Documentación Taller #3

Juan Esteban Navarro Camacho
jnavcamacho@gmail.com
Instituto Tecnológico de Costa Rica
Ingeniería en Computadores
Arquitectura de computadores II

***Resumen*—**

***Index Terms*—**

I. Preguntas

I-A. ¿En qué consiste un extensión SIMD llamada SSE?

Las instrucciones del conjunto de instrucciones de Intel, son instrucciones adicionales que pueden aumentar el desempeño, cuando se realizan las mismas operaciones en varios objetos. (Tradicionalmente representados como un vector)[1].

SSE es una tecnología de procesador que permite ejecutar una instrucción para varios datos (aplicar una operación a un conjunto de datos delimitado). Los procesadores anteriores solo procesan un elemento de datos único por instrucción (SISD una operación para un flujo de datos), es utilizado en aplicaciones intensivas, como gráficos 3D, para un procesamiento más rápido.

I-B. ¿Cuáles tipos de datos son soportados por este tipo de instrucciones?

La siguiente consulta se realiza mediante el sitio web oficial de Intel en donde describe los tipos de datos soportados.

- Paquetes de precisión simple de elementos de punto flotante de 32 bits. Representados por el tipo de dato: **__m128**.
- Paquetes de elementos de enteros de 16 bits. Representados por el dato **__m64**.
- Paquetes de elementos de enteros sin signo de 8 bits, representados **__m64**.

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

Los siguientes archivos de encabezado:

```
#include <emmintrin.h>
#include <smmintrin.h>
```

Figura 1. Archivos de encabezado de instrucciones SSE

GCC contiene por defecto conocimiento para ubicar dichos archivos de encabezado.

Al compilar el archivo **.c** con el siguiente comando:

```
gcc helloWorld.c -o helloWorld.out
```

Figura 2. compilación código de ejemplo

Y procesar luego el archivo ensamblador generado, se puede observar las instrucciones especializadas que muestran el uso de SSE instructions.

```
objdump -d helloWorld.out > helloWorld.asm
```

Figura 3. Generación del archivo ensamblador

```
awk '/[ \t]
(addps|andnps|andps|cmpps|cvtpi2ps|cvtps2pi
|cvttps2pi|divps|maxps|minps|movaps|movhlps
|movhps|movlhps|movlps|movmskps|movntps|mov
ntq|movups|mulps|orps|pavgb|pavgw|pextrw|pi
nsrw|pmaxsw|pmaxub|pminsw|pminub|pmovmskb|p
mulhuw|psadbw|pshufw|rcpps|rsqrtps|shufps|s
qrtps|subps|unpckhps|unpcklps|xorps)[ \t]/'
helloWorld.asm
```

Figura 4. Análisis de instrucciones generadas

Mostrando así en la siguiente figura, el uso de la instrucción para tratar datos empaquetados **movaps** referente a instrucciones SSE.

```
11e5: 0f 29 45 b0  movaps %xmm0,-0×50(%rbp)
122d: 0f 29 45 c0  movaps %xmm0,-0×40(%rbp)
1236: 0f 29 45 e0  movaps %xmm0,-0×20(%rbp)
123f: 0f 29 45 f0  movaps %xmm0,-0×10(%rbp)
1251: 0f 29 45 d0  movaps %xmm0,-0×30(%rbp)
```

Figura 5. Ensamblador filtrado

Referencia de la instrucción **movaps** *Move Aligned Packed Single-Precision Floating-Point Values*.

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

Si bien estas instrucciones soportan operaciones con paquetes de datos, es necesario analizar como la memoria esta organizada, desde el punto de vista de acceso de memoria. Si la memoria esta alineada, esta misma representara menos probabilidad de cache misses debido a la naturaleza de su localidad espacial.

II. Análisis

*II-A. Explicar las variables **oddVector**, **evenVector** y los diversos data en términos de cómo se definieron (instrucción utilizada), el tipo de dato que representan y por qué la cantidad de argumentos.*

Ambas variables **oddVector**, **evenVector** están definidos mediante la función **_mm_set_epi32**, la cual crea un paquete de cuatro valores de enteros en representación de 32 bits cada uno. De ahí la cantidad de valores suministrados como argumentos. El primero elemento en la lista de argumentos de la función **_mm_set_epi32**, representa el 3 elemento del paquete, similar a un arreglo de enteros [2].

II-B. ¿Qué sucede si las variables data se imprimieran con un ciclo y no uno por uno? ¿Por qué ocurre eso?

Se genera un error durante compilación:

```
r3_SIMD$ gcc helloWorld.c -o helloWorld.out
In file included from helloWorld.c:2:
helloWorld.c: In function 'main':
helloWorld.c:21:9: error: selector must be an integer constant in the range 0..3
   21 |   data = _mm_extract_epi32(result, index);
      |          ^~~~~~~~~~~~~~~~~
junavarro@junavarro-Latitude-5480:/media/junavarro/work1/backup/projects/tec/arqu
junavarro@junavarro-Latitude-5480:/media/junavarro/work1/backup/projects/tec/arqu
r3_SIMD$
```

Figura 6. Error de compilación

La documentación indica que solo deben de entrar valores definidos en 0 y 3 pues son los únicos permitidos en la función. Permitir el acceso sea definido por una variable, puede conducir a intentos de acceso inapropiados a paquetes de datos.

II-C. Compile el código fuente y adjunte una captura de pantalla con el resultado de la ejecución.

```
junavarro@junavarro-Latitude-5480:/media/junavarro/work1/backup/projects/tec/arqui2/taller3/
Taller3_SIMD$ gcc helloWorld.c -o helloWorld.out
junavarro@junavarro-Latitude-5480:/media/junavarro/work1/backup/projects/tec/arqui2/taller3/
Taller3_SIMD$ ls
a.out  Arqui2_Taller3.pdf  b.out  file2.asm  file.asm  helloWorld.asm  helloWorld.c  helloWo
rld.out
junavarro@junavarro-Latitude-5480:/media/junavarro/work1/backup/projects/tec/arqui2/taller3/
Taller3_SIMD$ ./helloWorld.out
Probando SSE
Result **********
5       7       9       11
junavarro@junavarro-Latitude-5480:/media/junavarro/work1/backup/projects/tec/arqui2/taller3/
junavarro@junavarro-Latitude-5480:/media/junavarro/work1/backup/projects/tec/arqui2/taller3/
Taller3_SIMD$
```

Figura 7. Compilación y ejecución del código

III. Solución de ejercicios

III-A. Ejercicio 1

Se define una matriz, mediante un arreglo de una dimensión, permitiendo así mediante índices y los valores de la cantidad de filas y columnas, abstraer un acceso matricial.

Esta matriz es alimentada mediante los valores suministrados utilizando **argv** definido en la función main.

En el archivo **Readme** se describe el comando a ejecutar para compilar y luego realizar pruebas.

Las siguientes imágenes describen el proceso tomado.

Se crea una matriz para contener todos los números.

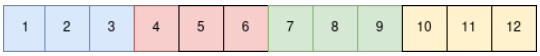

Figura 8. Matriz generada

Se generan los vectores asociados a la matriz de entrada

Figura 9. Vectores generados

Se seleccionan los mayores y se crea el vector de resultado.

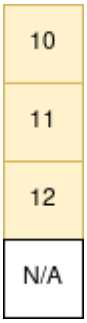

Figura 10. Vector de mayores generados

El archivo **README** describe los comandos necesarias para compilar y ejecutar pruebas.

III-B. Ejercicio 2

Se utilizo la función de suma de vectores para realizar la operaciones de multiplicación de matriz por vector.

La función utilizada para generar un vector con ya el coeficiente aplicado fue: **_mm_set_epi32(val3,val2,val1,val0);**

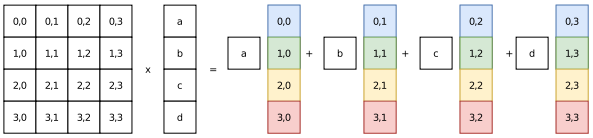

Figura 11. Operación matricial

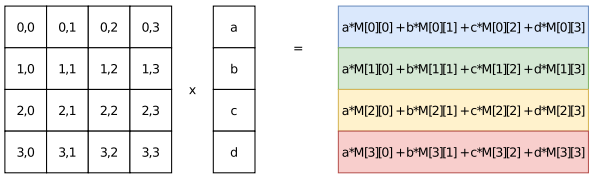

Figura 12. Resultado matricial

El resultado matricial se obtiene realizando tres sumas simultaneas vectoriales representado en la figura III-B.

El archivo **README** describe los comandos necesarias para compilar y ejecutar pruebas.

REFERENCIAS

[1] Intel. *Tecnología de las extensiones del conjunto de instrucciones Intel®*. ID del artículo 000005779. Nov. de 2021. URL: https://www.intel.la/content/www/xl/es/support/articles/000005779/processors.html.

[2] Intel. *The Intel Intrinsics Guide*. 2021. URL: vhttps://software.intel.com/sites/landingpage/IntrinsicsGuide/#text=_mm_set_epi32%5C&techs=SSE,SSE2,SSE3%5C&expand=4906.