIÓN DINÁMICA DE SALTOS

- Algoritmo con dos fases:
  - Fase de predicción:
    - Se realiza durante la etapa de búsqueda
    - Se realiza de acuerdo con la historia del salto
  - Fase de actualización
    - Se realiza durante la etapa de ejecución
    - Se realiza de acuerdo con la predicción y el resultado



2. TÉCNICAS DE PREDICCIÓN DE SALTOS

- La predicción de saltos condicionales requiere:
  - Predecir si se tomará el salto
  - Predecir la dirección destino del salto

- Las técnicas se pueden clasificar en dos grandes grupos:
  - Predicción estática. Son predictores que no utilizan ningún tipo de información en tiempo de ejecución sobre el comportamiento anterior de los saltos.
  - Predicción dinámica. Son predictores que pueden supervisar el comportamiento de los saltos mientras se ejecutan y realizan predicciones en función de las observaciones .

2.1. TÉCNICAS DE PREDICCIÓN ESTÁTICA DE SALTOS

- Suelen ser técnicas simples, pero limitadas en su eficiencia.
- Existen dos grandes grupos:
  - Predicción estática basada en reglas.
    - Usan reglas predeterminadas.
  - Predicción estática basada en perfiles.
    - Las técnicas basadas en perfiles se basan en la posibilidad de aproximar el comportamiento de un salto mediante distintas ejecuciones del programa sobre datos de prueba.
    - Pueden aprovechar también la información de alto nivel disponible en tiempo de compilación.
    - Pueden conseguir mejores rendimientos que las basadas en reglas.
    - Su desventaja es que al ser creadas en tiempo de compilación es necesario recompilar de nuevo para cambiarlas.
    - Si las estadísticas no están bien realizadas la predicción puede ser mala.

PREDICCIÓN ESTÁTICA BASADA EN REGLAS

- Existen diversas estrategias, entre ellas:
  - Predicción en una única dirección.
  - Predicción BTFNT (backwards taken/ forward not taken)
  - Predicción heurística basada en el programa [Ball-Larus, 1993]

PREDICCIÓN EN UNA ÚNICA DIRECCIÓN

- Consiste en predecir que la dirección de todos los saltos va siempre en la misma dirección (es decir siempre tomado o siempre no tomado).
- Se basa en que estadísticamente los saltos suelen ser tomados con más frecuencia (60%) que no tomados (40%).
  - El Intel 486 (1997) usaba la estrategia de predecir siempre que el salto no se toma porque simplifica la estrategia de extraer la instrucción, ya que se comienzan a ejecutar de forma automática las instrucciones situadas después del salto condicional.
  - La estrategia opuesta, suponer que el salto se tomará siempre tiene mejor tasa de aciertos que la anterior, pero requiere un hardware más complejo ya que la dirección de destino del salto no suele estar disponible cuando se hace la predicción.
- Siempre se puede usar el concepto de ventana de retardo.

PREDICCIÓN BT-FNT

- Es una variante del esquema anterior, consiste en predecir que los saltos hacia atrás se tomarán siempre y que los saltos hacia adelante no se tomarán.
- Se basa en que los saltos hacia atrás suelen corresponder a bucles que suelen ser iterados bastantes veces antes de terminar.
  - Muchos procesadores han utilizado este enfoque.
  - Es muy fácil de implementar pues basta con comprobar el signo del desplazamiento respecto al PC que va codificado en la propia instrucción.
  - El Intel Pentium IV ha usado esta estrategia.

PREDICCIÓN HEURÍSTICA BASADA EN EL PROGRAMA

- Consiste en predecir la dirección mediante una serie de reglas:
- Regla:
  - Salto de bucle: Si el destino del salto vuelve al comienzo del bucle se predice que será tomado.
  - Terminación de bucle: si se salta dentro de un bucle, y ninguno de los destinos es el comienzo del bucle se predice que el salto no será tomado.
  - Comienzo de bucle: se predice que el bloque posterior a un salto que sea comienzo de un bucle será tomado.
  - Llamada: si el bloque posterior contiene una llamada a subrutina se predice que el salto va a ese bloque posterior.
  - Almacenamiento: si un bloque posterior contiene una instrucción de almacenamiento se predice que el salto no va a ese bloque posterior.
  - ETC.
- Es necesario añadir algo de información al código de operación de los saltos cuando se usa esta estratega.

PREDICCIÓN ESTÁTICA BASADA EN PERFILES

- Esta estrategia de predicción requiere ejecutar el programa sobre datos de ejemplo para extraer y recopilar estadísticas que hay que suministrar al compilador.
  - El compilador usa estos perfiles para hacer predicciones estáticas que se insertan en el código binario del programa como ayudas a los saltos.
  - Una estrategia simple consiste en determinar la frecuencia de saltos tomados para cada instrucción de salto del programa (o si hay varias ejecuciones del programa usar datos promediados).
  - Si el promedio es superior al 50% el bit de ayuda al salto se marca para predecir que será tomado.
- Ventaja: esta técnica es muy fácil de implementar en términos de hardware.
- Desventaja: el sistema es muy rígido, ya que se establece y fija en tiempo de compilación y se requiere que la instrucción lea de la cache para poder conocer la predicción del salto.

2.2. TÉCNICAS DE PREDICCIÓN DINÁMICA DE SALTOS

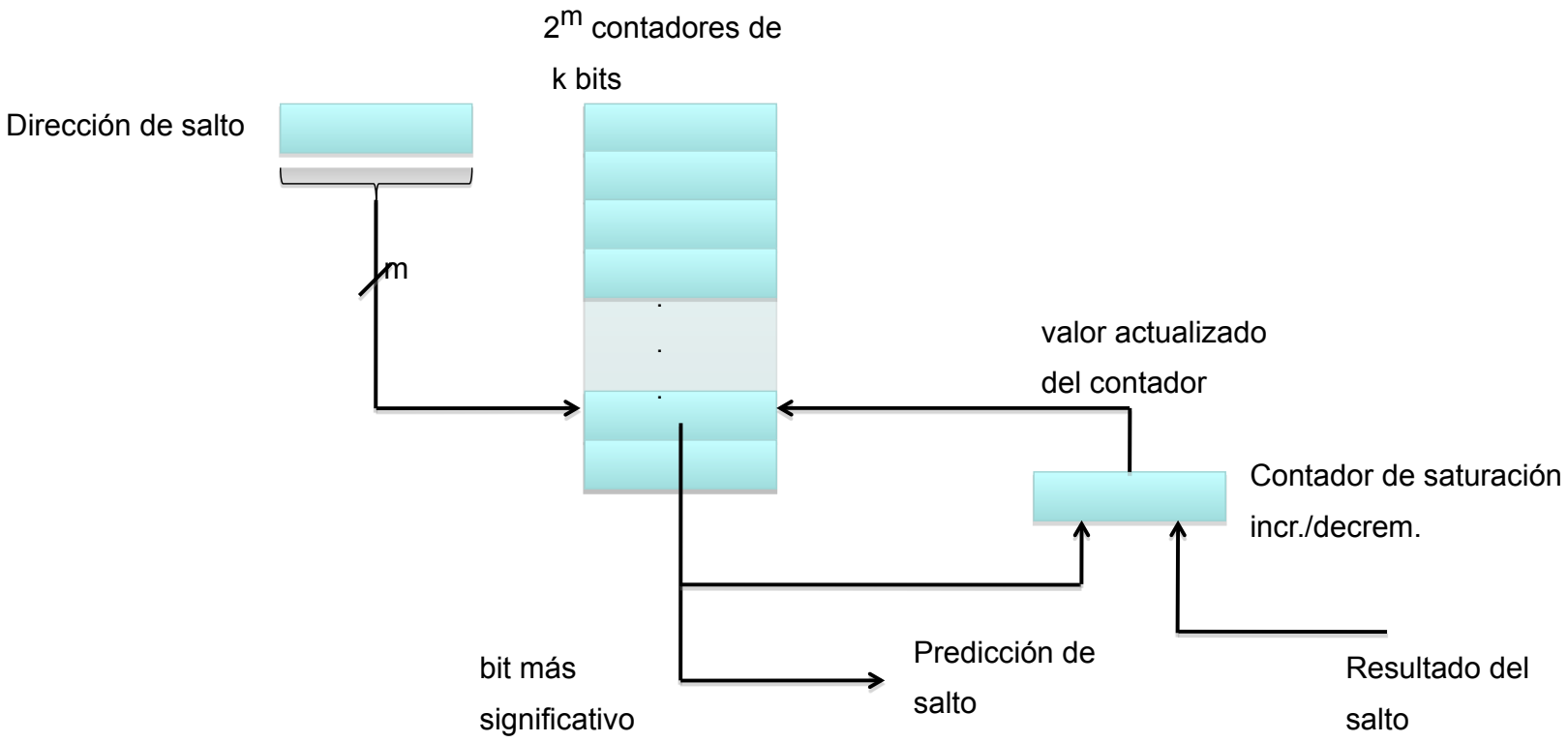
- Las técnicas de predicción estáticas alcanzan tasas de acierto del 70-80%. Pero si la información estadística no es buena la tasa de aciertos baja considerablemente.
- Los métodos de predicción dinámicos suelen lograr tasas de acierto entre el 80% y 95%.
- Métodos básicos:
  - Algoritmo de predicción de Smith (1981).
    - Predictor de 1-bit
    - Predictor de 2-bit.
  - Tablas de predicción de 2 niveles (1992).
    - Predictor de historia global
    - Predictor de historia local
    - Por conjuntos
  - Predictores de índice compartido (1993).
    - Gshare (1993)
    - Pshare (1996)
  - Predictores de reducción de interferencias
  - Predictores Tournament

PREDICCIÓN DINÁMICA: PREDICTOR DE SMITH

- El predictor de Smith (1981) fue uno de los primeros algoritmos de predicción dinámica.
- Consiste en:
  - Una tabla que registra para cada salto si las instancias anteriores se tomaron o no.
  - Se cuenta las veces que se tomó el salto en las últimas veces que se presentó, y el bit más significativo de este contador se utiliza como predicción del salto.
  - Si el salto se toma se incrementa en 1 el contador (salvo que esté en el valor máximo, es decir saturado)
  - Se le denomina también contador de saturación de k bits.

PREDICCIÓN DINÁMICA: PREDICTOR DE SMITH

Esquema



PREDICCIÓN DINÁMICA: PREDICTOR DE SMITH

- Si k=1 bit, tenemos el denominado predictor de un bit que utiliza sólo la información de la última vez que se presentó el salto.
- Si k=2 bits, es el denominado contador de saturación de 2 bits o predictor bimodal, que se utiliza en muchos predictores.
- Ejemplo:

Salto	Direc. salto	Cont 1bit	Predicción	Cont 2bit	Predicción
A	1	1	1	11	1
B	1	1	1	11	1
C	0	1	1 (fallo)	11	1 (fallo)
D	1	0	0 (fallo)	10	1
E	1	1	1	11	1
F	1	1	1	11	1

PREDICTOR DE SMITH: BUFFER DE PREDICCIÓN DE SALTOS (BPB) DE 1-BIT

- Fase de predicción
  - Se mantiene y actualiza una Tabla Histórica de Saltos (Branch Prediction Buffer BPB)
    - Los bits menos significativos del PC sirven como índice de una tabla de 1-bit
    - Este bit indica si se efectuó o no el último salto

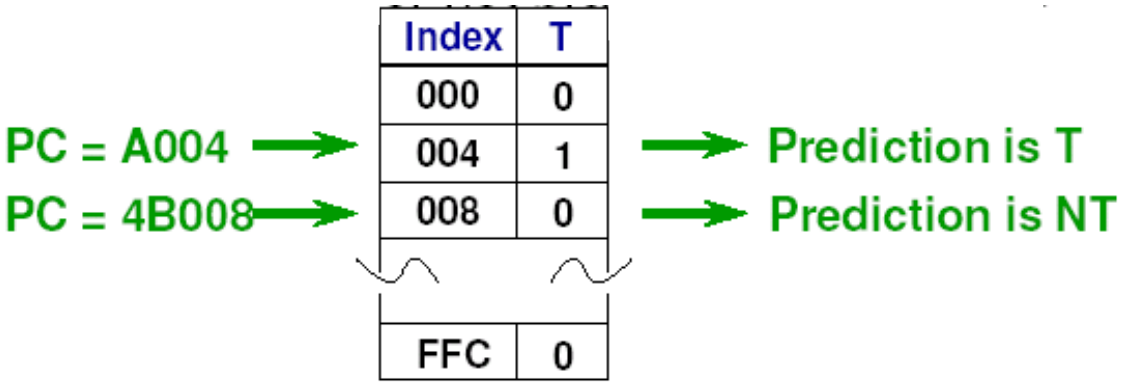


Imagen de Hennessy-Patterson, 1997

PREDICTOR DE SMITH: BUFFER DE PREDICCIÓN DE SALTOS (BPB) DE 1-BIT

- Fase de actualización
  - Después de la ejecución, se marca en la entrada apropiada si el salto ha sido tomado o no (T/NT)

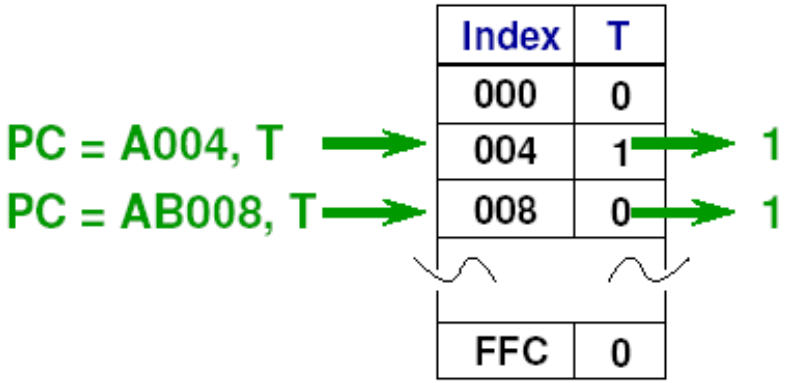


Imagen de Hennessy-Patterson, 1997

Problema: el BPB sólo es útil si el salto puede calcularse antes de la predicción del salto (esto no es posible en el MIPS)

PREDICTOR DE SMITH: BUFFER DE PREDICCIÓN DE SALTOS (BPB) DE 1-BIT

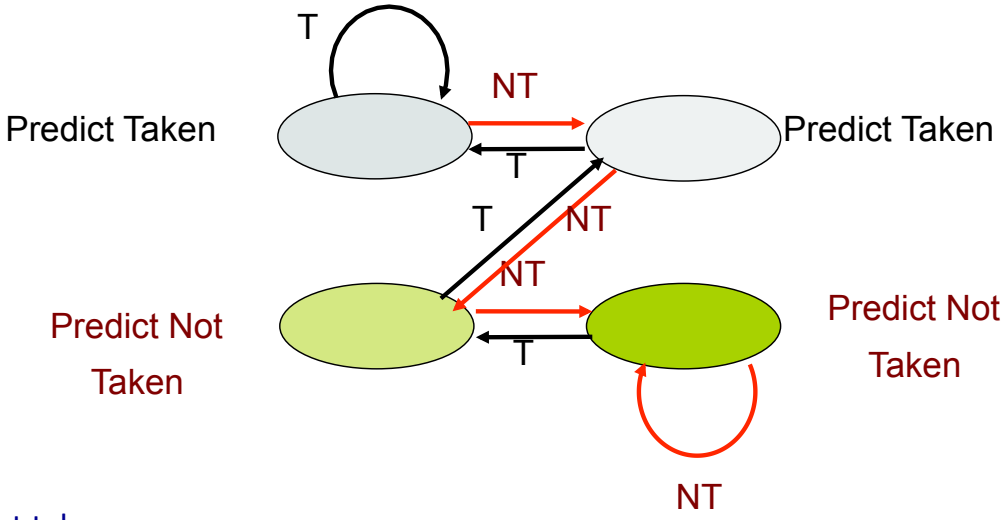
- **Problema**: en un bucle, la predicción con 1-bit origina 2 predicciones erróneas (en promedio un bucle se ejecuta 9 veces antes de finalizar):
  - Al final del bucle, cuando termina en vez de continuar el bucle como en los casos anteriores
  - La primera vez que pasa por el bucle, se predice que se saldrá en vez de predecir que se hace el bucle
  - Esto origina un 80% de precisión en la predicción

Branch Outcome	0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1
Prediction	? 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0

- **Solución**: esquema de 2-bits donde el cambio de predicción sólo se hace si hay dos errores de predicción

PREDICTOR DE SMITH: BUFFER DE PREDICCIÓN DE SALTOS (BPB) DE 2-BIT

- En el esquema de 2-bit la predicción sólo se cambia sí hay dos errores de predicción:



  - Red: stop, not taken
  - Black: go, taken
- Añade histéresis al proceso de toma de decisión
- Propuesto por J. Smith en 1981

PREDICTOR DE SMITH: BUFFER DE PREDICCIÓN DE SALTOS (BPB) DE 2-BIT

- Utiliza un array de contadores de 2-bits con saturación
- La predicción es el bit más significativo (MSb) del contador
- El contador se actualiza con la salida del salto

Code	Means
00	SNT
01	WNT
10	WT
11	ST

Index	Count	
000	10	
004	11	→ Taken
008	00	→ Not Taken
∿	∿	
FFC	01	

S - Strong NT - Not taken

W - Weak T - Taken

Imagen de Hennessy-Patterson, 1997

PREDICTOR DE SMITH: BUFFER DE PREDICCIÓN DE SALTOS (BPB) DE 2-BIT

- Estado inicial: weakly taken (la mayoría de los saltos se efectúan).
- Predice bien series monotónicas: sólo un error por iteración del ciclo
- No predice bien saltos con patrones 010101….01
  - Saltos con patrones complejos requieren predictores más sofisticados

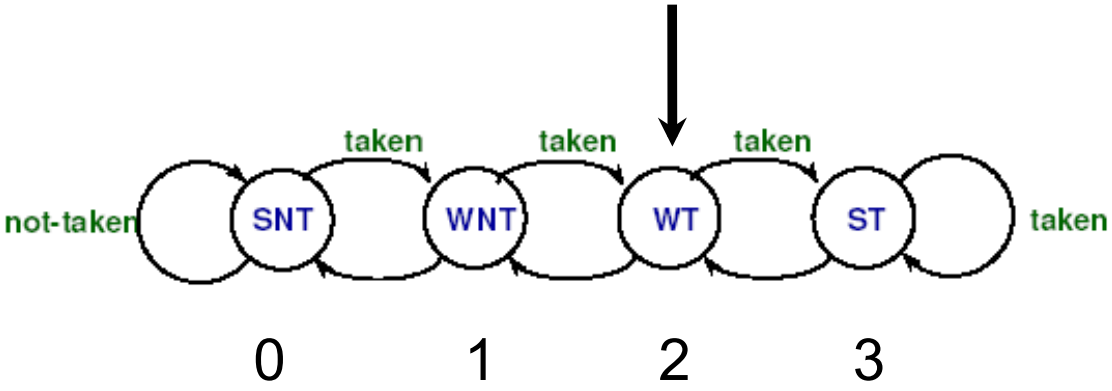


Imagen de Hennessy-Patterson, 1997

PREDICTOR DE 2-BIT (BIMODAL)

```
Code:
→ Loop:   ....
→ br1:  if (n/2) {
→          .......  }
→ br2:  if ((n+1)/2) {
→             .......  }
→          n--
→ br3: JNZ n, Loop
```

- Br1 prediction
  - Pattern: 1 0 1 0 1 0
  - counter: 2 3 2 3 2 3
  - Prediction: T T T T T T
- Br2 prediction
  - Pattern: 0 1 0 1 0 1
  - counter: 2 1 2 1 2 1
  - Prediction: T nT T nT T nT
- Br3 prediction
  - Pattern: 1 1 1 1 1 0
  - counter: 2 3 3 3 3 3
  - Prediction: T T T T T T

En rojo, las pred. erróneas

*Image from J. L. Hennesy and D. Patterson "Computer Organization" course

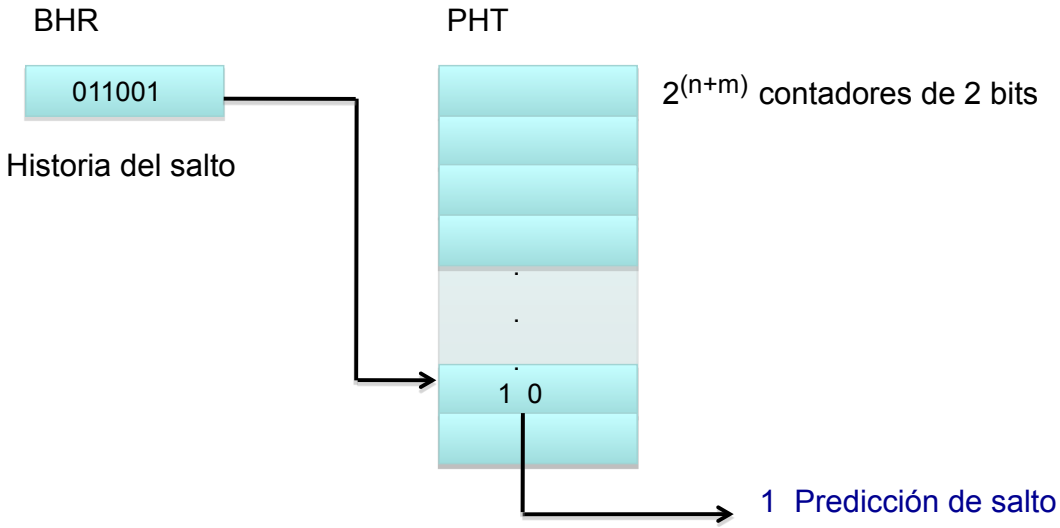
Obsérvese el alto número de predicciones erróneas en las dos primeras secuencias.

PREDICCIÓN DINÁMICA: TABLAS DE PREDICCIÓN DE 2 NIVELES

- Este predictor utiliza dos niveles independientes de información sobre el historial de los saltos para realizar la predicción.
- Puede utilizar historia global o local lo que da lugar a algoritmos diferentes
- También llamados correlados

PREDICTOR DE 2 NIVELES DE HISTORIA GLOBAL: 1º IDEA

- Esquema básico
  - BHR: Branch History Register
  - PHT: Pattern History Table



PROBLEMA: alto número de interferencias destructivas entre saltos

PREDICTOR DE 2 NIVELES DE HISTORIA GLOBAL

- La conducta de algunos saltos esta muy correlada con la conducta de otros saltos:

  if (x < 1)….

  if (x > 1) …

- Utilizando un Global History Register (GHR), la predicción del segundo if puede basarse en la dirección del primer if

  - GHR no es por salto específico, sino para el programa entero

- Para otros saltos la interferencia entre historias puede ser destructiva

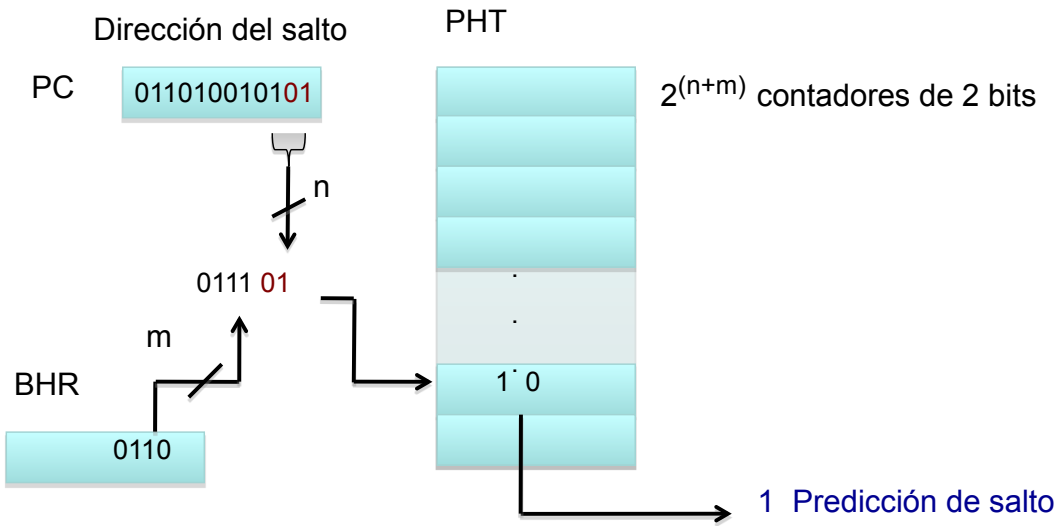
- La interferencia entre historias puede reducirse significativamente utilizando los esquemas g-select o g-share

PREDICCIÓN DINÁMICA: PREDICTOR DE HISTORIA GLOBAL (G-SELECT)

- El esquema anterior produce muchas interferencias destructivas por lo que se usa un esquema modificado denominado g-select
- Este predictor funciona de la siguiente forma:
- Primer nivel:
  - Los resultados de salto más recientes se almacenan en un registro de desplazamiento, que se desplaza cuando entra un nuevo salto, saliendo el más antiguo (se usan: 0 y 1, para indicar salto no tomado y tomado).
  - Se denomina registro de historial de salto (BHR, branch history register)
- Segundo nivel:
  - Este nivel es una tabla con contadores de saturación de 2 bits.
  - Se denomina tabla de historial de patrones (PHT, pattern history table).
  - La tabla se indexa con un algoritmo hash, con concatenación de la dirección del salto con el contenido del BHR.
  - El contador indexado en la tabla PHT proporciona la predicción de forma similar a como se hace en el predictor Smith de 2 bit o bimodal.

PREDICCIÓN DINÁMICA: PREDICTOR DE 2 NIVELES DE HISTORIA GLOBAL

Esquema g-select



PREDICCIÓN DINÁMICA: PREDICTOR DE 2 NIVELES DE HISTORIA GLOBAL

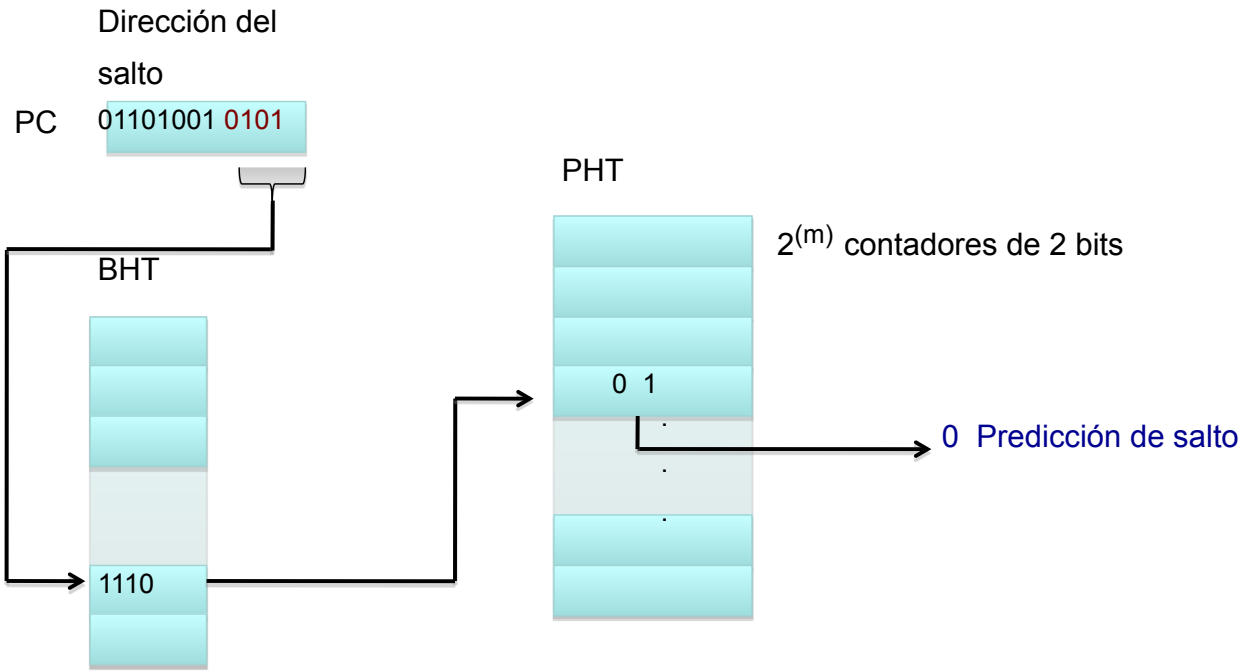
- Con m bits de historia global y n bits de la dirección de salto la PHT requiere 2^{n+m} entradas.
- Hay que equilibrar ambos, ya que si se usan más bits de dirección disminuyen los conflictos de saltos,
- Si se usa una historia más larga (más bit en el BHR) se permite que se correlacione con patrones más complejos
- Esta técnica explota la idea de que el resultado de un salto puede estar correlacionado con un salto anterior

PREDICCIÓN DINÁMICA: PREDICTOR DE 2 NIVELES DE HISTORIA LOCAL

- En este esquema se pasa de tener un único BHR global a tener un BHR por salto.
- La recopilación de BHR se denomina tabla de historial de saltos (BHT, branch history table).
- Se puede considerar que el esquema con un único BHR global es una simplificación de este.

PREDICCIÓN DINÁMICA: PREDICTOR DE 2 NIVELES DE HISTORIA LOCAL: 1º IDEA

Esquema

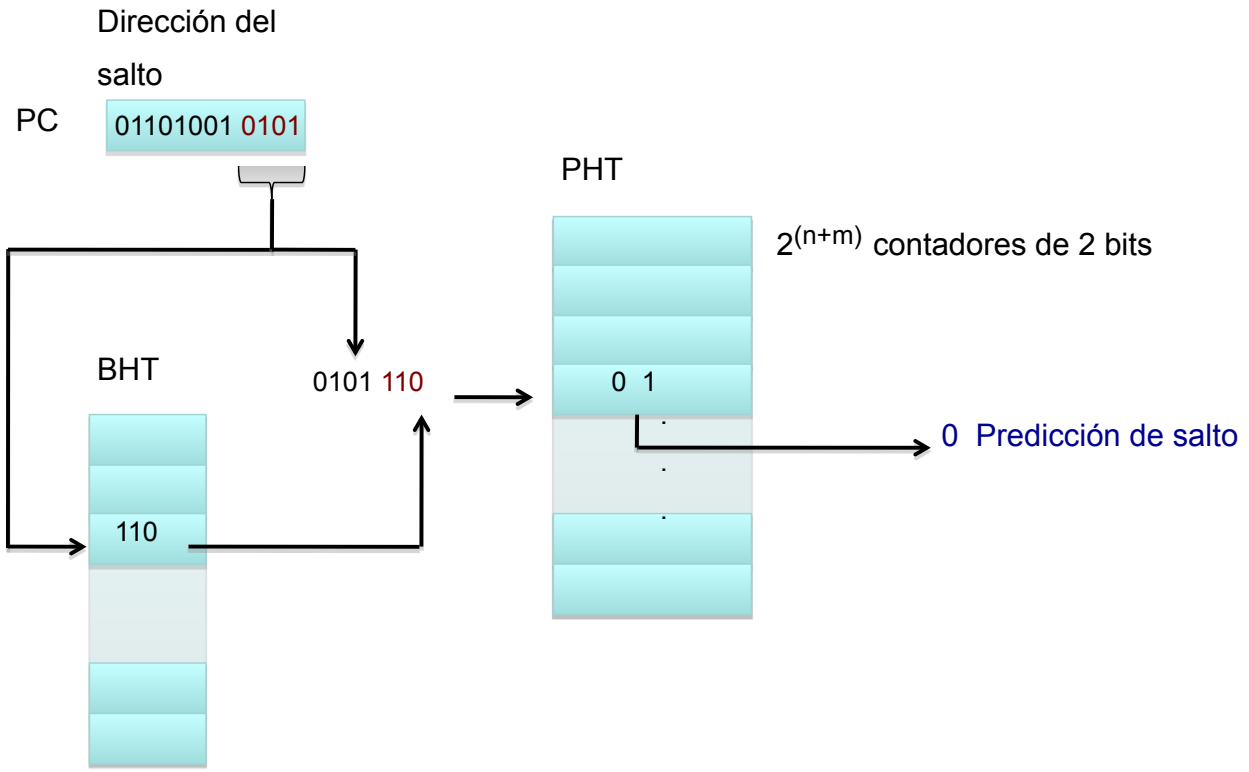


PREDICCIÓN DINÁMICA: PREDICTOR DE 2 NIVELES DE HISTORIA LOCAL

- La historia de cada salto se salva en un Branch History Register (BHR):
  - BHR: es un registro de desplazamiento actualizado con la salida del salto
- El BHR indexa un array de contadores saturados de 2-bits (bimodal)
- Cada contador predice el salto para una historia dada
  - Puede predecir bien patrones complejos
- El array de contadores puede ser
  - Privado: por BHR
  - Por-conjunto: compartido por todas las BHRs en el mismo conjunto
  - Global: compartido por todos los BHRs
- BHRs demasiado largos no son buenos
  - La historia pasada puede no ser ya relevante

PREDICCIÓN DINÁMICA: PREDICTOR DE 2 NIVELES DE HISTORIA LOCAL: L-SELECT

- Esquema



PREDICCIÓN DINÁMICA: PREDICTOR DE 2 NIVELES DE HISTORIA LOCAL: L-SELECT

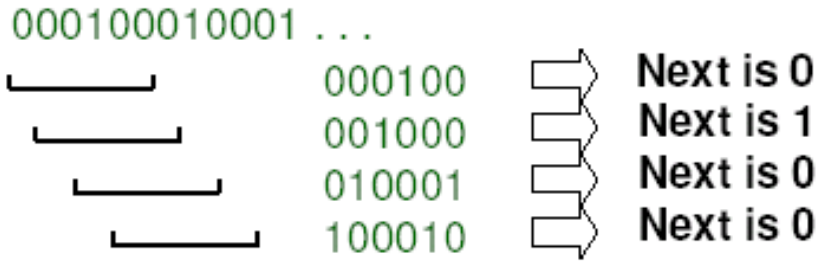
- Funcionamiento:
  - La dirección de salto se usa para seleccionar una de las entradas de la BHT que proporcional la historia local.
  - El contenido del BHR seleccionado en la BHT se combina con el PC en la misma forma que en el predictor de historia global para indexar la tabla PHT.
  - En la tabla PHT están los contadores de saturación de 2 bits.
  - Para actualizar el historial, se desplaza el resultado del registro del salto y se introduce el más reciente en la entrada BHR del BHT y se actualiza también en la PHT como en el predictor Smith de 2 bit.
- La asignación de tamaños en este predictor es más compleja que en el predictor de 2 niveles de historia global.
- Ejemplo: el Intel P6 (i686/pentium pro) utiliza un predictor de 2 niveles de historia local con longitud de historia de 4 bits.

PREDICCIÓN DINÁMICA: PREDICTOR DE HISTORIA LOCAL : EJEMPLO

- El salto del bucle interno en:
  - For (j=0; j<1000;j++)

    for (i=0; i<4; i++)

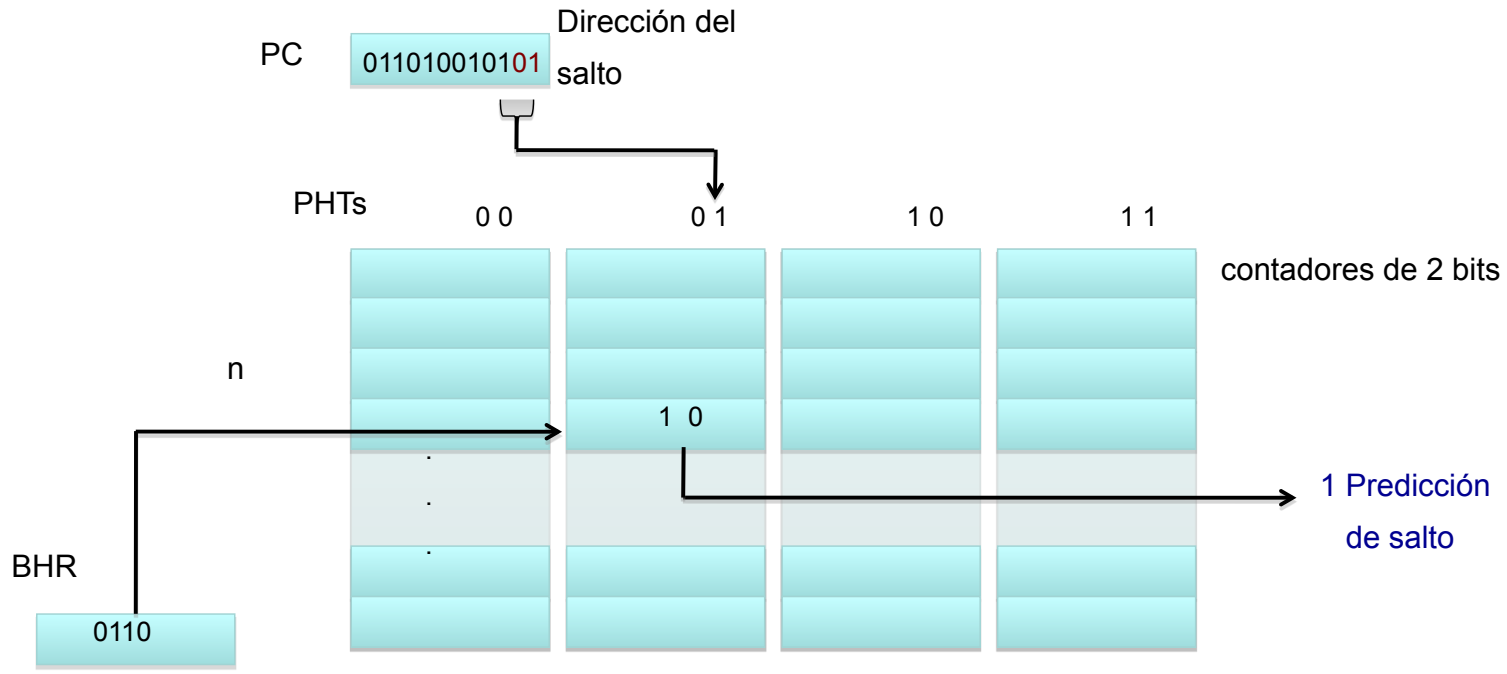
  Puede generar la secuencia: 000100010001…..
- Suponiendo una historia de longitud 6, en régimen permanente, se repetirán los siguientes patrones



- Los contadores apuntados por 000100, 010001, 100010 irán a NT (not taken)
- El contador apuntado por 001000 irá a T (taken)

PREDICCIÓN DINÁMICA: PREDICTOR DE 2 NIVELES DE HISTORIA LOCAL

- Otra visión del predictor local



PREDICCIÓN DINÁMICA: PREDICTOR DE 2 NIVELES POR CONJUNTOS

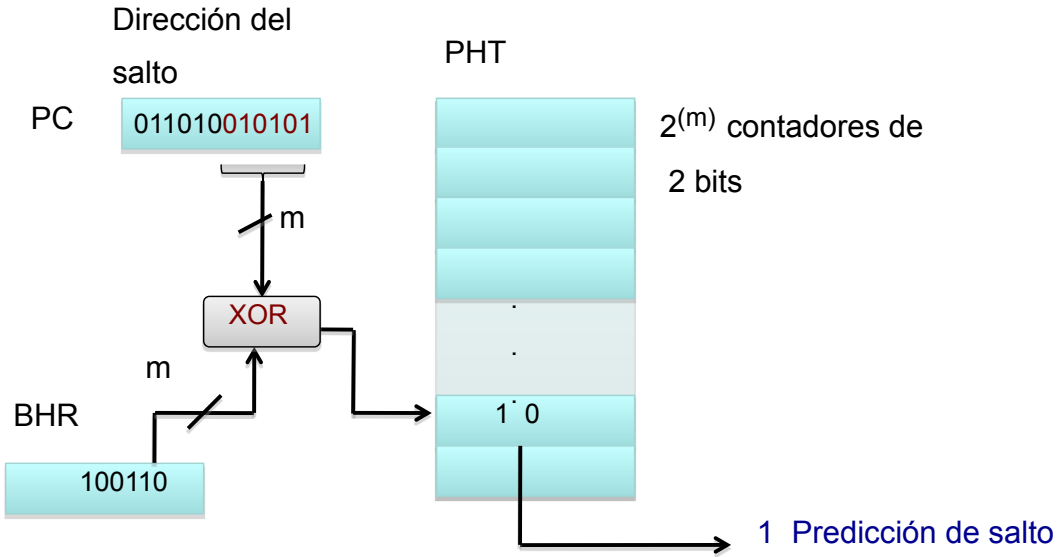
- Esta variante usa una BHT que usa una función de hash arbitraria para dividir los saltos en conjuntos distintos.
- Cada conjunto comparte un único BHR.
- En lugar de usar los bits menos significativos de la dirección para seleccionar el BHR de la BHT utilizan los más significativos.
- A este tipo de historial se le denomina historial de salto por conjunto y la tabla se denomina tabla de historial de salto por conjuntos (SBHT, set branch history table).
- Se pueden obtener diversas combinaciones.

PREDICCIÓN DINÁMICA: PREDICTOR DE ÍNDICE COMPARTIDO

- En general los predictores de 2 niveles son difíciles de equilibrar, en lo que se refiere al número de bits a utilizar en el BHR y el número de bits que se usan para indexar la PHT.
- Para un tamaño fijo de PHT, el uso de más bits de historial permite establecer correlaciones con saltos más lejanos pero con el coste de usar menos bits de la dirección de salto y aumentar los conflictos.
- Para evitar algunos de estos problemas Mc Farling propuso en 1993 una variación del predictor de 2 niveles de historia local que se denominó predictor Gshare.

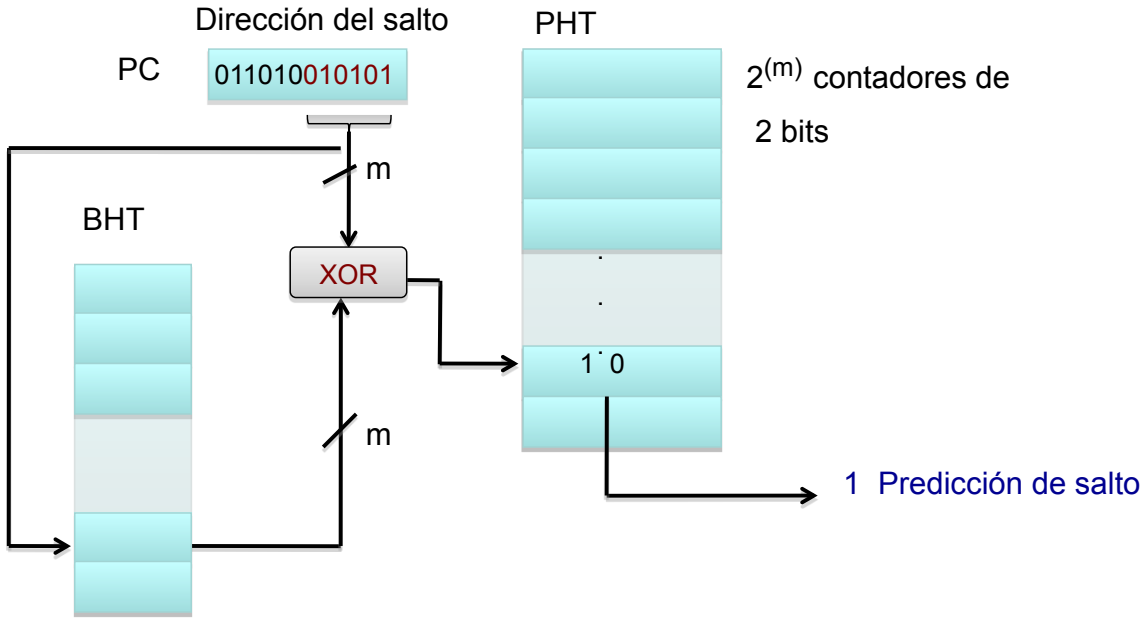
PREDICCIÓN DINÁMICA: PREDICTOR GLOBAL G-SHARE

- Esta solución intenta utilizar mejor los bits de índice aplicando la función Hash al BHR y al PC conjuntamente para seleccionar la entrada en la PHT.
- La función hash que se utiliza es un XOR a nivel de bit (a esto se le denomina compartición del índice).
- El hardware del predictor g-share es muy parecido al predictor de 2 niveles excepto en la generación del índice de la PHT.



PREDICCIÓN DINÁMICA: PREDICTOR P-SHARE (DE ÍNDICE COMPARTIDO)

- Evers (1996) propuso una variación del predictor G-share que utiliza una tabla de historial de salto por dirección para almacenar el historial local del salto.
- El P-share es análogo al G-share pero con la historia local del salto.
- Se usan los bits de orden inferior de la dirección de salto para indexar en la BHT de primer nivel.
- A continuación se hace un XOR entre el contenido del BHR indexado y la dirección de salto para formar el índice de la PHT.



PREDICCIÓN DINÁMICA: PREDICTORES DE ÍNDICE COMPARTIDO: EJEMPLOS

- Se utilizan habitualmente en los micros actuales.
- El IBM Power 4 utiliza una historia global (BHR) de 11 bits y una PHT de 16.384 entradas
- El Alpha 21264 usa un predictor de historia global con un historial global de 12 bits y una PHT de 4096 entradas.

PREDICCIÓN DINÁMICA: PREDICTORES DE REDUCCIÓN DE INTERFERENCIAS

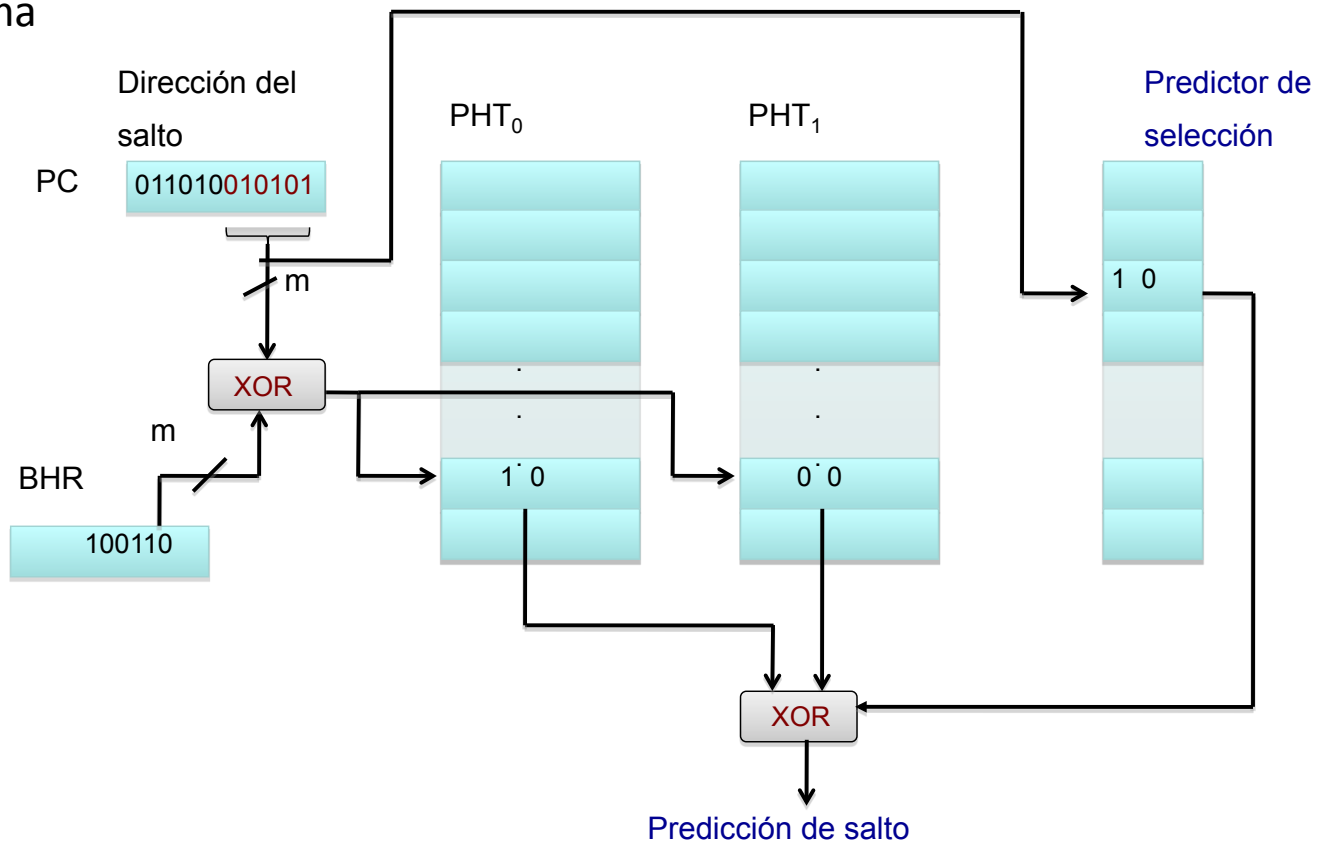
- La PHT que se utiliza en los predictores de dos niveles es una estructura sin etiquetas asignada de forma directa.
- El alias se produce entre dos parejas diferentes dirección/salto de la PHT.
- La PHT puede ser considerada como una memoria de tipo caché en la que se pueden producir fallos.
- Estos fallos en la PHT se pueden producir por diversas razones:
  - Por alias obligatorio la 1ª vez que se utiliza la pareja dirección/salto para indexar la PHT.
  - La capacidad de alias aparece debido a que el conjunto de trabajo actual de parejas direcc/ salto es mayor que la capacidad de la PHT.
  - Los conflictos de alias aparecen cuando dos parejas distintas direcc/salto asignan la misma entrada a la PHT.
- En general, la solución estándar en las caches es aumentar la asociatividad de la caché, pero los predictores son diferentes.

PREDICTORES DE REDUCCIÓN DE INTERFERENCIAS: PREDICTOR BI-MODE

- Utiliza múltiples PHTs para reducir los efectos de alias (1997), utiliza dos PHTs indexadas con un esquema g-share.
- Ambas utilizan el mismo índice.
- El predictor de selección se indexa con los bits inferiores de la dirección de salto y es un predictor de Smith de 2 bits, en el que el bit más significativo indica que PHT hay que usar para la predicción.
- Se basa en que la mayoría de los saltos se desplaza (en %) hacia ser tomado o no tomado.
- El predictor recuerda esto, de forma los que se inclinan a ser tomados se almacenan en una PHT y los que se inclinan a no ser tomados en la otra.
- Esto reduce las interferencias destructivas entre ambas PHTs.
- Una vez conocido el resultado del salto, se actualiza la PHT que ha marcado el predictor de selección.

PREDICTORES DE REDUCCIÓN DE INTERFERENCIAS: PREDICTOR BI-MODE

Esquema



PREDICTORES TOURNAMENT

- El motivo para correlar los predictores de salto es que los predictores de 2-bit fallan en saltos importantes; al añadir información global, se mejoran las prestaciones
- Los Tournament predictors: utilizan 2 predictores, 1 basado en información global y 1 basado en información local, y los combina con un selector
- La esperanza es seleccionar el predictor correcto para el salto correcto

SELECTOR

- El selector decide para cada salto que predictor es el mejor
- Cada salto es predicho por dos predictores: P1 y P2
- El puntero a la instrucción de salto también indexa un contador de saturación de 2-bits en el array del selector
  - Si P1 es correcto y P2 erróneo el contador se incrementa
  - Si P2 es correcto y P1 erróneo el contador se decrementa
  - En otro caso, el contador no se modifica
- El valor del selector será utilizado para predecir la próxima vez que se encuentre este salto
  - Si 11 o 10 => se elige P1
  - Si 00 o 01 => se elige P2

SELECTOR (CONTINUACIÓN)

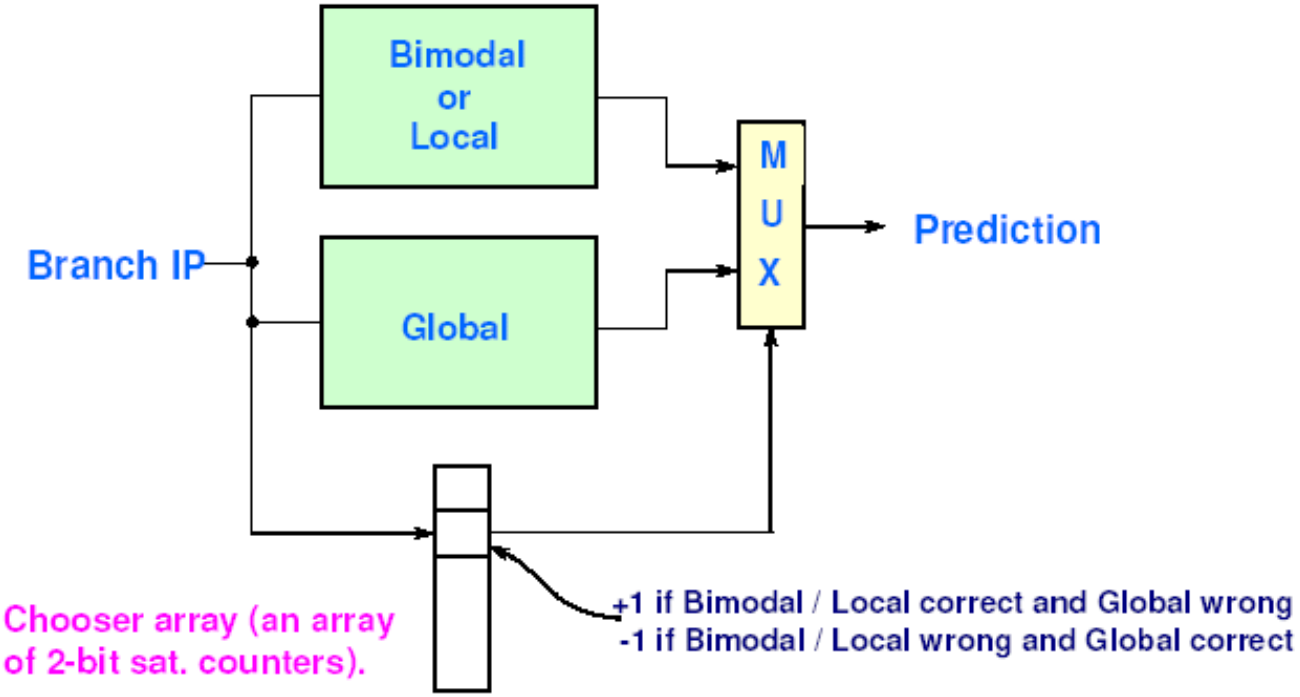


Imagen de Hennessy-Patterson, 1997

- El selector puede también ser indexado por el GHR

TOURNAMENT PREDICTOR EN EL ALPHA 21264

- 4K de contadores de 2-bit para elegir entre un predictor local y un predictor global
- Predictor Global, tiene también 4K entradas y esta indexado por la historia de los últimos 12 saltos; cada entrada del predictor global es un predictor standard de 2-bit
  - Patrón de 12-bit:
    - i-ésimo bit 0 => i-ésimo salto previo no efectuado;
    - i-ésimo bit 1 => i-ésimo salto previo efectuado;

TOURNAMENT PREDICTOR EN EL ALPHA 21264

- Predictor Local, consiste de un predictor de 2-niveles:
  - Top level es una tabla de historia local que consta de 1024 entradas de 10-bit; cada entrada de 10-bit corresponde a los resultados de los 10 saltos más recientes. Esta historia de 10-bit permite que patrones de 10 saltos sean descubiertos y predichos.
  - Next level Las entradas seleccionadas de la tabla de historia local se usan como índice de una tabla de 1K de entradas, que consisten en contadores de saturación de 3-bit, los cuales suministran la predicción local
- Tamaño total: 4K*2 + 4K*2 + 1K*10 + 1K*3 = 29K bits! (~180,000 transistores)

% DE PREDICCIONES DEL PREDICTOR LOCAL EN EL ESQUEMA TOURNAMENT PREDICTION

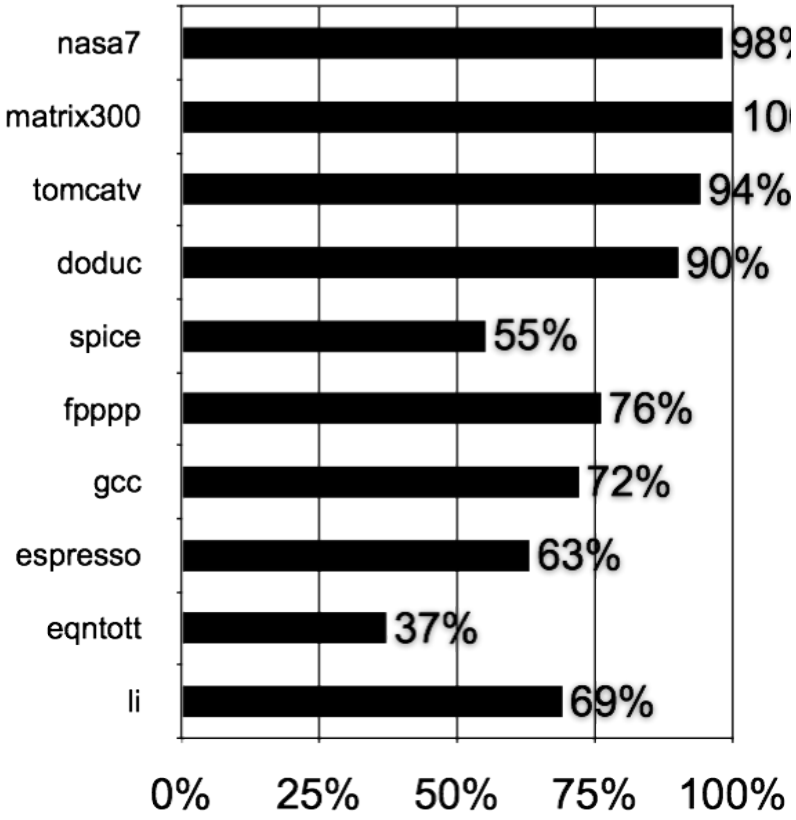


Imagen de Hennessy-Patterson, 1997

PRECISIÓN DE LA PREDICCIÓN DE SALTO

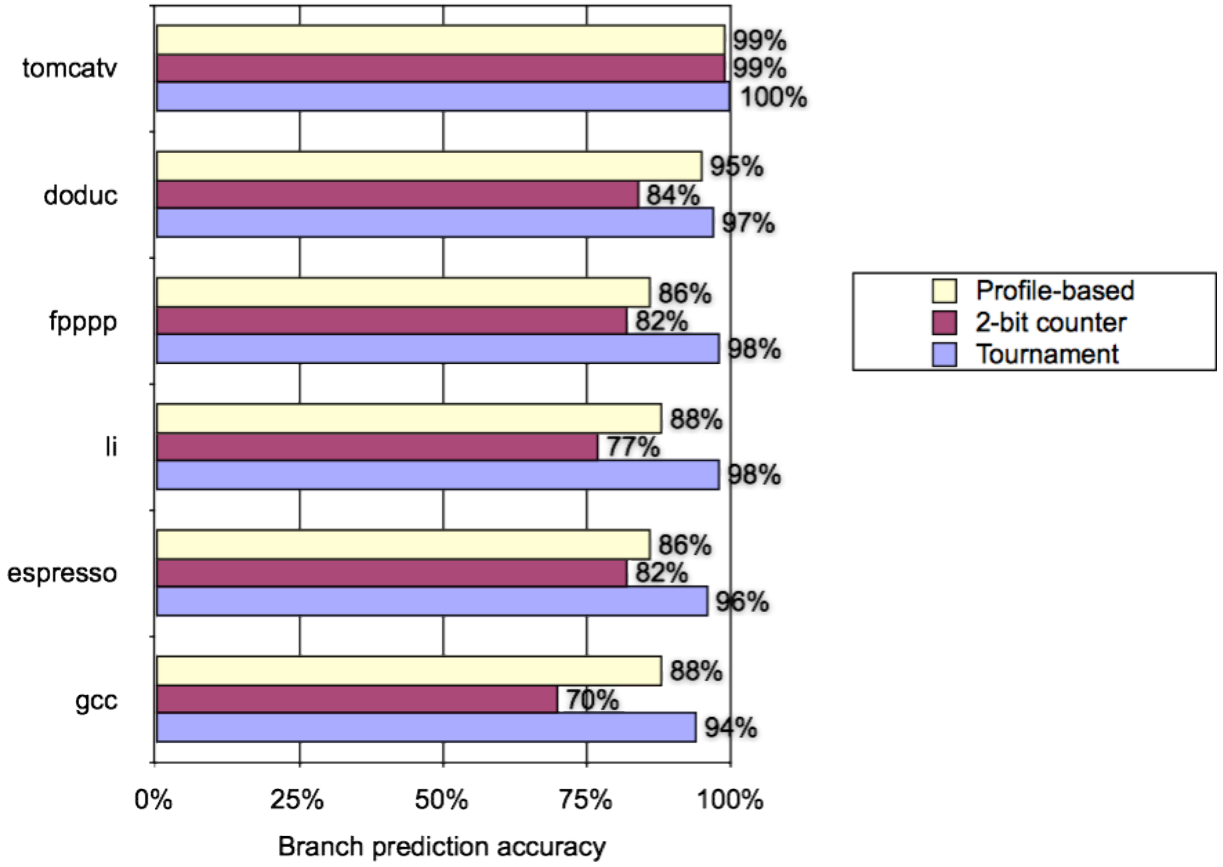


Imagen de Hennessy-Patterson, 1997

- Perfil: perfil de salto de la última ejecución

PRECISIÓN V. TAMAÑO (SPEC89)

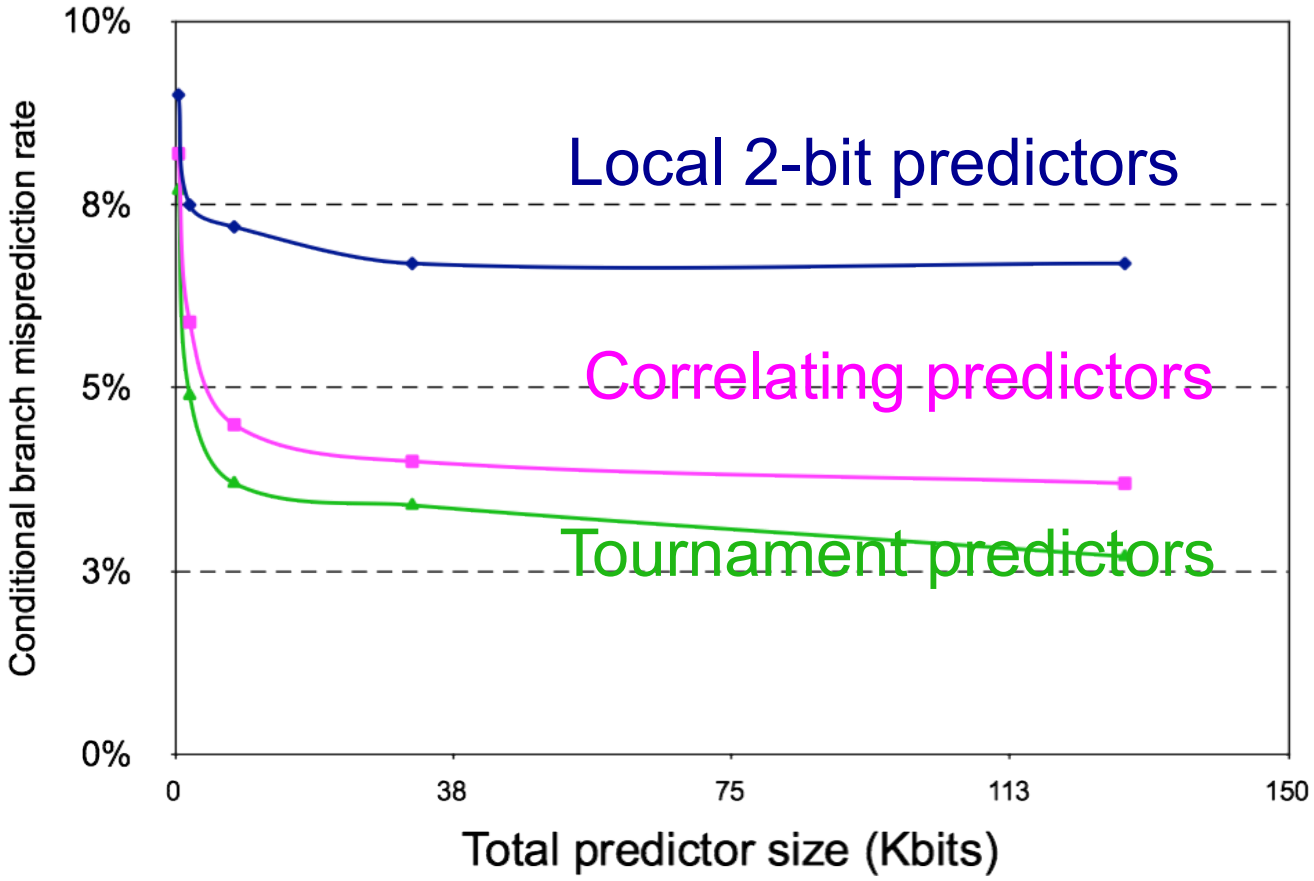


Imagen de Hennessy-Patterson, 1997

PREDICCIÓN DE LA DIRECCIÓN DE SALTO

- Además de predecir el sentido del salto es necesario predecir la dirección que corresponde a ese sentido del salto
- La idea básica es guardar las direcciones a las que se ha saltado anteriormente en el sentido correspondiente.

BUFFER DE OBJETIVOS DE SALTO (BTB)

(Branch Target Buffer BTB)

- Motivación: se necesita la dirección al mismo tiempo que la predicción.
- El PC se utiliza como un índice para obtener la predicción y la dirección objetivo del salto (si es efectuado)

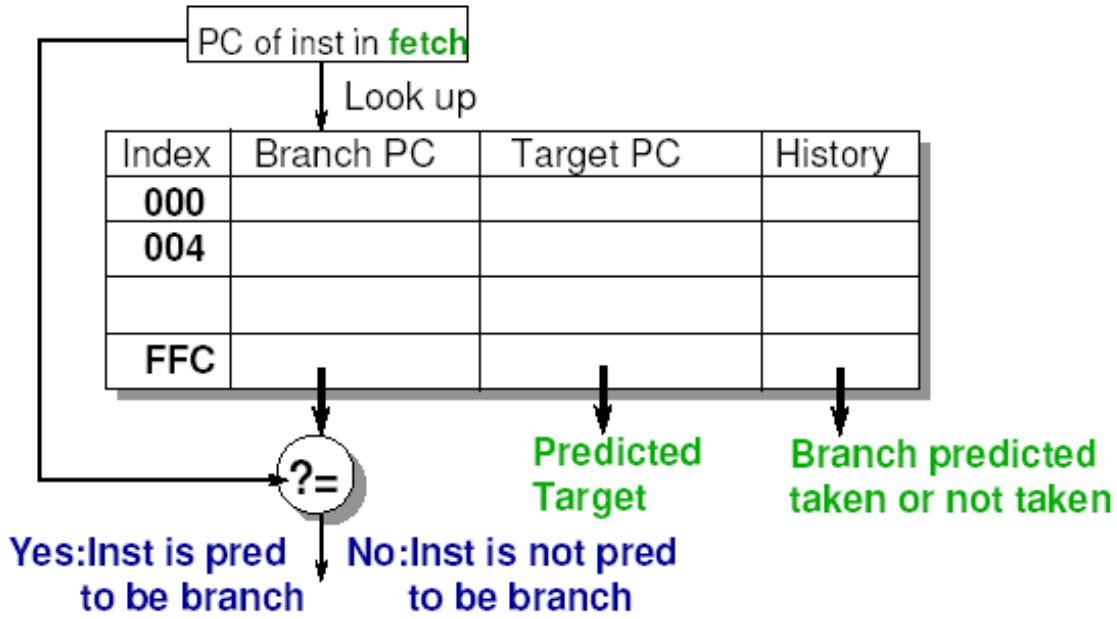


Imagen de Hennessy-Patterson, 1997

BUFFER DE OBJETIVOS DE SALTOS (BTB): ACCESOS

- Se requieren tres operaciones en el BTB:
  - Asignación (allocation)
    - Sólo las instrucciones identificadas como saltos efectuados (taken) son situadas en el BTB (después de su ejecución)
      - Un BTB hit implica que la instrucción es un salto
      - Tanto los saltos condicionales como los incondicionales son situados en el BTB
    - Los saltos no efectuados (not taken) no necesitan ser situados en el BTB
      - Un BTB miss predice implícitamente que no se efectúa el salto
    - Puede reemplazar una entrada válida
      - Ejemplo: si dos saltos en 0xA2020 y 0xAA020 son efectuados (taken)

BUFFER DE OBJETIVOS DE SALTOS (BTB): ACCESOS

- Actualización
  - Cuando se resuelve un salto (taken o not taken), la historia es actualizada en el BTB (si está situada en el BTB)
  - Cuando la dirección objetivo del salto se conoce, esta dirección se actualiza en el BTB (si es incorrecta)
- Búsqueda (para predicción)
  - La búsqueda en el BTB se hace en paralelo a la búsqueda de la instrucción
  - El BTB nos suministra
    - Una indicación de que la instr es un salto (si BTB hit)
    - Una predicción de la dirección objetivo del salto
    - Una predicción de la dirección del salto

UTILIZACIÓN DEL BTB

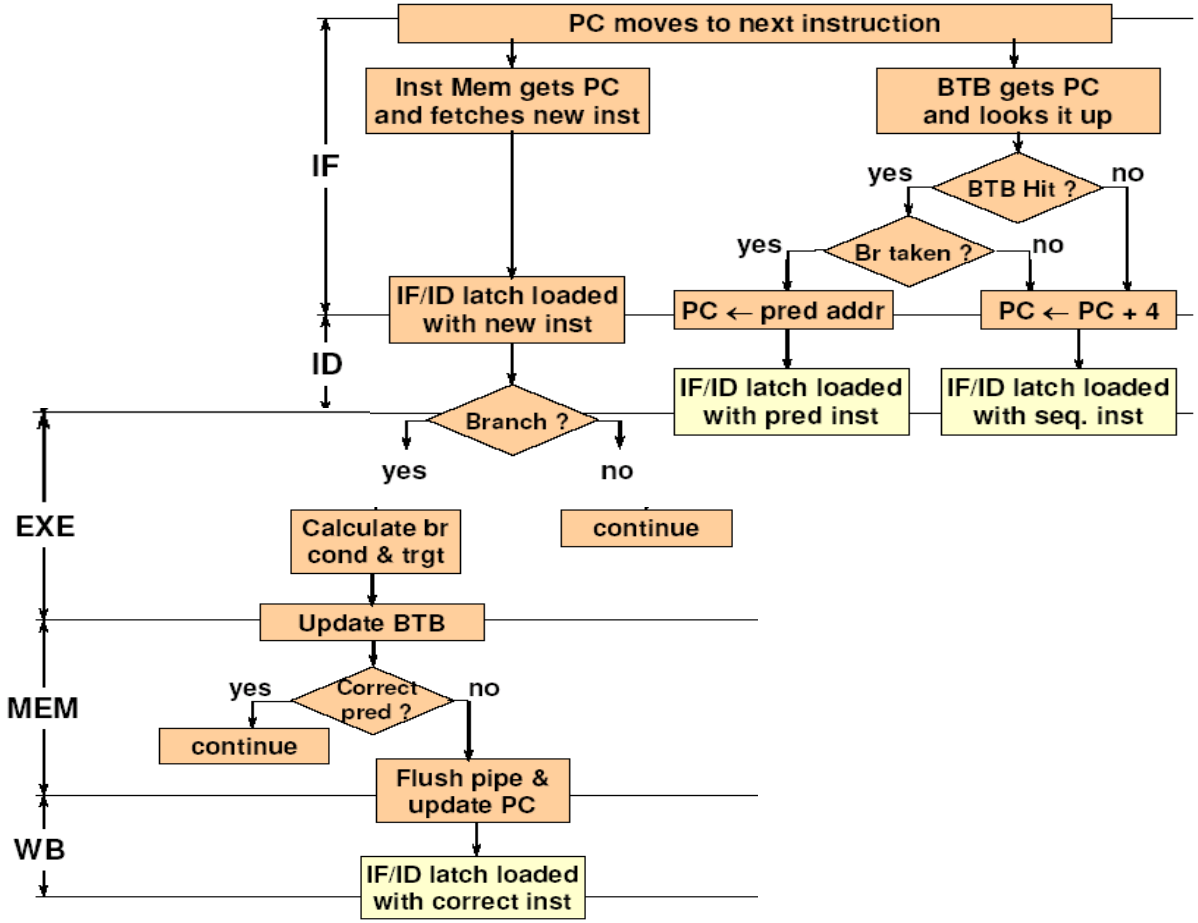


Imagen de Hennessy-Patterson, 1997

BTB DE DOS VÍAS (2-WAY)

- Mantiene 2 tablas BTB, se busca en ambas tablas
- Se reemplaza una entrada válida según regla LRU (least recently used)
- Ventaja: reduce la posibilidad de eliminar entradas válidas
- Desventaja: requiere un HW más complejo

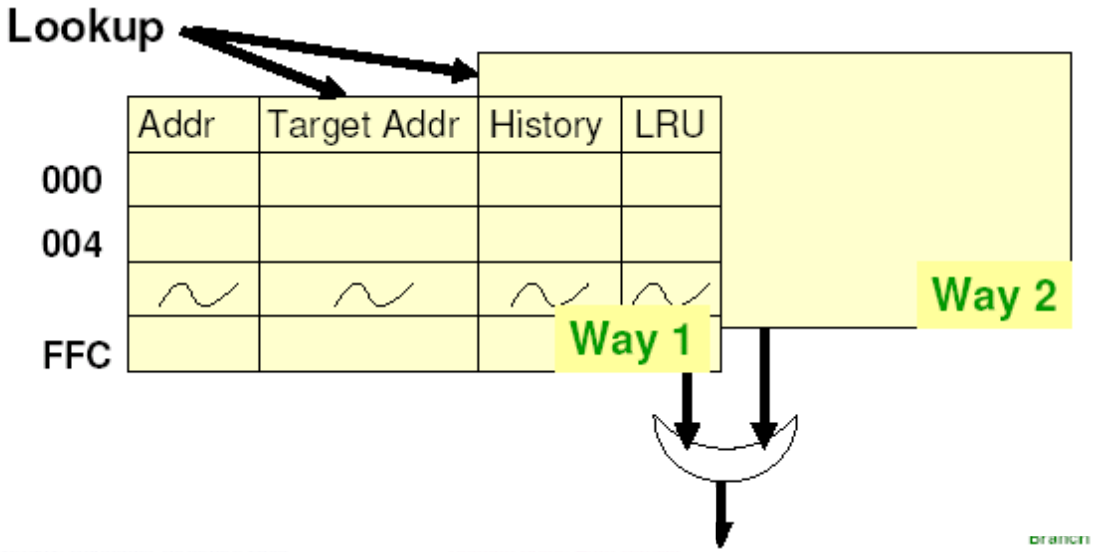


Imagen de Hennessy-Patterson, 1997

PREDICCIONES SEPARADAS DE OBJETIVO Y DIRECCIÓN DEL SALTO

- Las predicciones de dirección objetivo y de dirección se hacen de forma separada. La etiqueta puede ser parcial
- Motivo: Utilizar diferentes algoritmos de predicción

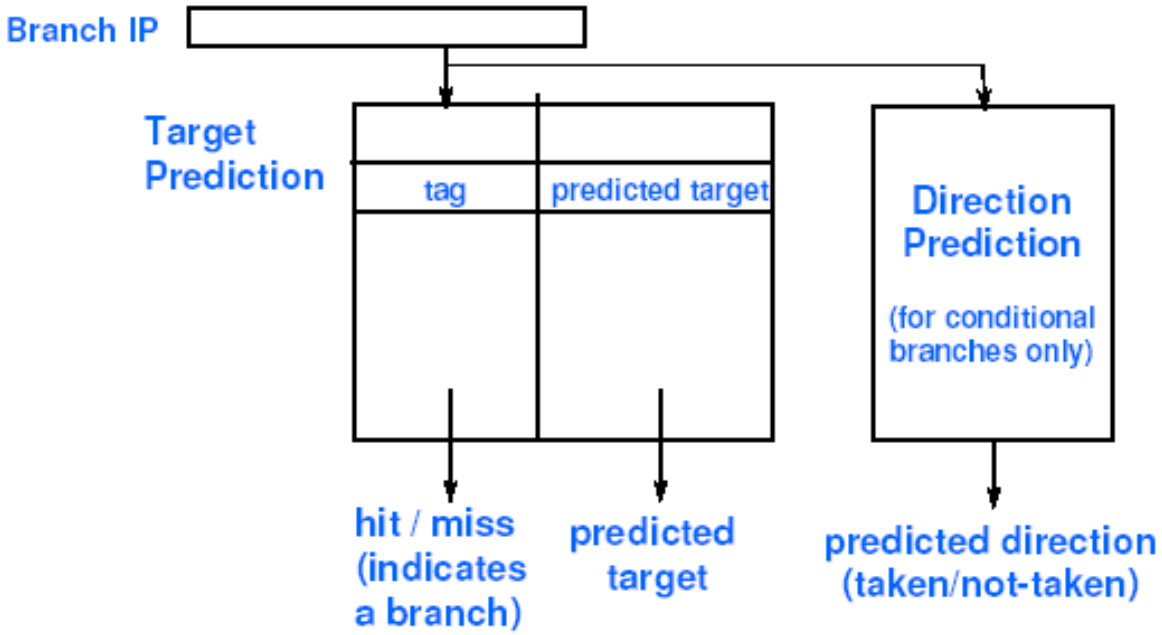


Imagen de Hennessy-Patterson, 1997

REFERENCIAS

- Curso: Professor David A. Patterson. Computer Science 1996.

- Curso: Prof. H.Shall y A. Gluska. Univ. Haifa.

- [Ball-Larus,1993] Ball, T., Larus, J. R. (1993) Branch Prediction for Free. Proceedings of the ACMSIGPLAN '93 Conference on Programming Language Design and Implementation,ACM SIGPLAN Notices, volume 28, pages 300-13
- [Hennessy-Patterson, 1997]David A. Patterson, John L. Hennessy (1997) Computer Organization & Design: The Hardware/Software Interface, Second Edition. Morgan Kaufmann.