A arquitetura Alpha

Trabalho de MO401

Luís Guilherme Fernandes Pereira*
Instituto de Computação
Unicamp
lpereira@gmail.com

RESUMO

Este trabalho pretende descrever diversas características da arquietutra Alpha, bem como formas de implementação dessas características no microprocessador Alpha-21264 através de um aprofundado estudo de caso.

1. INTRODUÇÃO

O trabalho presente consiste na descrição da arquitetura Alpha, de forma geral e específica do processador Alpha 21264.

A grande motivação, de que tratarei no fim do artigo, é o grande exemplo que se caracteriza essa arquitetura. Por ter diversas funcionalidades de pesquisa recente, torna-se uma grande fonte didática. Também sempre me fascinou a beleza de sua implementação.

Organizo o trabalho como se segue: inicialmente dou algumas notas preliminares, resumindo os conceitos de ILP, Exploração Dinâmica, emissão múltipla e previsão de desvios, necessárias para a descrição da arquitetura. Em seguida, descrevo a arquitetura Alpha como um todo, sem entrar em especificidades de implementação, tratando das linhas gerais que seu projeto segue. Após isso, aprofundo-me na implementação dessa arquitetura no microprocessador Alpha 21264, e no computador Alpha DS20L. Concluo com algumas considerações sobre esse projeto e com as razões de estudá-lo.

2. NOTAS PRELIMINARES

Antes de tratar da arquitetura Alpha, é bom definir alguns termos que farei uso. O primeiro, e mais importante, é paralelismo em nível de instrução (em inglês *Instruction Level Paralelism*, o que dá a sigla ILP). ILP consiste em utilizar diferentes unidades funcionais para executar operações básicas simultaneamente. Para isso, há duas abordagens: a exploração dinâmica (por *hardware*), e a abordagem via software, fazendo uso de *very long instruction words*.

*RA: 009206

Essa primeira abordagem é utilizada pela arquitetura Alpha. Ela faz uso de uma emissão múltipla de instruções superescalar: as instruções têm o tamanho normal, mas o processador realiza o *fetch* (a busca) de várias ao mesmo tempo. Também o preditor de desvios em *hardware* pode, eventualmente, executar os dois caminhos de um *branch* até descobrir qual é o correto. A previsão de desvios consiste em tentar descobrir o rumo que uma execução deve seguir, baseada em certas heurísticas, de forma a não atrasar a execução do código até a descoberta do caminho adequado.

A grande vantagem da abordagem dinâmica, é que permite que se execute código otimizado sem recompilar. Um código compilado para uma implementação anterior da mesma arquitetura pode ser executado com todas as otimizações em hardware. A grande desvantagem é que torna o *hardware* mais caro.

Mais informações sobre ILP podem ser encontradas no livro de Hennesy e Patterson ([Hennesy e Patterson 2003]).

3. A ARQUITETURA ALPHA

A arquitetura Alpha é uma arquitetura RISC de 64 bit, desenvolvida pela DEC (Digital Equipment Corp.) como sucessora do VAX,[DEC Alpha] tendo recebido várias influências da arquitetura MIPS.

O Alpha é um processador que segue a filosofia de projeto RISC, não utilizando operandos da memória. Em seu conjunto de instruções há instruções que trabalham com byte e com palavras. Além disso, ele tem instruções específicas para trabalhar com vídeos, e operações avançadas de ponto flutuante. Chama atenção o fato da escolha de uso de 64-bit ser motivada por uma espectativa do fim do espaço de endereçamento em 32 bits[Sites 1992], o que só começa a acontecer nos dias de hoje.

Já desde sua primeira versão, o 21064, o processador já era superescalar, e fazia uso de *superpipeline*, que consiste em dividir os estágios básicos do *pipeline* em outros estágios. Quando lançado, era o processador mais rápido do mundo.

O 21164 foi o primeiro a ter um cache L2 no chip e o 21264 foi o primeiro processador de alta freqüência a fazer uso de execução fora de ordem para obter ILP.

As diversas versões de chips utilizando arquitetura Alpha têm a mesma idéia mas diferentes funcionalidades e im-

plementações. Daremos prioridade para o 21264, que teve maior impacto no mercado. Fiz essa opção para poder tratar de detalhes da arquitetura, como instruções específicas, forma de exploração de paralelismo, características físicas, performance, etc.

4. ESTUDO DE CASO: ALPHA 21264

O chip Alpha 21264 é o chip utilizado nas servidoras Alpha DS10, Alpha DS20 entre outras. Há algumas Alpha DS20 no IC, como as servidoras `tatu` e `cotia` do LAD e algumas servidoras do LBI.

O Alpha-21264 tem:

- 4 unidades de execução inteira
- 2 unidades de propósito geral
- 2 ULAs de endereço
- 32 registradores de inteiros e 31 registradores de ponto flutuante.
- 48 registradores de reordenação de inteiros e 40 de ponto flutuante.
- 80 registradores de inteiros adicionais (cópia dos outros 80)

  Os tipos de instruções que o chip suporta são:

- Desvio (*branch*)
- Ponto flutuante
- Memória
- Memória / Função
- Memória / Desvio
- Operação
- *PALcode*
- Pré-*fetch* de cache
- MVI (instruções de *motion-video*)

Algumas instruções

- Ponto flutuante
  - `sqrts`: raiz quadrada com precisão simples
  - `sqrtt`: raiz quadrada com precisão dupla
  - Há instruções para mover dados de registradores de ponto flutuante para registradores de inteiros.
- MVI
  - `unpkbw`: desempacota bytes para palavras
  - `unpkbl`: desempacota bytes para
  - `packwb` : trunca os 4 palavras componentes do registro de entrada e as escreve nos 4 bytes menos significativos do registro de saída

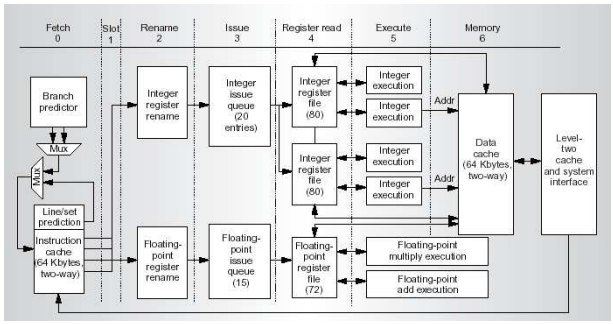

Figura 1: Pipeline do Alpha 21264

  - `packlb`: trunca as 2 palavras longas componentes do registro de entrada e as escreve nos 4 bytes menos significativos do registro de saída.
  - mínimo e máximo de bytes e palavras: (da forma `minXXX Ra, Rb, Rc`)
    - `MINUB8`
    - `MINUW4`
    - `MINSB8`
    - `MINSW4`
    - `MAXUB8`
    - `MAXUW4`
    - `MAXSB8`
    - `MAXSW4`
  - `perr`: essa instrução realiza estimativas de movimento de 8 pixels em apenas um *tick* do clock.

4.1 Características do paralelismo

O Alpha 21264 se caracteriza por ter um paralelismo super escalar especulativo. Ele realiza:

- *fetch* de até 4 instruções por ciclo;
- execução especulativa e dinâmica;
- predição de desvios;
- renomeação de registradores.

  O *pipeline* de instruções do Alpha 21264 tem 7 estágios:

- **0** Busca;
- **1** Transferência;
- **2** Renomeação;
- **3** Delegação;
- **4** Leitura dos registradores;
- **5** Execução;
- **6** Memória.

Além disso, há o estágio de mapeamento que realiza a renomeação de registradores.

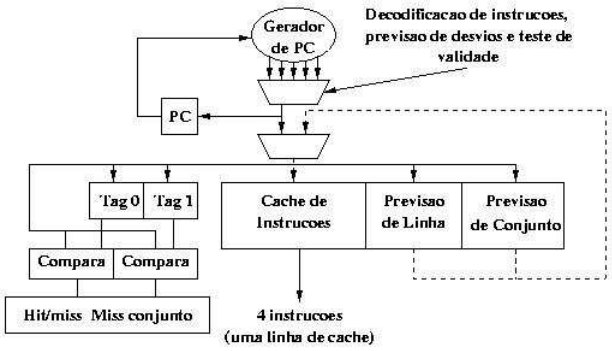

Figura 2: Busca (fetch) de instruções

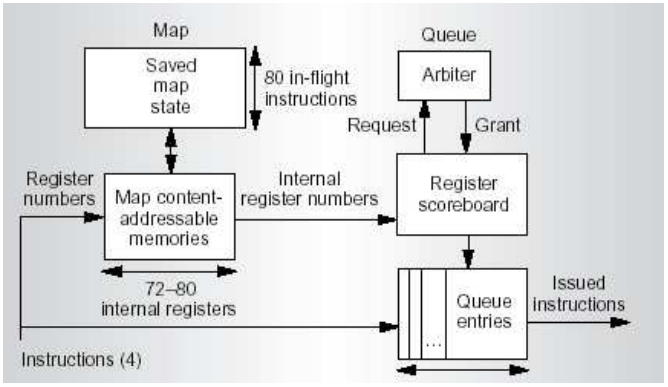

Figura 3: Renomeação de registradores

4.2 Busca

São buscadas quatro instruções por ciclo, buscando também antecipadamente 4 linhas de cache de 64 bytes (16 instruções).

A cache de instruções, de 64 KB, 2-associativa e 4 instruções por linha, que consegue *hit-ratios* de primeiro nível muito maiores que uma cache mapeada diretamente, e possui um mecanismo de previsão de conjunto, o qual procura descobrir qual dos conjuntos será acessado em seguida. Depois disso, um mecanismo de previsão de linha tenta prever qual linha daquele conjunto será acessada. Dessa forma, descobre-se o bloco de memória que se busca. O mecanismo de busca, então, traz este bloco para o *pipeline*, totalizando no máximo quatro instruções por ciclo.

Para cada linha da cache de instrução, é prevista a próxima linha e o próximo conjunto. Ao mesmo tempo em que se lê a próxima instrução usando a previsão, ele completa em paralelo a validação das instruções anteriores.
[Daniels, Dharmesh e Ziegler 1999, Keller, Halfhill 1998]

A previsão de desvios e o gerenciamento de memória serão melhor explicados posteriormente.

4.3 Transferência

Na transferência, os dados das instruções são mandados para o *hardware* de renomeação de registradores, além disso, são separadas instruções de inteiros e ponto flutuante

4.4 Renomeação de Registradores

A renomeação de registradores é crucial para a execução fora de ordem. Esse estágio elimina dependências de dados que não são verdadeiras. Isso faz com que o processador explore de forma mais agressiva o paralelismo.

Para cada instrução com um resultado em registrador, um registrador é alocado. Este registrador tornar-se-á visível para o usuário apenas quando a instrução terminar sua execução.

A memória que mantém o mapeamento entre registradores visíveis e os utilizados pelas instruções é acessada pelo seu conteúdo. Todos os estágios posteriores ao mapeamento utilizam os registradores mapeados nas operações. O processador mantém armazenado com cada registrador interno indicando o registrador visível ao usuário (se algum) que está associado atualmente com o registrador interno atual.

Como já dito, o Alpha 21264 possui além dos 31 registadores de inteiros e 31 registradores de ponto flutuante visíveis ao usuário (não especulativos), mais 41 registradores de inteiros e 41 registradores de ponto flutuante utilizados através do mapeamento de registradores para especulação antes do fim das instruções. São 144 registradores no total (sem contar as cópias dos 80 registradores inteiros).

Para os casos de erros na especulação, o renomeamento de registradores armazena o estado do mapeamento para cada instrução no *pipeline*, permitindo assim restaurar o estado da execução para antes do erro na especulação.

A instrução de movimentação condicional, a única que referencia três registradores, é separada em duas instruções pelo renomeamento. Ao final da execução, o renomeamento escolhe, através do bit de condição das instruções, qual dos registradores será efetivamente movimentado.

4.5 Delegação

A delegação se faz através de duas filas, uma de inteiros e outra de ponto flutuante, utilizando para isso um algoritmo de *scoreboard*

Esse algoritmo funciona da seguinte maneira: a cada ciclo ele escolhe os registradores que estão prontos para serem utilizados, usando *scoreboards*, baseados no número dos registradores internos. Os *scoreboards* mantêm o estado dos registradores e das operações que os utilizam, podendo essas ser de um único ciclo, de vários ciclos ou de número de ciclos variável. No máximo seis instruções podem ser despachadas por ciclo.

As unidades de inteiro estão divididas em dois *clusters* e as unidades de ponto flutuante estão em apenas um *cluster*. As instruções de inteiros precisam ser distribuídas de forma equilibrada entre os dois *clusters*. Quando uma instrução entra na fila de inteiros, um de dois árbitros é associado estaticamente, o qual não pode mais ser mudado. Estes árbitros, então, retiram as instruções da fila de inteiros quando os registradores necessários estão disponíveis. As duas instruções mais antigas da fila, com registradores disponíveis, são reti-

radas pelo árbitro correspondente em cada ciclo .

Cada fila pode retirar uma instrução por ciclo para cada unidade. Assim, a fila de inteiros pode retirar 4 instruções por ciclo e a fila de ponto flutuante pode retirar 2 instruções por ciclo. Os valores dos registradores podem ser recebidos direto das unidades funcionais ou através dos bancos de registradores.

Novas instruções de inteiros podem ser colocadas na fila quando há quatro ou mais posições disponíveis. Na fila de ponto flutuante, podem ser colocadas instruções enquanto houver espaço suficiente para tanto.

nstruções de leitura de inteiros são delegadas especulativamente, para atingir a latência mínima de três ciclos. Desta forma, os consumidores do valor carregado são beneficiados, pois podem acessá-lo assim que estiver disponível o resultado da especulação.

> A alternativa utilizada para lidar com os erros na especulação é uma reinicialização parcial dos *pipelines* de inteiros, o que é melhor que reinicializar todos os pipelines. Em caso de erros na especulação, dois ciclos de todos os pipelines de inteiros são desprezados. As instruções delegadas que estavam nestes ciclos são colocadas novamente na fila de inteiros novamente, tanto os consumidores da instrução de carga como as próprias instruções de carga. [Pilla 1999]

Porém, em aplicações com muitas instruções de carga de inteiros, o esquema citado pode vir a ser demasiado custoso. Assim, se uma instrução de *load* é prevista como *miss* a instrução de *load* não é delegada de forma especulativa.

> O mecanismo de previsão utilizado para as instruções de carga é o bit mais significativo de um contador de 4 bits que mantém o comportamento das instruções mais recentes. O contado saturado decrementa de dois em ciclos em que há um miss, incrementando de um quando há um hit. Este contador minimiza latências em aplicações com *hits* constantes e evita os custos da especulação em excesso em aplicações com muitos *misses*. [Pilla 1999]

Devido à latência maior necessária para as instruções de carga de ponto flutuante, quatro ciclos, estas instruções não são especuladas

4.6 Leitura de registradores

Os registradores são lidos no estágio de leitura de registradores, podendo utilizar *bypassing* de forma a usar dados dos registradores antes que eles estejam disponíveis. Isso se dá copiando do barramento os dados necessários para uma operação, diminuindo o tempo necessário para leitura.

4.7 Execução

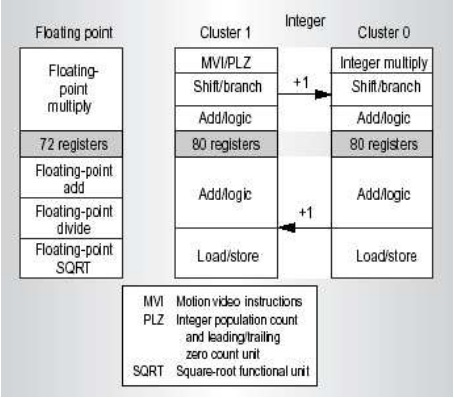

Figura 4: Clusters

Como já dito, o Alpha 21264 tem dois bancos de 80 registradores de inteiros, um sendo a cópia do outro. A atualização dos registradores gasta somente um ciclo de *clock*. São utilizados dois bancos pois há dois *clusters* e se todas as unidades de inteiros utilizassem os mesmos registradores, haveria problemas para manter a freqüência de operação elevada, o banco de registradores teria que ter tantas portas que seria difícil manter a freqüencia de operação em valores como 1GHz.

Cada um dos *clusters* está associado a uma das unidades de inteiros. Cada unidade funcional está sobre seus bancos de registradores. Das unidades de inteiros, duas unidades executam somas, operações lógicas, deslocamentos e desvios, uma delas com multiplicação e outra com multimídia. As demais executam somas, operações lógicas, cargas e armazenamentos. Uma das unidades funcionais de ponto flutuante executa adição, divisão, raiz quadrada e a outra executa multiplicação.

O Alpha 21264 trouxe novidades inexistentes até então, como *motion-video instructions*, unidade de multiplicação totalmente com *pipelines*, unidade de contador inteiro de população e contador de zeros no início/fim (PLZ), unidade de raiz-quadrada de ponto flutuante e instruções para mover valores de registradores diretamente entre registradores de ponto flutuante e de inteiros. Além disso, o processador inclui exceções precisas, processamento de NaN e de infinito, e suporte para transformar resultados fora do normal para zero.[HP 2000, Group 2000]

Classe da Instrução	Latência
Operações simples de inteiros	1
MVI / PLZ	3
Multiplicação de inteiros	7
Carga de inteiros	3
Carga de ponto flutuante	4
Addição de ponto flutuante	4
Multiplicação de ponto flutuante	4
Divisão de ponto flutuante (precisão simples)	12
Divisão de ponto flutuante (precisão dupla)	15
Raiz quadrada de ponto flutuante (precisão simples)	15
Raiz quadrada de ponto flutuante (precisão dupla)	30

4.8 Memória

Devido à natureza de execução fora de ordem das instruções, o estágio de memória precisa garantir que os resultados

sejam vistos na ordem das instruções no programa, desta forma mantendo sua semântica.

O Alpha 21264 possui um mecanismo de retirada de instruções em ordem, o qual mantém a impressão de que as instruções são executadas em ordem. As instruções são mantidas em um *buffer* circular de reordenamento com 80 posições, na ordem em que foram buscadas, sendo retiradas nesta mesma ordem. Caso uma instrução gere uma exceção, todas as instruções seguintes no buffer de reordenamento são eliminadas. Desta forma, o Alpha 21264 implementa um mecanismo de excessão precisa.

O mecanismo de retirada mantém ainda a informação sobre o uso de registradores por todas as instruções no pipeline. Cada entrada no buffer indica o registrador interno que possui o antigo conteúdo do registrador designado para a instrução.

Além das 80 instruções que podem estar em execução parcial no pipeline, o sistema de memória mantém até 32 instruções de carga e 32 instruções de armazenamento adicionais. Para tanto, o Alpha 21264 utiliza duas filas, uma fila de *loads* (LDQ) e uma fila de *stores*(STQ). Estas filas armazenam as instruções que acessam a memória na ordem em que foram despachadas. As instruções de *load* deixam a LDQ na ordem de busca, após a instrução correspondente deixar o *buffer* de reordenamento e o resultado do *load* ter sido retornado. As instruções de *store* deixam a STQ na ordem de busca, após serem retiradas do *buffer* de reordenamento e terem seus dados armazenados na memória *cache*.

A partir de todos esses dados, podemos perceber que o Alpha 21264 não necessita de apoio do compilador para explorar eficientemente o paralelismo em nível de instrução. [Kessler 1999]

4.9 Previsão de desvio

O Alpha-21264 tem uma unidade que trabalha exclusivamente com previsão de desvio. A Previsão de Desvios é realizada por três algoritmos em conjunto:

- A *Previsão Local* tem como principal objetivo prever o comportamento local dos desvios, ou seja, concentra-se em pequenos laços.
- A *Previsão Global* analisa o comportamento global dos desvios, mantendo os resultados mais recentes.
- O terceiro mecanismo decide qual das previsões anteriores será utilizada, isto é, se o mecanismo de busca vai seguir as indicações da previsão local ou global.

Os preditores de histórico local geralmente são tabelas, indexadas pelo PC, que contém informações de histórico sobre vários desvios. Entradas diferentes na tabela correspondem a instruções de desvio diferentes. Diferentes formatos de histórico local podem explorar diferentes formas de correlação local. Em alguns preditores as entradas da tabela de histórico local saturam contadores de previsão, e a previsão é o bit mais signifcativo do contador. Nesses preditores,

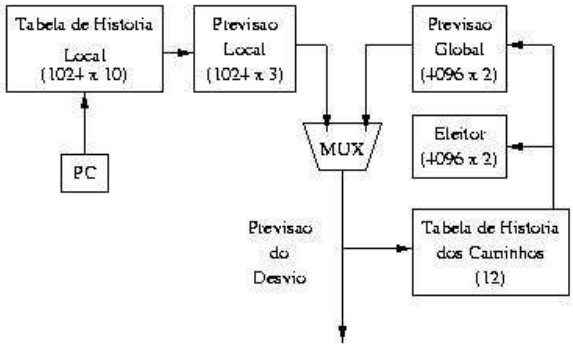

Figura 5: Preditor de desvios

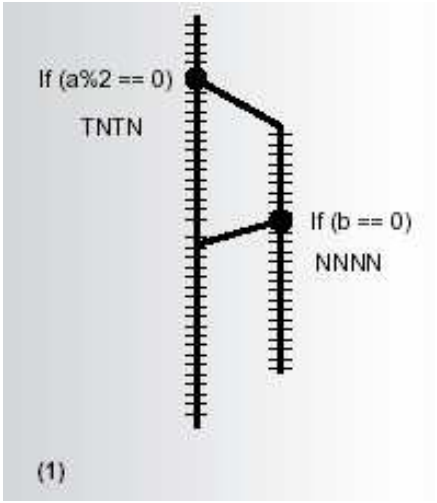

Figura 6: Exemplo de previsão de desvios

um desvio será previsto para ser tomado se ele é geralmente escolhido.

Porém, o 21264 tem preditores de dois níveis, que exploram previsão baseada em padrões. Cada entrada no primeiro nível da tabela de histórico local é um padrão de 10 bits indicando a direção do desvio selecionado nas últimas 10 execuções. A previsão de desvio local é um contador de previsão de uma segunda tabela indexado pelo padrão do histórico local. Nessa forma mais sofisticada de previsão, desvios serão previstos para serem tomados se desvios com o padrão do histórico local geralmente são tomados.

Como se vê na figura abaixo, o desvio `(b == 0)` nunca é tomado. Ele tem um padrão consistente cheio de zeros na tabela de histórico local. Depois de várias chamadas desse desvio, o contador de previsão com o índice 0 na tabela de previsão local treinará, e o desvio será previsto corretamente.

O primeiro desvio à esquerda `(a % 2 == 0)` alterna entre tomado e não tomado em cada chamada. Ele poderia não ser previsto corretamente com histórico local de apenas um nível. A alternância leva a dois padrões de histórico local na

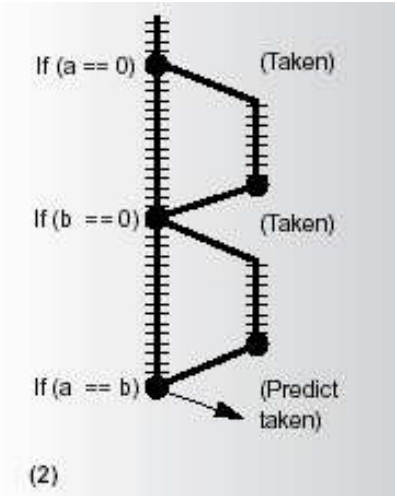

Figura 7: Outro exemplo de previsão de desvios

tabela de primeiro nível: `0101010101` e `1010101010`. Depois de várias chamadas desse desvio, os dois contadores de previsão nesses índices na tabela de previsão local treinarão, um a ser tomado e outro a não ser tomado. Qualquer padrão repetitivo de 10 chamadas de um mesmo desvio pode ser previsto dessa forma.

Com correlação global, um desvio pode ser previsto basado no comportamento de todos os desvios anteriores, ao invés do comportamento anterior de um único desvio. O 21264 explora correlação global seguindo o caminho, ou histórico global de todos os desvios. O histórico de caminhos é um padrão de 12 bits indicando a direção tomada/não tomada para os últimos 12 desvios executados, em ordem de *fetch*. O histórico global do preditor do 21264 é uma tabela de contadores indexados pelo histórico de caminhos.

A figura abaixo mostra um exemplo que pode utilizar correlação global. Se ambos os desvios `(a == 0)` e `(b == 0)` são tomados, podemos facilmente prever que `a` e `b` são iguais e portanto, o desvio `(a == b)` será tomado. Isso quer dizer que se o histórico de caminhos é `xxxxxxxx11`, o desvio deve ter previsão para ser tomado. Com execuções suficientes, os contadores de previsão global do 21264 nos índices `xxxxxxxx11` treinarão para prever a tomada do desvio e o desvio será previsto corretamente. Mesmo esse exemplo precisando apenas de uma profundidade de histórico de caminhos de 2, padrões de histórico de caminhos mais complicados podem ser encontrados com a profundidade de 12 existente.

Como desvios diferentes podem ser previstos melhor com seja previsão local ou global, o preditor de desvios do Alpha 21264 implementa tanto o preditor local quanto o global. O terceiro mecanismo, de previsão de escolha, escolhe a previsão local ou global como a previsão final. Esse mecanismo é uma tabela de contadores de previsão, indexado pelo histórico de caminhos, que dinamicamente escolhe entre a previsão global e local para cada chamada de desvio. O processador treina os contadores de previsão de escolha para preferir a previsão correta sempre que a previsão local difere da global. O mecanismo pode escolher diferentemente para cada invocação do mesmo desvio.

Ainda na figura acima, pode-se ver a utilidade do terceiro mecanismo. Esse pode escolher a previsão global para o desvio `(a == b)` sempre que os desvios `(a == 0)` e `(b == 0)` forem tomados, e pode selecionar a previsão local para o desvio `(a == b)` em outros casos. [Kessler 1999]

4.10 Gerenciamento de memória

O Alpha-21264 tem 128K de Cache L1 dos quais 64K são para dados e os 64K restantes são para instruções. O Cache L2 é opcional e totalmente controlado pelo chip. O 21264 suporta uma gama grande de dispositivos de memória Cache L2. A Compaq utiliza em sua DS20L 4MB de Cache L2 por CPU. Ela apresenta também duas vias de dados de 64 bits cada.

Ele apresenta também uma unidade de interface de barramento (*bus interface unit* - BIU). Ela recebe referências do sistema interno de memória, e responde com dados da Cache L2 ou do sistema (este em caso de falha).

A taxa de transferência da Cache L2 é de 16 bytes a cada ciclo e meio do processador, o que quer dizer 6.4 GiB/sec com 400MHz de taxa, e sua latência mínima é de 12 ciclos, com uma SRAM de 6 ciclos de latência.

O sistema de memória do 21264 suporta qualquer combinação de dois *loads* ou *stores* por ciclo sem conflito. Como explicado na seção de *pipeline*, o sistema tem duas filas, uma para *loads* e outra para *stores*. *Stores* são semelhantes aos *loads* mas não usam o vetor de cache até terminar sua execução. [Keller, Kessler 1999]

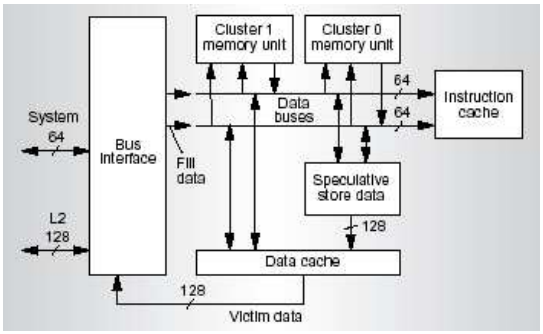

Figura 8: Datapath de memória do Alpha 21264

O sistema de memória interno também tem um arquivo de endereços de *miss* com oito entradas. Cada entrada indica um *miss* a um bloco de 64 *bytes* de cache

4.11 Características Físicas

Obteve-se características físicas do chip em si e de uma servidora que o utiliza: a Compaq Alpha DS20L.

A Compaq AlphaServer DS20L tem dois processadores Alpha 21264A de 833MHz, é possível montar 40 delas em um rack 1U, e pode ter até 2GB de memória com largura de banda de 2,7GB/s. É possível rodar TRU64 Unix ou Linux nessa máquina.

O chip é organizado de acordo com a figura abaixo. Percebe-se que boa parte do processador é dedicada ao tratamento de ponto flutuante. Há Cache L1 separado para Instruções e Dados, o Cache L2 é único para ambas. Há também uma caixa-preta de reordenação de instruções, já que o Alpha 21264 pode executá-las fora de ordem. Há dois bancos de registradores, espaço para tratamento de inteiros e um preditor de desvios.

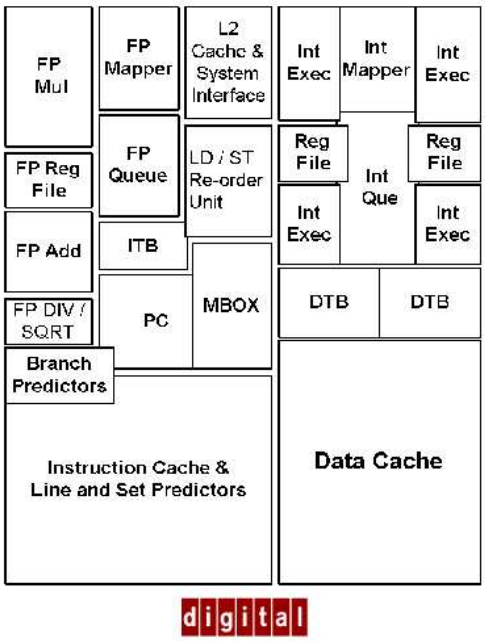

Figura 9: Planta do Alpha 21264

Segue uma tabela com dados técnicos do Alpha 21264A (o que é usado na Alpha DS20L):[HP 2000]

Dado	**Valor**
Suprimento de energia	Vss = 0,0V Vdd = 2,1V
Temperatura de operação	100°C máximo
Temperatura de armazenamento	-55°C a 125°C
Dissipação de energia	90W a 750MHz
Encapsulamento	PGA de 588 pinos
Número de transistores	15,2 milhões
Processo de fabricação	CMOS de seis camadas de metal com 0,25micron
Tamanho do *die*	aproximadamente 210mm^2
Performance (667MHz e 8Mb de Cache)	SPEC CPU2000: 444: SPECint2000, 577 SPECfp2000 SPEC CPU95 40,1: SPECint97, 83,6: SPECfp95

Nessa tabela, podemos ver que o 21264 dissipa 90W, o que é bastante comparado a outros processadores. Seu encapsulamento é feito com 588 pinos ligados por um *pin grid array*, seu processo de fabricação é com transistor CMOS de seis camadas de metal com 0,25 micrômetros de largura de porta (distância entre o dreno e a fonte). O *die* tem aproximadamente 210mm^2 nos quais se encontram 15,2 milhões de transistores.

5. CONSIDERAÇÕES FINAIS

A arquitetura Alpha é, como poucas, uma obra de arte da engenharia. Todo seu projeto foi manual, garantindo implementações fantásticas.

O desempenho dos processadores que dela faziam uso geralmente eram muito superiores a outros de mesmo *clock*. Contudo, o projeto manual, o acúmulo de funcionalidades em *hardware* e a grande quantidade de memória rápida fez com que o processador Alpha tornasse-se muito caro. O mercado de sistemas distribuídos, juntamente com o barateamento dos chips "domésticos", fez com que a Alpha perdesse território e que a COMPAQ descontinuasse sua fabricação.

Há um porquê de estudar essa arquitetura, mesmo depois de sua "aposentadoria". A quantidade de conhecimento acumulado em seu projeto é de um didatismo excelente para qualquer estudante de sistemas de computação. Simulá-la, em uma linguagem tal ArchC, ou realizar uma pesquisa tal como a feita para este artigo, firmam muitos conceitos de arquitetura de computadores. Afrora isso, suas implementações são exemplos de eficiência para implementações futuras, sendo que grande parte de seus engenheiros foram contratado pelas líderes de mercado Intel e AMD.

Finalmente, o fim anunciado da eficácia da lei de Moore traz de volta à tona muitas das idéias dessa brilhante arquitetura, já que não tem sido possível aumentar tão agressivamente o *clock* dos processadores. O aumento de desempenho, hoje em dia, deve vir acompanhado de melhorias de projeto, de um aumento de paralelismo, de uma cache mais efetiva. Em tudo isso, a Alpha tem coisas a nos ensinar.

6. REFERÊNCIAS BIBLIOGRÁFICAS

DANIELS, T.; DHARMESH, P.; ZIEGLER, M. *The Alpha 21264*. [S.l.], 1999.

DEC Alpha. http://en.wikipedia.org/wiki/DEC_Alpha. Artigo da Wikipedia.

GROUP, H. P. T. C. *Exploring Alpha Power for Technical Computing*. [S.l.], April 2000.

HALFHILL, T. R. Already there... alpha 21264 stakes off the claim - long before merced. *c't*, n. 15, p. 192, 1998.

HENNESY, J. L.; PATTERSON, D. A. *Computer Architecture: A Quantitative Approach*. 3rd. ed. San Francisco, CA: Morgan Kauffman, 2003.

HP. *HP Alpha Systems - AlphaServer comparison chart: AlphaServer DS series*. 2000. http://www.spec.org/osg/cpu2000/results/res2002q2/cpu2000-2002

KELLER, J. *The 21264: A Superscalar Alpha Processor with Out-of-order Execution*. Hudson, MA.

KESSLER, R. E. The alpha 21264 microprocessor. *IEEE MICRO*, v. 19, n. 0272-1732, p. 24–36, Março/Abril 1999.

PATTERSON, D. A.; HENNESY, J. L. *Computer Organization and Design: The Hardware/Software Interface*. 2nd. ed. San Francisco, CA: Morgan Kauffman, 1997.

PILLA, M. L. *Microprocessador Alpha 21264*. [S.l.], 1999.

SITES, R. L. Alpha axp architecture. *Digital Technical Journal*, v. 4, n. 4, Special Issue 1992. http://www.research.compaq.com/wrl/DECarchives/DTJ/DTJ801/DTJ8