Procesadores vectoriales comerciales

Roberto Fernández Rodríguez, Iván Menéndez González y Francesco Buccini

Arquitectura y Tecnología de Computadores (4º curso de Ingeniería en Informática).
EPSIG, Universidad de Oviedo. Campus de Viesques, E-33271. Gijón, Asturias.
{UO173025, UO69574, UO216184}@uniovi.es

Abstract. Esta memoria pretende abordar las capacidades de procesamiento vectorial tanto en general como en los procesadores comerciales actuales (de Intel, AMD, PowerPC, etc) así como otros de este tipo incluidos en los dispositivos avanzados más comunes hoy en día (no solamente los utilizados en computadores sino también en las tarjetas gráficas, videoconsolas, etc).

Palabras clave: paralelismo, procesador vectorial, supercomputador, ILLIAC, Cray, taxonomía Flynn, SIMD, extensiones multimedia, Intel MMX/SSE/AVX, AMD 3DNow!, PowerPC AltiVec, consola, Cell, tarjeta gráfica, PhysX, móvil, ARM Neon.

1 Introducción al procesamiento paralelo

La computación paralela es una forma de computación en la que múltiples operaciones son llevadas a cabo simultáneamente, partiendo el problema en subproblemas y luego resolviendo cada uno de éstos concurrentemente.

Al principio el paralelismo **[1]** sólo era utilizado en las arquitecturas **HPC** (*High Performance Computing*) pero posteriormente el interés por mejorar este tipo de computación fue creciendo al llegar a ciertas limitaciones físicas que impedían el escalado en frecuencia **[2]**. De hecho al ser cada vez más difícil mejorar el rendimiento aumentando la frecuencia **[3]** (debido al consumo de energía y consecuentemente de generación de calor), se empezó a aprovechar la mejor y mayor integración de los transistores para introducir otro tipo de mejoras, llegando así hasta los *cores* de hoy en día. Es decir, para ganar en rendimiento se ha pasado de escalar en frecuencia (aumentando la densidad de transistores según la ley de Moore **[4]**) a escalar en paralelismo (procesamiento multinúcleo).

Existen varias formas de computación paralela: a nivel de bit (**BLP**, *Bit-Level Paralelism*), a nivel de instrucción (**ILP**, *Instruction-Level Paralelism*), a nivel de datos (**DLP**, *Data-Level Paralelism*) y a nivel de tarea (**TLP**, *Task-Level Paralelism*). Este documento se referirá principalmente al paralelismo a nivel de datos.

La programación con este tipo de computación es más difícil que la tradicional ya que se introducen ciertos problemas como las condiciones de carrera (*race conditions*), la comunicación y sincronización entre procesos, exclusión mutua, etc.

La mejora máxima en el sistema que se obtiene al mejorar cierta parte del mismo mediante la introducción del paralelismo (su *speed-up*) puede calcularse con la ley de Amdahl **[5]**.

2 Procesamiento vectorial

Como su nombre implica, 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) **[6]** que es una técnica empleada para conseguir paralelismo a nivel de datos. Esta clasificación recibe el nombre de taxonomía de Flynn.

Casi todas las CPUs de hoy en día incluyen algunas instrucciones de procesamiento de tipo vectorial. En particular y como se verá más adelante, dispositivos móviles, consolas de videojuegos, tarjetas gráficas, etc hacen un uso intensivo de este tipo de procesamiento.

3 Taxonomía de Flynn

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):

	Single Instruction	Multiple Instruction
Single Data	**SISD** (Von Neumann)	**MISD** (sujeto a debate)
Multiple Data	**SIMD** (procesadores *array*)	**MIMD** (multiprocesadores)

Table 1. Taxonomía de Flynn

El planteamiento secuencial tradicional se corresponde con la clasificación **SISD** (previamente descrita por *John Hennessey* y *David Patterson*). El caso de **SIMD** es el más utilizado en el procesamiento paralelo y es imprescindible para comprender las mejoras que suponen los procesadores vectoriales.

En cuanto a **MISD** se puede decir que aunque existe es la forma menos utilizada de las cuatro porque no hay software que la explote de forma efectiva (aunque sí hardware), por lo que sigue estando sujeto a debate. Y por supuesto **MIMD**, la forma más utilizada hoy en día en computación paralela que suele implementarse mediante varios multiprocesadores que ejecutan cada uno de ellos instrucciones SISD (de hecho un supercomputador moderno suele ser un clúster de máquinas MIMD).

4 Instrucciones SIMD

SIMD (*Single Instruction Multiple Data*) 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.

Un ejemplo clásico de paralelismo de datos es la inversión de una imagen RGB para obtener un "negativo" de la misma. Para ello se suele iterar a través de un bucle que recorre los valores correspondientes a dicha imagen (sus píxeles), realizando la misma operación (inversión) una y otra vez. Esto mismo puede hacerse con una sola instrucción SIMD. Otros ejemplos actuales de utilización de este tipo de instrucciones son aplicaciones de investigación científica mediante *GPUs*, compresión de datos, criptografía, procesamiento gráfico en 3D (en consolas de videojuegos), etc.

La primera introducción de este tipo de instrucciones data de 1964 con el supercomputador ILLIAC, creado en la Universidad de Illinois. Esta máquina, dotada de 256 procesadores trabajando de forma paralela, era capaz de procesar grandes conjuntos de datos y sentaría las bases de lo que se conoce hoy en día como **procesamiento vectorial**. Sin embargo fue un fracaso comercial porque sólo se consiguió completar la cuarta parte de lo estipulado en su diseño, tardando 11 años y costando cuatro veces más de lo estimado. Para cuando esta máquina estaba lista en 1976 para enfrentarse a aplicaciones "reales" era ya ampliamente superada por supercomputadores comerciales existentes como el famoso Cray-1 (el primer computador considerado vectorial).

Hubo también ciertos intentos de creación de una máquina SIMD *pura* (es decir, que sólo ejecutase instrucciones SIMD). El problema es que el modelo SIMD no es lo suficientemente flexible como para ser planteado como de propósito general. Por eso estas instrucciones se integraron como parte de una máquina SISD (computación tradicional), ya que ésta se encarga del código que no es normalmente optimizable mediante el paralelismo de datos como por ejemplo instrucciones condicionales. Hoy en día es posible aplicar técnicas como **SWAR** (*SIMD within a register*) **[7]** que permiten utilizar registros de propósito general para ejecutar instrucciones SIMD en hardware sin soporte directo para ellas.

Por otro lado, 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, **3DNow!** de AMD y **AltiVec** (o **VMX**) de PowerPC **[8]**.

5 Procesadores vectoriales

Un **procesador vectorial** (*vector processor* o *array processor*) **[9]** es una CPU que ejecuta instrucciones que operan sobre un array unidimensional de datos (un vector).

Una máquina vectorial consta de una unidad escalar segmentada y una unidad vectorial. La unidad vectorial dispone de M registros vectoriales de N elementos y de unidades funcionales vectoriales (de suma, resta, multiplicación, división, de carga/almacenamiento, etc) que trabajan sobre registros de ese tipo.

El juego de instrucciones de este tipo de procesadores es evidentemente también vectorial. Un ejemplo de instrucción vectorial sería *addv v1,v2,v3*, es decir, una operación vectorial que equivale a un bucle escalar completo que procesaría los N elementos del registro vectorial.

La gran utilidad de crear CPUs que tratan con vectores de datos radica en la optimización de las tareas que tiende a requerir una misma operación al ser realizado con un gran conjunto de datos (por ejemplo, una suma o un producto escalar). Mientras que una CPU escalar debe completar todo el proceso de leer, decodificar y ejecutar cada instrucción y valor con un conjunto de datos, una CPU vectorial puede realizar una simple operación con un conjunto relativamente grande de datos en una sola instrucción. Se disminuye así tanto el ancho de banda de instrucciones requerido como el tiempo de decodificación.

Además, como normalmente los componentes del vector se almacenan de forma contigua, se puede explotar esta forma de procesamiento mediante un mecanismo de acceso lineal especializado que consigue traer de la memoria todos los datos necesarios de una sola vez, poniéndolos en un registro vectorial interno. A partir de ahí todo el procesamiento se realiza entre registros, consiguiendo elevar el rendimiento (haciendo mínimos los accesos a memoria) y disminuir los riesgos de control.

Los procesadores vectoriales son ampliamente utilizados actualmente en computación científica (donde se demandan velocidades de cómputo muy elevadas), en aplicaciones de predicción (meteorológica, de terremotos, de dinámica de fluídos, etc) así como en las tarjetas gráficas que a su vez están incluidas en las consolas de videojuegos más modernas de hoy en día.

Un caso muy similar al de estos procesadores son los denominados *procesadores matriciales*, en los que la unidad mínima de datos tiende a ser una matriz. La diferencia con respecto a los (segmentados) vectoriales es que éstos procesan componentes solapadamente en el cauce y los matriciales lo hacen simultáneamente. Otra diferencia que tienen estos últimos respecto de los vectoriales es su enorme coste.

Los **problemas del procesamiento vectorial** en general son:

- No todos los algoritmos son *vectorizables* (por ejemplo bucles con muchas instrucciones de control) **[10]**.
- Los procesadores vectoriales tienen registros mucho más grandes que requieren mayor consumo de energía y área de chip.
- La generación de código SIMD para un procesador de este tipo no se realiza automáticamente, debe hacerlo el programador manualmente teniendo en cuenta su problemática **[11]**. Por ejemplo, las instrucciones SSE tienen fuertes restricciones

en cuanto al alineamiento de datos por lo que cargar datos en/de registros SIMD puede ser muy complicado e ineficiente.
- Los juegos de instrucciones SIMD dependen de la arquitectura por lo que el programador debe proporcionar diferentes implementaciones vectoriales o incluso una no vectorial. Esto se da por ejemplo en los casos en los que los procesadores no tienen SSE por no estar basados en x86.

6 Extensiones de los procesadores vectoriales comerciales

Aunque el primer procesador vectorial data de 1976, hasta hace poco más de una década no se veían estos procesadores en los computadores personales. Dos grandes empresas fabrican la mayoría de procesadores de este tipo incluyéndolos en los ordenadores de hoy en día: *Intel Corporation* y *AMD*. Ambas crean todos sus procesadores vectoriales mediante distintas tecnologías, las cuales se estudian a continuación.

6.1 Intel MMX/SSE

Intel Corporation (*Integrated Electronics Corporation*) fue fundada en 1968. En 1971 desarrollaron su primer microprocesador, pero no fue hasta 1997 cuando apareció su primer procesador vectorial gracias a las instrucciones MMX. Más adelante crearían las instrucciones SSE **[12]**.

MMX

Es un conjunto de instrucciones SIMD desarrollado a partir de otro conjunto introducido en el *Intel i860*. Se introdujo en la gama *Pentium* con velocidades de 166, 200 y 233 MHz. 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.

Para simplificar el diseño MMX utiliza los ocho registros de la FPU, por lo que es muy difícil trabajar con ambos a la vez. Debido a esto, para maximizar el rendimiento, los programadores deben trabajar en un sólo modo, retrasando lo máximo posible el lento paso de un modo a otro (*context-switching* o cambio de

contexto). Además esta simplificación hacía que MMX fuera compatible con los sistemas operativos que había en aquella época.

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.

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).

SSE2

Introducidas en 2001, fueron utilizadas por la primera versión de Pentium IV. Fueron diseñadas con la misma finalidad que SSE y además para codificación de video, Internet, aplicaciones de ingeniería y científicas, etc.

Mantienen compatibilidad con SSE y MMX, pero permiten trabajar con operandos flotantes de doble precisión y al igual que MMX utilizan el concepto del tipo de datos compactados por el que se pueden almacenar dos enteros de 32 bits, cuatro enteros de 16 bits u ocho enteros de 8 bits.

Hay una instrucción en precisión doble correspondiente por cada instrucción SSE, excepto para las recíprocas. Como existen los escalares y los paquetes en precisión doble, añaden instrucciones de conversión para la conexión de ambos tipos.

SSE3

Introducidas en 2004 con la revisión de los *Pentium IV*. 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.

SSE4

Introducidas en 2007 con los procesadores de núcleo *Penryn* de la rama *Core 2*. 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.

AVX

Llamadas ***A**dvanced **V**ector EX**t**ensions*, son propuestas por Intel en 2008 e introducidas en el año 2011 con las familias de procesadores *Sandy Bridge* de Intel y *Bulldozer* de AMD.

En éstas los registros se incrementan de 128 bits a 256 bits, denominados *XMM0-XMM15* e *YMM0-YMM15*. Para que estas instrucciones sean compatibles con las instrucciones SSE, se opera con los 128 bits de los registros YMM.

Las instrucciones utilizan tres operandos en vez de dos para no sobrescribir los resultados. Por ejemplo, en SSE para realizar una suma se sumaban los registros "a" y "b" y se dejaba el resultado en uno de ellos; ahora se suman los dos registros y se deja el resultado en otro registro "c".

El incremento en la capacidad de los registros y el hecho de utilizar un registro extra al realizar las operaciones se debe a la utilización de un nuevo esquema de codificación llamado "*VEX prefix*".

Estas instrucciones incrementan el paralelismo y el rendimiento en operaciones en punto flotante. Debido a esto, estas instrucciones son apropiadas para el cálculo intensivo de operaciones en punto flotante para multimedia y aplicaciones científicas y financieras.

6.2 AMD 3DNow!

AMD (*Advanced Micro Devices*, Inc.), fue fundada en 1969. Al igual que *Intel*, aunque llevaban tiempo fabricando procesadores, no fue hasta 1998 cuando apareció el primer procesador vectorial con la introducción de las instrucciones *3DNow!* **[13]**.

3DNow!

Introducidas en 1998 con los procesadores K6-2. Fueron desarrolladas como una mejora del conjunto de instrucciones MMX de Intel, para poder operar con datos en coma flotante además de datos enteros.

Son un conjunto de 21 instrucciones (17 instrucciones de punto flotante, 2 instrucciones de enteros y 2 instrucciones para aumentar el rendimiento). Las dos instrucciones para aumentar el rendimiento son:

- **FEMMS**: entrada/salida rápida de MMX o estado punto-flotante.
- **PREFETCH/PREFETCHW**: prerecuperación de datos de 32 bytes (como mínimo) en la caché L1.

3DNow! Extensions

Introducidas en 1999 con la primera generación de procesadores Athlon. Se añaden 5 nuevas instrucciones 3DNow! y 19 instrucciones MMX.

Las nuevas instrucciones 3DNow! son utilizadas para acelerar el DSP (Procesamiento Digital de Señales) y las MMX para acelerar el streaming de contenidos electrónicos. Se considera en este período a las instrucciones MMX como un subconjunto de las instrucciones enteras de SSE.

3DNow! Professional

Introducidas en 2001 con los procesadores Athlon XP **[14]**. Estos procesadores mezclan las instrucciones 3DNow! y las SSE. Es el primer procesador de AMD que soporta todo el conjunto de instrucciones SSE1.

En total utiliza las 21 instrucciones originales de 3DNow!, las instrucciones 3DNow! extensions y 52 instrucciones SSE para compatibilizar todo el conjunto de instrucciones SSE1.

Para los procesadores Geode GX/LX se añadieron dos nuevas instrucciones que no se implementarían en los demás procesadores:

- **PFRSQRTV**: *Reciprocal Square Root Approximation* (aproximación recíproca de la raíz cuadrada para un par de valores flotantes de 32-bits).
- **PFRCPV**: *Reciprocal Approximation* (aproximación recíproca para un par de valores flotantes de 32-bit).

6.3 Extensiones multimedia en la actualidad

En Agosto de 2010 **[15]**, AMD deja de utilizar sus extensiones 3DNow!. Sólo mantiene dos instrucciones, PREFETCH y PREFETCHW, explicadas anteriormente.

Hoy en día, tanto Intel como AMD utilizan instrucciones SSE y AMD ofrece "Manuales de Programador" para portar todas las aplicaciones que usan 3DNow! a SSE. Y los procesadores sacados al mercado en 2011 son compatibles con las instrucciones AVX de Intel.

6.4 Extensiones multimedia en el futuro

Gracias al esquema de codificación "*VEX prefix*", en un futuro se introducirán nuevos conjuntos de instrucciones para manejar las instrucciones SIMD (algunos ya han aparecido con los procesadores AMD Bulldozer):

- **FMA**: es una extensión de instrucciones SIMD de 128 bits que permite trabajar con 4 registros independientes, 3 para operandos y el último para guardar el resultado.
- **CVT-16**: es una extensión de instrucciones SSE de 128 bits.
- **XOP**: es también una extensión de instrucciones SSE de 128 bits. Algunas de sus características son la permutación de bits en un vector y la comparación de vectores enteros.

7 Aplicaciones actuales de los procesadores vectoriales

7.1 Tecnología AltiVec de PowerPC

Como se ha explicado previamente, las instrucciones vectoriales se utilizan para realizar la misma operación sobre un conjunto de datos a la vez en vez de hacerlo de una en una (forma habitual del procesamiento paralelo).

El empleo de la tecnología de AltiVec **[16]** supone una aproximación para acelerar el procesamiento de largas secuencias de datos. Usando sus instrucciones se puede alcanzar un aumento significativo de la velocidad en las comunicaciones, en programas multimedia y en cualquier desarrollo de aplicación donde sea posible utilizar los diferentes niveles de paralelismo de datos, reduciendo al mínimo el ancho de banda necesario y los embotellamientos provocados por el acceso a memoria. Por lo tanto es una poderosa herramienta para los desarrolladores de software que quieran añadir eficacia y velocidad a sus aplicaciones.

La arquitectura PowerPC (desarrollada en conjunto por Motorola, IBM y Apple) se basa en la arquitectura POWER implementada para la familia de computadores RS/6000. Dicha arquitectura saca partido de los recientes avances tecnológicos en áreas tales como tecnología de procesos y diseño de compiladores y reduce el conjunto de instrucciones. Es asimismo flexible y escalable.

El repertorio de instrucciones *AltiVec* se ejecuta sobre dicho procesador PowerPC. *AltiVec* es un conjunto de instrucciones SIMD que opera con enteros y en coma flotante diseñado por y en propiedad de Apple Computer (*Velocity Engine*), IBM (*VMX*) y Motorola (*AltiVec*) y puesto en ejecución en las versiones de PowerPC (como el G4 de Motorola y los procesadores G5 de IBM). El repertorio de instrucciones AltiVec soporta tanto aplicaciones de audio como de vídeo.

La tecnología AltiVec se basa en la ejecución de instrucciones con múltiples operandos explotando todo lo posible el paralelismo de datos soportado por los microprocesadores. Amplía la arquitectura del sistema de instrucciones (ISA) de los microprocesadores PowerPC aumentando sus posibilidades hasta el límite, siendo capaz de ejecutar procesos de propósito general concurrentemente con una intensa computación algorítmica, soportando un amplio ancho de banda para el direccionamiento de procesos de datos en un único chip.

La idea principal detrás de AltiVec es el tratamiento en paralelo de un conjunto de datos agrupados en un vector. El orden de este agrupamiento viene determinado por la subdivisión que se hace de dicho vector, siendo 2 (quadwords), 4 (words), 8 (half-words) y 16 (bytes) las posibles subdivisiones. Es decir, sus operaciones pueden desarrollar múltiples conjuntos de datos en una sola instrucción.

La tecnología AltiVec se utiliza en muchas aplicaciones:

- Comunicaciones (módems multicanal, cifrado RSA, software para módem V.34)
- Realidad virtual
- Voz sobre tecnología IP (*VoIP*): transmite voz como paquetes de datos digitales comprimidos a través de Internet
- Concentradores de Acceso/DSLAMS

- Procesamiento de voz/sonido (G.711, G.721, G.723, G, 729A, y AC-3) para el uso en aplicaciones como asistencia telefónica y marcado automático. Se usa para mejorar la calidad de sonido sobre las líneas mediante las unidades en punto flotante (por ejemplo cancelación del eco en llamadas largas).
- Procesamiento de gráficos 2D y 3D (QuickDraw, OpenGL, VRML, juegos, etc) y en imágenes en general (JPEG, filtros, etc)
- Videoconferencia (H.261 y H.263)
- Alto ancho de banda para la transmisión de vídeo (MPEG-1, MPEG-2, MPEG-4, H.234)

La tecnología de AltiVec amplía la arquitectura de PowerPC a través de la adición de una unidad de ejecución de vectores de 128 bits, que funciona simultáneamente con las unidades enteras y de punto flotante existentes. Esta nueva unidad de ejecución de vectores proporciona operaciones altamente paralelas, permitiendo la ejecución simultánea de operaciones múltiples en un solo ciclo de reloj.

Se puede pensar en la tecnología AltiVec como un sistema de registros y de unidades de ejecución agregadas a la arquitectura de PowerPC de una forma análoga a la adición de unidades de punto flotante en cualquier procesador. Estas últimas fueron agregadas para proporcionar una ayuda a los cálculos científicos de alta precisión y ahora la tecnología AltiVec se agrega a la arquitectura PowerPC para acelerar estos aspectos.

7.2 Procesador Cell

Cell [17] es una arquitectura de microprocesadores desarrollada conjuntamente por Sony, Toshiba e IBM en una alianza conocida con el nombre de "*STI*". El diseño de su arquitectura y su primera implementación fueron llevados a cabo en el STI Design Center en Austin (Texas) durante un período total de cuatro años que comenzó en marzo de 2001, empleando para ello un presupuesto de 400 millones de dólares según IBM.

Cell es la abreviatura de *Cell Broadband Engine Architecture* ("arquitectura de motor Cell de banda ancha"), conocida también como *CBEA* por sus siglas al completo o *Cell BE*. Cell emplea una combinación de la arquitectura de núcleo PowerPC de propósito general y medianas prestaciones con elementos coprocesadores en cascada, los cuales aceleran notablemente aplicaciones de procesado de vectores y multimedia así como otras formas de computación dedicada.

La primera gran aplicación comercial de Cell fue la consola PlayStation 3 de Sony. También se puede encontrar este procesador en servidores duales Cell Blade, tarjetas aceleradoras PCI Express y adaptadores de TV de alta definición.

Este microprocesador de banda ancha se diseñó para cubrir el hueco existente entre procesadores convencionales de sistemas de escritorio (como Athlon, Pentium, PowerPC, etc) y los procesadores especializados de alto rendimiento (como las GPUs de ATI y nVidia).

Su propio nombre indica las especificaciones de su uso, principalmente como componente en sistemas de distribución digitales presentes y futuros. Como tal puede ser empleado en pantallas y equipos de grabación de alta definición así como en sistemas de entretenimiento para la era HDTV. De manera adicional, puede ser

apropiado para sistemas digitales de obtención de imágenes (médicas, científicas, etc) y para simulaciones físicas (por ejemplo para el modelado de ingeniería estructural).

En un análisis sencillo, la configuración más básica de Cell es un chip multinúcleo que se puede descomponer en cuatro partes:

1. Estructuras externas de **I/O**
2. **PPE** (Power Processing Element), el procesador principal consistente en un núcleo Power ISA de dos vías multihilo simultáneo
3. Ocho coprocesadores funcionales denominados **SPE** (*Synergistic Processing Elements*)
4. Un bus de datos circular especializado de gran ancho de banda que conecta la PPE, los elementos I/O y las SPEs denominado **EIB** (*Element Interconnect Bus* o bus de interconexión de elementos).

Para alcanzar el alto rendimiento necesario para tareas matemáticas intensivas (codificar/decodificar secuencias MPEG, generar/transformar datos 3D, realizar un análisis de Fourier con los datos, etc) el procesador Cell aúna las SPE y el PPE por medio del EIB para proporcionarles acceso tanto a la memoria principal como a dispositivos externos de almacenamiento.

El PPE, que es capaz de ejecutar un sistema operativo convencional, posee el control sobre las SPEs y puede comenzar, interrumpir y programar procesos para que se ejecuten en las mismas. Tiene para este fin instrucciones adicionales relativas al control de las SPEs a las que ha de enviar órdenes antes de poder ejecutar ninguna tarea de utilidad.

El PPE y la arquitectura de bus incluyen varios modos de operación que proporcionan diferentes niveles de protección de memoria permitiendo que ciertas áreas de la memoria queden protegidas frente a procesos específicos que se estén ejecutando en las SPEs o en la PPE.

Tanto la arquitectura de la PPE como las de las SPEs son de tipo RISC con instrucciones de un tamaño de palabra fijo de 32 bits. El PPE contiene un juego de registros de propósito general (GPR) de 64 bits, un registro de coma flotante (FPR) de 64 bits y un juego de registros de tipo Altivec de 128 bits. La SPE contiene solamente registros de 128 bits que pueden ser empleados para diversos tipos de datos escalares (que pueden ir desde los 8 hasta los 128 bits de tamaño) o para cálculos SIMD en una variedad de formatos enteros o de coma flotante.

Debido a la naturaleza de sus aplicaciones, el Cell está optimizado para la computación de datos de coma flotante de precisión simple. Las SPEs son capaces de ejecutar cálculos de doble precisión, pero a cambio de una penalización notable en el rendimiento.

7.3 Tarjetas gráficas

Aunque actualmente todos los ordenadores tienen algún tipo de GPU (que permite liberar a la CPU de los cálculos necesarios para generar gráficos 3D) se ha propuesto añadir más hardware de propósito específico, especialmente para juegos. La propuesta más importante ha sido la de un coprocesador físico conocido como **PPU** (*Physics Processing Unit*).

El PPU tiene como principal objetivo liberar al procesador de la carga que tiene al modelar características físicas de objetos, tal como se hizo en los años 90 con la implementacion del GPU, que liberaba al procesador de las tareas gráficas. Un ejemplo de lo que es capaz de hacer dicho procesador consiste en generar modelar el humo y las particulas de las explosiones en juegos de forma más realista (y no como animaciones).

La primera y única PPU construida por ahora es la **PhysX [18]** desarrollada por la empresa *Ageia* (recientemente comprada por nVidia). Hay tres chipsets que utilizan en la actualidad PhysX: Asus PhysX P1, BFG PhysX 128MB PCI y ELSA PHYNITE X100.

PhysX consiste en un núcleo RISC de propósito general que controla un array de procesadores VLIW (*Very Large Instruction Word*) SIMD de punto flotante. También se caracteriza por no tener una jerarquía de cachés como en una CPU o GPU tradicionales.

La empresa nVidia ha anunciado recientemente su intención de convertir la especificación de PhysX en un estándar abierto de forma que esta arquitectura pueda ser integrada en el futuro en cualquier GPU.

7.4 Móviles

La arquitectura **ARM** (*Advanced RISC Machine*, antiguamente denominada *Acorn RISC Machine*) es una arquitectura de tipo RISC de 32 bits desarrollada por *ARM Holdings* que se utiliza en multitud de sistemas *embedded*. Gracias a sus características de bajo consumo de energía (en comparación con el rendimiento), la arquitectura ARM es muy famosa en el campo de los dispositivos móviles (en el cual el ahorro de energía es fundamental) y cubre el 75% del mercado mundial de los procesadores de 32 bits para aplicaciones *embedded*.

Los procesadores *ARM* son utilizados en PDAs, teléfonos móviles, tabletas, reproductores multimedia, videojuegos, etc.

Referencias

[1] http://en.wikipedia.org/wiki/Parallel_computing
[2] http://en.wikipedia.org/wiki/Frequency_scaling
[3] http://www.nytimes.com/2004/05/08/business/intel-halts-development-of-2-new-microprocessors.html?pagewanted=all&src=pm
[4] ftp://download.intel.com/museum/Moores_Law/Articles-Press_Releases/Gordon_Moore_1965_Article.pdf
[5] http://es.wikipedia.org/wiki/Ley_de_Amdahl
[6] http://en.wikipedia.org/wiki/SIMD
[7] http://www.aggregate.org/SWAR/
[8] http://en.wikipedia.org/wiki/Central_processing_unit#Data_parallelism
[9] http://en.wikipedia.org/wiki/Vector_processor
[10] http://en.wikipedia.org/wiki/Vectorization_(parallel_computing)
[11] https://computing.llnl.gov/tutorials/parallel_comp/
[12] http://es.wikipedia.org/wiki/MMX
[13] http://es.wikipedia.org/wiki/SSE
[14] http://en.wikipedia.org/wiki/3DNow

[15]http://www.tommesani.com/3DNow.html
[16]http://blogs.amd.com/developer/2010/08/18/3dnow-deprecated/
[17]http://en.wikipedia.org/wiki/AltiVec
[18]http://es.wikipedia.org/wiki/Cell
[19]http://en.wikipedia.org/wiki/Physics_processing_unit

Bibliografía

-*Parallel Computer Architecture: A Hardware/Software approach*. David Culler, Jaswinder Pal Singh, y Anoop Gupta. Morgan Kaufmann, 1999.

-*Advanced computer architecture: Parallelism, scalability, programmability*. Kai Hwang. McGraw-Hill, 1993.

-*High-Performance Computer Architecture*. Harold S. Stone. Addison-Wesley, primera y tercera edición, 1987 y 1993.

-*Arquitectura de Computadoras y Procesamiento Paralelo*. Kai Hwang y Fayé A. Briggs. McGraw-Hill, 1987.

-*Computer Architecture, single and parallel systems*. Mehdi R. Zargham. Prentice-Hall, 1996.

-*Computer architecture: pipelined and parallel processor design*. Michael J. Flynn. Jones and Bartlett, 1995.

-*Organización y Arquitectura de Computadores, diseño para optimizar prestaciones*. William Stallings. Prentice Hall, cuarta edición, 1996.

-*See MIPS Run*. D. Sweetman. Morgan Kaufmann Publications, 2002.