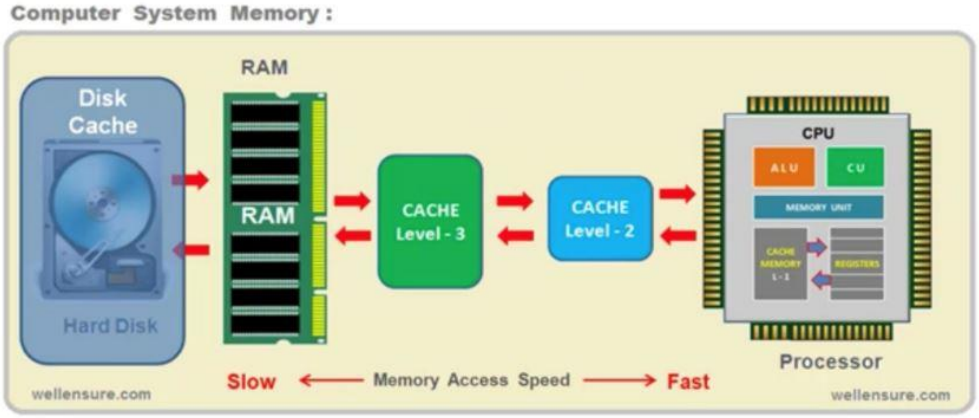
# Trabajo final Arquitectura de Computadores/ Interfaces y Arquitectura Hardware

*Student Outcome 06: An ability to develop and conduct appropriate experimentation, analyze and interpret data, and use engineering judgment to draw conclusions.*

## TEMA: El impacto de la localidad de caché en el rendimiento de la multiplicación de matrices

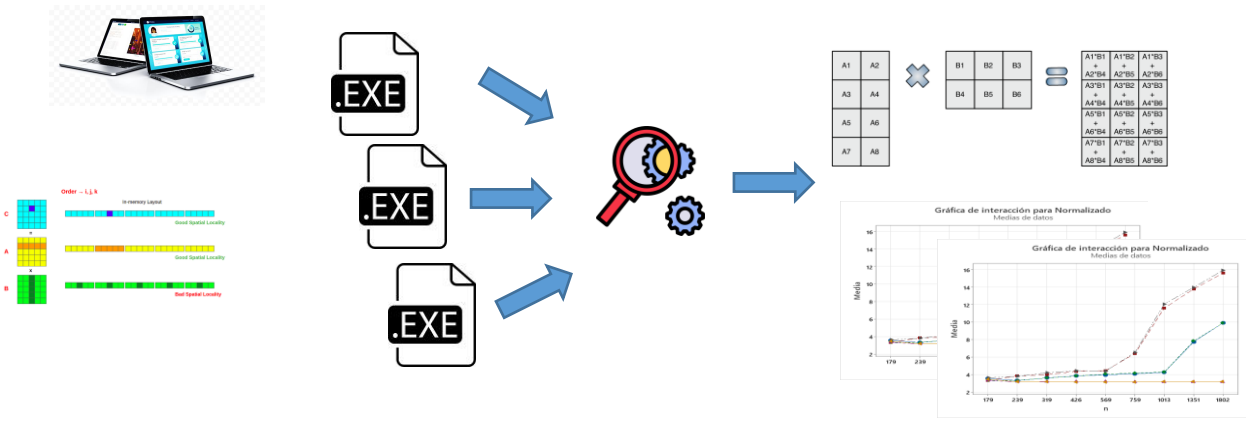



*Tomado de https://postImg.cc/DSTjyxF2*

## Introducción

A lo largo del curso se ha estudiado cómo la estructura y arquitectura de los computadores influyen en el rendimiento de los programas, enfocándose en cómo los componentes de hardware y software afectan el procesamiento, acceso y almacenamiento de datos. El análisis del desempeño, en particular la interacción con la memoria caché y el principio de localidad, es clave para mejorar el rendimiento. La memoria es un cuello de botella común, por lo que medir los tiempos de acceso es esencial para optimizar configuraciones y evaluar cómo diferentes programas, con igual complejidad, pueden comportarse de manera distinta en función de su aprovechamiento de la jerarquía de memoria. Un ejemplo de esto es el concepto de "memory mountain", que permite visualizar el rendimiento de la memoria en función del tamaño y patrón de acceso a los datos. Así, un buen diseño experimental ayuda a medir el impacto de distintos factores sobre el desempeño de los sistemas de cómputo

EL esquema general del diseño experimental es el siguiente:




## Objetivo:

Analizar el impacto en el rendimiento de la ejecución de diferentes algoritmos de multiplicación de matrices equivalentes, que comparten la misma complejidad algorítmica, pero presentan distintos niveles de aprovechamiento del principio de localidad. Para ello, se aplicará la metodología de conducción de experimentos y se apoyará en la teoría de arquitectura de computadores. Además, de acuerdo al análisis de los resultados, deberá proponer un esquema de multiplicación por bloques y evaluar su capacidad para mejorar el desempeño de los algoritmos deficientes, implementando esta mejora únicamente en el peor algoritmo para determinar si se obtiene un incremento en el rendimiento.

El trabajo se divide en dos partes: primero, medir el rendimiento de seis versiones de algoritmos de multiplicación de matrices y analizar la interacción con los demás factores; y segundo, analizar los resultados, proponer una mejora mediante la multiplicación por bloques, y comprobar su efectividad a partir de los datos recolectados en las fases previas.

## Factores de Estudio:

Versión del Algoritmo: a, b, c, d, e, y f
Tamaño de la matriz de datos:

**Gr 101:** n= 185, 450, 550, 650, 750, 900, 1160, 1480, 1700, 2000

**Gr 103:** n= 175, 420, 525, 625, 725, 875, 1150, 1475, 1725, 1985

**Gr 105:** n= 170, 440, 565, 670, 780, 915, 1185, 1470, 1700, 2050

**Gr 107:** n= 180, 430, 545, 635, 770, 890, 1140, 1490, 1700, 1990

Tipo de dato: Float y Double

Tipo de ISA: x86 y x64

## Condiciones del experimento

Para este trabajo tenga en cuenta que puede formar equipos con dos grupos de laboratorio (máximo 4 personas).

Se debe usar dos unidades experimentales, es decir dos equipos personales de cómputo. Los datos recolectados por cada uno de las unidades experimentales, los debe analizar de manera independiente y realizar un análisis comparativo.

La primera parte, el interés es cuantificar y analizar el grado de interacción o dependencia que tiene las seis versiones de código con el sistema de memoria.

Las versiones del programa.

En la figura siguiente aparecen 6 versiones del programa que cumplen con la misma función de calcular el producto entre dos matrices cuadradas de tamaño n, C=AxB:

(a) Version $ijk$

code/mem/matmult/mm.c

```
1  for (i = 0; i < n; i++)
2      for (j = 0; j < n; j++) {
3          sum = 0.0;
4          for (k = 0; k < n; k++)
5              sum += A[i][k]*B[k][j];
6          C[i][j] += sum;
7      }
```

code/mem/matmult/mm.c

(b) Version $jik$

code/mem/matmult/mm.c

```
1  for (j = 0; j < n; j++)
2      for (i = 0; i < n; i++) {
3          sum = 0.0;
4          for (k = 0; k < n; k++)
5              sum += A[i][k]*B[k][j];
6          C[i][j] += sum;
7      }
```

code/mem/matmult/mm.c

(c) Version $jki$

code/mem/matmult/mm.c

```
1  for (j = 0; j < n; j++)
2      for (k = 0; k < n; k++) {
3          r = B[k][j];
4          for (i = 0; i < n; i++)
5              C[i][j] += A[i][k]*r;
6      }
```

code/mem/matmult/mm.c

(d) Version $kji$

code/mem/matmult/mm.c

```
1  for (k = 0; k < n; k++)
2      for (j = 0; j < n; j++) {
3          r = B[k][j];
4          for (i = 0; i < n; i++)
5              C[i][j] += A[i][k]*r;
6      }
```

code/mem/matmult/mm.c

(e) Version $kij$

code/mem/matmult/mm.c

```
1  for (k = 0; k < n; k++)
2      for (i = 0; i < n; i++) {
3          r = A[i][k];
4          for (j = 0; j < n; j++)
5              C[i][j] += r*B[k][j];
6      }
```

code/mem/matmult/mm.c

(f) Version $ikj$

code/mem/matmult/mm.c

```
1  for (i = 0; i < n; i++)
2      for (k = 0; k < n; k++) {
3          r = A[i][k];
4          for (j = 0; j < n; j++)
5              C[i][j] += r*B[k][j];
6      }
```

code/mem/matmult/mm.c

Figure 6.46 **Six versions of matrix multiply.** Each version is uniquely identified by the ordering of its loops.

En la parte de análisis debe incluir una sección donde explique en detalle y a la luz la teoría relevante, la relación del comportamiento de la memoria y el tiempo de respuesta del experimento. Tenga en cuenta entre otros conceptos, los principios de localidad, la jerarquía de memoria de cada tipo de procesador y el AMAT.

Para efectos de evaluación y calificación se aplicará una rúbrica la cual considera los siguientes aspectos que sigue la mayoría de los pasos de la metodología general de diseño de experimentos:

Restricciones obligatorias y aclaraciones:

a) En las pruebas preliminares debe evidenciar y comprobar que cada uno de los algoritmos realiza la operación correctamente.
b) Recuerde normalizar los datos de tiempo de ejecución, ya que se va manejar diferentes tamaños de matrices. Debe reportar tanto los valores de tiempo total de ejecución como los valores normalizados.
c) Lea muy bien los pasos de la metodología de diseño de experimentos y tenga en cuenta la rúbrica de evaluación.

Para efectos de evaluación y calificación se aplicará una rúbrica la cual considera los siguientes aspectos que sigue la mayoría de los pasos de la metodología general de diseño de experimentos:

| Aspecto a evaluar | | Factor /evidencia de logro |
|---|---|---|
| Justificar y concebir un experimento de acuerdo a una una claridad del problema a resolver, teorías relevantes e identificación de los factores de tratamiento y factores secundarios. | Objective 5% | Es capaz de identificar el problema a resolver y definir los objetivos del experimento. |
| | Methods 30% | Es capaz de reconocer teorías y conceptos relevantes y utilizarlos para explicar los resultados esperados del experimento. |
| | | Identifica la variable de respuesta y los niveles de los factores primarios o de tratamiento. |
| | | Identificación de factores secundarios y de ruido y la manera de controlar sus efectos. |
| | | Selección del tipo de experimento de acuerdo con la hipótesis / cuestiones por resolver, los factores y las limitaciones existentes |
| Llevar a cabo un experimento siguiendo los procedimientos definidos para adquirir datos sobre las variables de estudio usando los instrumentos o herramientas de medición apropiada y tomando medidas para minimizar la incertidumbre. Reportar y consignar adecuadamente los resultados. | Condut 25% | Es capaz de minimizar la incertidumbre experimental |
| | | Es capaz de registrar y representar datos de manera significativa. |
| Analizar e interpretar los resultados de un experimento para exponer conclusiones que permitan contestar preguntas o hipótesis del fenómeno computacional | Analysis 25% | Es capaz de analizar los datos de forma adecuada. |
| | | Es capaz de emitir un juicio sobre los resultados del experimento. |

| | | |
|---|---|---|
| | | |
| | Conclusion 15% | Puede llegar a una conclusión razonable y justificarla. |

Fecha límite de la entrega del documento: Martes semana 18 - 18:00.

**Entregables**

La tarea en INTU será el único medio permitido para entregar el trabajo final.

Por favor suba entregue en una archivo .ZIP lo siguiente:

1. Documento metodología, análisis y resultados DoE(formato .DOCX)

2. Datos resultados (formato .XLS)

3. Anexo: Archivos fuentes: Python script Colab, c++ files

**Referencias:**

[1] 1.Computer Systems: A Programmer's Perspective, Randal E. Bryant, David R. O'Hallaron. Addison Wesley Pub Co Inc; Edición: 0002 (Febrero de 2010).

[2] Diseño y análisis de experimentos Douglas C. Montgomery

[3] Writing Fast Programs: A Practical Guide for Scientists and Engineers, John Riley.

[4] Diseño y conducción de Experimentos en Arquitectura de Computadores - Terminología y Conceptos-