Instituto Tecnológico de Costa Rica **Escuela de Ingeniería en Computadores** **Programa de Licenciatura en Ingeniería en Computadores** **Curso:** CE-4302 Arquitectura de Computadores II **Profesores:** Luis Alonso Barboza Artavia Ronald García Fernández **Semestre:** II 2025	**Taller 02 Extensiones *SIMD* e *Intrinsics*** **Fecha de asignación:** 09/10/25 (grupo 02) 10/10/25 (grupo 01) **Fecha de entrega:** 30/10/25 (grupo 02) 31/10/25 (grupo 01) **Grupos de trabajo: 2 personas (máximo)**

1. Descripción

En el procesamiento de texto, una operación fundamental es la conversión de caracteres entre minúsculas y mayúsculas.

En la codificación ASCII, los caracteres alfabéticos tienen valores numéricos específicos, La conversión de una letra minúscula a su equivalente en mayúscula se puede realizar restando un valor fijo. Este taller busca que las personas estudiantes mediante técnicas de vectorización y programación *SIMD* con *intrinsics* implementen un conversor sencillo de minúsculas a mayúsculas y viceversa.

A su vez se pretende exponer a las personas estudiantes al uso de herramientas tipo *LLM* para la traducción código de un ISA a otro.

2. Requisitos

a- Sistema operativo basado en Linux (no virtualizado)
b- GCC con soporte para C++ e *Intrinsics* para una plataforma objetivo
c- Acceso a compiler explorer

3. Investigue

A continuación, se formulan una serie de preguntas de guía para investigar sobre la implementación de algoritmos empleando intrinsics y extensiones SIMD.

a- ¿En qué consisten las funciones intrinsics? ¿Cuáles son funciones y aplicaciones comunes? ¿Cómo se evalúa su desempeño sobre otro tipo de implementación?
b- ¿En C++ cómo es posible usar intrinsics para una arquitectura objetivo?
c- ¿Cómo se puede determinar el soporte de una extensión SIMD para una plataforma específica?
d- En C++ ¿Qué técnicas existen para medir el desempeño de un programa?
e- En C++ cómo es posible generar cadenas de caracteres aleatorias y cómo es posible definir como se guardan en memoria respecto a su alineamiento.
f- Al implementar un micro-benchmark. ¿Cómo se puede garantizar un estado “limpio” en los cachés entre ejecuciones independientes de funciones? Explique técnicas para limpiar la caché y minimizar el “ruido” entre pruebas sucesivas según, por ejemplo, ISA y SO.

4. Ejemplo y ejercicio de aplicación de instrucciones *SIMD* e *intrinsics*

Para efectos de este taller se le brinda el siguiente ejercicio el cual tiene que ser resuelto de forma analítica:

En el procesamiento de texto, una operación fundamental es la conversión de caracteres entre minúsculas y mayúsculas.

En la codificación ASCII, los caracteres alfabéticos tienen valores numéricos específicos. La conversión de una letra minúscula a su equivalente en mayúscula se puede realizar restando un valor fijo de 0x20, como se muestra en la tabla 1.

Caracter	Valor Hexadecimal
`'A'`	`0x41`
`'Z'`	`0x5A`
`'a'`	`0x61`
`'z'`	`0x7A`
`('a'-'A')`	`0x20`
`('z'-'Z')`	`0x20`

Tabla 1. Ejemplo de relación de caracteres ASCII mayúscula y minúscula

De acuerdo con esta información, se solicita lo siguiente:

a- Implemente el pseudocódigo para una función `case_converter_serial(text[m], case)` que tome una cadena de texto de entrada (un arreglo de caracteres/bytes) de longitud m y la convierta a mayúsculas o viceversa según el valor de `case`. La función debe modificar la cadena directamente (*in-place*). Sólo los caracteres que sean letras minúsculas o mayúsculas (según el rango ASCII `['a','z']` o `['A', 'Z']`) deben ser convertidos; los demás caracteres (mayúsculas o minúsculas según el valor de `case`, números, símbolos) deben permanecer sin cambios.

b- Utilizando un diagrama, describa como se comportará un algoritmo vectorizado para la conversión de minúsculas a mayúsculas y viceversa. Considere que el texto se procesa en bloques (vectores) de 256 bits.

c- Basándose en el diagrama del punto b, diseñe los *intrinsics*/instrucciones necesarias que permitan realizar la conversión de texto empleando vectores de 256 bits.

d- Empleando técnicas de vectorización y los *intrinsics* del punto c implemente el seudocódigo que permite transformar el texto de longitud m. Considere todos los posibles casos de longitud de m y explique cómo es posible no acceder a regiones de memoria no validas y procesar todo el texto.

e- Análisis de Rendimiento Vectorizado: Escalabilidad y Limitaciones

Considerando el algoritmo serializado (punto a) y el vectorizado de 256 bits (punto d) discuta:

i- Escalabilidad *SIMD*: Si el ancho del vector *SIMD* aumentara a 512 bits y el hardware de memoria/ejecución escalara adecuadamente, ¿Cómo impactaría esto el desarrollo de Software e ISA?

ii- Identifique dos cuellos de botella de hardware (a parte de las unidades de procesamiento *SIMD*) que podrían limitar el rendimiento real de la versión vectorizada de 256 bits.

iii- Explique brevemente como afecta el desempeño y la seguridad, la relación entre el tamaño de la cadena de texto y el tamaño del vector *SIMD*

5. Implemente

a- Implemente un generador de cadenas de texto aleatorias (UTF-8) donde se puede especificar el tamaño, en donde se pueda configurar como están alineados los datos, **ambos** deben poder ser definidos en tiempo de ejecución, el generador debe ser capaz de definir un porcentaje de caracteres alfabéticos (0-100%).

b- Basándose en lo realizado en las secciones 3 y 4, implemente C++ un algoritmo serial que resuelva el problema de conversión mayúscula – minúscula y viceversa, llame a este archivo **`case_converter_serial.cpp`**, incluya lo necesario para medir su desempeño (tiempo de ejecución, memoria, y cualquier otro aspecto que considere relevante).

c- Implemente el algoritmo *SIMD* mediante el uso de *intrinsics* para su plataforma elegida (Es necesario justificar como se eligió, además de emplear los *intrinsics* equivalentes a las funciones que se brindan en el enunciado y las propuestas en su solución en un archivo **`case_converter_SIMD.cpp`** **debe ser capaz de manejar datos no alineados.**

d- Valide la correctitud de los resultados de la versión *SIMD*/*intrinsics* usando la misma cadena en ambos casos.

e- Empleando el generador de cadenas de texto realice mediciones sobre ambas implementaciones, con diferentes tamaños de cadenas, porcentajes de caracteres alfabéticos y alineamientos (al menos 50 diferentes tamaños, 2 alineamientos y 10 diferentes porcentajes) tiene que justificar el rango de valores empleados en función de la plataforma y soporte de *SIMD*. (sugerencia implemente un script para automatizar esta tarea y realizar gráficas ver figura 2).

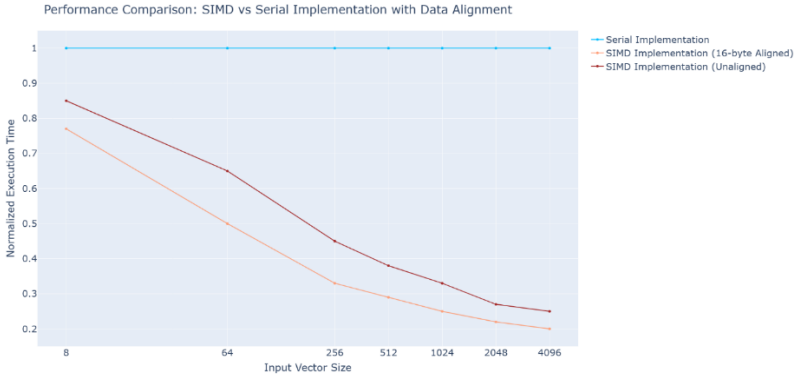

Figura 2. Ejemplo de resultados de tiempo de ejecución de algoritmos de conversión mayúscula y minúscula con un porcentaje de caracteres alfabéticos de 20%.

f- Mediante el uso de herramientas tipo LLM realice una traducción de su implementación a otro ISA, para esto emplee compiler explorer para validar sus resultados.

6. Justifique

Basándose en la implementación del punto 5 y lo investigado en este taller:

a- ¿Cómo afecta el alineamiento de los datos el uso de *intrinsics*?
b- Realice un análisis comparativo de las gráficas de desempeño al aumentar el tamaño de la cadena de caracteres en ambas implementaciones y el porcentaje de caracteres alfabéticos (serial e *intrinsics*)
c- ¿Qué consideraciones son necesarias cuando el tamaño de la cadena no es múltiplo del tamaño de vector *SIMD*?
d- Realice un análisis comparativo del código con *intrinsics* implementado y el obtenido usando *LLMs*, incluya en este aspectos como cantidad de líneas, soporte de documentación.

7. Entregables

a- Un documento **PDF** donde muestre las respuestas a las preguntas planteadas en secciones 3 y 4, junto con los resultados de la sección 5, 6, para las secciones en donde utilice *LLM* debe presentar el análisis hecho sobre los resultados de la herramienta y la validación respectiva de los mismos.

b- En un archivo taller_*SIMD*.zip el código fuente con las implementaciones del punto 5, un archivo **README** con los detalles para ejecutar su código, graficar sus resultados, y el enlace a *compiler explorer* con su traducción de ISA. NO incluya ejecutables de ser así se le asignará un 0 de nota.

Si tienen dudas puede escribir al profesor al correo electrónico. Los documentos serán sometidos a control de plagios. La entrega se debe realizar por medio del TEC-Digital en la pestaña de evaluación.

No se aceptan entregas extemporáneas después de la fecha de entrega a las 23:59 como máximo.