**Jesus Zuñiga Mendez**
**Estructuras de computadoras Digitales II**
**Tarea # 1 Predicción de saltos**
**I Semestre 2023**

## Predictor por perceptrones de Jimenez

Los resultados obtenidos con el predictor de perceptrones fueron los siguientes

| Bits de Historia Global | Bits del PC para indexar | | | | |
|---|---|---|---|---|---|
| | **4** | **8** | **12** | **16** | **20** |
| **2** | 58.137% | 72.269% | 86.573% | 88.467% | 88.573% |
| **6** | 54.593% | 69.013% | 85.068% | 87.138% | 87.271% |
| **8** | 53.728% | 68.472% | 84.522% | 86.745% | 86.886% |
| **16** | 54.069% | 67.490% | 83.968% | 86.216% | 86.312% |
| **20** | 54.121% | 67.958% | 83.853% | 86.059% | 86.173% |

- Si la cantidad de bits del PC se mantiene constante, ¿cómo cambia la precisión ante cambios en el tamaño del registro de historia?

  La variación observada en el porcentaje de aciertos es muy pequeña, para el PC constante, siendo la diferencia entre el porcentaje mayor y el menor en la mayoría de casos alrededor del 4%, sucede algo particular al aumentar el número de bits en la historia ya que se obtienen mejores resultados con historias pequeñas que con historias grandes, probablemente porque el vector xi está relacionado con el tamaño de la historia, por lo que yout se puede calcular con una mayor densidad de pesos con historias pequeñas, este comportamiento se puede considerar contraintuitivo ya que se esperaría que a mayor cantidad de bits en la historia mejorarían las predicciones pero se demuestra que el comportamiento es distinto.

- Si la cantidad de bits del registro de historia global se mantiene constante,¿cómo cambia la precisión ante cambios en la cantidad de bits del PC utilizados para indexar?

  El cambio reflejado es bastante grande, ya que la mejor predicción se da con una historia de 2 bits, y si esta se mantiene constante la diferencia entre un PC de 4 y un PC de 20 es poco más del 30%, esto se comporta según lo esperado ya que a mayor cantidad de bits a indexar se tiene un registro más

amplio de los cambios en cada branch, por lo que los pesos de ese branch corresponden a un grupo más pequeño de PCs dando la oportunidad de mejorar las predicciones con cada salto mejorando así el porcentaje de aciertos

## Predictor propuesto JZMFilter2023BP

- Una descripción del algoritmo, debe explicar qué componentes utiliza, cómo realiza las predicciones y cómo se actualiza.

  **Componentes:** El algoritmo propuesto funciona gracias a una tabla de tamaño $2^n$ donde n es la cantidad de bits del PC seleccionado, esta tabla en cada entrada tiene un vector de x posiciones, donde x es la cantidad de bits de la historia deseado, la idea de esto es que se pueda almacenar una historia individual para cada PC almacenado,esta es la única estructura que almacena datos.
  **Predicción:** La predicción se hace con una especie de selección de pesos, primero se saca el vector almacenado en el PC consultado, pj: [0,0,1,1], siendo 1 los saltos tomados “T” y 0 los saltos no tomados “N”, se evalúa la diferencia entre Ts y Ns y si la diferencia es positiva, es decir hay más Ts que Ns se predice que ese branch según su historia se debe tomar, por el contrario si es una diferencia negativa, es decir hay más Ns que Ts, se predice que ese Branch no se toma, para los casos en los que la diferencia es cero quiere decir que hay igual cantidad de Ts que de Ns por lo que se hace una evaluación con una especie de Mux de dos entradas donde si s=00 ó s = 01 se predice N y para los otros dos casos se predice T, esto fundamentado en que un branch se toma muy frecuentemente o no se toma muy frecuentemente.
  **Actualización:** Consiste únicamente en un registro desplazante que representa la historia de cada PC indexado, para lo que se obtiene la historia en el PC consultado y se modifica almacenando en el bit más significativo un 1 si el salto real es T y un 0 si el salto es N y el resto de bits los desplaza hacia la izquierda para después almacenar esta historia en el índice de la tabla correspondiente al PC

- Una justificación del algoritmo, debe explicar por qué el algoritmo propuesto podría funcionar.

  El algoritmo funciona ya que en esencia es un filtro que identifica si una rama se está tomando frecuentemente o no se está tomando frecuentemente, y los casos medios los evalúa en función del comportamiento esperado más lógico, es decir si las últimas dos ramas se tomaron entonces se debe tomar la siguiente y si la últimas dos ramas no se tomaron entonces la siguiente

tampoco se debe tomar, para los casos donde la última si se tomó pero la penultima no y viceversa se supone un comportamiento periodico donde se esta tomando las ramas o no alternadamente. El algoritmo debe funcionar ya que si el PC es relativamente grande se están indexando en la tabla los PCs similares más distribuidamente, por lo que se puede tener una historia pequeña que indica como es el comportamiento más reciente de una rama y ya que las ramas estás más distribuidas se puede evitar un poco el aliasing. en la sección de resultados hay una captura de pantalla del resultado obtenido con los tamaños indicados en el cálculo de presupuesto.

- El cálculo del presupuesto de hardware, el cual debe ajustarse al límite de 64kb.

  En el codigo tambien se calcula el presupuesto por lo que se puede ver en cada simulación, este se calcula de la siguiente manera:

  **Tamaño de la tabla**: Es igual a $2^n$ con n = PC = 14 por lo que el tamaño de la tabla es de 16384 campos

  **Tamaño del vector de historia:** es el tamaño de la historia deseada en este caso 4

  **Presupuesto utilizado:** 16384 * 4 = 65536 = 64kb

  Lo anterior mencionado son los únicos elementos del predictor que necesitan memoria.

- Resultados

```
PS C:\Users\jezch\OneDrive\Documentos\Estructuras Compus2\Tarea1\Entregables> python .\branch_predictor.py
No se pudo importar el predictor bimodal.py
No se pudo importar el predictor gshared.py
Parámetros del predictor:
        Tipo de predictor:                      JZMFilter2023BP
        Bits del PC para indexar:                       14
        Tamaño de los registros de historia global:     4
Informacion sobre presupuesto:
Otorgado:    65536    Gastado:    65536    Sobrante:    0
Tracer cargado con éxito!!!
Prediciendo saltos:...
                                                                                                                  100%
Resultados de la simulación
        # branches:                                             16416279
        # branches tomados predichos correctamente:             5218223
        # branches tomados predichos incorrectamente:           993372
        # branches no tomados predichos correctamente:          9336396
        # branches no tomados predichos incorrectamente:        868288
        % predicciones correctas:                               88.660%
Predictor evaluado con 16416279 lineas en 174.2505819797516 segundos.
PS C:\Users\jezch\OneDrive\Documentos\Estructuras Compus2\Tarea1\Entregables>
```

Cómo se puede observar el porcentaje de acierto es relativamente alto, alcanzando un 88,6% por lo que se puede considerar que el algoritmo aunque es relativamente sencillo realiza un buen trabajo de predicción en un tiempo corto