Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE-0521 Estructuras de computadoras digitales II
I Ciclo 2024

.

Tarea # 1

Estudio sobre predictores de saltos

Estudiantes Jimena González Jiménez, B83443
Juan Ignacio Hernández Zamora, B93826
Profesor: Erick Carvajal Barboza

7 de Agosto de 2024

1. Introducción

En esta tarea se trabajó con tres predictores. Un pshared, uno basado en perceptrones y uno de creación propia. Se programó un simulador para cada uno. El pshared contiene una tabla de historia local y una tabla de patrones donde se encuentran los predictores de 2 bits. Todas estas estructuras son parametrizables. Luego, se programó el predictor de perceptrones con base en el artículo proporcionado con el enunciado. Este trabaja en un umbral de: $\theta = [1{,}93h + 14]$, el cual es proporcionado en el mismo artículo [1]. El cual ayuda a decidir si los pasos tomados para actualizar las predicciones son los correctos. Por último, se tiene el predictor UCR-IE0521. El cual fue de autoría del grupo. Este se realizó con un presupuesto de 32kb.

2. Predictor p-shared

Este predictor corresponde al estudiado en clases. Es un predictor que funciona con historia local. Este está compuesto por dos tablas. Una de registros de historia y otra de patrones, como se muestra en la figura.

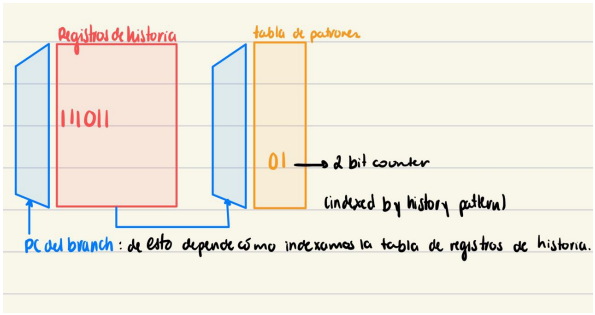

Figura 1: Diagrama básico de cómo se relacionan las tablas del predictor Pshared

La tabla de registros de historia se indexa con base en el PC del branch. La cual contiene también un registro deslizante, que es el que contiene la historia de los saltos que han ocurrido. Cabe recalcar que estos tienen un PC que termina en los números del lado izquierdo de la tabla.

PC del branch	Resultado más reciente
00	registro desplazable
01	registro desplazable
10	registro desplazable
11	registro desplazable

Luego, con base en la información de esta tabla se indexa la tabla de patrones:

Índice (Historia local)	
0000	pred. de 2 bits
0001	pred. de 2 bits
0010	pred. de 2 bits
...	pred. de 2 bits
1111	pred. de 2 bits

La cual contiene la historia local y predictores de dos bits. En resumen, con el PC del branch se

accede a la primera historia y se obtiene un valor. Luego, utilizando esa historia se indexa una segunda tabla donde ya entra el predictor.

2.1. Resultados

Una vez que se probó el predictor, se obtuvo la siguiente tabla de resultados:

Tamaño Historia local	Bits PC para indexar				
	4	8	12	16	20
2	66,836	74,22	84,939	86,597	86,692
6	68,543	76,427	88,027	89,488	89,58
8	69,71	77,388	89,109	90,518	90,589
16	84,616	85,82	92,497	93,507	93,559
20	87,913	86,464	92,391	93,401	93,447

La cual tiene combinaciones para 4, 8, 12, 16 y 20 bits del PC. Mientras que para la historia local tiene tamaños de 2, 6, 8, 16 y 20. Como se puede observar, para primer combinación de 4 bits del PC y dos de tamaño de la historia local se obtuvo un porcentaje de predicciones correctas del 66,836 %. El cual se aproxima mucho al proporcionado en el enunciado de 66,84 %. Por lo cual se puede concluir que el predictor está funcionando correctamente.

En segundo lugar, hay que recalcar que entre mayor sean las cantidades de bits de historia local y del PC aumenta la precisión del predictor. Esto, debido a como se pudo observar en los ejemplos vistos en clase, el predictor puede entrenarse mucho mejor. Ya que el problema en estos ejemplos es que se hacían muy pocas predicciones, lo cual entorpecía un poco el proceso de aprendizaje del predictor. Por otro lado, el tener más bits de PC y de historia global también aumenta la precisión del predictor ya que puede guardar más historial, lo cual se ve que se traduce en mejores predicciones. Por último, si se mantuvieran constante ambas cantidades de bits, no se obtendría la mejora en precisión que se observa en la tabla y en la gráfica. Por las razones mencionadas anteriormente.

Por último, se graficaron los resultados de la tabla. Donde se ve aun más claro el aumento en la precisión del predictor con el aumento de los bits PC para indexar y el tamaño de la historia local.

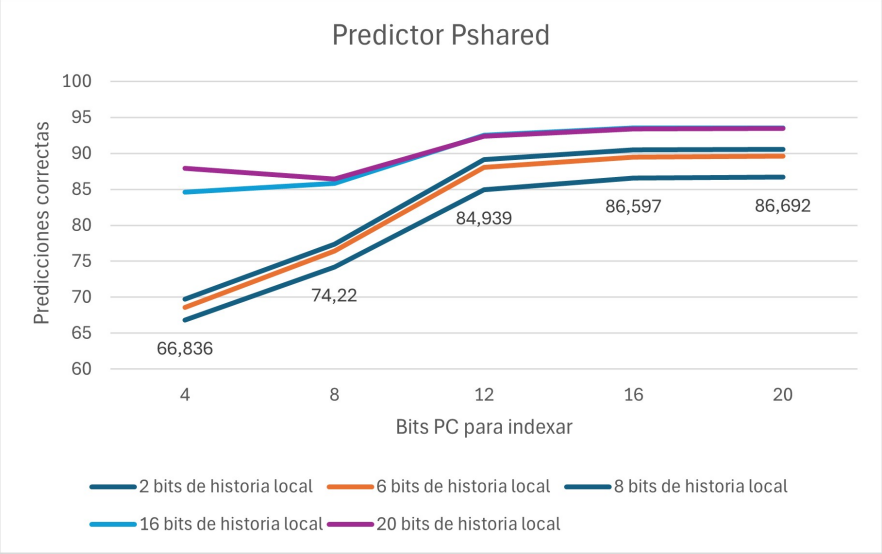

Figura 2: Gráfica de resultados para el predictor pshared.

3. Predictor basado en perceptrones

También se trabajó un predictor basado en perceptrones. El cual funciona con base en la red neuronal mas simple: un perceptrón. Es un predictor bastante más preciso ya que funciona con una historia de branches más larga, lo cual es posible gracias a que los recursos de hardware escalan linealmente con el largo de la historia. Mientras que para otra clase de predictores estos escalan exponencialmente.

Según el artículo proporcionado por el profesor [1], este diseño se evaluó respecto a dos predictores ya conocidos. También se muestra que para un presupuesto de 4k bytes de hardware, el método mejoró la exactitud del predictor para el benchmark del spec 2000 10,1 % arriba del predictor g-shared.

Cabe recalcar que en artículo se habla que el predictor funciona mejor en branches linealmente separables. Por lo tanto, es complementario a predictores ya existentes y funcionaría bien como parte de un predictor híbrido.

Aún así, este tipo de predictor tiene la debilidad de que es más complejo computacionalmente comparado con los contadores de dos bits. Tampoco posee la habilidad de aprender funciones linealmente inseparables. De igual manera, a pesar de estas debilidades, el predictor de perceptrones tiene un buen desempeño. Con una menor cantidad de predicciones incorrectas en todos los presupuestos de hardware. A diferencia del predictor global spec 2000.

3.1. Resultados

Bits historia global	Bits PC para indexar				
	4	8	12	16	20
2	71,496	84,303	90,859	91,466	91,48
6	74,266	90,029	93,665	93,847	93,86
8	74,783	91,282	94,306	94,445	94,445
16	76,459	93,697	95,942	96,062	96,07
20	77,019	94,242	96,242	96,369	96,377

Al igual que con el predictor pshared, se hizo una tabla con los resultados de la simulación del predictor de perceptrones. De primera entrada, se puede notar que este predictor es bastante más exacto que el anterior. Igual que para el predictor anterior, se trabajó con 4, 8, 12, 16 y 20 bits del PC para indexar y 2, 6, 8, 16 y 20 bits para el tamaño de los registros de historia global.

En este caso, según el enunciado la primera simulación tenía que tener un 71,47 % de predicciones correctas. Mientras que la realizada dio un 71,496 % de predicciones correctas. Por lo que se puede asegurar que el predictor está funcionando adecuadamente.

Al igual que con el predictor anterior, se puede observar que al aumentar la cantidad de bits de historia global y del PC para indexar, aumenta la cantidad de predicciones correctas. Lo cual indica que si se quedaran estáticos no habría un cambio significativo en la cantidad de predicciones correctas. Contrario a lo que se observa en ambos experimentos. Otro aspecto importante que se mencionó en el artículo proporcionado[1], es que este tipo de predictor únicamente necesita un crecimiento lineal del hardware para la historia global. A diferencia de otros predictores que requieren un crecimiento exponencial. Lo cual se ve reflejado en el mayor porcentaje de predicciones correctas por cantidad de bits del PC y de historia global. Ya que en este caso el aumento se está dando exponencialmente. Que se ve mucho mejor ilustrado en la siguiente gráfica:

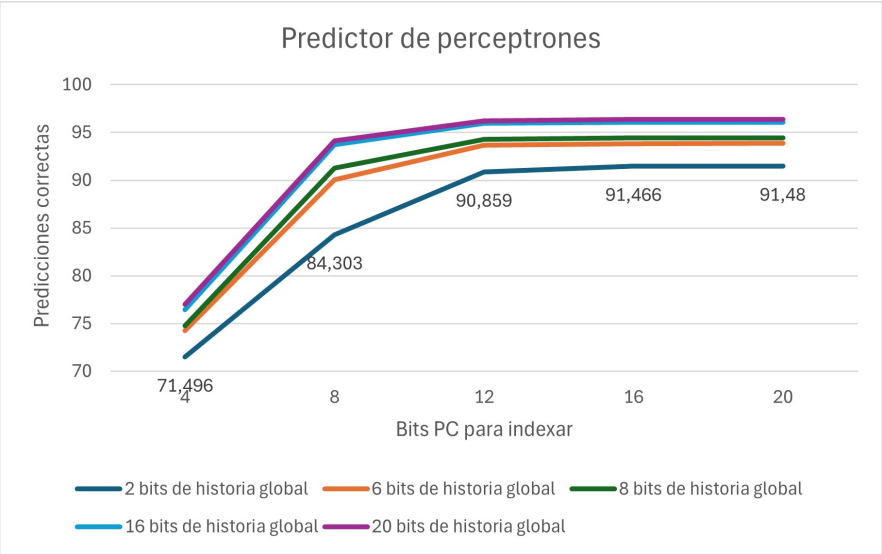

Figura 3: Gráfica de resultados para el predictor de perceptrones.

4. Predictor UCR-IE0521

El predictor propuesto pretende implementar la indexación del contador de programa de manera que se reduzca el *aliasing*, para esto se utiliza una matriz de tres dimensiones y se divide la lectura del PC tal que:

$$Indice_{PC1} = PC_{N..,0} \,\%Tamao_{Dim1} \tag{1}$$

$$Indice_{PC2} = PC_{M+N...N+1} \,\%Tamao_{Dim2} \tag{2}$$

Donde se indexa según N dígitos para la primera dimensión y M para el segundo. Los dígitos son diferentes. Se muestra una representación de la matriz propuesta:

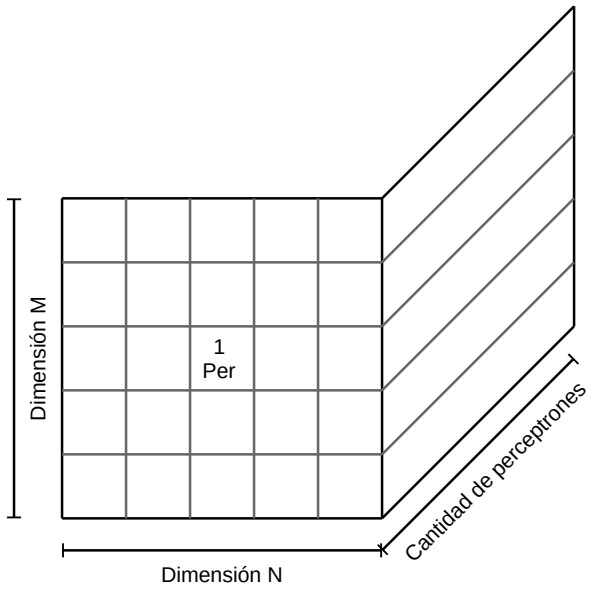

Figura 4: Matriz de perceptrones en tres dimensiones.

Ahora, sobre este modo de indexado del *PC* se implementó un predictor de perceptrones al que se le limitó, en términos de bits, el tamaño de los números enteros que pueden tomar los pesos.

El ajuste del predictor se realizó mediante la variación del tamaño de la historia y la magnitud de las dimensiones. La cantidad de perceptrones es la de la historia global más uno.

El mejor desempeño obtenido para este predictor fue de:

$$\text{Mejor resultado} \;=\; 75{,}327\,\% \tag{3}$$

4.1. Cálculo del presupuesto

A la hora de obtener cuánta memoria consume el predictor propuesto, basta con obtener la cantidad de pesos de todos los perceptrones, multiplicar la cantidad de *bits* que ocupa cada uno, y sumarle la longitud de la historia global.

Se presenta la tabla de cálculo utilizada para obtener el presupuesto consumido:

Cuadro 1: Tabla de cálculo del presupuesto

Longitud de la historia global (*bits*)	14
Longitud de la dimensión 1	5
Longitud de la dimensión 2	4
Cantidad de perceptrones ($2^{Dim_1+Dim_2}$)	512
Pesos por perceptrón (Dimensión 3 = BHT+1)	15
Cantidad de pesos en la tabla	7680
Tamaño de *int* seleccionado (*bits*)	4
Peso total de la tabla (*bits*)	30734

La longitud de los valores enteros se obtuvo con una suma truncada y los valores se representan en binario con complemento a dos. Así, para los cuatro bits utilizados, los pesos pueden estar entre $[-7, 7]$.

Nótese que el valor de presupuesto utilizado, 30734 *bits*, es menor al máximo dado en el enunciado de la tarea de 32768 *bits*.

5. Verificación de resultados

Con la entrega de la tarea se presenta un *README* en el que se incluyen las instrucciones para replicar los resultados, así como la hoja de cálculo.

Además, se la tarea se encuentra disponible en un repositorio de *GitHub* en la siguiente dirección electrónica.

6. Conclusiones

Para finalizar, fue muy interesante realizar las simulaciones y ver los resultados de las mismas en gráficas y en tablas. Ya que se muestra muy claro las diferencias de desempeño de ambos predictores. Al igual que se pueden observar características del predictor de perceptrones ya en acción. Una duda que surgió debido a que en el artículo asignado se menciona que es más complejo computacionalmente que otros predictores. Lo cual me deja con la duda, este artículo es del 2001. ¿Actualmente ya se están utilizando esta clase de predictores en la industria o sigue siendo algo exclusivo de la academia?

Referencias

[1] D. Jimenez and C. Lin, "Dynamic branch prediction with perceptrons," in *Proceedings HPCA Seventh International Symposium on High-Performance Computer Architecture*, pp. 197–206, 2001. 1, 3, 3.1