# MEMORIA PRACTICA3

## a) Medir el tiempo de ejecución del bucle

```
for(int k=0; k<200; k++){
        for(int i = 0; i < 2176; i++){
                for(int j = 0; j < 2176; j++){
                        A[j][i] = A[j][i] * k + B[i][j] * C[j][i] + 3.14 * D [i][j]
                }
        }
}
```

| Opción -O0 | Opción -O2 |
|---|---|
| 25’942.4 ms | 20’014.5 ms |

## b) Desarrollar el algoritmo de Padding y realizar las modificaciones correspondientes en el código. Para el mismo número de iteraciones de los bucles, medir tiempo de ejecución del bucle.

Calculamos el stride del siguiente algoritmo:

```
for(int k=0; k<200; k++){
        for(int i = 0; i < 2176; i++){
                for(int j = 0; j < 2176; j++){
                        A[j][i] = A[j][i] * k + B[i][j] * C[j][i] + 3.14 * D [i][j]
                }
        }
}
```

| **Calculamos el valor de FA para cada matriz** | | | |
|---|---|---|---|
| FA(A) | 2176 * j + i | | |
| FA(B) | 2176 * i + j | | |
| FA(C) | 2176 * j + i | | |
| FA(D) | 2176 * i + j | | |
| **Calculamos el stride de cada matriz** | | | |
| | x=i | x=j | |
| Stride(A,x) | 1 | 2176 | |
| Stride(B,x) | 2176 | 1 | |
| Stride(C,x) | 1 | 2176 | |
| Stride(D,x) | 2176 | 1 | |
| **Calculamos el valor de SetStride de cada matriz** | | | |
| | x=i | x=j | |
| SetStride(A,x) | 0 (64) | 8 | |
| SetStride(B,x) | 8 | 0 (64) | |
| SetStride(C,x) | 0 (64) | 8 | |
| SetStride(D,x) | 8 | 0 (64) | |

| | | | |
|---|---|---|---|
| **Seleccionamos todos los arrays Xi con stride >> W en el bucle más interno del anidamiento Ik** | | | |
| NewStride(A,j) | Stride(A,j)+W = 2176+16 = 2192 | | |
| NewStride (B,i) | Stride(B,i)+W = 2176+16 = 2192 | | |
| NewStride (C,j) | Stride (C,j) + W = 2176 + 16 = 2192 | | |
| NewStride (D,i) | Stride(D,i) + W = 2176 + 16 = 2192 | | |
| **Calculamos el SetStride de los NewStride** | | | |
| SetStride(NewStride(A,j)) | 9 | | |
| SetStride(NewStride(B,i)) | 9 | | |
| SetStride(NewStride(C,j)) | 9 | | |
| SetStride(NewStride(D,i)) | 9 | | |
| **Calculamos la ocupación de la caché debida a las referencias a un array X en un bucle Y** | | | |
| | x=i | x=j | |
| GCD(SetStrike(A,x),64) | 1 | 1 | |
| GCD(SetStrike(B,x),64) | 1 | 1 | |
| GCD(SetStrike(C,x),64) | 1 | 1 | |
| GCD(SetStrike(D,x),64) | 1 | 1 | |

Para obtener estos resultados, debemos de aumentar el tamaño de las columnas de las matrices A y C en 16. Una vez aplicado estos cálculos, obtenemos los siguientes tiempos:

| Opción -O0 | Opción -O2 |
|---|---|
| 20’225.8 ms | 19’526.5 ms |

**c) Para los resultados obtenidos en los casos de ejecución anteriores, realizar la comparación de tiempos de ejecución con las distintas opciones de optimización en la compilación (defecto, O0 y O2)**

Si observamos los tiempos de cómputo, comprobamos que al aplicar la opción -O2 se produce un tiempo inferior al obtenido con la opción -O0.

Además, el tiempo de cómputo al realizar el padding también disminuye si lo comparamos a los tiempos obtenidos sin aplicar el padding.

**d) Justificar resultados, aportando características hardware del equipo donde se han realizado las pruebas**

Cuando no se aplica el padding, la ocupación de la caché se realiza del siguiente modo: Se rellenan 64 filas de la caché y se utiliza una. Esto provoca un mayor tiempo de ejecución del algoritmo.

Al aplicar el padding, se rellenan 64 filas de la caché y se utilizan las 64 filas de la caché. Esto produce una reducción notable del tiempo de ejecución del algoritmo, tal y como se muestran en los datos obtenidos anteriormente.

Al utilizar GCD, tenemos que el número de set donde se pueden ubicar referencias a X en Ik son 64/GCD(X,Ik).

Dado que el valor de GCD con padding es 1, tenemos que la ocupación de la caché es máxima; en cambio, si no aplicamos padding la ocupación de la caché no es máxima

Las características hardware del equipo empleado son las siguientes:

| | |
|---|---|
| Procesador del equipo | Intel Core i7-8565U |
| Entorno de desarrollo | Ubuntu 20.04.3 |
| Herramienta de desarrollo | C++ |
| Compilador empleado | G++ version 9.3.0 |

**Características de la memoria caché L1**

| | |
|---|---|
| data | 4x32 KB (4 bloques de 32 KB) |
| line size | 64 B |
| Associativity | 8-way set |

Numero de palabras por bloque: 16
Calculamos el número de líneas por vía:

$$\frac{32\,KB}{8} = \frac{4\,KB}{via} \rightarrow \frac{\frac{4\,KB}{via}}{\frac{64\,B}{via}} = \frac{64\,lineas}{via}$$