Lab 5: Impacto del Paralelismo a Nivel de Instrucción

Ejecución del Programa:

En esta tarea simplemente hemos ejecutado el código que se nos ha dado al comienzo de la práctica y nos hemos fijado en el tiempo de ejecución de la función. El resultado que hemos obtenido ha sido de 4397 ns. Además, el tiempo de ejecución total del programa “bpred1” ha sido de 0,0241 segundos.

Saltos y Predicción de saltos:

Tras haber ejecutado el código una vez, hemos hecho uso de la herramienta “perf” para ver el comportamiento de los saltos. Esto ha hecho que el tiempo de ejecución aumente un poco hasta los 5182 ns.
En cuanto a las estadísticas de saltos, hemos obtenido que de 10.409.177 saltos, el 14,71% (1.530.732) de estas instrucciones son predicciones fallidas.

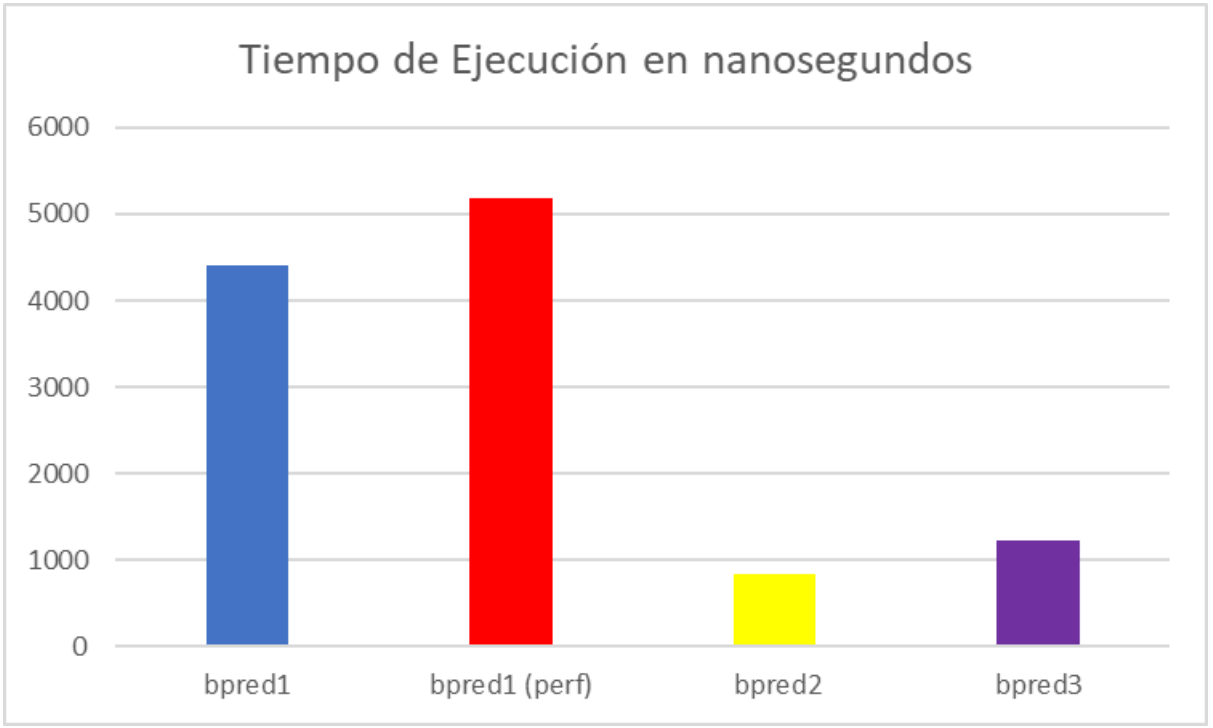
Impacto en secuencias Ordenadas:

En este apartado hemos visto cómo influye al tiempo de ejecución del programa ordenar el vector en números positivos y negativos antes de ejecutar la función. Para ello, el tiempo que tomamos ha sido tras ordenar el vector y no antes. El resultado de la ejecución de la función que hemos obtenido es de 840 ns. Este tiempo es mucho menor comparado al del código base. Esto se debe a que, gracias a que tenemos el vector ordenado, el predictor va a acertar casi en la totalidad de las veces. de esta forma aumenta la velocidad considerablemente aunque no va a reducir el número de fallos. El speedup de este método es de 5,235.
Hay que destacar también que el tiempo de ejecución del programa es mucho mayor al de “bpred1”, puesto que se ejecuta una función adicional (el algoritmo para ordenar el vector). Este tiempo de ejecución es de 0,0899 segundos.

Eliminación de Saltos:

Finalmente, en la última tarea hemos visto cómo afecta al tiempo de compilación la eliminación de saltos. Para ello hemos hecho uso de un operador condicional ternario para generar una máscara de bits y así aplicarla sobre el vector para así obtener solo los valores positivos. El resultado de la ejecución ha sido de 1232 ns. Nuevamente mucho menor que el del código base pero no tan bajo como el de la tarea anterior. Esto se debe a que gracias a la máscara, nos libramos de todos los valores negativos y hacemos la suma de los valores positivos. El speedup de este método es de 3,569.

Debajo adjuntamos un gráfico con los valores obtenidos:



Conclusión:

En este laboratorio no hemos tenido muchos problemas a la hora de aplicar los distintos métodos de optimización de tiempo en los saltos, aunque sí que es cierto que aplicar el operador condicional ternario y hacer uso de una máscara y operaciones bit a bit para eliminar los saltos requirió un poco más de tiempo para poder entender qué era lo que se pedía en el guión y conseguir una ejecución correcta del programa.