Conjunto de instruções multimídia

Henrique Dante de Almeida
IC, Unicamp

Junho, 2008

hdante@gmail.com

RESUMO

Este artigo apresenta uma visão geral dos conjuntos de instruções multimídia. Serão descritos: os conceitos envolvidos, os conjuntos de instruções SSE3 e Altivec, exemplos de arquiteturas e resultados de performance.

1.CONCEITOS

Os conjuntos de instruções multimídia têm como objetivo fornecer suporte de hardware para aceleração de certas operações em quantidades grandes de dados, tipicamente utilizadas em aplicações de áudio, vídeo, computação científica e outros[1, 12]. A idéia principal envolvida nos conjuntos de instruções multimídia é explorar o paralelismo de dados[3, 8], isto é, a propriedade de certos programas requererem operações repetitivas em grandes quantidades de dados que possuem pouca dependência entre si (em contraste com paralelismo em nível de instruções, cuja ênfase está em operações independentes não necessariamente repetitivas que possuem pouca dependência entre si). Na classificação de Flynn[3], o paralelismo de dados é explorado pelos modelos que operam em múltiplos dados, em especial ao modelo de única instrução operando em múltiplos dados (SIMD). Este modelo pode ser descrito da seguinte forma: um conjunto de dados que exibe paralelismo é dividido em pedaços independentes e cada parte é processada paralelamente por uma unidade funcional distinta. Todas as unidades funcionais executam a mesma instrução.

Como exemplo, um problema que exibe paralelismo de dados é a soma de dois vetores:

$$\vec{A} = \vec{B} + \vec{C}$$

Cada coordenada do vetor pode ser calculada com uma soma escalar independente:

$$a_i = b_i + c_i$$

Assim, se o vetor possui tamanho n e a máquina SIMD pode possui n unidades funcionais, ela pode calcular a soma de um vetor calculando paralelamente a soma de cada um de seus elementos.

A denominação "multimídia" dos conjuntos de instruções SIMD em arquiteturas escalares de computadores de mesa e embarcados vem de motivo histórico: durante a década de 90, houve o interesse de melhorar a performance em codificações de vídeo e computação gráfica[12]. Na época, diversos conjuntos de instruções foram criados (MDMX para o MIPS, MAX-2 para PA-RISC, VIS para o SPARC, etc.), inicialmente bastante limitados ao processamento gráfico, de forma a não causar impacto nas arquiteturas[1, 12]. Posteriormente conjuntos de instruções se tornaram mais gerais (como o Altivec para o PowerPC e o SSE4 para o x86), permitindo a utilização em domínios diversos que exibem paralelismo de dados, como processamento de sinais[1, 13], aparelhos para telecomunicações[13], processamento de texto[14], e outros.

2.ARQUITETURA

A inclusão de instruções SIMD nas arquiteturas escalares é feita modificando o estágio de execução e acrescentando (ou adaptando) registradores. Por exemplo, no Pentium III e PowerPC 970 (super escalares), uma nova unidade funcional é responsável pela execução das instruções SIMD.

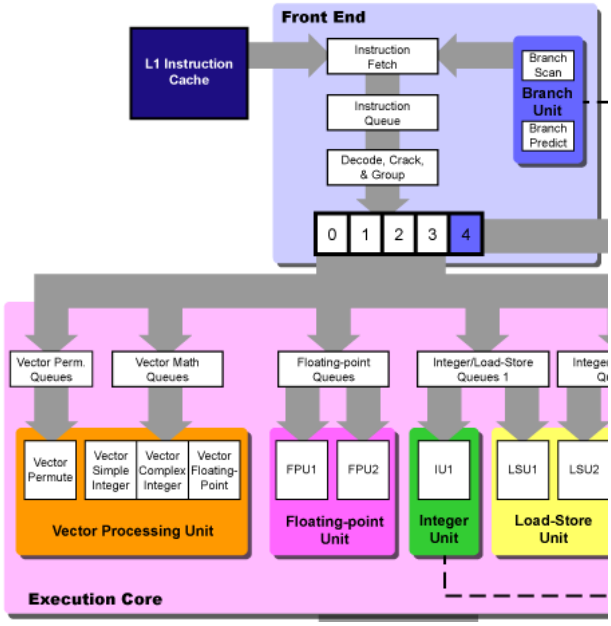

Ilustração 1: Trecho de esquemático da arquitetura do PowerPC 970, mostrando a unidade Altivec

Na ilustração 1 pode-se notar que a unidade SIMD do PowerPC 970 funciona como qualquer outra unidade funcional. Esta característica de preservar a arquitetura escalar do processador original ao acrescentar as instruções multimídia também aparece nos outros processadores[1, 12].

Os conjuntos de instruções úteis a serem implementados paralelamente são de diversas categorias. Por exemplo, a biblioteca para Altivec fornecida pela Freescale possui núcleos úteis para telecomunicações, redes, multimídia e outros (Tabela 1). Sob o ponto de vista de arquitetura, os conjuntos podem ser classificados da seguinte forma:

- Operações lógicas e aritméticas em inteiros – inclui operações em inteiros modulares e de saturação, deslocamentos e rotações
- Operações em números de ponto flutuante – inclui operações de arredondamento e conversão

- Operações de memória, cache e auxiliares de acesso – incluem operações de pre-carga de cache, permutação de elementos, intercalação, seleção, etc.
- Operações de controle – inclui operações de comparação, desvio e execução condicional
- Operações complexas e tipos de dados úteis em certos domínios – podem pertencer a qualquer uma das categorias acima

Aplicação/Categoria	Bibliotecas
Telecomunicações	> Fixed-Point and Floating Point Fourier Transform (FFT) Megafunction > Complex, Real, and Real Delayed Least Mean Squared (LMS) Finite Impulse Response (FIR) Functions > Autocorrelation > Global System for Mobile Communications (GSM) Convolution Encoder > GSM/3G Viterbi Decoder > Error Correction Codes (Cyclic Redundancy Check 8,12,16,24)
Redes	> Open Shortest Path First (OSPF) > Transmission Control Protocol/Internet Protocol (TCP/IP) > Encryption Protocols (AES, DES, 3DES, MD5)
Multimídia	> 2D Discrete Cosine Transform (DCT) > 2D Inverse Discrete Cosine Transform (IDCT) > MPEG-2 (Moving Picture Experts Group) > MPEG-1 Audio Layer-3 (MP3) > JPEG (Joint Photographic Experts Group) > Quantization > Dequantization > Sum of Absolute Differences (SAD)
Impressão	> Ghostscript® Library Elements > Color Conversion (RGB to YCbCr) > FS Dithering
Bibliotecas de sistema (libc)	> Link-level Support for Standard C Functions (memcpy, strcmp etc.)
Primitivas matemáticas	> Log, Exp, Sin, Cos, Log, Sqrt
Suporte ao sistema operacional	> Linux (TCP/IP) > VxWorks® Elements

Tabela 1: Biblioteca de núcleos para uso com Altivec

Para facilitar a discussão, serão apresentados dois exemplos práticos de implementação SIMD: o Altivec e o SSE3.

2.1 Altivec

O Altivec foi lançado no fim da década de 90 pela IBM, Motorola e Apple[15]. Foi estreado no processador da Motorola PowerPC 7400 (PowerPC G4)[15]. A inclusão nos computadores da Apple tinham, como objetivo, acelerar decodificação de vídeo[15].

A arquitetura das instruções Altivec[5] contém 32 registradores de 128 bits. Estes 128 bits podem representar diversos tipos de vetores: inteiros com ou sem sinal, de 8, 16 e 32 bits, números de ponto flutuante de 32 bits, pixels RGB de 16 bits ou, em certos casos especiais, um um inteiro de 128 bits. Instruções de acesso à memória podem operar em 8, 16, 32 ou 128 bits. Não há suporte para números de 64 bits.

O conjunto de instruções é extenso, possuindo mais de 150 instruções. As instruções podem usar até 3 registradores de origem e 1 de destino. As operações podem operar tanto entre dois vetores, como dentro de um vetor. O conjunto de instruções não implementa instruções excessivamente complexas sob o ponto de vista de hardware. Por exemplo, não há instruções para divisão e acesso desalinhado à memória. Exemplos de operações estão na Tabela 2.

As implementações atuais de Altivec utilizam ao menos uma unidade funcional especializada em permutações e outra para o restante das operações[5]. Ambas as unidades suportam o início de uma operação SIMD por ciclo.

2.2 SSE3

O SSE3 foi lançado no Intel Pentium 4 "Prescott", em 2004. O conjunto de instruções vem de uma seqüência de diversas outras iterações evolutivas de instruções SIMD nos processadores x86. As iterações foram gradativamente aumentando o domínio de problemas suportados pela unidade SIMD[4]. Considerando o agregado SSE, SSE2 e SSE3, o número de instruções passa de 220.

A arquitetura contém dois conjuntos de registradores, chamados XMM e MMX. O conjunto XMM contém 8 registradores de 128 bits, quando operado em modo de 32 bits ou 16 registradores de 128 bits, quando operado em modo de 64 bits. O conjunto MMX contém 8 registradores de 64 bits. Normalmente, há duas instruções com o mesmo mnemônico, uma operando em registradores XMM e outra no MMX. Os tipos de dados são inteiros de 8, 16, 32, 64 e 128 bits e números de ponto flutuante com 32 e 64 bits. As instruções SSE3 operam em dois registradores de origem ou um registrador de origem e um endereço de memória. O registrador de destino é implícito, igual a um dos registradores de origem. O SSE3 possui instruções bastante complexas, como divisão, diversas operações de cache e sinalização de exceções. Exemplos de instruções do agregado SSE1/2/3 (levando em consideração a operação com registradores XMM) estão na Tabela 3.

Nas implementações atuais, o SSE3 utiliza as mesmas unidades funcionais das operações escalares de ponto flutuante[7, 9, 10, 11]. São duas ou mais unidades, especializadas em operações simples (unidade "aditiva") e complexas (unidade "multiplicativa"). É possível emitir simultaneamente ao menos uma instrução simples e outra complexa. No entanto, certas operações complexas podem ter vários ciclos de latência.

Instrução	Operação	Categoria
vaddfp	Soma de vetor de números de ponto flutuante	Ponto flutuante, aritmético
vaddshs	Soma com saturação de inteiros de 16 bits com sinal	Inteiro, aritmético
vsububm	Subtração modular de inteiros de 8 bits sem sinal	Inteiro, aritmético
vmaxsw	Valor máximo de inteiros com sinal de 32 bits	Inteiro, aritmético
vmaddfp	Multiplica números de ponto flutuante e soma os resultados com um terceiro vetor	Ponto flutuante, aritmético
vand	Operador E-lógico, bit a bit, de 128 bits	Lógico
vcmpgefp	Comparador "maior ou igual" de números de ponto flutuante	Ponto flutuante, comparação
vslh	Desloca à esquerda inteiros de 16 bits, utilizando um segundo vetor com a contagem de deslocamento para cada elemento	Inteiro, deslocamento
vrfip	Função "teto" em números de ponto flutuante	Ponto flutuante, arredondamento
vmsumubm	Multiplicação de inteiros de 8 bits, seguido do agrupamento do resultado em 4 grupos de 4 elementos, seguido da somatória de cada um dos 4 grupos, seguido da conversão dos 4 resultados para 32 bits, seguido da soma modular com um terceiro vetor de inteiros de 32 bits	Inteiro, aritmético
vcfux	Converte inteiros de 32 bits para números de ponto flutuante, possivelmente deslocando a posição da vírgula com um imediato	Conversão
vrsqrtefp	Calcula valor aproximado do inverso da raiz quadrada de números de ponto flutuante	Ponto flutuante, aritmético
vperm	Gera uma permutação de um subconjunto de elementos de 8 bits de dois vetores, utilizando um terceiro vetor como controle	Permutação
lvxl	Carrega um vetor da memória, no endereço alinhado em 128-bits dado pela soma de dois registradores escalares e marca no sistema de cache que este endereço não será reutilizado em breve	Memória
vpkuwus	Converte inteiros sem sinal de 32 bits em inteiros de 16 bits, possivelmente saturando	Conversão
dst	Indica que uma certa região de memória vai ou não vai ser lida em breve	Memória
lvsl	Prepara um vetor de permutação para ajudar na carga de um vetor que se encontra desalinhado pela direita na memória, no endereço desalinhado dado pela soma de dois registradores escalares	Permutação

Tabela 2: Exemplos de diversos tipos de instruções Altivec, com descrições informais

Instrução	Operação	Categoria
addps	Soma de vetor de números de ponto flutuante de 32 bits	Ponto flutuante, aritmético
paddsw	Soma com saturação de inteiros de 16 bits com sinal	Inteiro, aritmético
addsubpd	Subtrai os primeiros elementos de ponto flutuante de 64 bits de dois vetores e soma os segundos elementos	Ponto flutuante, aritmético
pmaxsd	Valor máximo de inteiros com sinal de 32 bits	Inteiro, aritmético
divps	Divide números de ponto flutuante de 32 bits	Ponto flutuante, aritmético
pand	Operador E-lógico, bit a bit, de 128 bits	Lógico
cmpps	Comparador genérico de números de ponto flutuante de 32 bits	Ponto flutuante, comparação
movshdup	Copia o segundo elemento de 32-bits de um vetor de origem para o primeiro e o segundo elemento do vetor de destino e copia o quarto elemento de 32-bits do vetor para o terceiro e o quarto elemento	Cópia
sqrtps	Calcula a raiz quadrada de números de ponto flutuante de 32 bits	Ponto flutuante, aritmético
pmaddwd	Multiplicação de inteiros de 16 bits, seguido do agrupamento do resultado em 4 grupos de 2 elementos de 32 bits, seguido da somatória de cada um dos 4 grupos	Inteiro, aritmético
cvtpi2ps	Converte inteiros de 32 bits para números de ponto flutuante de 32 bits	Conversão
rsqrtss	Calcula valor aproximado do inverso da raiz quadrada de números de ponto flutuante de 32 bits	Ponto flutuante, aritmético
pshufd	Gera um arranjo de inteiros de 32 bits	Permutação
lddqu	Carrega um vetor desalinhado na memória, acessando, para isto, dois blocos alinhados de 128 bits e, em seguida, extraindo o vetor	Memória
packssdw	Converte inteiros sem sinal de 32 bits em inteiros de 16 bits, possivelmente saturando	Conversão
prefetch0	Indica que uma certa região de memória deve estar na cache	Memória
movaps	Carrega ou armazena um vetor alinhado em 128 bits, da memória, ou copia dois vetores	Memória, cópia

Tabela 3: Exemplos de instruções SSE, SSE2 e SSE3 operando em registradores XMM

3. PERFORMANCE

O conjunto de instruções multimídia tem um histórico de aceleração bem sucedida de codificação de multimídia, com mínimo impacto na arquitetura escalar[12]. Informalmente, pode-se estimar o ganho de velocidade em certas operações mais gerais, como multiplicação de matrizes[17] (em [16], por exemplo,, melhora de 2 a 5 vezes, dependendo do esforço em otimizar o acesso à memória). Trabalhos mais formais quantificam os ganhos em função de domínios específicos. Por

exemplo, em [6], foi feito um estudo para ganho de performance em processamento de sinais com Altivec (filtros, FFT, IDCT, aritmética vetorial). Os ganhos de velocidade variaram de 1,60 a 11,66 e os ganhos de números de instruções executadas foram de 1,82 a 10,25. Os ganhos foram atribuídos ao paralelismo, à utilização de instruções complexas específicas, ao pipeline SIMD superior ao pipeline escalar de ponto flutuante e à utilização de pré-carga de cache. Outro estudo do gênero, com conjuntos de instruções mais antigos (MMX, 3DNow!, VIS, etc.) também demonstrou melhora de 2 a 5 vezes, em núcleos multimídia diversos[2].

4.CONCLUSÃO

Os conjuntos de instruções multimídia aproveitam o paralelismo de dados presente em certas aplicações. Foram, inicialmente, acrescentadas em processadores escalares para acelerar a codificação de vídeo. Atualmente, são utilizadas em diversos domínios. As implementações atuais de SSE3 e Altivec estão presentes em processadores super escalares e representam unidades funcionais de execução. Os ganhos de performance variam, tipicamente, de 2 a 5 vezes

5.REFERÊNCIAS

[1] N. T. Slingerland, A. J. Smith, 2004, Multimedia extensions for general purpose microprocessors: a survey

[2] N. T. Slingerland, A. J. Smith, 2000, Measuring the Performance of Multimedia Instruction Sets

[3] J. L. Hennessy, D. A. Patterson, 2006, Computer Architecture – A quantitative Approach, 4rd. edition

[4] Intel 64 and IA-32 Architectures Software Developer's Manual, http://www.intel.com/products/processor/manuals/

[5] AltiVec Technology Programming Environments Manual (ALTIVECPEM), http://www.freescale.com/files/32bit/doc/ref_manual/ALTIVECPEM.pdf

[6] H. Nguyen, L. K. John, 1999, Exploiting SIMD Parallelism in DSP and Multimedia Algorithms Using the AltiVec' Technology

[7] Isaiah revealed: VIA's new low-power architecture, http://arstechnica.com/articles/paedia/cpu/via-cpu-isaiah.ars/2

[8] SIMD architectures, http://arstechnica.com/articles/paedia/cpu/simd.ars

[9] Inside AMD64 Architecture, http://www.hardwaresecrets.com/article/324/9

[10] K10 architecture, http://en.wikipedia.org/wiki/Image:AMD_K10_Arch.svg

[11] Inside Intel Core Microarchitecture, http://www.hardwaresecrets.com/article/313/4

[12] G. Erickson, 1996, RISC for Graphics: A Survey and Analysis of Multimedia Extended Instruction Set Architectures

[13] Altivec Technology (ALTIVECFACT), www.freescale.com/files/32bit/doc/**fact_sheet**/**ALTIVECFACT**.pdf

[14] Z. Lei, 2008, Intel XML Parsing Accelerator with Intel Streaming SIMD Extensions 4 (Intel SSE4)

[15] Altivec, http://en.wikipedia.org/wiki/AltiVec

[16] Optimizing for SSE – A case study, http://www.cortstratton.org/articles/OptimizingForSSE.php

[17] Intel Pentium III Processor Small Matrix Library, http://developer.intel.com/design/PentiumIII/sml/