1.Investigación

1.1. ¿En qué consiste un extensión SIMD llamada SSE?

SSE son las siglas de Streaming SIMD Extensions es una de las formas que se utiliza para la optimización de multiprocesadores modernos que permite la instrucción única de varios datos. Se amplió a través de las generaciones de procesadores Intel para incluir SSE2, SSE3/SSE3S y SSE4. Cada iteración ha aportado nuevas instrucciones y un mayor rendimiento.

SSE maneja solo vectores de 128 bits pero es compatible con la mayoría de los procesadore Intel y AMD. Existen versiones para vectores de mayor tamaño: AVX está desarrollado para 256 bits, pero no tiene tanto soporte, especialmente en procesadores más viejos y AVX512, para 512 bits, es soportado solo por procesadores de muy alto desempeño.

1.2. ¿Cuáles tipos de datos son soportados por este tipo de instrucciones?

Dependiendo de la versión que se esté utilizando varía, pero en general las instrucciones tienen un sufijo que define el tipo de dato, algunos de estos son:

- s single-precision floating point
- d double-precision floating point
- i128 signed 128-bit integer
- i64 signed 64-bit integer
- u64 unsigned 64-bit integer
- i32 signed 32-bit integer
- u32 unsigned 32-bit integer
- i16 signed 16-bit integer
- u16 unsigned 16-bit integer
- i8 signed 8-bit integer
- u8 unsigned 8-bit integer
- ps packed single-precision floating point
- pd packed double-precision floating point
- sd scalar double-precision floating point
- epi8/16/32/64 extended packed 8/16/32/64-bit signed integer
- epu8/16/32/64 extended packed 8/16/32/64-bit unsigned integer
- si256 scalar 256-bit integer

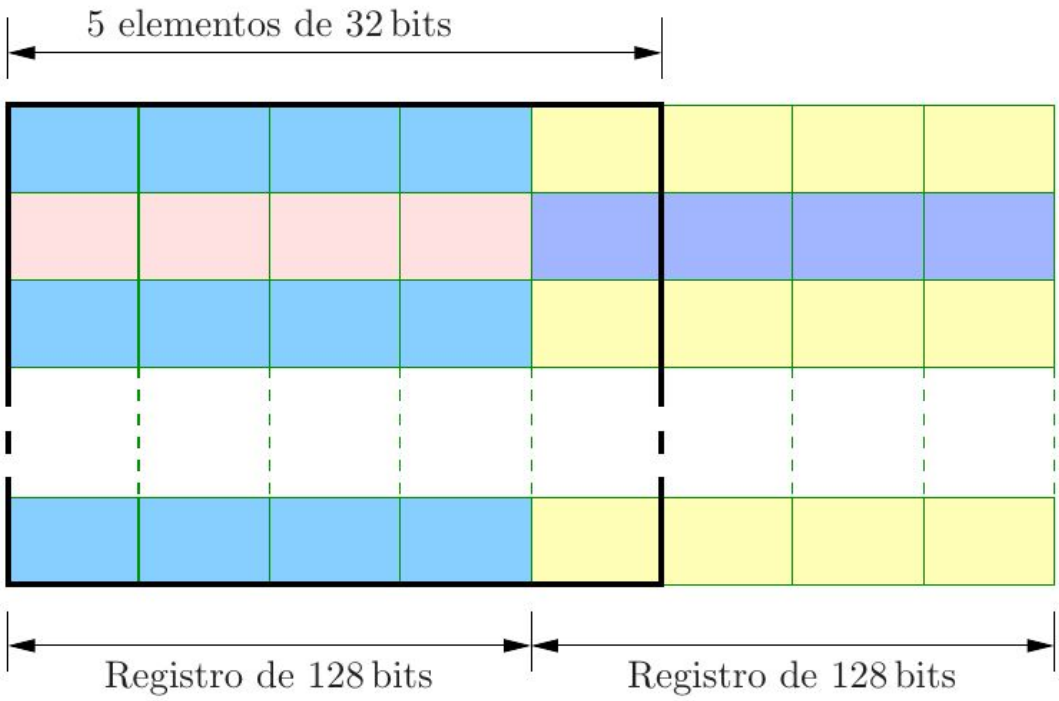
[https://scc.ustc.edu.cn/zlsc/sugon/intel/compiler_c/main_cls/intref_cls/common/intref_avx_details.htm]

1.3. ¿Cómo se realiza la compilación de un código fuente en C que utilice el set SSEx de Intel?

Desde C/C++ se accede a este tipo de instrucciones a través de funciones básicas llamadas intrinsics, que se encuentran en bibliotecas como emminstrin. Para agregarlas al compilador se puede usar una bandera como -msse4.

1.4. ¿Qué importancia tienen la definición de variables y el alineamiento de memoria al trabajar con un set SIMD vectorial, como SSE?

El alineamiento de la memoria es esencial para el buen rendimiento de SIMD. Cuando la memoria no está alineada el procesador debe cargar, separar y reunir la memoria para obtener solo los datos que desea operar y efectivamente alinear los datos. Si este trabajo se hace previamente a las operaciones SIMD con padding o si los vectores son del tamaño correcto, se puede saltar este paso y al cargar los datos se obtienen solo los datos que se desean operar.



2. Análisis

2.1 Explicar las variables oddVector y evenVector

Los vectores se definen como __mm128i, vectores de 128 bits. A estos se les asigna, por medio de _mm_set_epi32, 4 valores enteros firmados de 32 bits. Para mantener los vectores alineados con la memoria en este caso, se debe agregar 4x elementos ya que 4*32=128.

2.2 ¿Qué sucede si las variables data se imprimieran con un ciclo y no uno por uno?

Al imprimir todo el vector a la vez, el resultado es el valor del vector entero, que es un valor de 128 bits, en vez de los sets de 32 bits que esperamos.

2.2.1 ¿Por qué ocurre eso?

La variable es un vector, pero la computadora no sabe imprimir este vector. A diferencia de una lista, para la computadora este es un solo bloque de memoria, por lo que es necesario dividir el vector manualmente en los componentes que se desea. Esto se lleva a cabo por medio de **_mm_extract_epi32** o copiando los espacios de memoria con **memcpy**.

2.3 Compile el código fuente y adjunte una captura de pantalla con el resultado de la ejecución.