Resumo do Artigo: *Larrabee: A Many-Core x86 Architecture for Visual Computing*

Larrabee: A Many-Core x86 Architecture for Visual Computing
Seiler, L.; Carmean, D.; Sprangle, E.; Forsyth, T.; Dubey, P.; Junkins, S.; Lake, A.; Cavin, R.; Espasa, R.; Grochowski, E.; Juan, T.; Abrash, M.; Sugerman, J.; Hanrahan, P.
Micro, IEEE
Volume 29, Issue 1, Date: Jan.-Feb. 2009, Pages: 10-21

Jonathas Campi Costa - RA: 085380

Larrabee é uma arquitetura baseada na arquitetura x86, extendida de unidades de processamento vetorial (VPU) e um conjunto de funções fixas (funções implementadas diretamente em hardware) com objetivo de obter um desempenho superior em aplicações gráficas. É composta de um vetor de núcleos de processamento, onde cada núcleo é uma extensão da arquitetura x86 e estes compostos de uma unidade escalar baseada no processador Pentium, cuja pipeline é curta e possui uma execução barata, e uma unidade vetorial, com largura de 512 bits e freqüentemente referenciada como uma unidade vetorial de largura 16 (16 palavras de 32 bits cada). Os núcleos se comunicam através de uma rede de intercomunicação de 1024 bits chamada de *Interprocessor ring network*, que além da comunicação, provê suporte as operações de I/O e acesso as funções fixas. Adicionalmente há um decodificador de instruções, uma memória cache do tipo L1, para permitir um acesso de baixa latência a memória tanto para a unidade escalar como para a vetorial, com 32 Kbytes para instruções e 32 Kbytes para dados, e um subconjunto local, de 256 Kbytes, de memória cache do tipo L2 de uma memória global, o que permite leituras em paralelo. Entre as instruções de cache é possível citar: *prefetching* em L2, de L2 para L1, e *evicting* de L1 após acesso, o que permite o uso eficiente de L1 para dados do tipo *streaming*. Vale notar que a unidade escalar possui o conjunto completo de instruções do Pentium e portanto é capaz de executar até mesmo o kernel de um sistema operacional.

A VPU foi concebida para trabalhar com 16 palavras de 32 bits cada, capaz de executar operações em instruções de precisão inteira, em ponto flutuante e precisão dupla em ponto flutuante. A escolha de largura foi baseada em um estudo do *trade-off* entre o aumento da densidade computacional e a dificuldade em obter alta utilização de unidades muito largas. As VPUs são constituídas de registradores de máscara, unidade lógica aritmética vetorial de largura 16, unidades para executar instruções de *replicate* e *swizzle* (instruções para aumentar a eficiência de armazenamento do cache e reduzir o número de instruções requeridas para implementar algoritmos gráficos mais comuns), registradores vetoriais, e conversores numéricos. É passível de nota a capacidade de executar diferentes *shaders*[1] em paralelo, mesmo que estes possuam acesso a vetores baseado em cálculo de índices; isso porque a VPU permite que cada "faixa de palavra", dentre as 16 presentes, possua seu próprio endereço de leitura e escrita.

Diferentemente das GPUs atuais, Larrabee implementa o processo de rasterização e interpolação em software ao invés de hardware pois, com o advento das versões mais novas das principais APIs gráficas, interpolação não é mais efetuada em precisão inteira, logo pode ser eficientemente efetuada pela VPU do Larrabee, e a rasterização que é mais rápida se implementada em hardware, teria um custo alto para a arquitetura como um todo, se fosse implementada emhardware. O processador de texturas, uma das partes de maior custo de processamento no pipeline gráfico, é implementado como uma função fixa em hardware.

Para obter o máximo em performance da arquitetura Larrabee, foi desenvolvido um um renderizador. O renderizador é baseado em um algoritmo do tipo *sort-middle*, isto é, um algoritmo que após transformar as primitivas de entrada para um espaço diferente, efetua uma ordenação 3D e uma comparação de profundidade para eliminar as primitivas. Sua arquitetura é simples, a imagem a ser renderizada é dividida em *tiles*, e os triângulos das primitivas de entrada presentes em um determinado *tile* são armazenadas em *bins*; o conjunto de *bins* da imagem é chamado de *Bin set*, e fica armazenado na memória principal do computador e não no chip. A esse conjunto de tarefas referencia-se como *front end* da arquitetura de renderização. Uma vez que os *bins* tenham sido preenchidos, durante as fases do *front end*, cada *bin* é então processado por um núcleo separado, assim a comunicação entre os núcleos só é necessária ao final do processamento do *tile*, o que permite um uso maior da capacidade paralela do Larrabe. No processamento de cada *bin* são realizadas diversas otimizações para garantir a largura de banda no acesso a memória, além da utilização de diversos threads por núcleo: um thread principal por núcleo, chamado *setup thread*, lê as primitivas do *tile*, processa-as enviando grupos de 16 pixels, chamados de *qquad*, para outros threads, chamdos de *work threads*, que efetuam as demais operações gráficas (*pixel shadding, depth text, z-culling*, etc) necessárias sobre os pixels. Após os procedimentos descritos anteriormente, o único passo restante são as operações de textura, executadas por hardware dedicado. Como o acesso as funções de textura podem possuir uma latência de até mesmo centenas de ciclos de clock, uma estrutura de fila circular é utilizada entre os *shaders* de cada *qquad*, chamados de *fibers*, para diminuir a latência total.

Para validar a arquitetura do Larrabee e do renderizador, foram selecionados três jogos, a saber: *Gears of Wars* (GW), *F.E.A.R* (FEAR) e *Half Life 2 episode 2* (HL2); todos ajustados para o nível máximo de resolução: 1600x1200 pixels, e com amostragem de 4, 4 e 1 respectivamente para cada jogo. A performance foi medida em unidades de Larrabee, *i.e.*, um núcleo do Larrabee executando a um clock de 1GHz. Entre os principais resultados foi verificado a capacidade de escala linear no desempenho com a adição do número de núcleos, uma comparação entre o modelo de renderização adotado pelo Larrabee, o *sort-middle*, com o das atuais GPUs, o *immediate-mode*, principalmente no que diz respeito a utilização da banda de memória, pois o subsistema de memória pode ser uma das partes mais vorazes no consumo de energia. Com essa comparação, pode-se corrobarar-se a escolha do modelo adotado pelo Larrabee em detrimento ao modelo das atuais GPUs, que pode chegar a usar até 7 vezes mais largura de banda, como é o caso do jogo FEAR. É interessante notar também a presença de um balanceador de carga dinâmico, escrito em software, presente em todos os estágios da arquitetura do Larrabee que, diferente das GPUs atuais, só está presente nos estágios de *vertex* e *shaders*.

Finalmente uma discussão sobre o modelo nativo de programação de muitos núcleos, com suporte a APIs como p-Threads, OpenMP, e APIs nativas; além de um compilador nativo em C/C++, com extensões de vetorização; e a capacidade de programar o Larrabee nativamente, para implementação de qualquer aplicação. Como exemplos são aprensetadas outras possíveis aplicações: *real-time ray tracing*, processamento de imagens, simulações físicas e computação numérica. É digno de nota a previsão presente no último paragrafo do artigo, a convergência entre os modelos de programação, GPU e CPU, para um modelo comum.

[1] Pequenos programas que executam sobre os pixels.