Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE0521 Estructuras de Computadoras Digitales II
I ciclo 2024

Tarea 1

Branch Prediction

Elias Alvarado Vargas B80372
Grupo 02

Profesor: Erick Carvajal Barboza

3 de Mayo de 2024

Índice

1. Objetivos

Implementar esquemas de predicción de saltos por medio de simulaciones en un lenguaje de alto nivel.

2. Descripción

- Programar un simulador de predictores de saltos, en C/C++ o Python, que permita obtener métricas de rendimiento utilizando como entrada un *trace* donde se indica la dirección de memoria donde ocurre el salto, así como el resultado del mismo (tomado o no).
- Implementar un predictor de saltos estilo p-shared.
- Implementar un predictor de saltos basado en perceptrones.
- Realizar una serie de experimentos que permitan explorar el espacio de diseño de los predictores de salto.
- Proponer, implementar y evaluar un algoritmo de predicción de saltos nuevo.

3. Resultados

En forma general para esta sección se muestra el despliegue de resultados al ejecutar el programa **branch_predictor.py** que se detalla en la sección *Ejecución* y los experimentos de los predictores.

3.1. Predictor de Saltos Tipo P-Shared

Este predictor debe contar con una tabla de historia local y una tabla de patrones donde se almacenen los predictores de 2 bits. Los tamaños de todas las estructuras se encuentran parametrizadas en su ejecución.

El despliegue de resultados para este predictor que como prueba se tiene n(bits por indexar) de 4 y g(historia local) de 2 se presenta en la siguiente Figura 1

```
PS C:\Users\elias\OneDrive\Desktop\Tarea01_Estructuras\Tarea01_Estructuras\Entrega_Tarea1> python .\branch_predictor.py --bp 2
-n 4 -g 2
Parámetros del predictor:
        Tipo de predictor:                                      P-Shared
        Entradas en el Predictor:                               16
        Entradas en la Historia Global:                         2
Resultados de la simulación
        # branches:                                             16416279
        # branches tomados predichos correctamente:             2879994
        # branches tomados predichos incorrectamente:           3331601
        # branches no tomados predichos correctamente:          8092038
        # branches no tomados predichos incorrectamente:        2112646
        % predicciones correctas:                               66.836%
```

Figura 1: Despliegue Resultados para P-Share. Elaboración propia

3.1.1. Experimentos y Gráfica

Los experimentos como tal se muestran en la Figura 2 para todas las combinaciones de:

- Bits del PC para indexar = 4, 8, 12, 16, 20
- Tamaño de los registros de historia local = 2, 6, 8, 16, 20

```
PS C:\Users\elias\OneDrive\Desktop\Tarea01_Estructuras\Tarea01_Estruc
turas\Entrega_Tarea1> & C:/Users/elias/AppData/Local/Microsoft/Window
sApps/python3.11.exe c:/Users/elias/OneDrive/Desktop/Tarea01_Estructu
ras/Tarea01_Estructuras/Entrega_Tarea1/experimento_pshare.py


Tabla con Resultados para Predictor P-Share

                Bits del PC para indexar
Tamaño
HL
        4       8       12      16      20
2       66.836% 74.220% 84.939% 86.597% 86.692%
6       68.543% 76.427% 88.027% 89.488% 89.580%
8       69.710% 77.388% 89.109% 90.518% 90.589%
16      84.616% 85.820% 92.497% 93.507% 93.559%
20      87.913% 86.464% 92.391% 93.401% 93.447%
```

Figura 2: Tabla Resultados para P-Share. Elaboración propia

Luego la gráfica del porcentaje de predicciones correctas para este predictor P-Share se muestra en la siguiente Figura 3

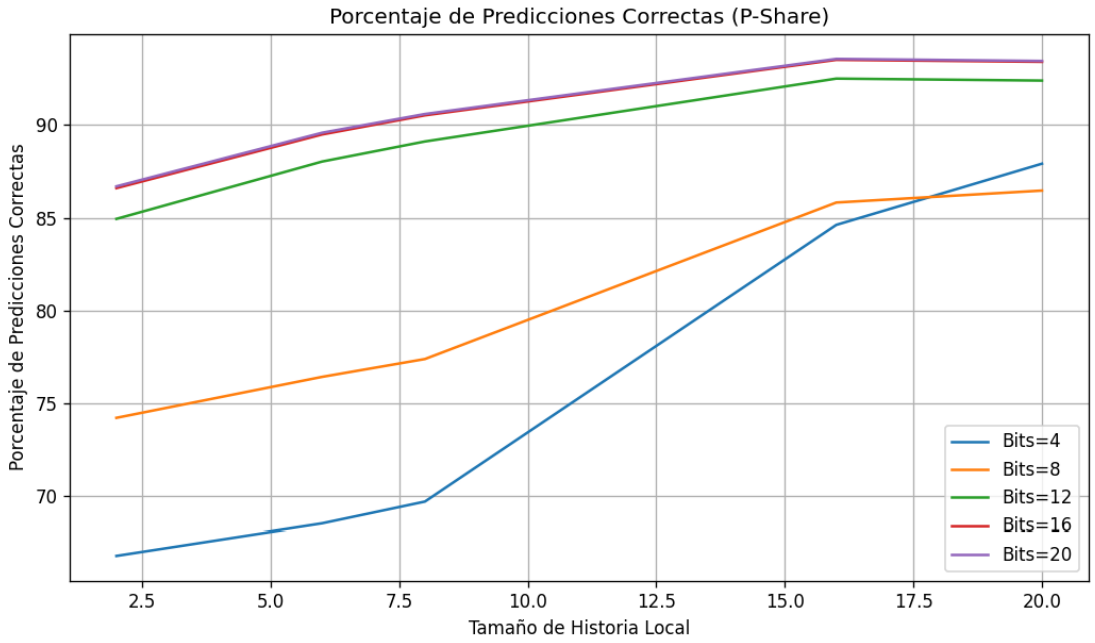

Figura 3: Gráfica Predicciones Correctas P-Share. Elaboración propia

3.1.2. Análisis de Resultados

El análisis de resultados con respecto al predictor P-Share, el tamaño de la historia local produce de forma directa la capacidad del predictor en captar branches a lo largo del tiempo, entonces si se mantiene constante la cantidad de bits del PC, aumentar el tamaño del registro de historia debe mejorar la precisión del predictor, y esto porque permite captar y utilizar más datos de la historia para hacer predicciones.

Ahora para cuando la cantidad de bits del PC presenta la cantidad de entradas que puede tener la tabla de historia local que se encuentra asociada a cada dirección de retorno del branch, puede esto mejorar la precisión, sin embargo hay un punto en el que agregar más bits al PC puede disminuir la precisión.

3.2. Predictor de Saltos Tipo Perceptrones

Este predictor se encuentra descrito en el artículo *Dynamic branch prediction with perceptrons*, de Daniel Jiménez. Para acceder al artículo mediante este link `https://drive.google.com/file/d/1N3K1PFxwYAQ0LCSGgDriJMcmzwpUxbYt/view`. Donde los pesos para cada perceptron se representan mediante números, el threshold utilizado es $\Theta = [1{,}93h + 14]$.

En la siguiente Figura 4 se muestra el despliegue de resultados para este predictor, la prueba usada es n(bits por indexar) de 4 y g(historia local) de 2

```
PS C:\Users\elias\OneDrive\Desktop\Tarea01_Estructuras\Tarea01_Estructuras\Entrega_Tarea1> python .\branch_predictor.py --bp 3
-n 4 -g 2
Parámetros del predictor:
        Tipo de predictor:                                      Perceptron
        Entradas en el Predictor:                               16
        Entradas en la Historia Global:                         2
Resultados de la simulación
        # branches:                                             16416279
        # branches tomados predichos correctamente:             2743725
        # branches tomados predichos incorrectamente:           3467870
        # branches no tomados predichos correctamente:          8996727
        # branches no tomados predichos incorrectamente:        1207957
        % predicciones correctas:                               71.517%
```

Figura 4: Despliegue de Resultado para Perceptron. Elaboración propia

3.2.1. Experimentos y Gráfica

Para los experimentos del predictor basado en perceptrones, se observa en la Figura 5 para todas las combinaciones de:

- Bits del PC para indexar = 4, 8, 12, 16, 20
- Tamaño de los registros de historia global = 2, 4, 8, 16, 20

```
PS C:\Users\elias\OneDrive\Desktop\Tarea01_Estructuras\Tarea01_Estruc
turas\Entrega_Tarea1> & C:/Users/elias/AppData/Local/Microsoft/Window
sApps/python3.11.exras/Tarea01_Estructuras/Entrega_Tarea1/experimento
ras/Tarea01_Estructuras/Entrega_Tarea1/experimento_perceptron.py


Tabla con Resultados para Predictor Perceptron

                Bits del PC para indexar
Tamaño
HG
        4       8       12      16      20
2       71.517% 84.303% 90.863% 91.481% 91.495%
6       74.299% 90.030% 93.671% 93.861% 93.877%
8       74.817% 91.285% 94.311% 94.463% 94.475%
16      76.483% 93.698% 95.946% 96.077% 96.086%
20      77.039% 94.154% 96.248% 96.385% 96.393%
```

Figura 5: Tabla Resultados Perceptrones. Elaboración propia

Luego en la Figura 6 se muestra el porcentaje de predicciones correctas.

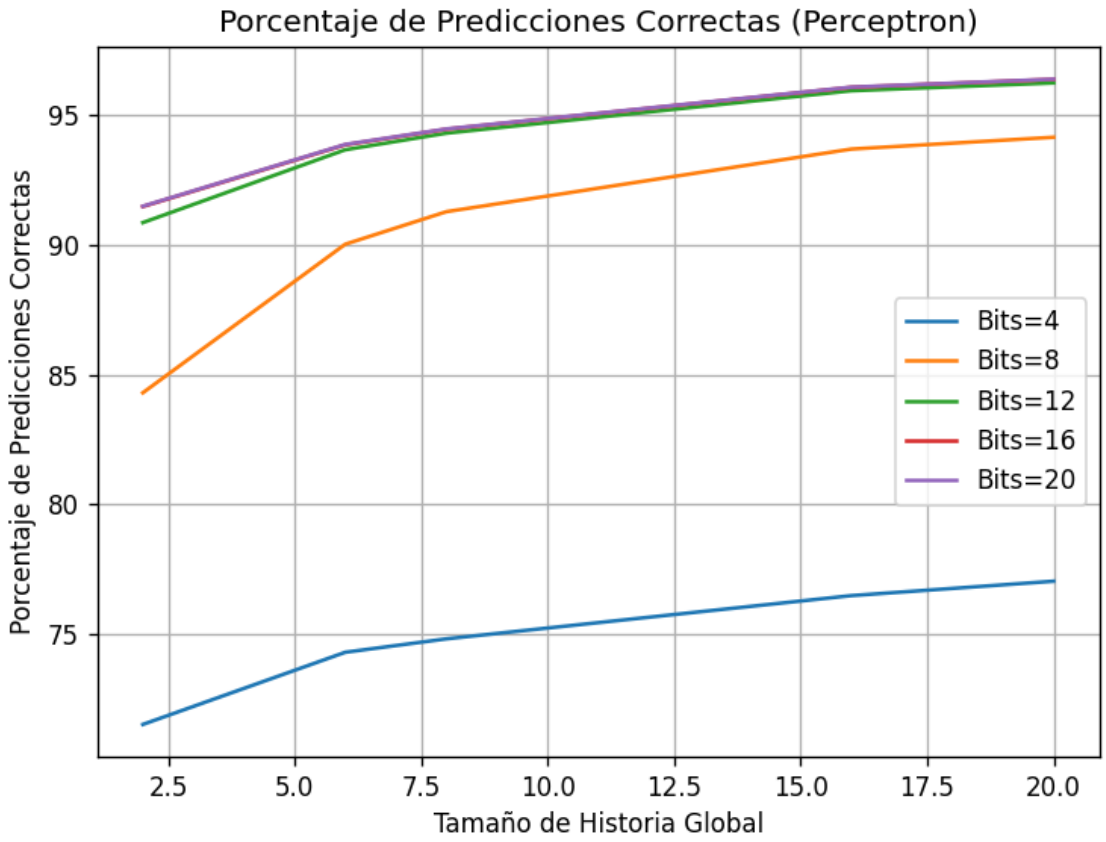

Figura 6: Gráfica Predicciones Correctas para Perceptron. Elaboración propia

3.2.2. Análisis de Resultados

En el caso de este predictor Perceptron, el tamaño del registro de historia afecta la complejidad y cantidad que puede aprender este predictor, por lo tanto un registro de historia más grande proporciona más datos de entrada y que potencialmente mejore en las predicciones de los branches, además entre más grande es este registro de historia puede aumentar la complejidad y el tiempo en el que entrena.

Además si hay un cambio en la cantidad de bits del PC utilizados, esto afecta de cierta manera la granularidad con la que el predictor puede diferenciar entre las direcciones, entonces un mayor número de bits del PC permite una indexación más precisa de entradas en el registro de historia global, al final debería de cierta forma como efecto en mejorar la precisión en la medida en que el predictor pueda distinguir mejor entre diferentes comportamientos de branches.

3.3. Predictor de Saltos propuesto

Como parte de la descripción del algoritmo implementado, es basado en la historia global de predicciones y de la tabla de branches. Por lo tanto, en cada celda de la tabla almacena un valor entre el 0 y 1, que representa la probabilidad de salto en caso que se tome o no se tome. Luego con el uso de cinco contadores se registra el número de las predicciones correctas e incorrectas, en el código se presenta las funciones que hacen una predicción y luego una actualización.
La justificación del predictor propuesto ie0521_bp tiene como base en la idea de que los saltos recientes son más importantes para poder predecir saltos futuros, mediante el uso de probabilidades de que un salto se tome en el futuro, y la forma en que se almacena en las tablas de branches, lo anterior permite al predictor poder aprender y adaptarse a diferentes patrones.
El cálculo realizado para obtener el correcto presupuesto, el enfoque va a no estar tan cerca del límite por lo que se tiene:

- bits_index = 14
- global_history = 6

El cálculo del presupuesto para esta combinación es la siguiente:

$$2^6 * 14 * 32 = 28672 bits \tag{1}$$

Entonces utilizando estos valores de bits por indexar y de la historia global, en la siguiente Figura 7 presenta el despliegue de resultados para este predictor.

```
PS C:\Users\elias\OneDrive\Desktop\Tarea01_Estructuras\Tarea01_Estructuras\Ent
rega_Tarea1> python .\branch_predictor.py --bp 4 -n 14 -g 6
Parámetros del predictor:
        Tipo de predictor:                                      ie0521_bp
        Entradas en el Predictor:                               16384
        Entradas en la Historia Global:                         6
Resultados de la simulación
        # branches:                                             16416279
        # branches tomados predichos correctamente:             4956002
        # branches tomados predichos incorrectamente:           1255593
        # branches no tomados predichos correctamente:          9566394
        # branches no tomados predichos incorrectamente:        638290
        % predicciones correctas:                               88.463%
```

Figura 7: Despliegue de Resultados. Elaboración propia

3.3.1. Análisis de Resultados para predictor propuesto

En esta sección se analiza mediante la tabla y gráfica que de igual forma se realizaron en los predictores anteriores para tener una idea del comportamiento y determinar de forma paralela los parámetros para el presupuesto que se propuso.

En la siguiente Figura 8 presenta la tabla con los resultados del predictor ie0521_bp que fue de gran ayuda para poder tener una mejor visión del comportamiento y de elegir los valores del presupuesto que fueron de n(14) y g(6) como se explicó antes no completa la cantidad de los 32768 bits sino que 28672 bits de forma que cumple con el presupuesto y no arriesga en estar bajo el punto crítico.

```
sApps/python3.11.exe c:/Users/elias/OneDrive/Desktop/Tarea01_Estructu
ras/Tarea01_Estructuras/Entrega_Tarea1/experimento_propuesto.py


Tabla con Resultados para Predictor ie0521_bp

               Bits del PC para indexar
Tamaño
HG
        4       8       12      16      20
2       69.658% 76.492% 85.593% 86.949% 87.011%
6       69.573% 77.706% 87.280% 88.670% 88.740%
8       69.439% 77.789% 87.441% 88.817% 88.886%
16      68.675% 77.893% 87.515% 88.820% 88.891%
20      68.727% 77.883% 87.527% 88.806% 88.876%
```

Figura 8: Tabla Resultados Propuesto. Elaboración propia

La gráfica de este predictor ie0521_bp mostrando el porcentaje de predicciones correctas se muestra en la Figura 9

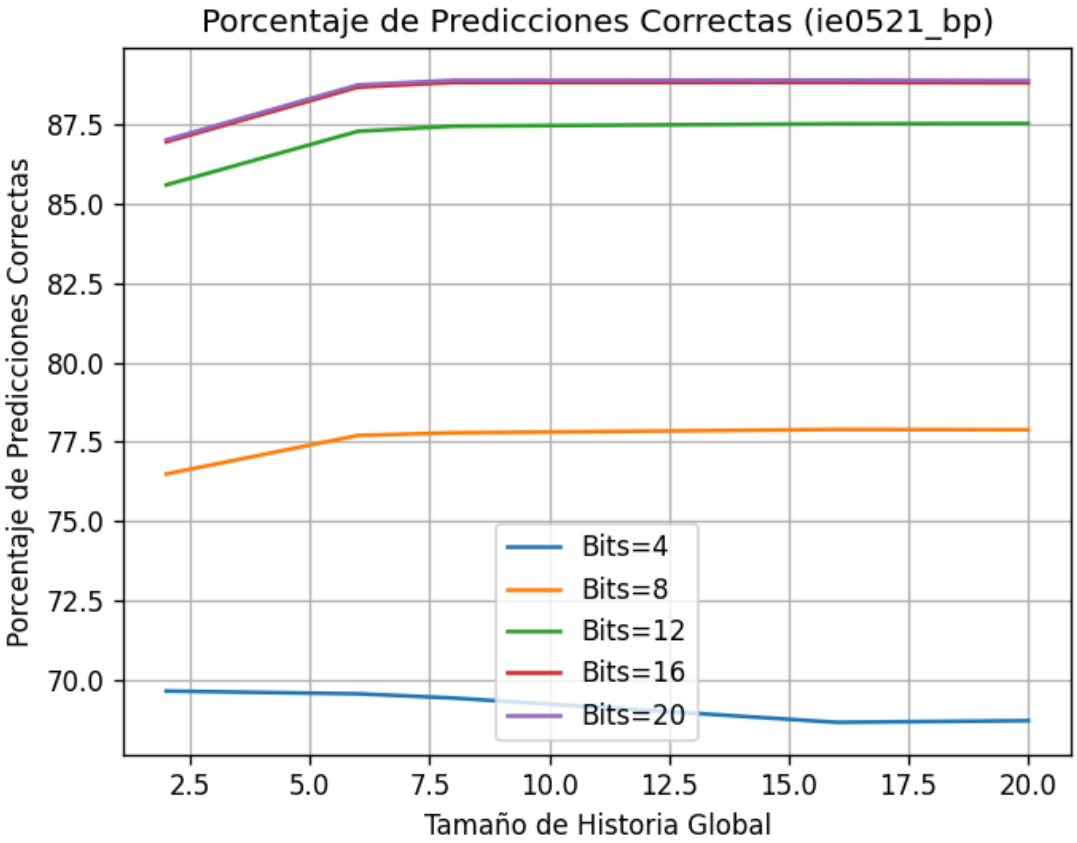

Figura 9: Gráfica porcentaje de predicciones correctas ie0521_bp. Elaboración propia

4. Conclusiones

- Como parte de los predictores implementados tienen un funcionamiento esperado, para analizar los resultados se implementan tres scripts para cada uno, la ejecución lleva un tiempo y al final presenta la gráfica con las predicciones realizadas.
- Para el predictor de saltos implementado P-Share, se tiene un porcentaje de predicciones correctas de 66.836 % para el escenario de bits por indexar con un valor de 4 y del tamaño de historia local de 2 por lo que verificando con la tabla de experimentos el valor dado coincide por lo que se puede concluir que este predictor funciona de forma correcta.
- Con respecto al predictor Perceptron el porcentaje de predicciones correctas da como resultado 71.517 % para un escenario igual al mencionado en el anterior predictor, por lo que verificando con el resultado dado que es de 71.47 % tiene un porcentaje de error muy pequeño, esto puede producirse por la inicialización u otros factores pero como tal el predictor funciona de forma correcta.
- Para el predictor propuesto fue un reto ya que se creó un tipo de predictor híbrido por decir lo una forma entre el pshare,gpshare y perceptrones, pero funcional. Otro aspecto que se destaca fue el comprender como puede ajustarse al presupuesto de 37 kbits, por lo que analizando se logra determinar los parámetros para que se encuentre dentro del mismo y no sobrepase lo establecido con el diseño.

5. Ejecución de los Predictores

En esta sección presenta un poco la estructura de los archivos.

Los archivos que vienen adjuntos

- branch_predictor.py: Ese el codigo por ejecutar, las instrucciones parametrizadas para el predictor P-Share y Perceptron se encuentran en el README adjunto. Para el predictor propuesto también viene la línea con los parámetros hard-codeados con los valores de tamaño de cada estructura
- La definición de las clases y funciones de los predictores se encuentran en los archivos: pshared.py, perceptron.py y ie0521_bp.py
- Los experimentos donde se tiene las tablas y gráficas generadas se encuentran en los archivos: experimento_pshare.py, experimento_perceptron.py y experimento_propuesto.py. Al ejecutar estos programas se muestran estos resultados y cabe destacar que tiene un poco de duración, al final se muestra la gráfica en base a la tabla obtenida.

Otro aspecto importante, para la entrega no se agregaron los predictores bimodal y g share dados, ya que no se toman en cuenta pero para seguir convención solamente imprime en pantalla que no están disponibles en caso de utilizarse ya que en el enunciado se menciona que no es necesario incluirlos en la entrega