Conjunto de Instruções Multimídia

Jonathas Campi Costa
Instituto de Computação
Universidade Estadual de Campinas - Unicamp
Campinas, Brasil
RA: 085380
jon.costa@gmail.com

ABSTRACT

Apresenta-se neste artigo uma visão geral dos diferentes conjuntos de instruções multimídia existentes no mercado de processadores. São abordados os principais conceitos da tecnologia por detrás do conjunto de instruções bem como seus principais representantes; além de análises de desempenho e abordagens de implementação.

General Terms

SIMD theory, MMX, SSE, 3DNow!, Altivec.

1. INTRODUÇÃO

Durante os anos 90 houve um grande aumento no uso da computação como suporte as operações multimídia, isto é, o uso do computador na criação de informação multimídia (video, imagem, som, etc.); aliado a esse fato, as *workstations* e os computadores pessoais eram utilizados cada vez mais como instrumentos de cálculos avançados. Analisando essa tendência, os principais fabricantes de processadores utilizaram uma idéia já conhecida para atender a uma nova demanda: o uso de instruções vetoriais.

A implementação de uma arquitetura vetorial completa (a presença de registradores vetoriais em todo os estágios do *pipeline*) sobre uma arquitetura i386, por exemplo, se possível (devido a falta de flexibilidade na execução de códigos de propósito geral) ainda seria altamente complexa e custosa, do ponto de vista operacional, logo a solução encontrada pelos fabricantes de processadores foi a implementação de um subconjunto das operações tipicamente existentes em uma arquitetura puramente vetorial[9], sobre uma arquitetura do tipo SISD [3]. Para tal conjunto de operações foi dado o nome de Conjunto de Instruções Multimídia. É importante notar que existem diferenças significativas entre as instruções multimídia e vetorial [9]; *e.g.* o número de elementos em uma instrução vetorial não está presente no código da operação (*opcode*) como nas instruções multimídia, e sim em um registrador separado.

Analisando mais atentamente esse conjunto de operações multimídia podemos classifica-la, segundo a classificação proposta por Flynn [3], como pertencentes a um hardware do tipo SIMD, isto é, *Single Instruction Multiple Data*; processadores em que uma mesma instrução é aplicada sobre diferentes fluxos de dados, empacotados (o conceito de empacotamento de dados será analisado mais adiante). Essas instruções permitem ao hardware a operação simultânea de diferentes ALUs (*Arithmetic Logic Unit*), ou equivalentemente, a divisão de uma grande ALU em muitas ALUs menores que podem executar paralelamente [9].

A idéia dos projetistas de hardware foi unir o melhor de dois mundos, ou seja, unir o paralelismo existente em nível de instruções das máquinas tipo SISD com o paralelismo no nível dos dados, típico da máquinas SIMD.

O uso de instruções multimídia, referenciado de agora em diante como instruções SIMD também, pode ser visto como uma forma de aproveitamento de situações em que o paralelismo está presente e pode ser utilizado. Como um exemplo do uso de instruções SIMD podemos citar a coerência espacial em aplicações de computação gráfica [4].

Em aplicações de computação gráfica, tipicamente aplicações de *rasterização* e processamento de imagem, a coerência espacial está muito presente, isto é, a probabilidade de que o conjunto de pixels vizinhos a um certo pixel em questão possua atributos diferentes é muito pequena [4]. Logo, quando desejamos aplicar uma instrução sobre a imagem, a mesma instrução será aplicada ao mesmo conjunto de pixels com iguais propriedades, portanto utilizando uma única instrução sobre o mesmo conjunto de dados. Se o conjunto de pixels suportado pela operação em questão for de cardinalidade n, podemos dizer que a instrução SIMD possui n unidades funcionais onde cada unidade opera sobre um pixel a mesma instrução.

Um exemplo mais comum é o uso de instruções SIMD para aritmética de vetores; como um vetor pode ser decomposto por suas coordenadas, pode-se efetuar operações aritméticas como soma, subtração, etc., sobre as diferentes coordenadas dos vetores envolvidos nas operações. Por exemplo, para a soma de dois vetores: $\vec{Z} = \vec{X} + \vec{Y}$ pode ser executada diretamente sobre as coordenadas dos vetores: $z_i = x_i + y_i$, onde cada soma será efetuada por uma unidade funcional distinta mas a partir da mesma instrução, *i.e.*, a instrução de soma.

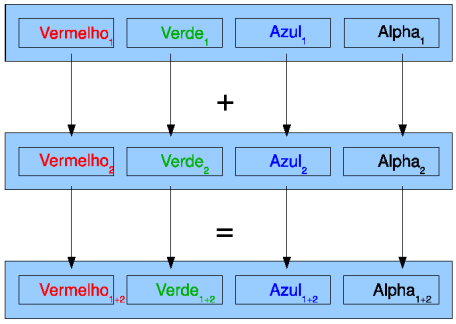

Figure 1: Diagrama representando a soma entre dois pixels diferentes utilizando os registradores vetoriais.

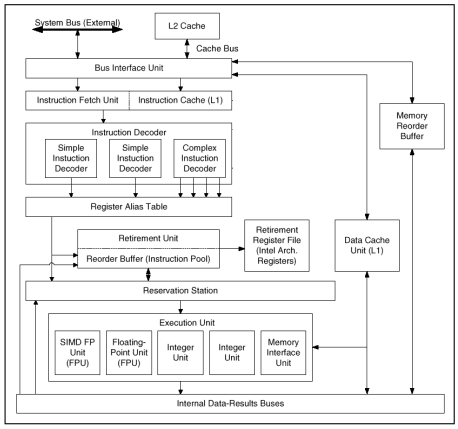

Figure 2: Diagrama do pipeline básico de execução da arquitetura P6. Figura retirada de [1].

Nos dois exemplos citados acima podemos observar claramente a maior vantagem do uso das instruções SIMD: a diminuição da latência no acesso a memória ao ler todos os dados necessários uma única vez e efetuar a mesma operação sobre eles[5].

2. REGISTRADORES VETORIAIS

A base da arquitetura vetorial e das instruções SIMD são os registradores vetoriais. Um registrador vetorial é um registrador em que os dados estão organizados na forma de um vetor, isto é, os dados podem ser comparados aos valores dos escalares que compõem as coordenadas de um vetor. Assim, enquanto que em arquiteturas do tipo SISD, a CPU opera sobre escalares um a um, em uma arquitetura do tipo SIMD a CPU opera sobre uma linha desses escalares, todos do mesmo tipo, executando uma mesma operação sobre todos, como uma unidade.

Esses vetores são representados em um formato de dados chamado: empacotado (*packed data*). Por empacotado podemos entender que os dados são agrupados em diferentes formatos, por exemplo, para um registrador vetorial de 128-bits, podemos empacotar os dados como 4 inteiros de 32-bits cada, ou 8 inteiros de 16-bits cada.

Utilizando dessa abordagem de organização dos dados, é possível efetuar operações sobre os dados de forma eficiente (a latência no acesso aos dados é diminuída, como anteriormente exeplicado). Como abordado anteriormente, a soma de dois pixels pode ser efetuada em uma operação de adição apenas, bastando organizar os elementos do pixel (cores vermelha, verde, azul e o canal de composição) em um registrador vetorial. Podemos observar tal arranjo na Figura 1.

3. ARQUITETURA PARA INSTRUÇÕES MULTIMÍDIA

Em geral, a adição das instruções multimídia é efetuada através da alteração do estágio de execução das arquiteturas escalares [11, 9, 5, 1], incluindo uma unidade especializada para a execução das instruções SIMD. Podemos observar a presença dessas unidades nas arquiteturas dos processadores da família P6 da Intel (Figura 2), na arquitetura do processador Athlon da AMD (Figura 3) e na arquitetura do processador PowerPC 970 (Figura 4), por exemplo.

Uma implementação interessante foi a do primeiro conjunto de instruções multimídia da Intel, o MMX [18]. As instruções MMX foram implementadas sobre a unidade de ponto flutuante já disponível nos primeiros membros da arquitetura P5, isto é, os registradores vetoriais foram implantados sobre os registradores de ponto flutuante logo, os registradores MMX, como veremos mais adiante, que eram implementados com largura de 64-bits para trabalhar com dados em precisão inteira, eram representados internamente como números em ponto flutuante inválidos, já que os registradores de ponto flutuante da arquitetura P5 possuiam largura de 80-bits. Isso era uma forma de diferenciar o conteúdo dos registradores também.

No início, cada fabricante de processadores criou e implementou seu próprio conjunto de instruções SIMD, como por exemplo os conjuntos MAX, VIS, MDMX, etc; enquanto que na arquitetura i386 esse conjunto de instruções acabou por tornar-se um padrão de mercado, o padrão SSE; apesar de atualmente existirem algumas variações, como veremos mais adiante.

Para determinar quais seriam as melhores intruções a implementar, os fabricantes de processadores selecionaram um conjunto de aplicações multimídia que melhor representava o que eles acreditavam ser um conjunto representativo de aplicações multimídia geral [2]. Analisando essas aplicações, criaram, além das instruções básicas de aritmética e instruções de manipulação lógica e de alinhamento, instruções para suportar operações comuns a muitas das aplicações. Essas operações variam em número e complexidade de fabricante para fabricante.

Em geral, podemos dividir o conjunto de instruções SIMD implementadas pelos fabricantes em quatro grandes grupos:

- Instruções aritméticas

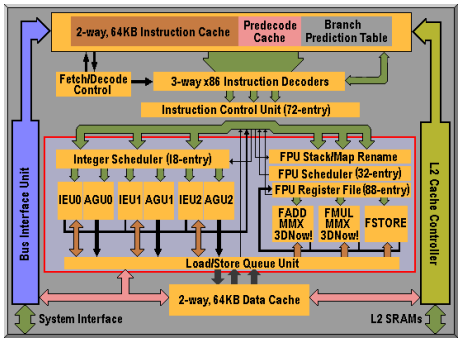

Figure 3: Diagrama da arquitetura do Athlon. Figura retirada de [10].

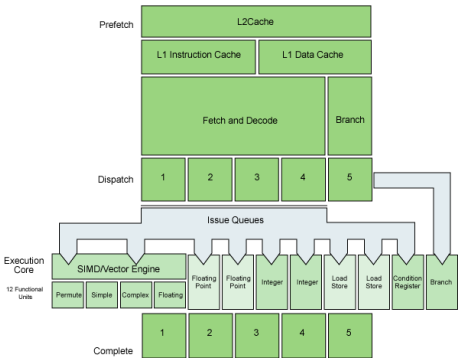

Figure 4: Diagrama da arquitetura do PowerPC 970. Figura retirada de [13].

  - Podemos dividir as instruções aritméticas em dois subgrupos: as de ponto flutuante e as de precisão inteira. Aqui estão incluídas as principais operações aritméticas, *e.g.*: saturação (*clampf*), módulo, soma, subtração, divisão e multiplicação (alguns fabricantes implementam essas duas últimas operações apenas através de *shifts* para esquerda e direita, respectivamente [6]). Em ponto flutuante podemos citar também operações específicas para arredondamento e conversão.

- Instruções lógicas
- Instruções para conversão de dados e reordenação
  - Instruções para organizar os dados em pacotes de 8-, 16-, 32-, 64-, ou 128-bits, e reordenação dos dados dentro de um pacote.
- Instruções de memória.
  - Instruções para acesso, leitura e escrita e, em alguns casos, instruções de armazenamento parcial através do uso de máscaras [6].

A seguir são apresentadas exemplos de implementações de instruções SIMD.

4. PRINCIPAIS REPRESENTANTES

Como previamente abordado, cada fabricante de processadores optou por implementar seu próprio conjunto de instruções multimídia. A seguir apresentamos as principais implementações e algumas de suas características mais importantes.

4.1 MAX-1

MAX é um acrônimo para *Multimedia Acceleration eXtensions*, um conjunto de instruções multimídia desenvolvidas para a arquitetura PA-RISC da Hewlett-Packard. Foi o primeiro conjunto de instruções multimídia disponível para o público em geral, em 1994. Projetadas através da análise de uma aplicação MPEG [6], as instruções mais frequentes foram divididas em simples primitivas e implementadas em hardware.

As instruções MAX operam sobre tipos de dados SIMD de 32-bits, formados por múltiplos inteiros de 16-bits alinhados e armazenados (empacotados) em registradores de propósito geral. Assim, uma mesma instrução pode ser executada sobre 2 inteiros de 16-bits cada. Podemos observar as principais instruções MAX na tabela 1.

4.2 VIS

VIS é um acrônimo para *Virtual Instruction Set*, conjunto de instruções SIMD para os processadores `UltraSPARC I` desenvolvidos pela Sun Microsystems em 1995. Assim como a maioria dos conjuntos de instruções multimídia, foi desenvolvido motivado pela necessidade da melhora de desempenho de aplicações multimídia como MPEG e visualização computacional [7]. Como premissa básica no desenvolvimento foi aplicada que qualquer potencial instrução deveria ser executada em um único ciclo de *clock* ou deveria ser facilmente *pipelined*[14].

Table 1: Principais Instruções MAX

Instrução	Descrição
HADD	Soma paralela com aritmética modular.
HADD,ss	Soma paralela com saturação e sinal.
HADD,us	Soma paralela som saturação sem sinal.
HSUB	Subtração paralela com aritmética modular.
HSUB,ss	Subtração paralela com saturação e sinal.
HSUB,us	Subtração paralela com saturação sem sinal.
HAVE	Média paralela.
HSHLADD	*Shift* paralelo para esquerda e soma saturada com sinal.
HSHRADD	*Shift* paralelo para direita e soma saturada com sinal.

As instruções VIS operam sobre tipos de dados SIMD de 64-bits, formados por dados de precisão inteira (2 inteiros de 32-bits ou 4 inteiros de 16-bits). Para efetuar suas operações, utiliza a já presente unidade de ponto flutuante, maximizando o paralelismo no nível de instruções e maximizando o número de registradores utilizados.

Entre as principais características, podemos citar:

- duas instruções VIS podem ser lidas e decodificadas por ciclo de *clock*,
- as instruções VIS são totalmente *pipelined*,
- as instruções VIS possuem baixa latência.

4.3 VIS 2.0

A extensão das instruções VIS [8]. Implementadas primeiramente nos processadores `UltraSPARC III` veio como uma solução para um problema recorrente do VIS, a reordenação dos dados dentro de um pacote SIMD (armazenamento dos dados no registrador SIMD).

Assim como seu ancestral, possui uma execução totalmente *pipelined*, como exceção das instruções de raíz quadrada e divisão. É composta de um *pipeline* de 12 estágios e 32 registradores SIMD. Efetua operações sobre dados de precisão inteira somente, a versão VIS 3.0 pretende corrigir essa depenência.

4.4 MMX

É o primeiro conjunto de instruções multimídia da Intel. Apareceu pela primeira vez na arquitetura P5, a arquitetura dos primeiros Pentiums, em 1997. Assim como os conjuntos de instruções anteriormente descritos, as instruções MMX foram desenvolvidas a partir do estudo de um conjunto de aplicações multimídia: MPEG-1, MPEG-2, audio, reconhecimento e compressão de voz, jogos 3D, modens, etc. Analisando essas aplicações, um conjunto de características críticas para a execução foram obtidas e então transformadas para instruções MMX [14].

Uma das principais premissas no desenvolvimento das instruções MMX foi a necessidade de não alterar o hardware existente até o momento para que os sistema operacionais da época pudessem executar no novo processador com as novas instruções. Assim, as instruções MMX foram mapeadas na arquitetura e nos registradores de ponto flutuante existente,

Table 2: Principais Instruções MMX

Instrução	Descrição
PADD-	Soma paralela com aritmética modular.
PADDS-	Soma paralela com saturação e sinal.
PADDSU-	Soma paralela som saturação sem sinal.
PSUB-	Subtração paralela com aritmética modular.
PSUBS-	Subtração paralela com saturação e sinal.
PSUBUS-	Subtração paralela com saturação sem sinal.
PACKSSWB	Empacotamento.

o que significa que o uso de instruções MMX e de ponto flutuante não eram permitidas ao mesmo tempo.

As instruções MMX definem então um modelo SIMD simples e flexivel capaz de trabalhar com dados de precisão inteira empacotados em registradores SIMD de 64-bits [18]. É composta por 8 registradores SIMD de 64-bits cada, nomeados de MMX0 até MMX7, com dois tipos de acesso aos dados: mode de acesso de 64-bits e modo de acesso de 32-bits. É capaz de armazenar 8 bytes, ou 4 palavras de 16-bits cada, ou 2 palavras de 32-bits cada.

Seu conjunto de instruções é composto por 47 intruções agrupadas nas seguintes categorias: transferência de dados, aritmética, comparação, conversão, desempacotamento, lógica, *shift* ou deslocamento e *Empty MMX state instructions - EMMS*[1].

Alguns das principais instruções podem ser vistas na tabela 2. Na tabela 2, um instrução do tipo XXX- pode ser interpretada como atuando sobre bytes (B), palavras de 16-bits (W), e palavras duplas de 32-bits (D). Por exemplo, PADDD é uma soma paralela em registradores SIMD de 64-bits contendo dois dados de 32-bits cada.

4.5 3DNow!

A resposta da concorrente AMD as instruções MMX da Intel em 1998. A AMD licenciou o conjunto de instruções e registradores MMX da Intel e adicionou suporte particionado a tipo de dados em ponto flutuante [14]. Como isso, em um processador AMD com suporte a instruções multimídia do tipo 3DNow! é possível efetuar operações em ponto flutuante em paralelo no formato SIMD.

Com a introdução dos processadores Athlon, em 1999, a AMD introduziu 19 novas instruções ao seu conjunto 3DNow!, batizando-o de AMD Enchanced 3DNow!.

4.6 SSE

O conjunto de instruções multimídia SSE da Intel é a evolução natural das instruções MMX [18]. Introduzido em 1999 com o Pentium III sua principal diferença em relação as instruções MMX é a capacidade de trabalhar com dados em ponto flutuante além de uma nova unidade (um estado arquitetural) para execução de operações SSE. Essa nova unidade reduziu a complexidade da implementação bem como

[1]Essa instrução esvazia o estado MMX do processador e deve ser chamada ao final de uma rotina utilizando instruções MMX e antes de iniciar outras rotinas que utilizem instruções de ponto flutuante.

da programação, permitindo aos desenvolvedores utilizar SSE e MMX ou x87 concorrentemente.

As instruções SSE são compostas por 8 registradores SIMD de 128-bits cada, nomeados de XMMX0 até XMMX7, em modo não-64-bits [18], e 16 registradores XMM em modo 64-bits. Capaz de trabalhar com 4 dados em ponto flutuante (*IEEE single precision floating-point*) empacotados.

Um fato interessante é que, apesar da definição dos registradores de 128-bits de largura, presente no SSE, as unidades de execução presentes no Pentium III possuem largura de 64-bits. A unidade de execução SSE do Pentium III traduz a instrução SSE de 128-bits em dois pares de micro-ops de 64-bits cada [14].

4.7 SSE2

É a evolução do conjunto de instruções SSE. Introduzidas inicialmente em 2000, com o processador Pentium IV, ela extendeu o conjunto de instruções em precisão inteira do MMX para registradores de 128-bits, com o auxílio de 68 novas instruções que utilizam os mesmo registradores XMM0-7, do original SSE [14, 18]. Sua principal funcionalidade foi a adição da capacidade de trabalhar com dados em ponto flutuante de precisão dupla, destinados a aplicações científicas, na maior parte do do tempo.

4.8 SSE3, SSSE3, SSE4, SSE4.1, SSE4.2 e SSE4a

O conjunto de instruções SSE3 foi introduzido com o processador Pentium IV com suporte a *Hyper-Threading*, enquanto que as instruções SSE4 foram introduzidas nos processadores de da geração de $45nm$ [18].

A principal diferença entre as instruções SSE2 e SSE3 é a inclusão de novas instruções para aritmética horizontal [18], *i.e.*, artimética entre partes diferentes de registradores SIMD, e não todo o registrador; e instruções que melhoram a sincronização entre agentes multi-thread.

As instruções SSSE3 adicionaram 18 novas instruções para aritmética horizontal, intruções para acelerar o cálculo do produto vetorial, instruções para alinhamento dos dados, e para execução de valores absolutos entre outras [18].

O conjunto de instruções SSE4 é composto por dois subconjuntos, o das instruções SSE4.1 e SSE4.2. Seu desenvolvimento teve como motor impulsionador o desempenho de aplicações em mídia, imagem e 3D. SSE4.1 adicionou instruções para melhorar a vetorização via compilador e melhorou significantemente o suporte a computação de palavras de largura dupla empacotadas [18]. SSE4.2 adicionou instruções para melhorar o desempenho em aplicações com uso de *strings* e *Application-target accelerator (ATA) instructions*[2].

As instruções SSE4a são a implementação de 4 instruções do conjunto SSE4 da Intel mais 2 novas instruções, pela concorrente AMD na arquitetura do K10.

4.9 Altivec

[2]Acelera o desempenho de softwares na procura de padrões de bits.

É o conjunto de instruções SIMD para os processadores Power desenvolvidos em conjunto entre a Motorola e a IBM [20]. As instruções Altivec são compostas de 32 registradores de 128-bits que armazenam dados fontes para as unidades Altivec. Esses registradores oferecem suporte a 16 dados paralelos de 8-bits cada em precisão inteira (como ou sem sinal) ou caracteres, ou 8 dados paralelos de 16-bits cada em precisão inteira (com ou sem sinal) ou 4 dados paralelos de 32-bits em precisão inteira e ponto flutuante. Possui um conjunto extenso de instruções, totalizando mais de 150 diferentes instruções.

4.10 Outras extensões

Entre as outras extensões existentes podemos citar: *MIPS V ISA Extension*, *MDMX - Mips Digital Media Extension*, *MIPS-4D ASE - Application Specific Extension* todos para processadores MIPS, *MVI - Motion Video Instructions* para processadores Alpha e *NEON* para processadores ARM.

5. DESEMPENHO

O ganho de desempenho em aplicações utilizando o conjunto de instruções multimídia é, em geral, avalidado através da comparação do algoritmo utilizando as instruções multimídia contra aquele que não as utiliza [15]. O que a literatura mostra [19, 16] é que, em geral, pode-se obter melhoras de 2 a 5 vezes em desempenho. Tal variação no ganho de desempenho médio é fruto da implementação do algoritmo, da otimização no acesso a memória e também nas características do problema a ser paralelizado (vetorizado) [17].

6. CONCLUSÃO

O conjunto de instruções multimídia trouxe aos computadores pessoais e as *workstations* a vantagem dos computadores vetoriais: o paralelismo dos dados. Mesmo que, divididas em vários sabores (implemetações de fabricante para fabricante), um pabrão para a arquitetura i386 foi estabelecido com as instruções SSE. Em geral, possuem um ganho de desempenho da ordem de 2 a 5 vezes mas, deve-se levar em conta que esse ganho de desempenho é fortemente baseado na habilidade dos programadores já que os compiladores atuais são imaturos no sentido de utilizar toda a capacidade SIMD dos atuais processadores do mercado.

É importante notarmos que as primeiras instruções multimídia surgiram na época em que os processadores não dispunham de desempenho o sucifiente para as aplicações multimídia existentes, mas hoje com o avanço das atuais GPUs fica a pergunta: quanto de energia criativa os projetistas de processadores devem inserir no projeto dos processadores atuais com instruções multimídia se as principais aplicações multimídias são hoje aceleradas pelas GPUs? Uma possível resposta para essa pergunta pode ser o uso de inúmeros processadores com conjuntos de instruções multimídia como um único processador de propósito gráfico; por exemplo a arquitetura Larrabee da Intel [12].

7. REFERENCES

[1] J. K. And. Pentium iii processor implementation tradeoffs, 1999.

[2] G. Erickson. Risc for graphics: A survey and analysis of multimedia extended instruction set architectures, 1996.

[3] M. Flynn. Some computer organizations and their effectiveness. *IEEE Trans. Comput.*, C-21, 1972.

[4] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice*. Addison Wesley, second edition in c edition, 1996.

[5] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach (The Morgan Kaufmann Series in Computer Architecture and Design)*. Morgan Kaufmann, fourth edition, 2006.

[6] R. Lee. Accelerating multimedia with enhanced microprocessors. *IEEE Micro*, 15:22–32, 1995.

[7] S. Microsystems. *VIS Instructions Set User's Manual*. Sun Microsystems, 1997. Manual.

[8] S. Microsystems. *The VIS Instructions Set*. Sun Microsystems, 2002. A White Paper.

[9] D. A. Patterson and J. L. Hennessy. *Computer Organization and Design: The Hardware/Software Interface*. Morgan Kaufmann, fourth edition, 2009.

[10] PCTECHGUIDE. Amd athlon, 2009. [Online; accessed 20-June-2009].

[11] S. K. Raman, V. Pentkovski, and J. Keshava. Implementing streaming simd extensions on the pentium iii processor. *IEEE Micro*, 20(4):47–57, 2000.

[12] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, M. Abrash, P. Dubey, S. Junkins, A. Lake, J. Sugerman, R. Cavin, R. Espasa, E. Grochowski, T. Juan, and P. Hanrahan. Larrabee: a many-core x86 architecture for visual computing. *ACM Trans. Graph.*, 27(3):1–15, August 2008.

[13] C. Shamieh. Understanding 64-bit powerpc architecture: Critical considerations in 64-bit microprocessor design, 2004. [Online; accessed 20-June-2009].

[14] N. Slingerland and A. J. Smith. Multimedia extensions for general purpose microprocessors: a survey. Technical Report UCB/CSD-00-1124, EECS Department, University of California, Berkeley, Dec 2000.

[15] J. Stewart. An investigation of simd instruction sets. Technical report, School of Information Technology and Mathematical Sciences, University of Ballarat, Nov 2005.

[16] C. D. W. F. Stratton. Optimizing for sse: A case study, 2002. [Online; accessed 20-June-2009].

[17] D. Talla, L. K. John, and D. Burger. Bottlenecks in multimedia processing with simd style extensions and architectural enhancements. *IEEE Transactions on Computers*, 52(8):1015–1031, 2003.

[18] I. A. Team. *Intel Architecture Software Developer's Manual*. Intel Corporation, 2007.

[19] H.-M. C. Trung Hieu Tran and S.-B. Cho. Performance enhancement of motion estimation using ss2 technology. In *World Academy of Science, Engineering and Technology*, volume 30, July 2008.

[20] Wikipedia. Altivec — wikipedia, the free encyclopedia, 2009. [Online; accessed 15-June-2009].