ARQUITECTURA E INGENIERÍA DE COMPUTADORES

Tema 2.3

Miquel Gómez Corral Ing. Informática ETSINF, UPV

Tema 2.3

Predicción dinámica de Saltos

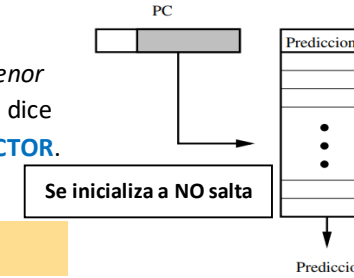
Hasta ahora la mejor predicción que podíamos hacer era el `predict not taken`. Bastante ok xq a latencia 1 solo perdemos 1 ciclo por cada salta. *Pero eso en los casos que vemos, en otros puede ser peor.*

Se pretende realizar una mejor predicción del salto, se necesita una solución dinámica (*implementada por hardware*).

Hay que saber si solo queremos saber si la **condición se cumple**, si saber si se cumple y **su dirección**, el como se predice (a partir del PC…), **cuantos bits se utilizan**, **como se almacena** (*tablas caché*) y **cuando se realiza** esta predicción (tras decodificar: en ID o después, o antes de decodificar: IF)

BRANCH PREDICCIÓN BUFFERS (*BPB*)

De momento **solo Condición**, usando **algunos bits**, y **almacenándolo** con **correspondencia directa**. (*Nos da algo igual en que etapa se sabe).*

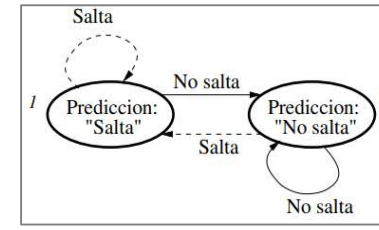


Se usa una tabla (***buffer/cache***) donde se **guarda parte el PC** (*algunos bits, los de menor peso*) para saber qué instrucción es cual y después un bit o bits de predicción que te dice que es lo que supuestamente "hará". **LO QUE SE GUARDA ES EL ESTADO DEL PREDICTOR**. En función de eso predigo si salta o no. *El estado es el estado de un autómata.*

Predictor 1 bit *Caso más sencillo*

Estado del autómata: Condición obtenida en la **última ejecución del salto**.

Predicción: **Estado** del **autómata**.



Si ahora ha saltado pongo que en la siguiente salta, si vuelve a saltar no cambia nada y se queda igual… Si ahora no salta lo cambio, y la siguiente predicción será predicha a no salto… Si vuelve a saltar se cambia y así todo el rato…

Problema: Si usas **pocos bits de PC** es muy posible que otras instrucciones compartan la parte baja de su PC, por lo que varias estarían leyendo y cambiando la predicción de salto.

- **Sol**: **Tamaño mayor** de tabla o utilizar **correspondencia asociativa** o asociativa por **conjuntos** (*y no directa*).

Predictor 2 bit *Histéresis y Saturación*

Las instrucciones de salto que implementan un bucle siguen un patrón predecible, Si el bucle tiene n iteraciones, el salto es efectivo $n - 1$ veces y una vez (*la última iteración*) no salta. Hay que aprovecharlo. Solución: **4 Estados**.

La Predicción **debe fallar dos veces antes de ser modificada**. 2 versiones: En la 1r, sí o sí tienes que **NO hacer algo 2 veces** SEGUIDAS para cambiar de estado. En el segundo se puede quedar en un estado intermedio y **solo con hacer 1 fallo más cambiar**. *El 1r si pasa 2 veces vas al **STRONGLY opuesto**, en el 2nd **al Weakly**.*

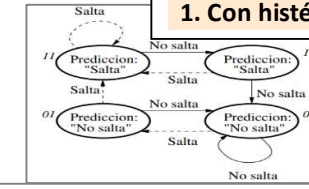
Strongly not taken (*estado 00*): Clara tendencia a no jump. SEGURO Q NO.

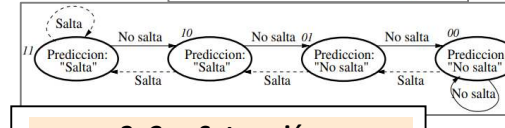
Weakly not taken (*estado 01*): Alguna tendencia a no jump. NO PERO POT SER.

Weakly taken (*estado 10*): Cierta tendencia a j. SÍ PER POT NO SER.

Strongly taken (*estado 11*): Clara tendencia a j. SEGURO QUE SÍ.

1. Con histéresis





2. Con Saturación

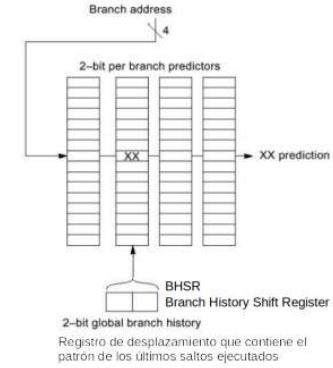
Es como un contador, si es 0 o 1 SALTA, si es 3 o 4 NO (si salta resta, si no suma)

Predictores de dos niveles

Predictor bimodal: Obtiene la predicción a partir de una sola tabla indexada por el PC de la instrucción de salto considerada, en la que se almacena el comportamiento de dicha instrucción → Lo de antes de 1 y 2 bits.

Predictor de dos niveles: Añade, además, el patrón de ejecución (*Salta/No Salta*) de los saltos ejecutados para obtener la predicción. *Vas a **tener en cuenta** no solo **tu comportamiento** sino también el de otros saltos*.

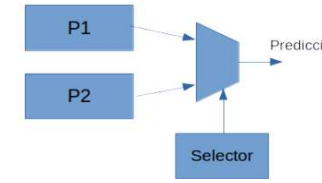
Esto es porque en algunos saltos, por ejemplo, cuando hay varios "ifs" seguidos, el que en uno se salte o no NO depende de sus anteriores saltos, depende a lo mejor de los saltos de otros saltos. *Ejemplo en las diapo 22.*



Ahora en vez de guardarte solo lo que ha hecho tu salto en la última, te apuntas lo que han hecho los 2 o 3 saltos (*vamo a decir 2*) que se han ejecutado antes. Así, cuando se vuelva a producir esa combinación de saltos, se repita, ya sabes si tu salto saltará o no (*muy seguramente*).

La caché ahora es más grande para almacenar todas las combinaciones, los bits de "BHSR" **Branch History Shift Register**. Los **XX** del dibujo son el **estado en el que está el predictor** (*los autómatas estos*), y para **elegir qué columnas** tienes que ver lo que han hecho los de antes Si es **SÍ y SÍ** (**0,0** o **1,1**) la 1r, si es **SÍ** y **NO** (**0,1** o **1,0**) la 2nd…)

Predictor híbrido (*Contador*)



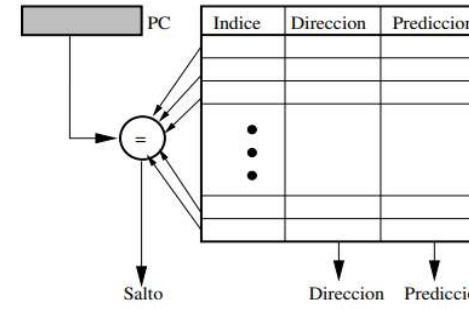
Cada predictor es apropiado para diferentes patrones. Por lo que vamos a Combinar varios predictores, seleccionando el más adecuado para caso (o patrón). Ejemplo: ***Tournament Predictor***

Mecanismo de selección: Elige el predictor que haya **dado mejores resultados hasta el momento**. Si el P2 acierta cont ++, si acierta el P2 cont -- (*se pegan entre ellos por el contador, si los dos aciertan o fallan no cambia, pero si uno acierta y el otro no, **cambia a favor del que acierte***)

Si es negativo pillas P2, si es positivo pillas P1. *Como es binario, puedes ver el primer bit pa ver si es pos o negativo.*

BRANCH TARGET BUFFERS (*BTB*)

Aquí **Condición + Dirección**, usando **el PC**, y **almacenándolo** con **correspondencia totalmente asociativa** Y en la **etapa** de búsqueda de la instrucción **IF**. Se el PC pa decirte si salta o no, a donde (y si es un salto xq se hace en IF).



Caché: Ahora esta cosa es totalmente asociativa, xq pones el PC entero. Tiene el **índice (PC)**, la **dirección donde saltas** y **si salta**.

Hasta ahora se hacía como mejor en Predict no Taken y latencia 1 ID. Ahora hay que **cambiar el procesador** (la ruta segmentada) para que se haga en IF, latencia 0. NO penalización. *Saber antes de decodificar*.

Cuando ahora la instrucción que llega es **un salto**, se mira la caché,

- **NO está**: Si no está pues se predice "NO" y se ejecuta normal.
- **Sí está**: Si sí que está hay 2 opciones: *Predicción sí salta* o *No salta*.
  - **Salta**: Si es sí salta, pues **vas ejecutando el salto** y te pones ya a buscar en la dirección a la que saltas nuevas instrucciones.
  - **NO salta**: Si es no salta, pues **sigues ejecutando normal por donde ibas**.

Predicción errónea: Si la predicción es errónea, tendrás que **cancelar todas las instrucciones que se han empezado a ejecutar**, devolver el **PC a donde toca** y **actualizar la tabla de la cache** con la nueva predicción.