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:** I 2025	**Taller 02 Extensiones SIMD e *Intrinsincs*** **Fecha de asignación :** 20/05/25 (grupo 02) 21/05/25 (grupo 01) **Fecha de entrega :** 03/06/25 (grupo 02) 04/06/25 (grupo 01) **Grupos de trabajo: 2 personas (máximo)**

1. Descripción

El procesamiento eficiente de archivos de texto de gran tamaño requiere operaciones optimizadas, como el conteo de ocurrencias de un carácter específico, donde se busca contar la cantidad de veces que un carácter aparece en una cadena de longitud arbitraria, utilizando técnicas de vectorización y programación SIMD mediante *intrinsics*.

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.

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.

Una operación típica al manipular archivos de texto de gran tamaño es el conteo de ocurrencias de una palabra o un carácter (1 byte, UTF-8). En la figura 4 se muestra como ejemplo para una cadena de 16 caracteres **`str.`** Se puede realizar la búsqueda del carácter **`';'`** mediante una comparación entre cada elemento de **`str`** y el valor de **`';'`** (**`char_x`**), obteniendo una cadena de salida **`out_str`** con 2 valores posibles **`0x00`** y **`0xFF`**.

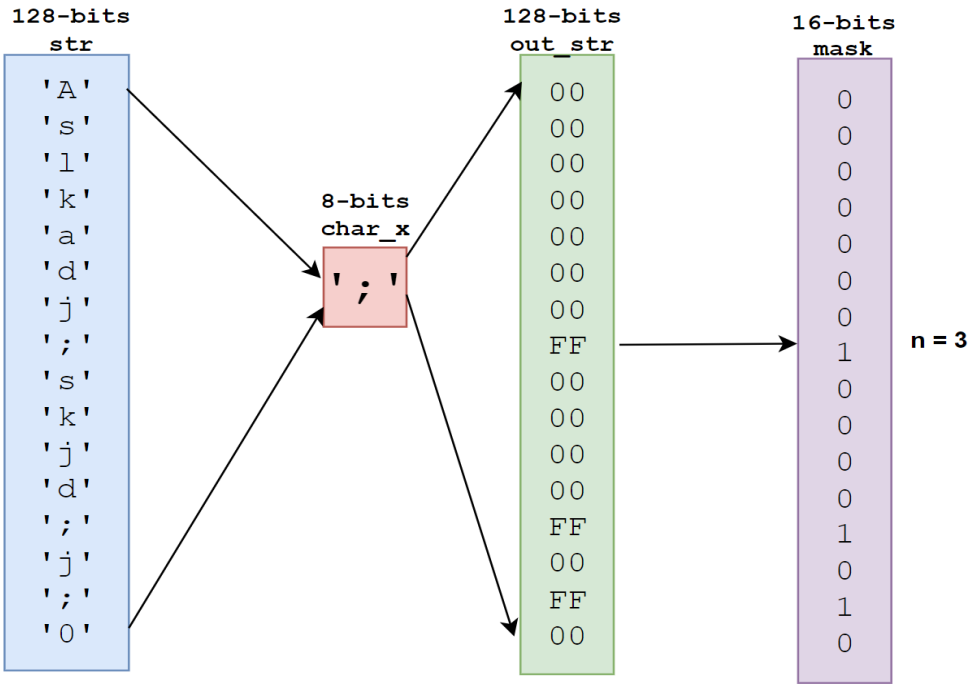

Figura 1. Ejemplo de búsqueda/conteo del carácter **`';'`**

De esta forma se puede interpretar el valor **`0xff`** como un **`1 lógico`** y **`0x00`** como un **`0 lógico`**, y así reducir **`out_str`** a un vector **`mask de 16 bits`** sobre el cual sólo es necesario contar el número bits con valor **`'b1`** (n = 3 para este ejemplo)**.**

Respecto a la aplicación anterior y suponiendo que dispone de 2 *funciones* para obtener el vector mask en una variable (**`mask = get_mask(simd_vector)`**) y contar el número de bits en '1' de una variable (**`bits_set = count_ones(mask)`**), se le solicita lo siguiente:

a- Implemente el seudocódigo que realice la operación de manera 'serializada' (sin vectorizar, ni usar mask, etc) para obtener la cantidad de ocurrencias de un carácter **`char_x`** en una cadena **`str`** de longitud **M**.

b- Basándose en la figura 4, diseñe un *intrinsic*/instrucción que permite replicar el valor de **`char_x`** en un vector *SIMD* de 128 bits **`vect_x`**.

c- Basándose en la figura 1, diseñe un *intrinsic/*instrucción que permite comparar dos vectores *SIMD* de **128 bits** `vect_x` y `vect_y` .

d- Empleando técnicas de vectorización y los *intrinsics* de los puntos **b** y **c implemente** el seudocódigo que permite encontrar la cantidad de ocurrencias de un carácter `char_x` en una cadena `str` de longitud **M**, considere todos los posibles casos de longitud de **M** y explique cómo manejarlos de manera "segura".

5. Implemente

i. 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, el tamaño de la misma debe poder ser definido en tiempo de ejecución.

ii. Basándose en lo realizado en las secciones 3 y 4, implemente C++ un algoritmo serial que resuelva el problema de cuenta de caracteres (**char_count_serial.cpp**) incluya lo necesario para medir su desempeño (tiempo de ejecución, memoria, y cualquier otro aspecto que considere relevante).

iii. 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 **char_count_simd.cpp debe ser capaz de manejar datos no alineados.**

iv. Valide la correctitud de los resultados de la versión SIMD/intrinsics usando la misma cadena en ambos casos y el mismo carácter de búsqueda.

v. Empleando el generador de cadenas de texto realice mediciones sobre ambas implementaciones, con diferentes tamaños de cadenas y alineamientos (al menos 50 diferentes tamaños y 2 alineamientos) 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).

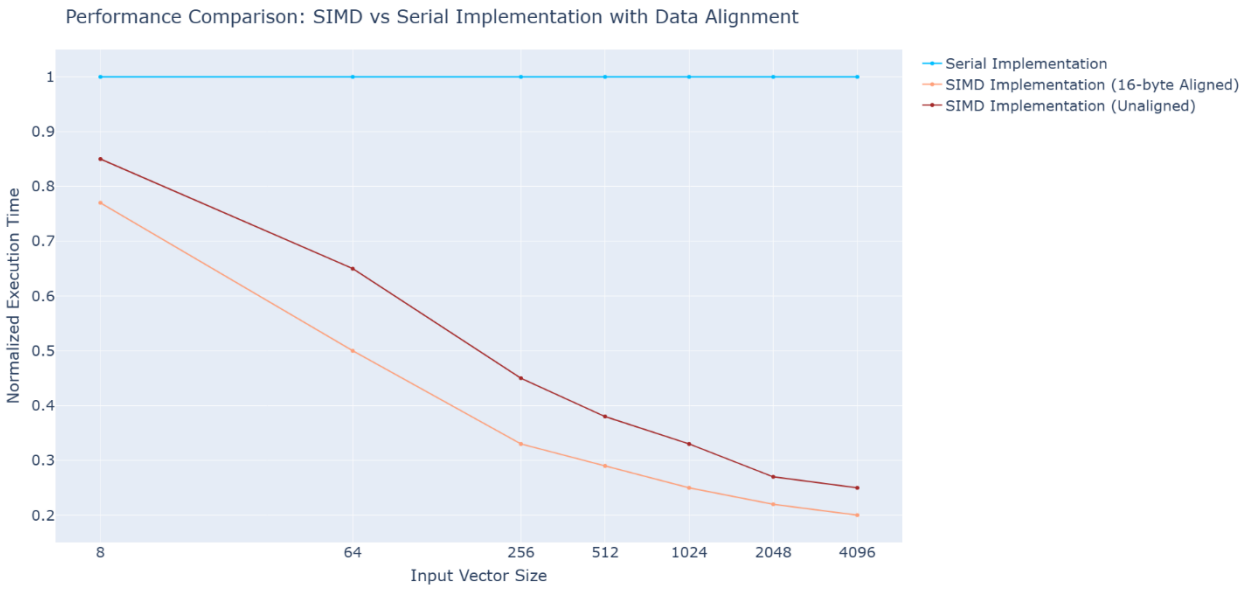

Figura 2. Ejemplo de resultados de tiempo de ejecución de algoritmos de búsqueda del carácter ';' con alineamiento de datos de 16 bytes

vi. 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 (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 aspecto 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 penalizara con 15%.

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.