Instrucciones SSE y MMX

Kevin Andrés Hernández Rostrán

Escuela de Computación, Instituto Tecnológico de Costa Rica

Cartago, Costa Rica

kevinah95@gmail.com

Resumen

Significado y tipos de procesamientos vectoriales, los tipos SISD tomados en la taxonomía Flynn donde se contienen los tipos de instrucciones MMX y SSE especificando qué son, para que se usan, su implementación en los procesadores Intel y su manejo de memoria y datos (enteros o punto flotante).

Introducción

Se llama **vector** a una secuencia de datos escalares del mismo tipo almacenados en memoria, normalmente en posiciones contiguas, aunque no siempre. Para ilustrar este hecho, su- póngase una matriz bidimensional almacenada en memoria por filas. En esta matriz, podríamos considerar vectores a las filas, columnas o diagonales; en este caso, sólo las filas estarán en posiciones contiguas de memoria.

Un **procesador vectorial** es un conjunto de recursos para efectuar operaciones sobre vectores. Estas operaciones consistirán en funciones aritméticas y lógicas aplicadas sobre las componentes de los vectores.

La diferencia entre un procesador vectorial y uno escalar estriba en que el procesador vectorial puede decodificar instrucciones cuyos operandos son vectores completos. La conversión de un programa correspondiente a un procesador escalar a otro vectorial se llama vectorización.

Las máquinas vectoriales proporcionan operaciones que trabajan sobre vectores. Una instrucción vectorial es equivalente a la ejecución de un bucle completo de instrucciones ordinarias, donde cada iteración trabaja sobre cada una de las componentes del vector. Las operaciones vectoriales tienen algunas ventajas sobre las escalares:

- En las operaciones vectoriales, cada resultado es independiente de los anteriores.
- Una simple instrucción vectorial sustituye a muchas escalares.
- Si se utiliza una instrucción vectorial, evitaremos el riesgo de control del salto del bucle.

Procesamiento vectorial

Los procesadores vectoriales se ocupan de múltiples datos en el contexto de una instrucción, contrastando con las CPUs más comunes de hoy en día que tienen procesadores escalares/superescalares y tratan un dato por cada una. A estos dos esquemas se les conoce respectivamente como **SISD** (Single Instruction, Single Data) que se corresponde con la arquitectura Von Neumann y **SIMD** (Single Instruction, Multiple Data) que es una técnica empleada para conseguir paralelismo a nivel de datos. Esta clasificación recibe el nombre de taxonomía de Flynn.
En particular dispositivos móviles, consolas de videojuegos, tarjetas gráficas, etc hacen un uso intensivo de este tipo de procesamiento.

Michael J. Flynn (profesor de Stanford) creó en 1972 una primera clasificación de los sistemas según su forma de computación (incluyendo tanto la secuencial como paralela):

- SISD (Single Instruction Single Data): planteamiento secuencial tradicional.
- SIMD (Single Instruction Multiple Data): es el más utilizado en el procesamiento paralelo y es imprescindible para comprender las mejoras que suponen los procesadores vectoriales.
- MISD (Multiple Instruction Single Data): sujeto a debate porque no hay software que la explote de forma efectiva, pero sí hay hardware.
- MIMD (Multiple Instruction Multiple Data): multiprocesadores.

Instrucciones SIMD

Consiste en una misma instrucción aplicada a un conjunto de datos (generalmente grande). Para lograrlo se conectan varias unidades de procesamiento a una unidad de control común de forma que todas reciben y ejecutan la misma instrucción (de manera síncrona) pero operan sobre diferentes datos. Es decir, explotan el paralelismo a nivel de datos.

A medida que la multimedia se fue incorporando mayoritariamente a los medios digitales, la importancia de las instrucciones SIMD en las CPUs de propósito general fue creciendo enormemente. Poco después de que comenzara a ser común incluir unidades de coma flotante en procesadores de uso general, también comenzaron a aparecer especificaciones e implementaciones de unidades de ejecución SIMD para dichas CPUs. Algunas de estas primeras especificaciones SIMD, como **MMX** de Intel, fueron solamente para números enteros. Esto resultó ser un impedimento significativo para algunos desarrolladores de software, ya que muchas de las aplicaciones que normalmente se beneficiaban de SIMD trataban sobre todo con números de coma flotante. Progresivamente, estos primeros diseños se fueron refinando hasta llegar a las actuales y modernas especificaciones SIMD como el **SSE** de Intel.

Instrucciones MMX

Es un conjunto de instrucciones SIMD desarrollado a partir de otro conjunto introducido en el Intel i860.

Permite incrementar el rendimiento en aplicaciones multimedia y de comunicaciones aumentando la separación interna de las cachés, lo cual reduce el tiempo de acceso a memoria y proporciona mayor rapidez de acceso a datos y código recientemente utilizado. Las instrucciones y las cachés pueden ser utilizadas a la vez.

Utiliza el algoritmo **BTB** (Branch Target Buffer) para aumentar el rendimiento del conjunto de instrucciones a utilizar. Se añade otro pipeline para que se puedan ejecutar dos instrucciones MMX (o una instrucción entera y otra instrucción MMX) a la vez en el mismo ciclo de reloj, aumentando así la productividad.

Intel también agregó 87 nuevas instrucciones y 8 registros nuevos a la arquitectura, conocidos como **MM0** al **MM7**. Se refieren a registros de la **FPU**

(Floating-Point Unit) x87. Por tanto cualquier cosa que se realice en la pila FPU afecta a los registros MMX. Estos registros son fijos y no relativos como la pila FPU, por lo que se puede acceder a ellos aleatoriamente.

En cada registro entra un número entero de 64 bits aunque se puede aplicar el concepto del tipo de datos compactado por el que se pueden almacenar dos enteros de 32 bits, cuatro enteros de 16 bits u ocho enteros de 8 bits.

Como los registros de la pila tienen 80 bits y los de MMX tienen 64 bits, cuando se está trabajando con MMX se ponen los 16 bits primeros de la pila a "1", haciendo que sea muy fácil saber si se está trabajando con enteros o con coma flotante.
Como sólo trabaja con enteros, mejora el rendimiento multimedia. MMX era utilizado para el cálculo de operaciones en 2D y 3D, pero con la introducción de la GPU (Unidad de Procesamiento Gráfico) esto pierde sentido e Intel ve necesario que realice cálculos en punto flotante. Por ello aparecen las nuevas instrucciones **SSE**.

Instrucciones SSE

Las instrucciones SSE (Streaming SIMD Extensions), introducidas en 1999, son una extensión de las instrucciones MMX para procesadores Pentium III.

Operan con paquetes de operandos en coma flotante simple y son adecuadas para decodificación de MPEG-2 (códec DVD), procesamiento de gráficos tridimensionales y software de reconocimiento de voz.

Hay varios tipos de instrucciones SSE:

- De transferencia de datos.
- De conversión.
- Aritméticas.
- Lógicas.

Con SSE se crean 70 instrucciones nuevas y ocho registros nuevos: xmm0 al xmm7. Los registros son de 128 bits y al contrario que MMX, no es necesario inhabilitar la FPU para volver a habilitarla después (que era lo provocaba pérdida de velocidad).

Instrucciones SSE2

Fueron diseñadas con la misma finalidad que SSE y además para codificación de video, Internet, aplicaciones de ingeniería y científicas, etc.

Permiten trabajar con operandos flotantes de doble precisión, mantienen compatibilidad con SSE y MMX

Instrucciones SSE3

Se agregaron 32 nuevas instrucciones con el fin de mejorar la velocidad de ejecución de las aplicaciones. El cambio que introdujeron fue la capacidad de trabajar horizontalmente y no como en las anteriores que debía ser verticalmente. Esto se refleja en que se puede operar con números que se encuentran en el mismo registro. Por ello se introdujeron instrucciones para sumar y restar múltiples valores almacenados en un registro, que mejoran notablemente el uso del procesamiento digital de señales y gráficos 3D por computadora. También se introdujeron instrucciones para realizar conversiones de punto flotante a entero sin tener que cambiar el modo global de redondeo.

Instrucciones SSE4

En su primera versión (SSE 4.1), se agregaron 47 instrucciones orientadas a mejorar el rendimiento en la manipulación de datos multimedia, juegos, criptografía y otras aplicaciones. La segunda versión (SSE 4.2), incorpora 7 nuevas instrucciones adicionales orientadas a trabajar con procesadores de texto y acelerar algunas operaciones en aplicaciones científicas.

Conclusiones

El procesamiento vectorial significó un cambio drástico por dos factores: la implementación de multimedia o gráficos y la precisión de los datos en las estructuras (punto flotante). Los procesos se optimizaron en su tiempo de ejecución al utilizar vectores en vez de escalares, permitiendo accesos rápidos a memoria y resolución de algoritmos de gran magnitud para el procesamiento de imágenes, gráficos y multimedia digital. Pero no todo fue provechoso en el procesamiento vectorial debido a que no todos los algoritmos eran vectorizables, tienen registros mucho más grandes que requieren mayor consumo de energía y área de chip, los juegos de instrucciones SIMD dependen de la arquitectura por lo que el programador debe proporcionar diferentes implementaciones vectoriales o incluso una no vectorial (por ejemplo en los casos en los que los procesadores no tienen SSE por no estar basados en x86). Pero Intel Corporation y AMD apostaron a lo que hoy conocemos como consolas, videojuegos, procesamiento de voz y decodificación de multimedia.

Referencias

[1] R. Hyde, The Art of Assembly Language Programming, San Francisco: No Starch, 2003.

[2] I. C. 1999, Intel Architecture Software Developer's Manual, 2005.

[3] D. A. Patterson, Computer Organization and Design: The Hardware/Software Interface, 2 Edition ed., Morgan Kaufmann, 1997.

[4] Kunzman y Kale, Programming Heterogeneous Systems, 2011.