Clusters

Leandro Rodrigues Magalhães de Marco
Instituto de Computação – Unicamp
RA 009089

Clusters

- Cluster: conjunto de computadores ligados em rede
- Cada computador é um nó
- Processar várias partes de um problema maior
- Garantir confiabilidade

Cluster

- "Os clientes inventaram os clusters a partir do momento em que não conseguiam colocar todo o trabalho em um simples computador, ou precisavam de um backup. A data do prmeiro Cluster é desconhecida, mas eu me surpreenderia se não fosse nos anos 60 ou no fim dos anos 50" – *In Search of Clusters – Greg Pfister*

Cluster

- Lei de Amdahl
- Diferença entre Cluster e multiprocessamento
- Cluster & redes (sempre ligados)
- Primeiro Cluster comercial: ARCNET, desenvolvido pela Datapoint
- Sucesso comercial: VaxCluster
- PVM: Clusters a partir de PCs

Cluster

- High-Availability (HA) Clusters
  - Aumentar disponiblidade de serviços oferecidos
  - Categorias:

    - Active/Active
    - Active/Passive
    - N+1
    - N+M

- Loading Balance Cluster
  - Front-ends distribuem a carga de trabalho

- High performance (HPC) Cluster
  - Dividir tarefas em inúmeros nós

- Grid Computing
  - Computadores não confiam uns nos outros
  - Tarefas independentes

Cluster

- **distcc:** compilação paralela quando se utiliza o GCC
- **MPI:** biblioteca que implementa um protocolo de comunicação entre processos rodando em diferentes nós de um Cluster.
- **Linux Virtual Server, Linux-HA:** clusters que implementam soluções de balanceamento de carga entre nós
- **MOSIX, openMosix, Kerrighed, OpenSSI**: cluster integrados ao Kernel do Sistema Operacional e permitem migração automática de processos entre nós homogêneos.
- **MSCS:** Solução da Microsoft para clusters de alta-disponibilidade em ambiente Windows.

Conclusão

- Clusters surgiram como uma solução natural para problemas que ocorriam devido há limitações das máquinas. Limitações estas que existem até hoje e muito provavelmente irão existir no futuro.
- A evolução dos Clusters anda de mãos dadas com a evolução das redes de computadores e da criação de padrões de comunicação. Talvez a maior revolução trazida pela tecnologia de Cluster, seja a possibilidade de montar supercomputadores com máquinas comuns.

Arquiteturas VLIW

Uma Alternativa Para a Exploração de ILP

Rafael Augusto Scaraficci (RA:009649)

rafael.scaraficci@students.ic.unicamp.br

Universidade Estadual de Campinas - UNICAMP

Instituto de Computação - IC

Roteiro

- História
- Fundamentos da Arquitetura VLIW
- Escalonamento Estático
- Vantagens e Desvantagens
- Perspectivas Futuras

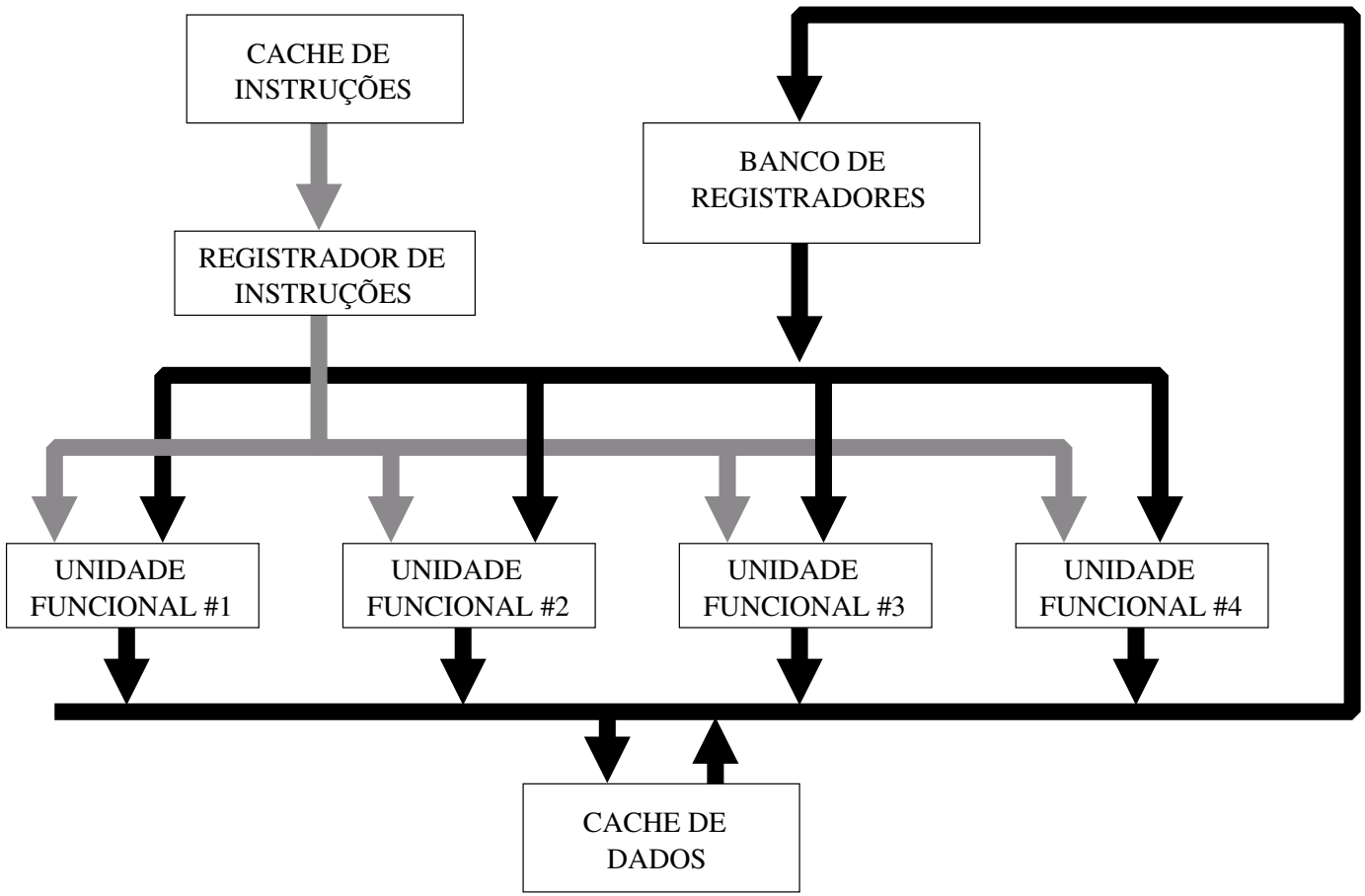
História

- Microcódigo Horizontal $\rightarrow$ VLIW (final de 70)
- *Trace Scheduling* - Joseph Fisher (81)
- Multiflow e Cydrome (84)
- VLIW - Software Dedicado (meados de 90)
- C6X da Texas Instruments, Crusoe da Transmeta, Intanium da Intel

Fundamentos do VLIW

CARACTERÍSTICAS	VLIW
Tamanho da Instrução	Tamanho único
Formato da Instrução	Regular, posição consistente dos campos
Semântica da Instrução	Várias operações simples e independentes
Registradores	Vários, propósito geral
Acesso à Memória	Arquitetura do tipo *load-store*
Foco do Projeto de Hardware	Simples, explora múltiplas unidades funcionais, lógica de despacho de baixa complexidade

Fundamentos do VLIW



Escalonamento Estático

- Essencial para um bom desempenho da arquitetura
- *Basic Block Scheduling*
  - Poucas Instruções Independentes (4 – 5)
  - Baixa Eficiência
- *Extended Basic Block Scheduling*
  - Ultrapassa a Barreira do Bloco Básico
  - Maior Número de Instruções Independentes
  - Maior Eficiência

Escalonamento Estático

- *Loop Unrolling*
- *Software Pipelinig*
- *Trace Scheduling*
- *SuperBlock Scheduling*
- *HyperBlock Scheduling*

Vantagens e Desvantagens

- Vantagens
  - Hardware Simples $\rightarrow$ Baixo Consumo de Energia
  - Despacho de Múltiplas Operações
  - Arquitetura Exposta ao Compilador
- Desvantagens
  - Compilador VLIW é Complicado
  - Não Roda Código De Outras Plataformas
  - $\downarrow$Op/Instr $\rightarrow$ Desperdício de Memória

Perspectivas Futuras

- Supercomputadores $\rightarrow$ DSPs, Processadores Dedicados
- Processadores de Propósito Geral (Sem Perspectiva)
- Áreas de Pesquisa
  - Compiladores
  - Arquitetura

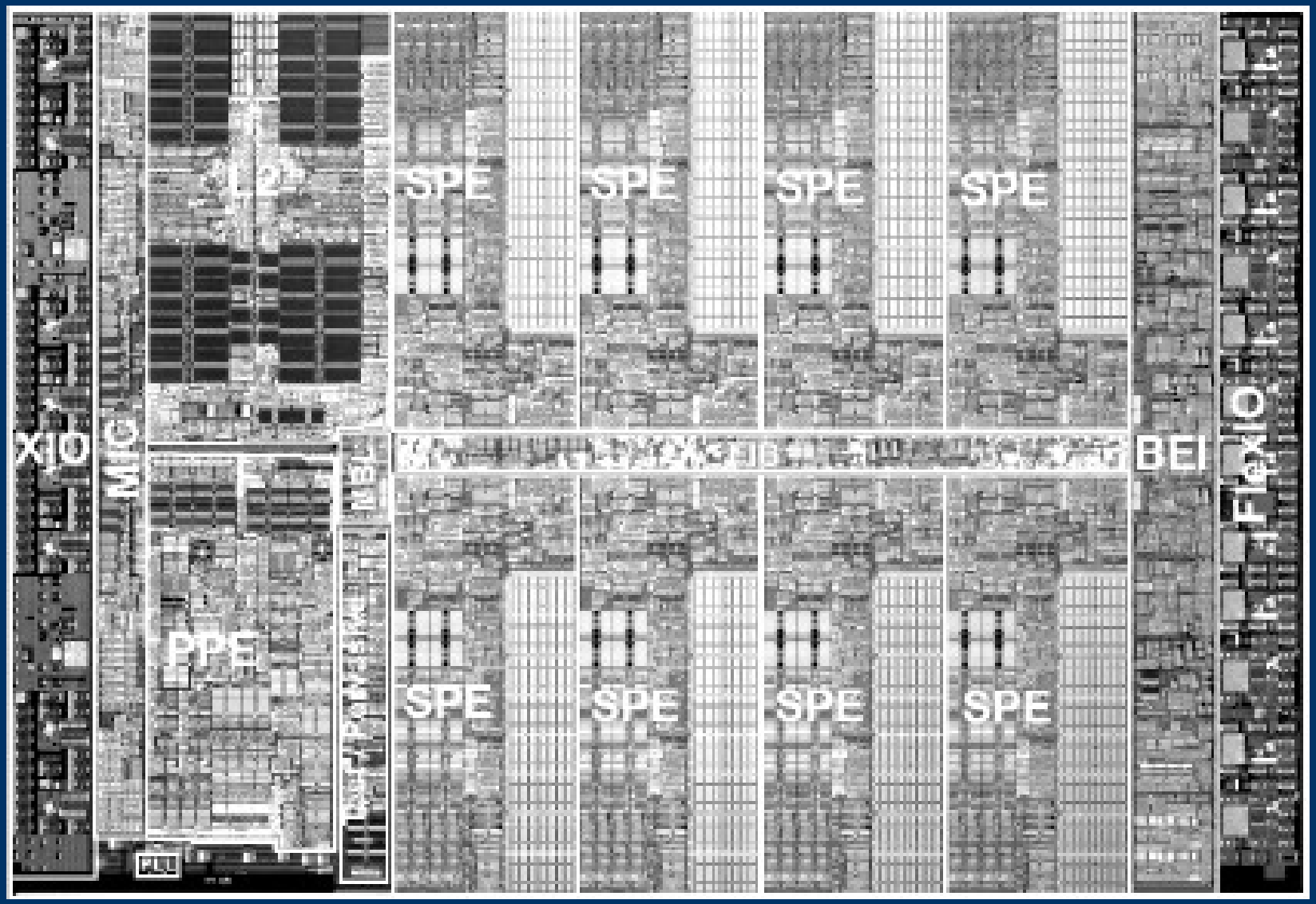
A arquitetura Cell

Douglas José Soares Rodrigues

Organização

- Power Processor Element (PPE)
  - Processador principal baseado na arquitetura POWER
  - Controlador das SPEs
  - Cache L2
- Synergistic Processing Elements (SPE)
  - Responsável por tratar workload
  - Possui uma Local Store, ao invés de cache.
- Element Interconnect Bus (EIB)
  - Conecta PPE + SPE + I/O
  - I/O: 2 interfaces Rambus independentes

Die do Cell



Comparativo Cell x Emotion Engine

	Sony Emotion Engine	Cell Processor
CPU Core ISA	MIP64	64-bit Power Architecture
Core Issue Rate	Dual	Dual
Core Frequency	300MHz	~4GHz (est.)
Core Pipeline	6 stages	21 stages
Core L1 Cache	16KB I-Cache + 8KB D-Cache	32KB I-Cache + 32KB D-Cache
Core Additional Memory	16KB scratch	512KB L2
Vector Units	2	8
Vector Registers (#, width)	32, 128-bit + 16, 16-bit	128, 128-bit
Vector Local Memory	4K/16KB I-Cache + 4K/16KB D-Cache	256KB unified
Memory Bandwidth	3.2GB/s peak	25.6GB/s peak (est.)
Total Chip Peak FLOPS	6.2GFLOPS	256GFLOPS
Transistor Count	10.5 million	235 million
Power	15W @ 1.8V	~80W (est.)
Die Size	240mm^2	235mm^2
Process	250nm, 4LM	90nm, 8LM + LI

A Arquitetura x86-64 e o Processador Athlon 64

João Paulo Porto
016377

“O x86 não é complicado. Ele só não faz sentido.”

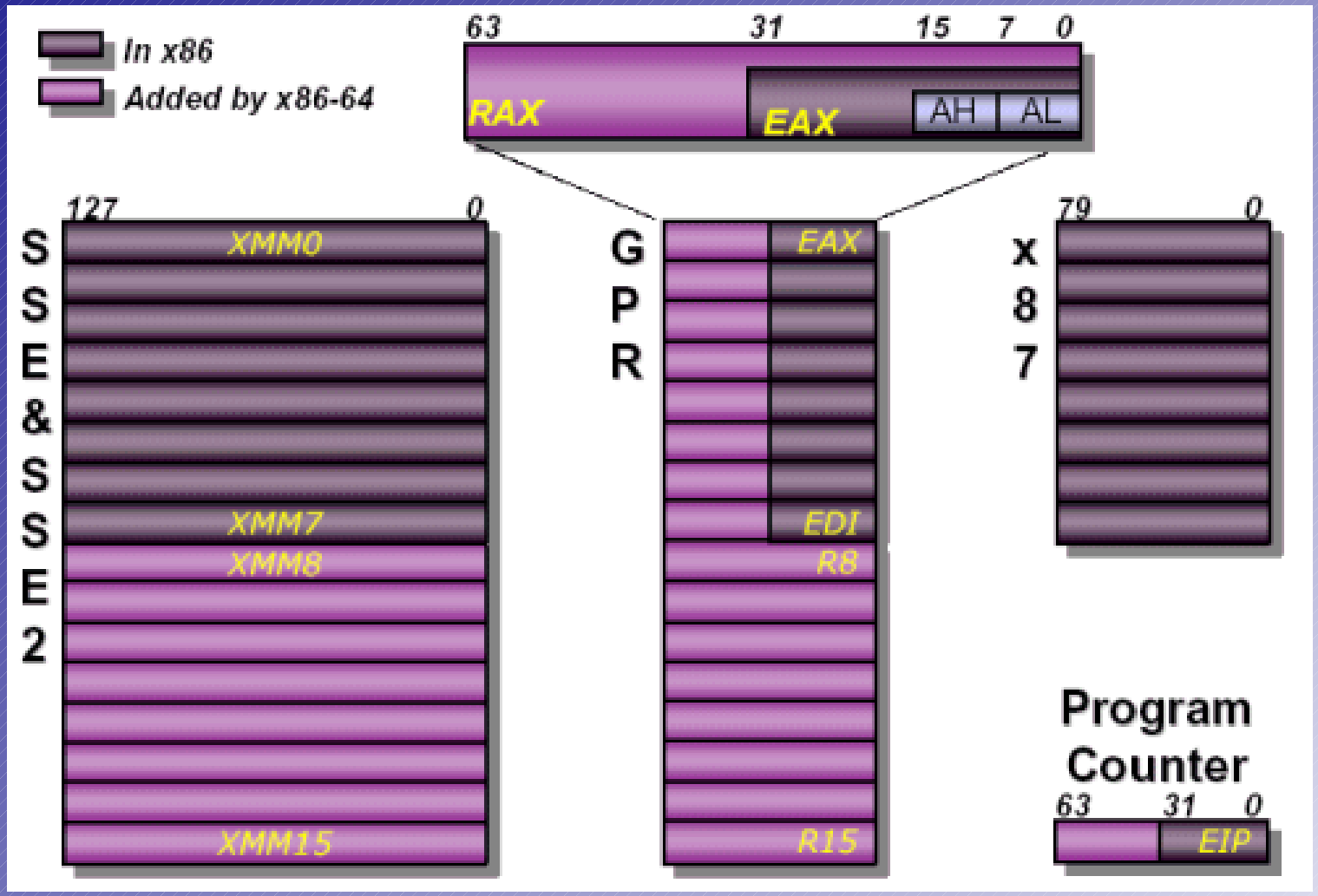
A Arquitetura

- História
  - Início
  - Meio
  - Fim?

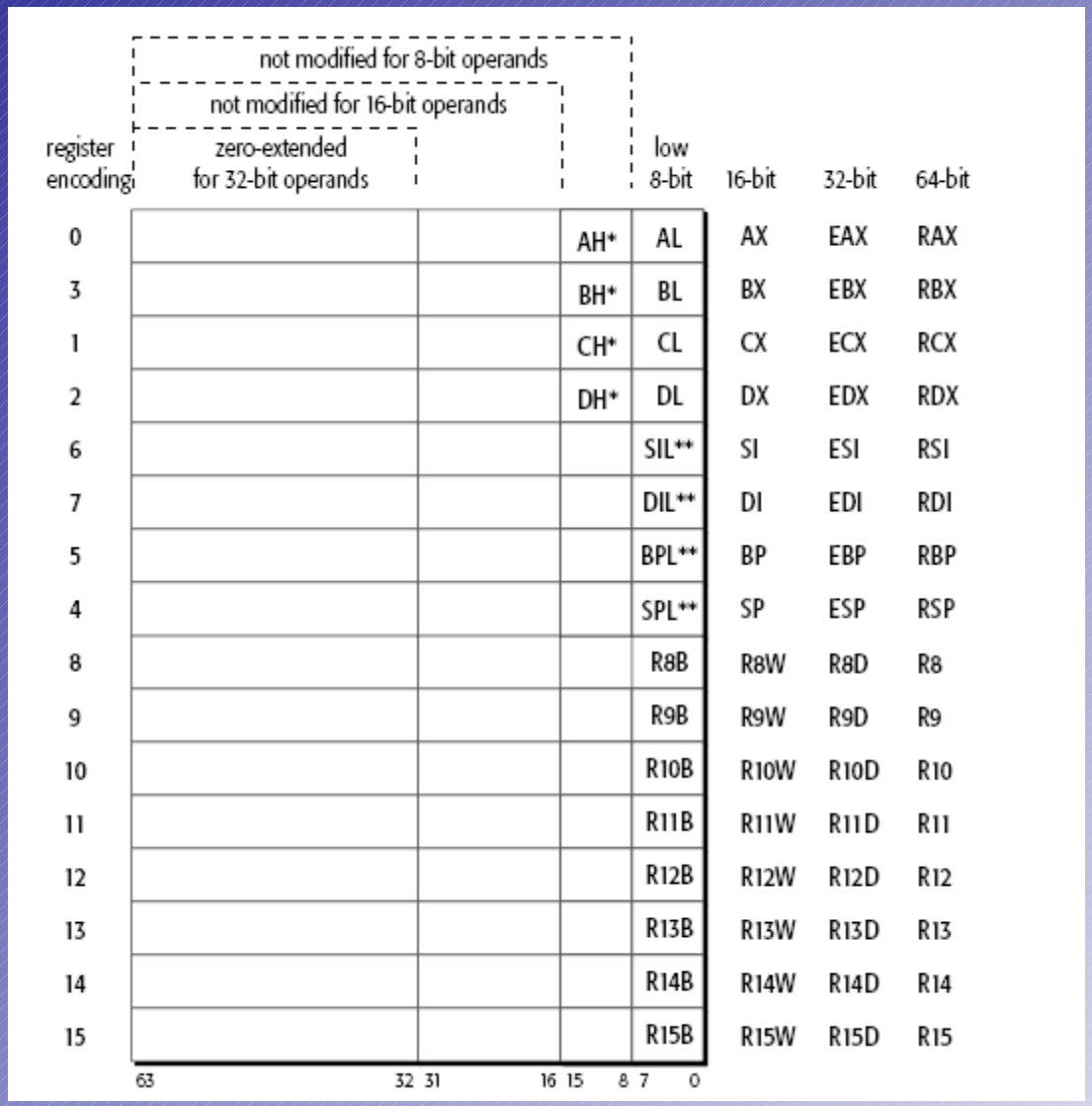
A arquitetura

- Melhorias
  - *Long Mode*
  - Registradores!!!
  - Memória Virtual
  - Cache?

A Arquitetura - Registradores



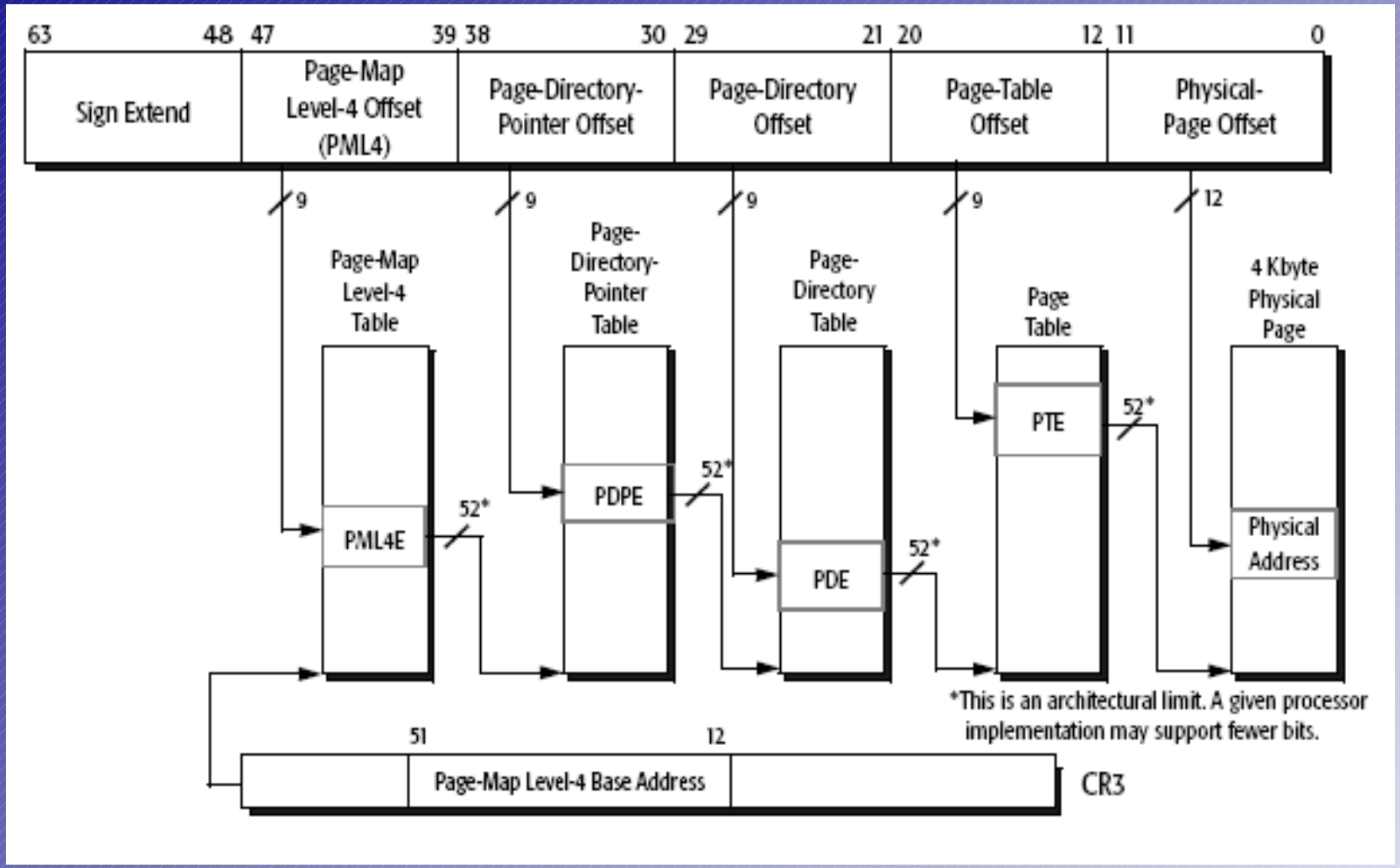
A Arquitetura - Registradores



A Arquitetura – Memória Virtual

- Endereços virtuais de 52 bits
  - O que fazer para 64 bits?
- Endereços físicos de 48 bits.

A Arquitetura – Memória Virtual



Eu Quero 64 Bits!!!!!!

- 1 => real mode (praticamente um 8086!)
- 2 => protected mode (já é um 286!)
- 3 => long mode inicializado (um Athlon com algo a mais)
- 4 => long mode habilitado e ativo (ufa!)
- ... e meu joguinho de DOS?

Modos de Operação - Resumo

Operating Mode		Operating System Required	Application Recompile Required	Defaults		Register Extensions	Typical GPR Width (bits)
				Address Size (bits)	Operand Size (bits)		
Long Mode	64-Bit Mode	New 64-bit OS	yes	64	32	yes	64
	Compatibility Mode		no	32		no	32
				16	16		16
Legacy Mode	Protected Mode	Legacy 32-bit OS	no	32	32	no	32
				16	16		
	Virtual-8086 Mode			16	16		16
	Real Mode	Legacy 16-bit OS					

AMD 64 Athlon

A Arquitetura – Onde Está a Cache?

- Nenhuma imposição *arquitetural!*
- Vários tipos de memória
- Coerência => MOESI

A Arquitetura – Tipos de Memória

Memory Access Allowed		Memory Type				
		UC/CD	WC	WP	WT	WB
Read	Out-of-Order	no	yes	yes	yes	yes
	Speculative	no	yes	yes	yes	yes
	Reorder Before Write	no	yes	yes	yes	yes
Write	Out-of-Order	no	yes	no	no	no
	Speculative	no	no	no	no	no
	Buffering	no	yes	yes	yes	yes
	Combining[1]	no	yes	no	yes	yes

Note:

1. *Write-combining buffers are separate from write buffers.*

AMD 64 Athlon

Cade o Processador??

- Falar é fácil, mas...
  - Onde está a implementação????

O Processador Athlon 64

- Implementa a arquitetura
- Implementa algo a mais
- Parece CISC mas...

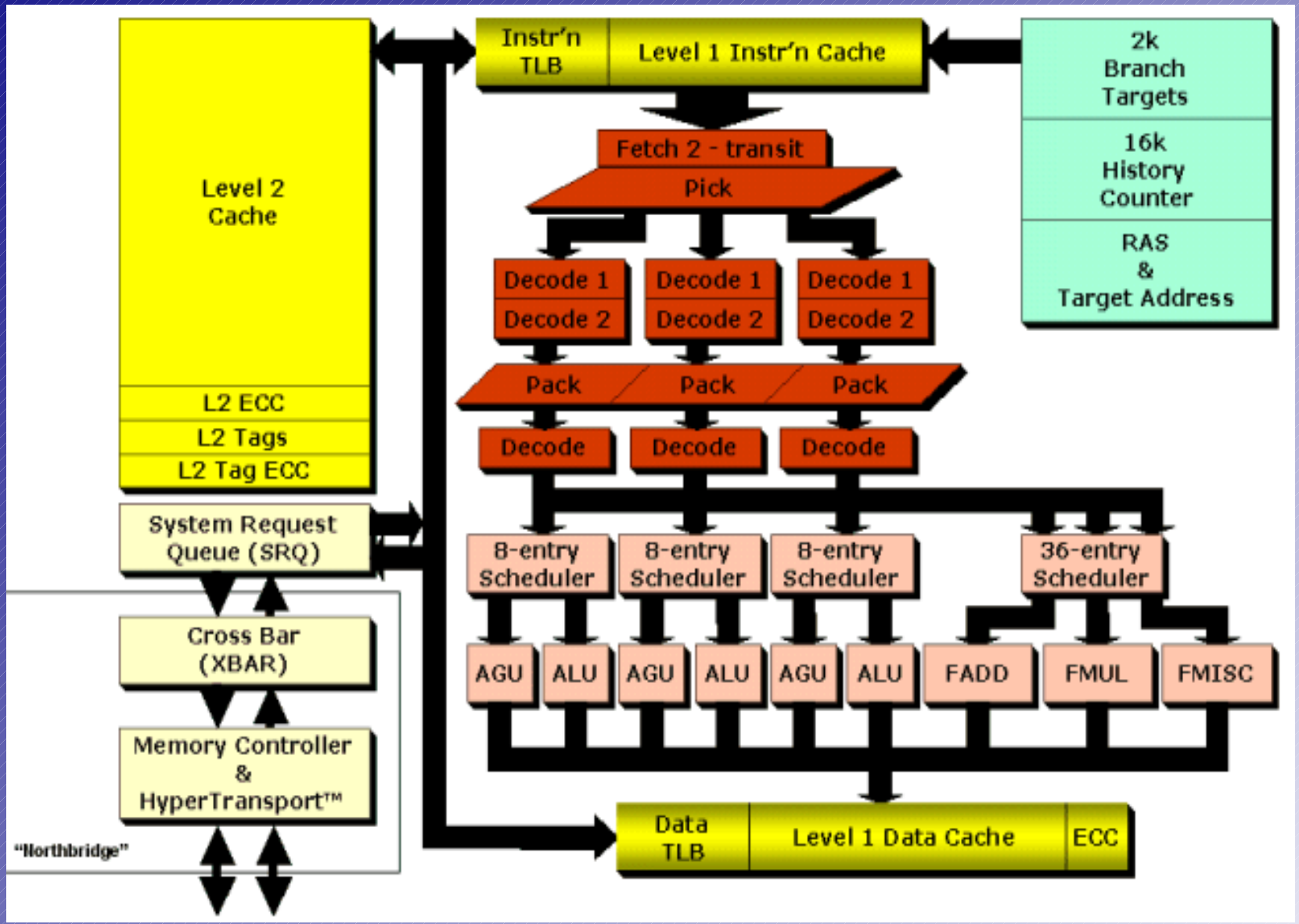
O Processador Athlon 64

- ... RISC É!
  - Casca de tradução
- Na verdade, o Athlon 64 é
  - RISC
  - Superescalar
  - Fora de ordem

O Processador Athlon 64

- HyperTransport
- Cool'n'Quiet
- Controlador de memória externo... pra quê?

O Processador Athlon 64 - Datapath



O Processador Athlon 64 - Pipeline

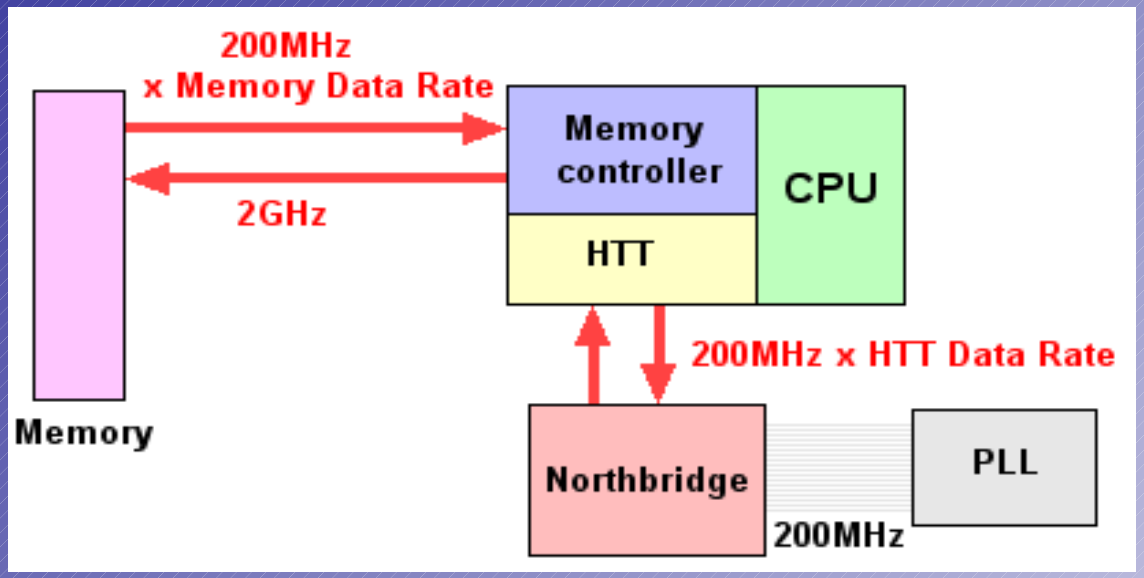
- Três grandes estágios
  - Fetch / Decode
  - Execute
  - Load / Store
- Dois Pipelines independentes

O Processador Athlon 64 - Pipeline

ULA	FPU
Fetch 1	
Fetch 2	
Pick	
Decode 1	
Decode 2	
Pack	
Pack / Decode	
Dispatch	Dispatch
Schedule	Stack Rename
Exec	Register Rename
Data Cache 1	Write Schedule
Data Cache 2	Schedule
	Register Read
	FX0
	FX1
	FX2
	FX3

O Processador Athlon 64

- Controlador de memória integrado



O Processador Athlon 64 – Cache

- Não precisa, mas tem!
  - L1I e L1D
  - L2
  - Memória
- L1 com L2: relação exclusiva

O processador Athlon 64

	L1	L2
Tamanho	Código: 64KB Dados: 64KB	512KB(NewCastle) 1024KB(Hammer)
Associatividade	Código: 2 way Dados: 2 way	16 way
Tamanho da Linha	Código: 64 bytes Dados: 64 bytes	64 bytes
Política de Escrita	*Write Back*	N/D
Latência	3 ciclos	N/D
Largura do Barramento	N/D	128 bits
Relação com L1		Exclusiva

Referências

•Ramos, F., Sogumo F., e Silva, F., Arquitetura AMD 64. Em *Trabalho de Graduação da Disciplina mc722*, obtido 22/06/2006 em http://www.ic.unicamp.br/~rodolfo/Cursos/mc722/2s2004/g07-amd64-texto.pdf

•Vários. Intel 8086. Em *http://en.wikipedia.org/wiki/8086*, acessado em 22/06/2006

•Delattre, F.,. CPUID.COM – AMD K8 Architecture. Em *http://www.cpuid.com/reviews/K8/index.php* acessado em 22/06/2006

•Vários. *AMD64 Architecture Programmer's Manual Volume 1: Application Programming*.

•Vários. *AMD64 Architecture Programmer's Manual Volume 2: System Programming*.

•Vários. *AMD64 Architecture Programmer's Manual Volume 3: General Purpose and System Instructions*

Introdução à Computação Quântica

Tony Minoru Tamura Lopes - ra017502

Instituto de Computação
Universidade Estadual de Campinas

MO401 - Julho de 2006

Sumário

Motivação

Necessidade de processadores mais rápidos

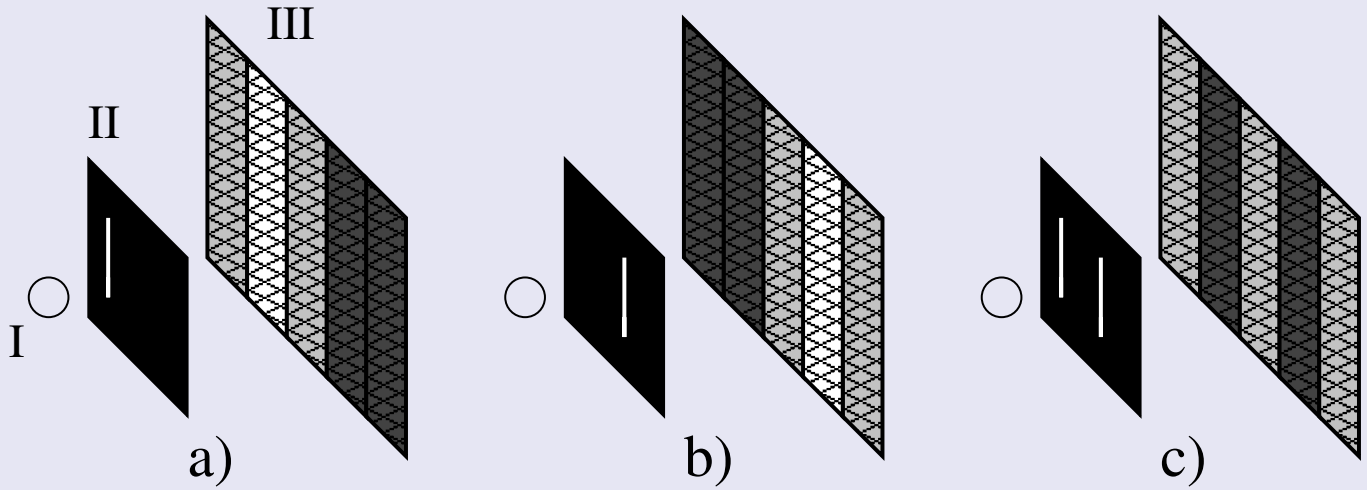
- Miniaturização dos chips
- Limite teórico em 2020
- Poucos átomos para representar um bit
- Necessidade da Quântica

Revolução nas Bases da Computação

- Nova forma de básica de informação
- Novos limitantes inferiores
- Novo paradigma de Computação

Fenômenos Quânticos

Experimento das Duas Fendas



- **Interferência**
- Incerteza de Heisenberg
- Probabilidades

Fenômenos Quânticos

Estado Quântico

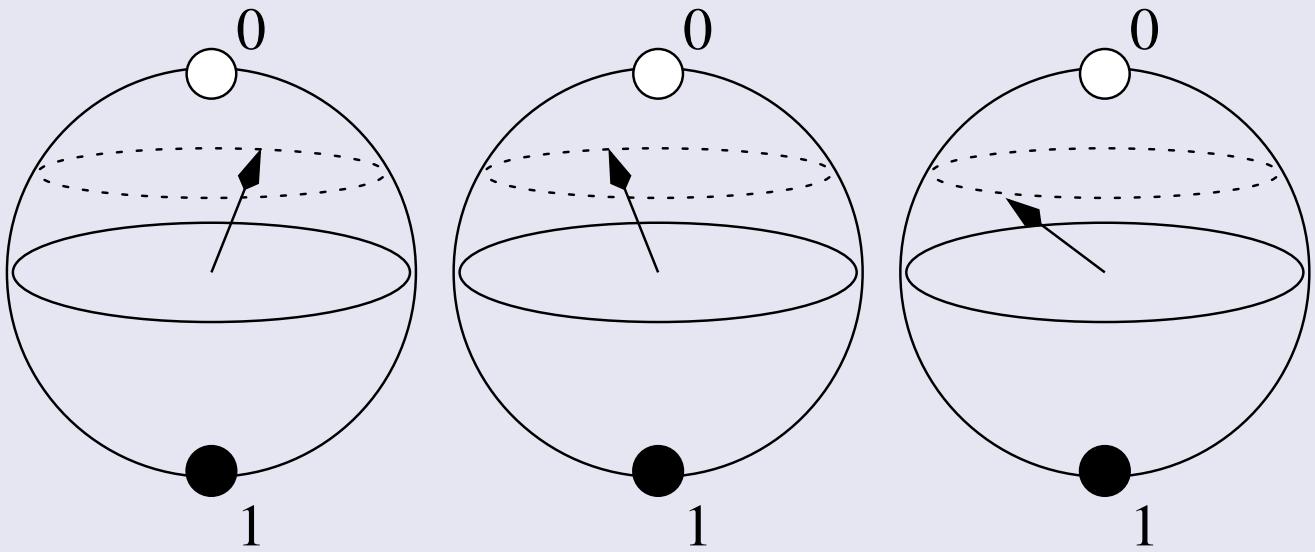
Passagem pela fenda é um estado quântico $|\psi\rangle$!

$$|\psi\rangle = c_0|\text{passa pela fenda } 1\rangle + c_1|\text{passa pela fenda } 2\rangle$$

- *Espaço vetorial complexo de dimensão finita* (Hilbert)
- $c_0 = c_1 = \frac{1}{\sqrt{2}}$
- $|c_0|^2 + |c_1|^2 = 1$
- **Superposição**

Qubit

Qubit



- Qualquer sistema quântico com dois estados

Qubit

Representação do Qubit

- $|Q\rangle = \alpha|0\rangle + \beta|1\rangle$
- $\alpha, \beta \in \mathbb{C}$
- $|\alpha|^2 + |\beta|^2 = 1$
- Fases diferentes, mesma probabilidade.

Transformação do Qubit

- Funcionamento básico
- $In = (\alpha\ \beta)^T$
- $Out = XIn = (\alpha'\ \beta')^T$
- Transformação X é unitária

Múltiplos Qubits

Registrador Quântico

- $|\psi\rangle = c_0 |0\rangle|0\rangle|0\rangle + c_1 |0\rangle|0\rangle|1\rangle + c_2 |0\rangle|1\rangle|0\rangle + \cdots + c_7 |1\rangle|1\rangle|1\rangle$
- 2^n valores ao mesmo tempo!
- 2^n operações!
- **Paralelismo Quântico**

Estados Emaranhados

- $|Q\rangle = \frac{1}{\sqrt{2}} (|0\rangle_1|1\rangle_2 + |1\rangle_1|0\rangle_2)$
- **Não-Localidade**

Necessidades

Necessidades

- Reversibilidade
- Portas lógicas universais

Portas Quânticas

Rotação de 1-Qubit

$$U_{\sqrt{\text{NOT}}}|0\rangle = \left(\frac{1}{2} + \frac{i}{2}\right)|0\rangle + \left(\frac{1}{2} - \frac{i}{2}\right)|1\rangle$$
$$U_{\sqrt{\text{NOT}}}|1\rangle = \left(\frac{1}{2} - \frac{i}{2}\right)|0\rangle + \left(\frac{1}{2} + \frac{i}{2}\right)|1\rangle$$

E para a operação NOT:

$$\text{NOT}|0\rangle = U_{\sqrt{\text{NOT}}}U_{\sqrt{\text{NOT}}}|0\rangle = |1\rangle$$
$$\text{NOT}|1\rangle = U_{\sqrt{\text{NOT}}}U_{\sqrt{\text{NOT}}}|1\rangle = |0\rangle$$

Portas Quânticas

NOT-Controlado

$$U_{\mathrm{CN}}|00\rangle = |00\rangle$$
$$U_{\mathrm{CN}}|01\rangle = |01\rangle$$
$$U_{\mathrm{CN}}|10\rangle = |11\rangle$$
$$U_{\mathrm{CN}}|11\rangle = |10\rangle$$

- Lógica condicional
- Produz estados *emaranhados*

Fatoração de Inteiros

- Segurança em Comunicações
- Intratabilidade é conjectura
- Melhor algoritmo clássico: 193 dígitos
- 2000 dígitos?

Algoritmo de Shor

Período de Função

- $f_n(a) = x^a \bmod n$, $mdc(x, n) = 1$
- $f_n(x + r) = f(x)$ e r é o período da função
- $\mathrm{mdc}(x^{r/2} - 1, n)$ quase sempre um fator não-trivial

Exemplo

- $n = 15$ e $x = 8$
- $f_{15}(a) = 8^a \ mod\ 15$ para $a = \{0, 1, 2 \dots\}$
- $\{1, 8, 4, 2, 1, 8, 4, 2, 1, \dots\}$
- $f_{15}(a + 4) = f(a)$, portanto, $r = 4$
- $\mathrm{mdc}(8^{4/2} - 1, 15) = \mathrm{mdc}(63, 15) = 3$
- 5 e 3 são os fatores.

Algoritmo de Shor

Obtendo r

- Obter r é $O(2^L)$ classicamente

$$|f_n\rangle = \frac{1}{\sqrt{n}} \sum_{x=0}^{n-1} |x\rangle |f_n(x)\rangle$$

- $|x\rangle$ emaranhado a $|f_n(x)\rangle$
- Ao medir $|f_n(x)\rangle$ temos y_0
- $|x\rangle$ se torna superposição de x tais que $f_n(x) = y_0$

Algoritmo de Shor

Obtendo r

$$|\psi\rangle = \frac{1}{\sqrt{k}} \sum_{p=0}^{k-1} |x_0 + p.r\rangle$$

$$\mathcal{F}|\psi\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{2\pi i \frac{x_0 j}{r}} |j\frac{n}{r}\rangle$$

- FFT é polinomial em Quântica
- Extrai-se r após no máximo $O(L)$ medições de $c = j\frac{n}{r}$

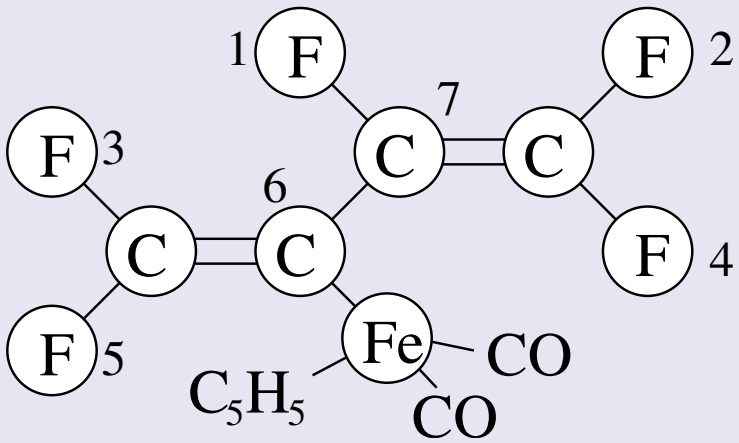
Estado Atual

Criptografia Quântica

- Canal seguro de comunicação é viável, **Não-Clonabilidade**

Computação Quântica

- Implementações de Algoritmos
- Ressonância Magnética de Núcleo



Conclusões

Conclusões

- Área ampla
- Muitos fenômenos "bizarros"
- Necessidade de Estabilidade
- Computação Quântica ainda distante

MO401
IC - Unicamp

Trace Scheduling

Márcio Paixão Dantas
Junho, 2006

Roteiro

1. Introdução e Motivação
2. Trace Scheduling (TS)
3. Variações do TS
    1. Tree Compaction
    2. Singly Rooted Directed Acyclic Graph (SRDAG)
    3. Improved Trace Scheduling Compaction (ITSC)
4. Comparação entre algoritmos
5. Conclusão

Introdução

- Problema:
  - Otimização de microcódigo
- Importância:
  - Aumento do código --> dificuldade de identificar oportunidades de otimização
  - Desenvolvimento de Compiladores

--> Técnicas que produzam código eficiente

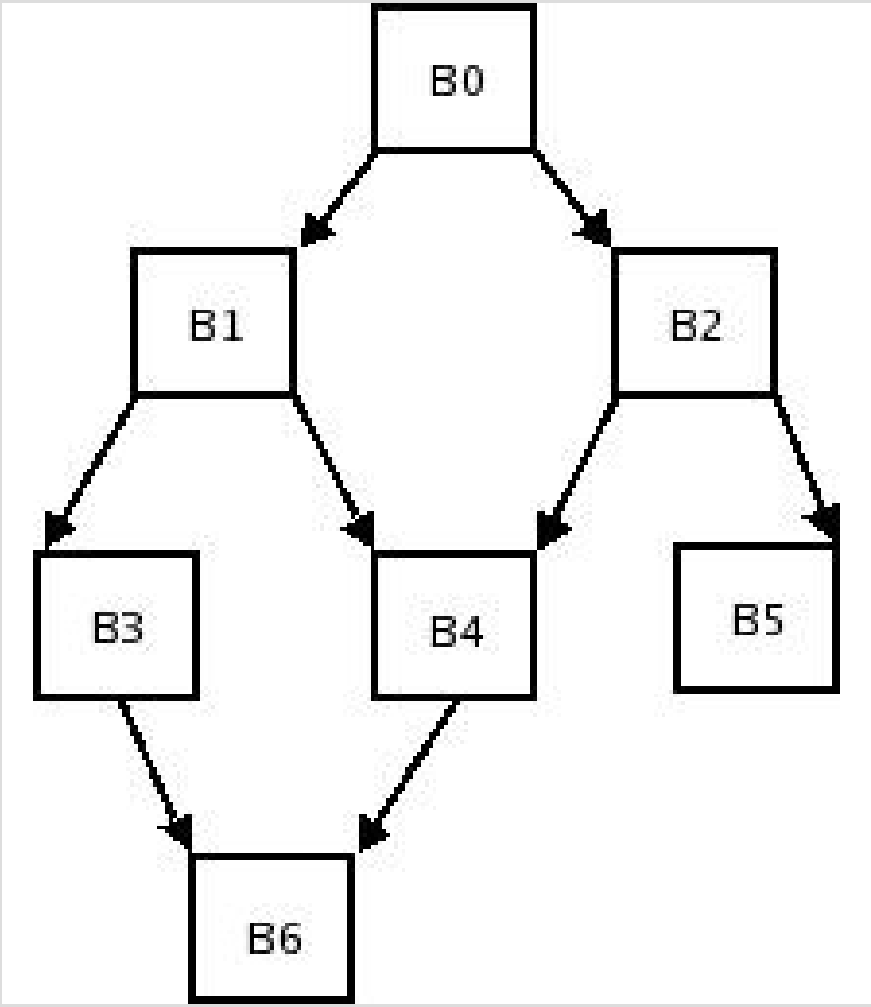
Introdução

Historicamente:

- Compactação Local
  - List Scheduling

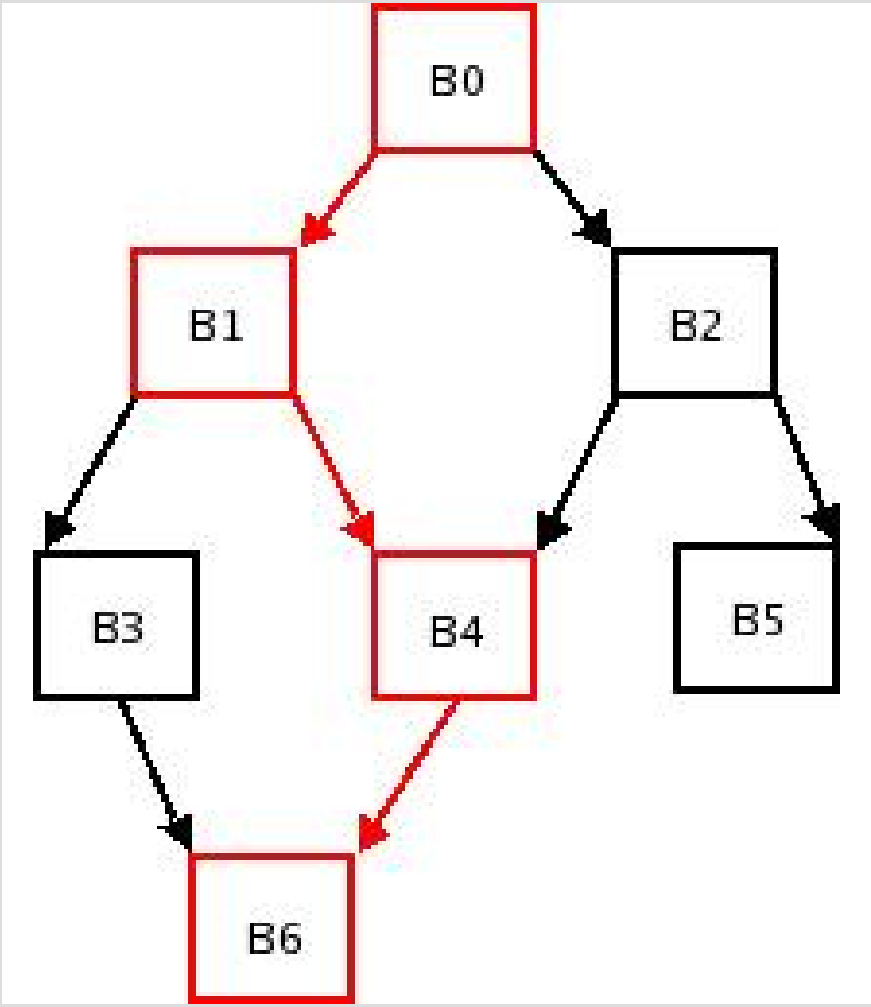
- Compactação global
  - Comp. Local + Movimentação entre blocos
  - Trace Scheduling (Fisher, 1981)

Trace Scheduling (TS)



Grafo de Fluxo de P

Trace Scheduling (TS)



Grafo de Fluxo de P

Funcionamento Básico:

- Escolha de *Traces* (caminhos), T, com maior probabilidade de serem executados
- Compactação de T
- *Bookkeeping*

Trace Scheduling (TS)

- Compactação de T
  - Definição de relação de precedência entre operações;
  - Construção de um DAG de precedência de dados;
  - Imposição das movimentações legais de MOP's no DAG;
  - Compactação de T como se fosse um bloco básico por *List Scheduling.*

Trace Scheduling (TS)

- Vantagens
  - Tempo de execução bem melhor que abordagens anteriores

- Problemas
  1. Cresc. tamanho de P pode ser exponencial;
  2. Estrutura do código otimizado pode ser bastante diferente da estrutura original;
  3. Desempenho ruim quando caminhos apresentam ciclos.

Tree Compaction

- Objetivo
  - Amenizar problemas 1 e 2.
- Idéia básica:
  - Realizar mesmas compactações feitas por TS, exceto aquelas que implicam cópia de blocos.

Tree Compaction

- Definições
  - *Top tree*
    - Árvore com um único nó com grau de entrada zero (raiz da árvore).
  - *Bottom Tree*
    - Árvore com um único nó com grau de saída zero (raiz da árvore).

Tree Compaction

Algoritmo

1. Particionar blocos de P em subconjuntos de *top trees*;
2. Aplicar *Trace Scheduling* em cada árvore;
3. Particionar blocos em *bottom trees*;
4. Fazer 2 até que todos os blocos estejam compactados.

Tree Compaction

- Vantagens
  - No pior caso, crescimento no tamanho do programa é da ordem de n^2, onde n é o número de desvios condicionais.
  - Complexidade computacional menor que TS;
  - Facilidade de implementação.

- Desvantagem
  - Tempo de execução pior que TS.

SRDAG

- Objetivos
  - Aumentar contexto de informação;
  - Escolher corretamente caminhos com maior probabilidade.

- Idéia básica
  - Usar um SRDAG para compactar cada bloco de P, ao invés de usar e compactar caminhos inteiros.

SRDAG

Algoritmo

Enquanto houver bloco não-compactado:

1. Seleciona raiz;
2. Seleciona SRDAG;
3. Compacta raiz;
4. Atualiza probabilidades e DAG original.

SRDAG

- Vantagens
  - Melhora tempo de execução e uso de memória em relação ao TS.
- Desvantagem
  - Maior complexidade computacional

ITSC

- Objetivos:
  - Reduzir espaço requerido;
  - Reduzir tempo de execução, sobretudo em caminhos contendo ciclos.

- Usa:
  - Conjunto modificado de regras de movimentação de MOP's entre blocos;
  - Algoritmo de TS melhorado;
  - Algoritmo de URCR (*unrolling, compacting, rerolling*).

ITSC

- Vantagens
  - Melhora no tempo de execução e tamanho do código.
- Desvantagem
  - Complexidade computacional um pouco maior maior que TS.

Comparação entre Algoritmos

B. Su and S. Ding. Some experiments in global microcode compaction. In MICRO 18: Proceedings of the 18th annual workshop on Microprogramming, pages 175–180. ACM Press, 1985.

- Arquiteturas testadas: PUMA e VAX.
- Em média
  - Tree Compaction --> piores tempos;
  - ITSC e SRDAG --> desempenho muito parecido;
- Todos os métodos globais produziram melhores resultados que compactação local.
- Nenhum método superou as otimizações manuais.

Conclusão

- Superioridade de métodos globais de otimização evidenciada.
- Técnicas baseadas em TS podem alcançar resultados próximos ou até mesmo iguais aos de compactação manual.
- Escolha de qual técnica de TS a ser utilizada depende bastante da arquitetura- alvo.

Conclusão

- Superioridade de métodos globais de otimização evidenciada.
- Técnicas baseadas em TS podem alcançar resultados próximos ou até mesmo iguais aos de compactação manual.
- Escolha de qual técnica de TS a ser utilizada depende bastante da arquitetura- alvo.
- Perguntas??

Arquitetura de Computadores

MO401 – Prof. Dr. Paulo Cesar Centoducatte

Execução Especulativa

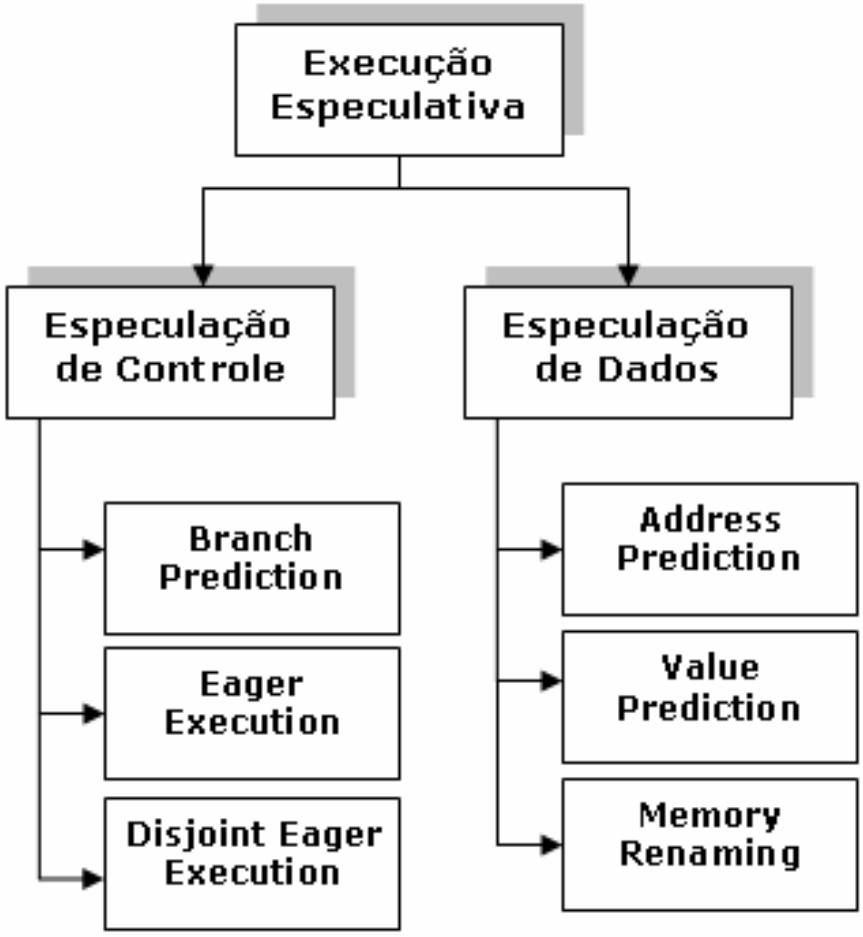
Daniel Carlos Guimarães Pedronette

RA: 050269 – daniel.pedronette@students.ic.unicamp.br

Execução Especulativa

- **Objetivos:**
  - **Aumentar grau de Paralelismo a Nível de Instrução**
  - **Evitar a paralisação do Pipeline**
- **Reduzir efeitos:**
  - **Dependência de Controle**
  - **Dependência de Dados**

Classificação e Taxonomia



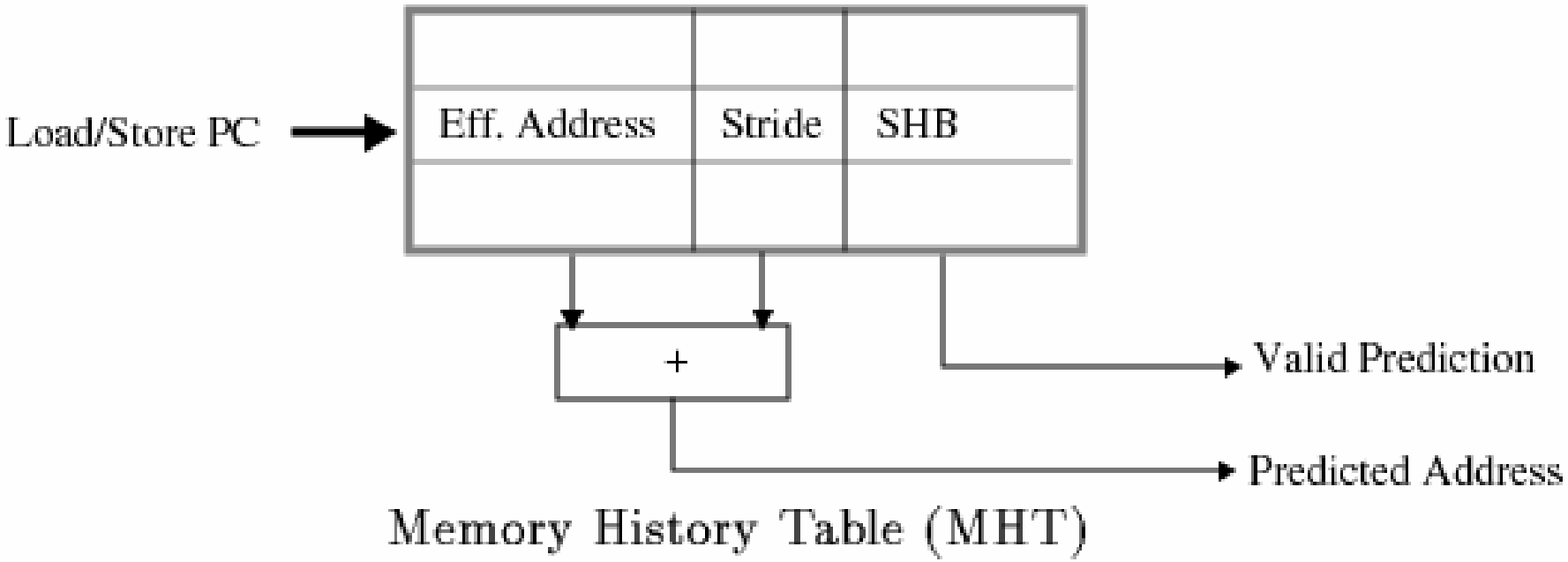
Especulação de Dados

- **Address Prediction:** previsão da posição de memória em que dados ou instruções estão armazenados.
- **Value Prediction:** previsão do valor que está armazenado em um registrador ou em um endereço de memória.
- **Memory Renaming:** ‘encaminhamento’ de valores previamente armazenados para instruções *loads*.

Address Prediction

- Cálculo do Endereço Efetivo para execução da instrução *load*
- Procura explorar conceito da localidade
- Endereços costumam seguir uma progressão aritmética
- Utilização de uma tabela: Memory History Table (MHT), contendo endereço anterior, passo e contador.

Address Prediction



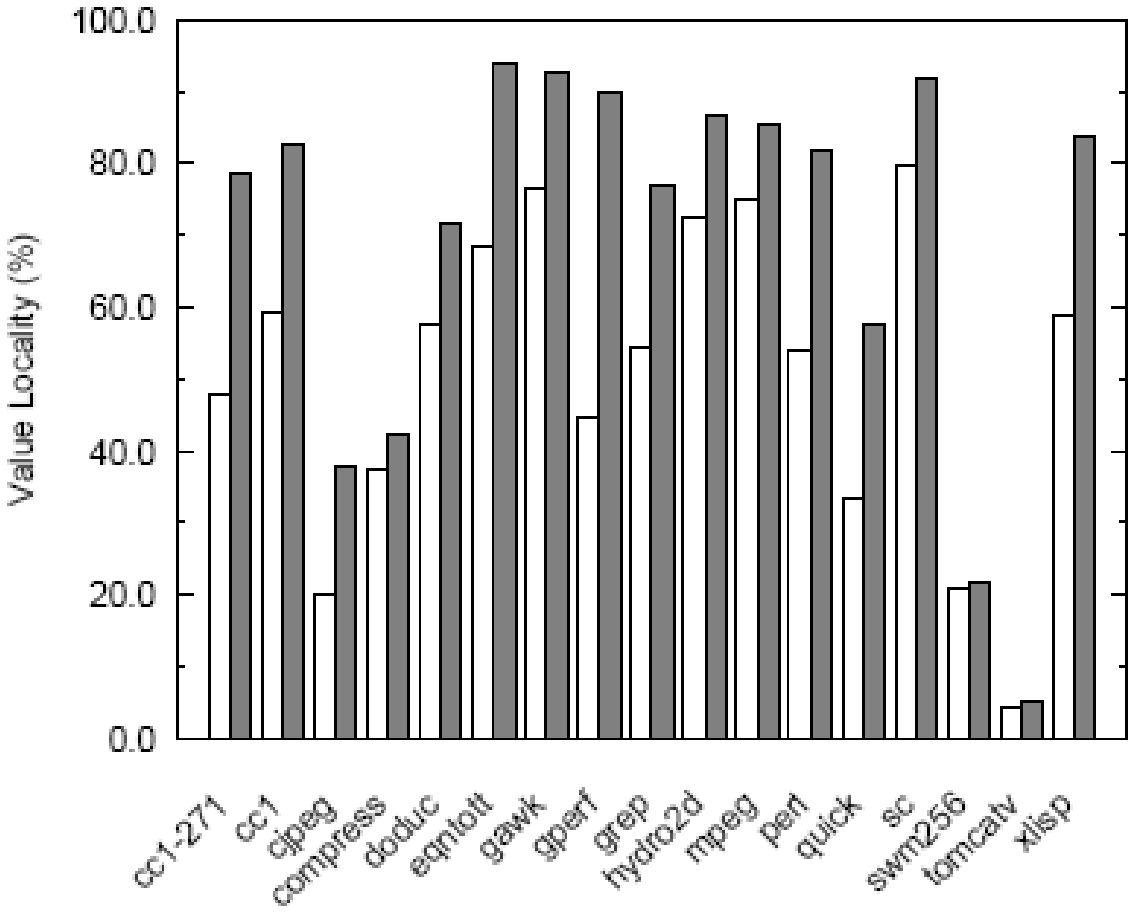
Address Prediction

- Instruções executadas especulativamente devem ser verificadas.
- Em caso de acertos, o contador da MHT é incrementado e o endereço atualizado.
- Em caso de erro, o contador é decrementado e endereço e passo são modificados para que possam refletir a nova realidade.

Value Prediction

- Motivação
  - Localidade temporal
  - Muitas vezes valor lido já foi obtido em instruções anteriores
- Causas:
  - Redundância de dados, como ocorre em matrizes esparsas.
  - Constantes de programa

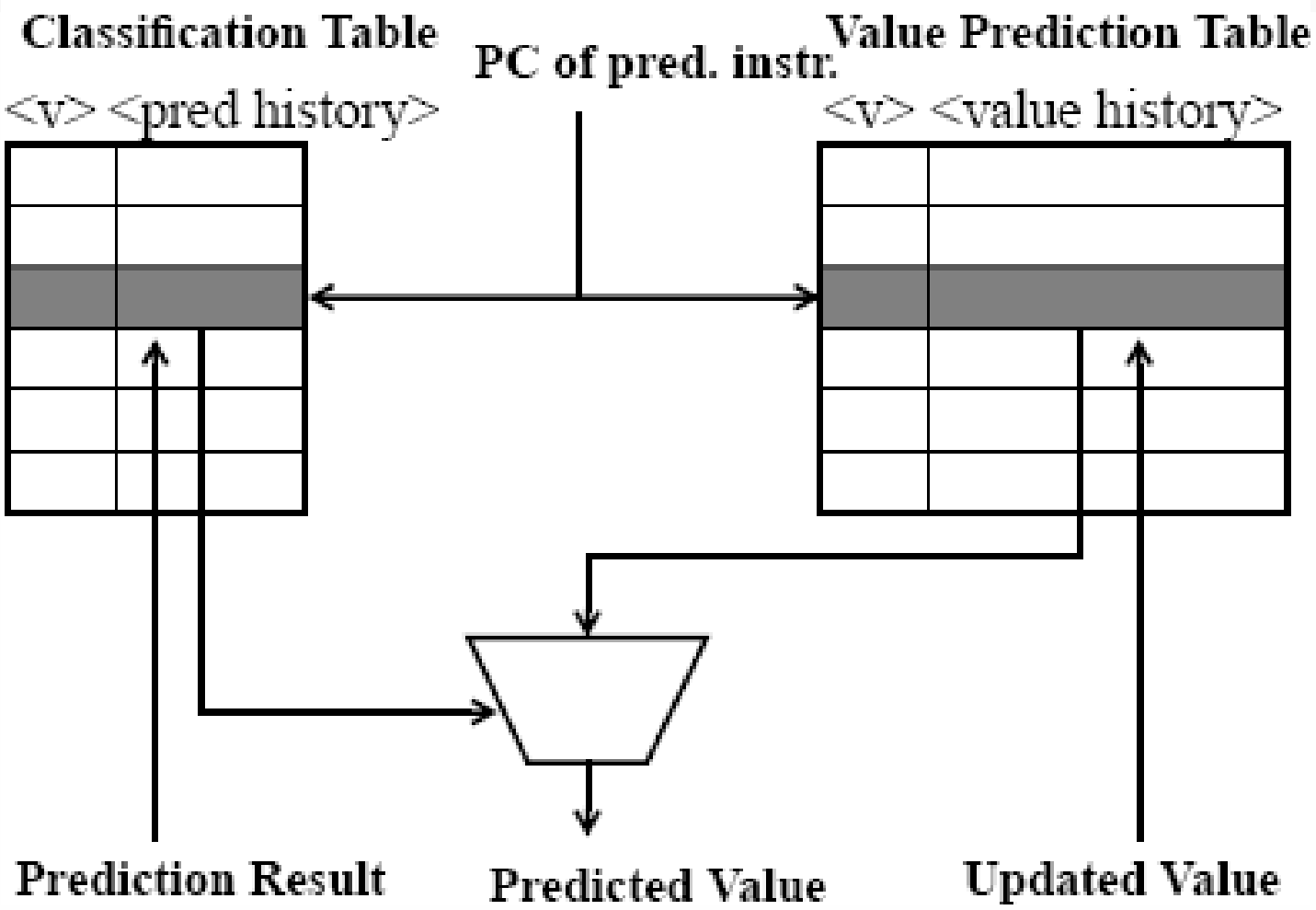
Value Speculation



Value Speculation

- A unidade de predição de valores é composta por duas tabelas de:
  - Predição
  - Verificação
- Indexadas através dos últimos bits de endereço da respectiva instrução

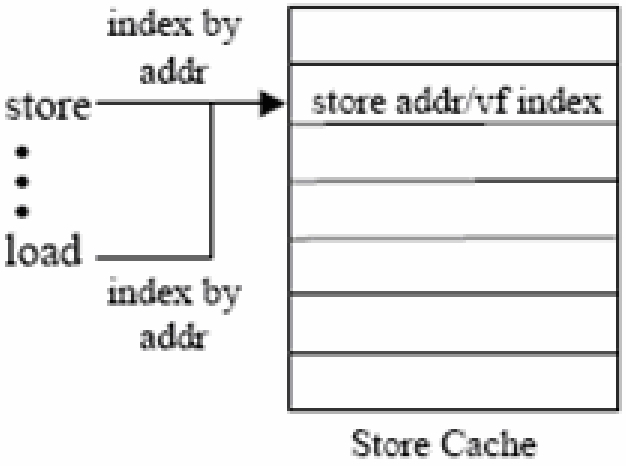
Value Prediction



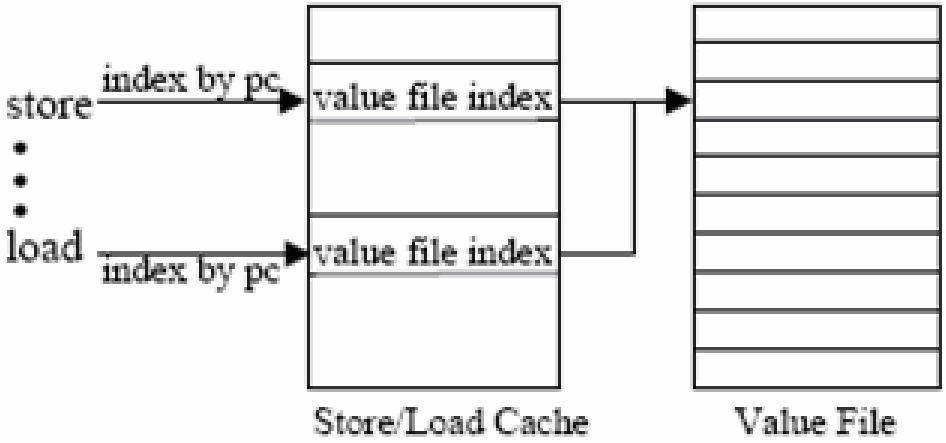
Memory Renaming

- Motivação
  - Localidade temporal
  - Instruções *loads* lêem dados armazenados por *stores* recentes.
  - Uma vez idenficada essa situação, o dado pode ser ‘encaminhado’.

Memory Renaming



Finding Store/Load Relationships



Finding the Value File Entry

Execução Especulativa

- Dúvidas?
- Obrigado!

Previsão de Desvios

Daniel Nicácio

dnicacios@yahoo.com.br

Introdução

- Perigos Estruturais
- Desvios
  - Incondicionais
  - Condicionais
- Motivação
- Previsor de desvio (branch predictor)
  - Estático
  - Dinâmico

Previsores Estáticos

- Desvio Seguido
  - taxa de erros de 9% a 59%
- Base na orientação do desvio
  - taxa de erros de 30% a 40%
- Baseado em perfil
  - taxa de erros de 5% a 22%

Previsores Dinâmicos

- Previsão de desvio básico e *buffers* de previsão de desvios.
- Previsores de desvio com correlacionamento
- Previsores por torneio

Técnica Alternativa

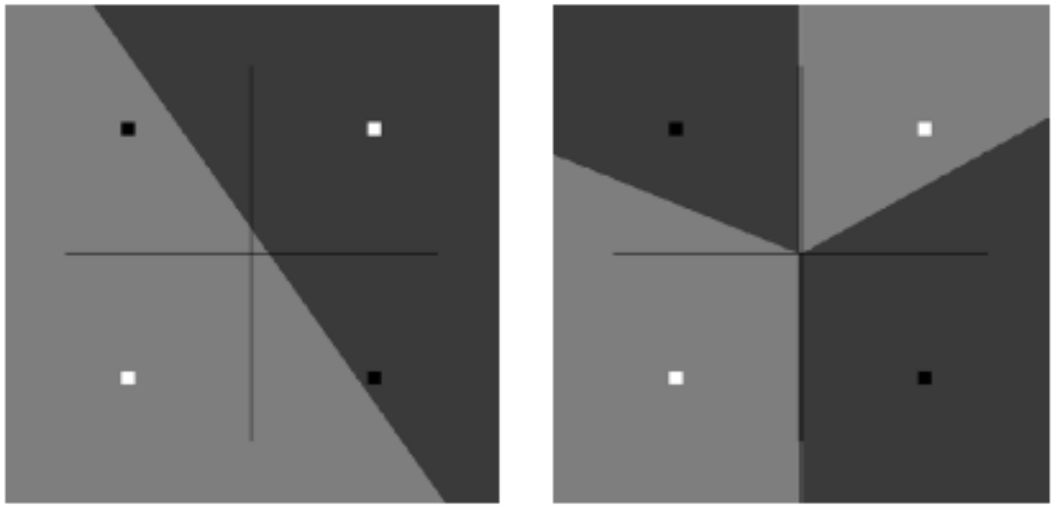
- instruções predicadas
  - Transforma a dependência de controle em dependência de dados.

Métodos Recentes

- Previsor bimodal
- Previsor e-gskew
- O previsor 2bc-gskew
  - Hibrido – e-gskew + bimodal
- Métodos Neurais
  - Perceptron
  - Perceptrons em dois níveis

Métodos Recentes

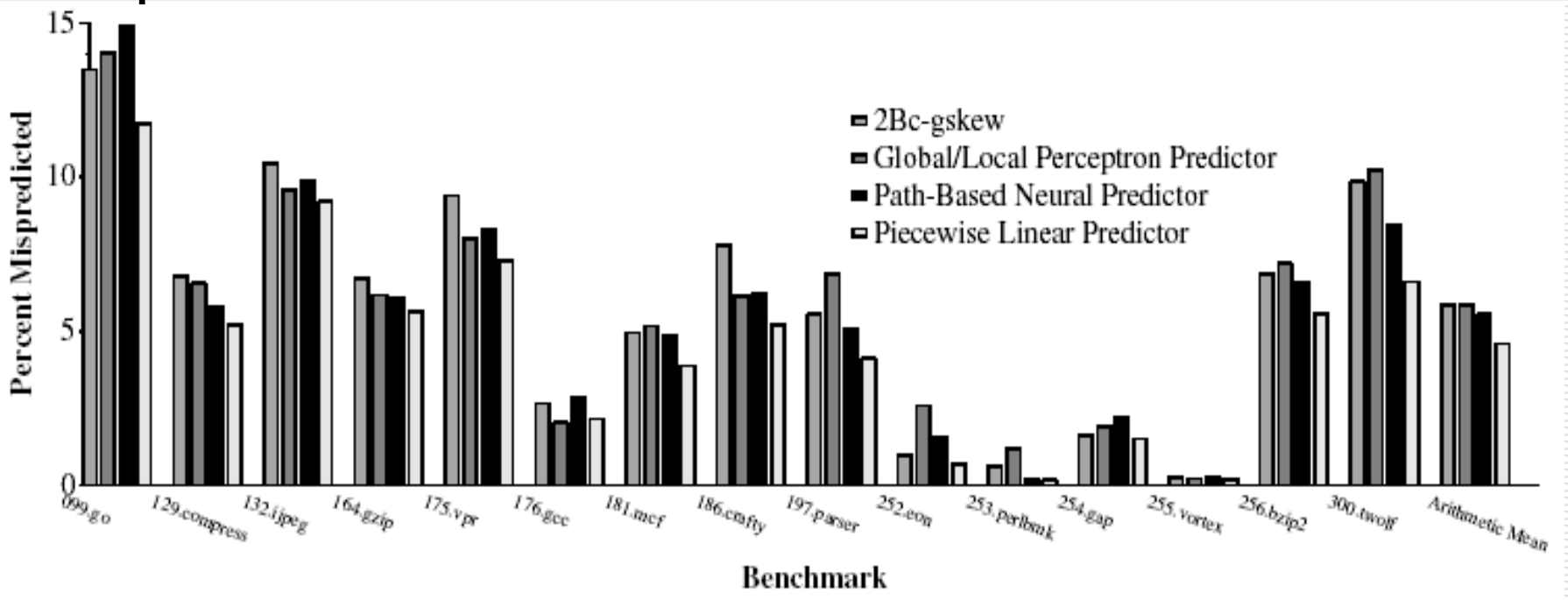
- O previsor Piecewise
  - A função XOR não é aprendida por um perceptron (a), mas pode ser aprendida utilizando uma superfície de decisão linear (b)



(a) (b)

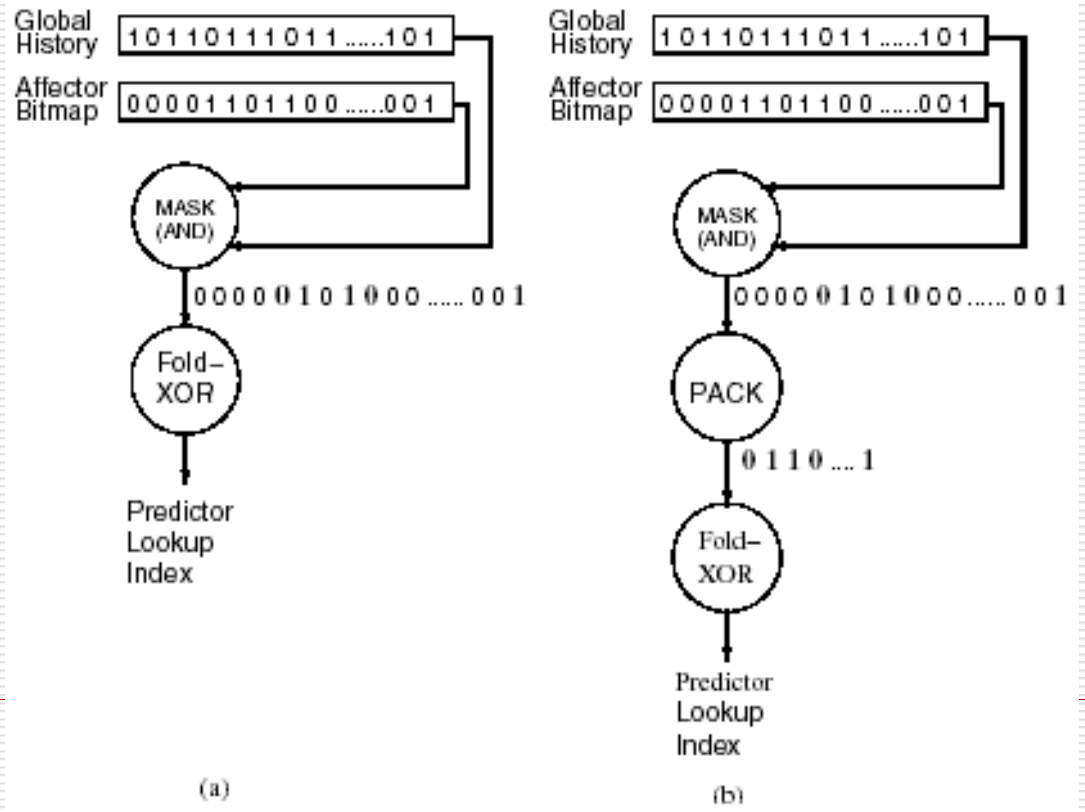
Métodos Recentes

- Comparação do piecewise com outros previsores



Métodos Recentes

- Identicando desvios correlacionados através de um longo histórico global
  - Affectors
  - Zeroing
  - Packing



Métodos Recentes

- **O previsor *Prophet/Critic***
  - **O Profeta prediz o resultado do desvio**
  - **O crítico critica o profeta se baseando em desvios futuros**

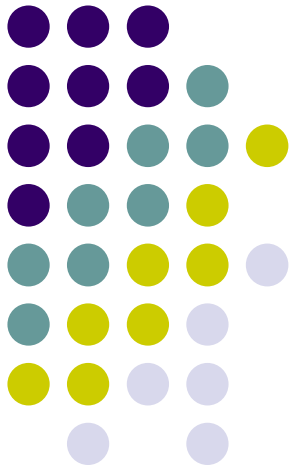
Métodos Recentes

- Adição de código para aumentar a precisão dos previsores de desvio
  - O endereço do desvio indexa a tabela de histórico
  - Desvio seguido → endereço xxx001
  - Desvio não seguido → endereço xxx000
  - Incluir no-ops
  - Heurística de inclusão

Conclusões

- Previsão de desvios é impressindível para a evolução da computação
- Grande evolução nos últimos dez anos
- Previsores com taxas de erro < 5%
- Futuro próximo → penalidade de desvios praticamente nula

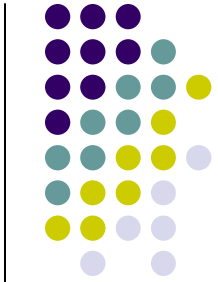
BlueGene/L



Introdução

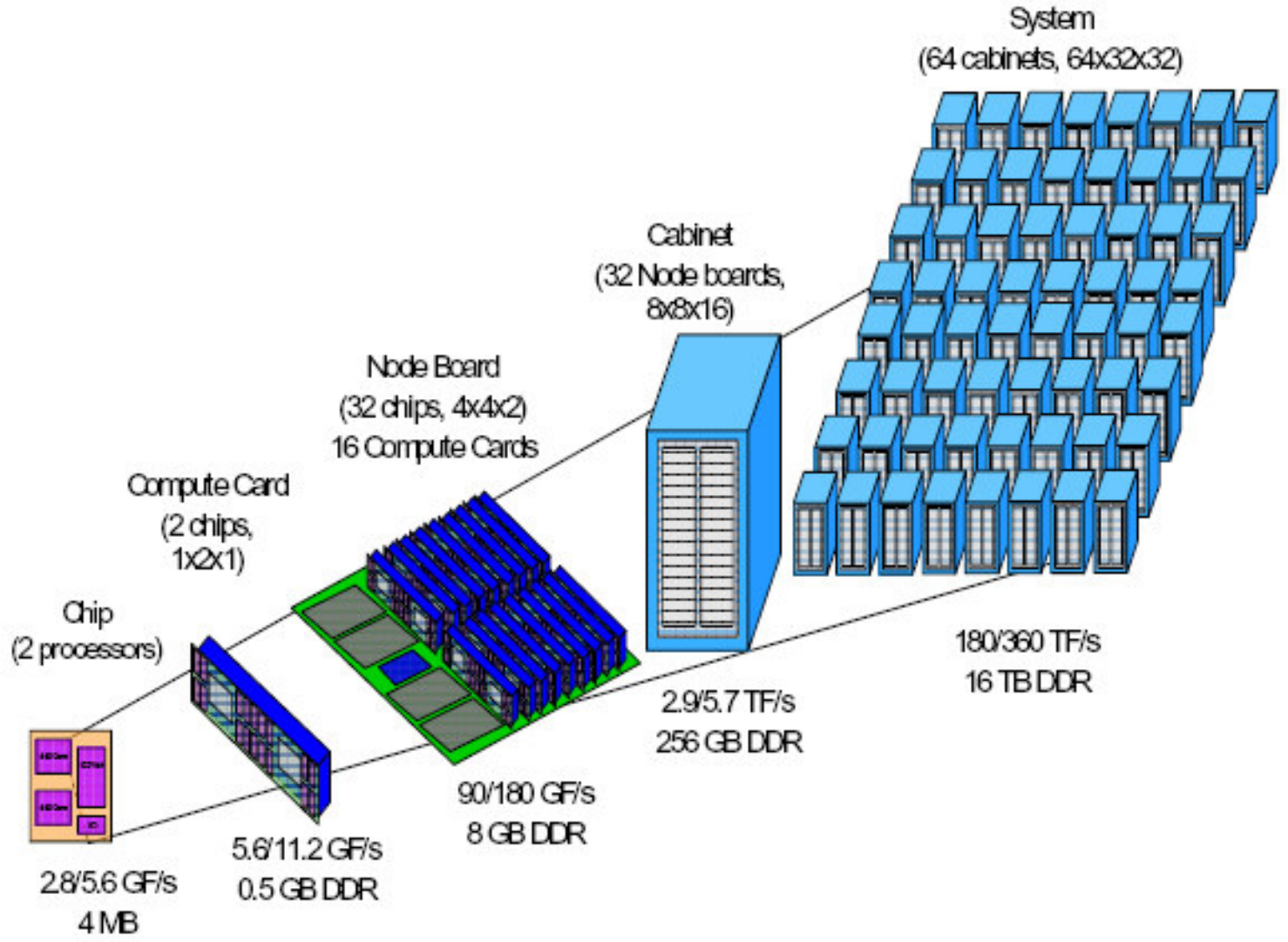
- Supercomputador massivamente paralelo
  - Desevolvido pela IBM, em parceria com o Lawrence Livermore National Laboratory
- Pico de performance: 360 teraFLOPS
- Até 65.536 nós
  - Processadores de baixo custo/consumo
- Modelo de passagem de mensagem
- Primeiro e segundo lugares na lista top500
  - Primeiro: 131072 processadores (LLNI)
  - Segundo: 40960 processadores (IBM Thomas J. Watson)

Arquitetura

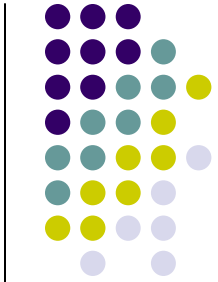


- Nós formados por 2 processadores PowerPC
  - 700MHz
  - 2 unidades de ponto flutuante
  - Tecnologia System on a chip
- Rede de interconexão em toro, dimensões 64x32x32
  - Comunicação ponto a ponto entre os nós
- Mais 4 redes adicionais, com finalidades específicas

Packing

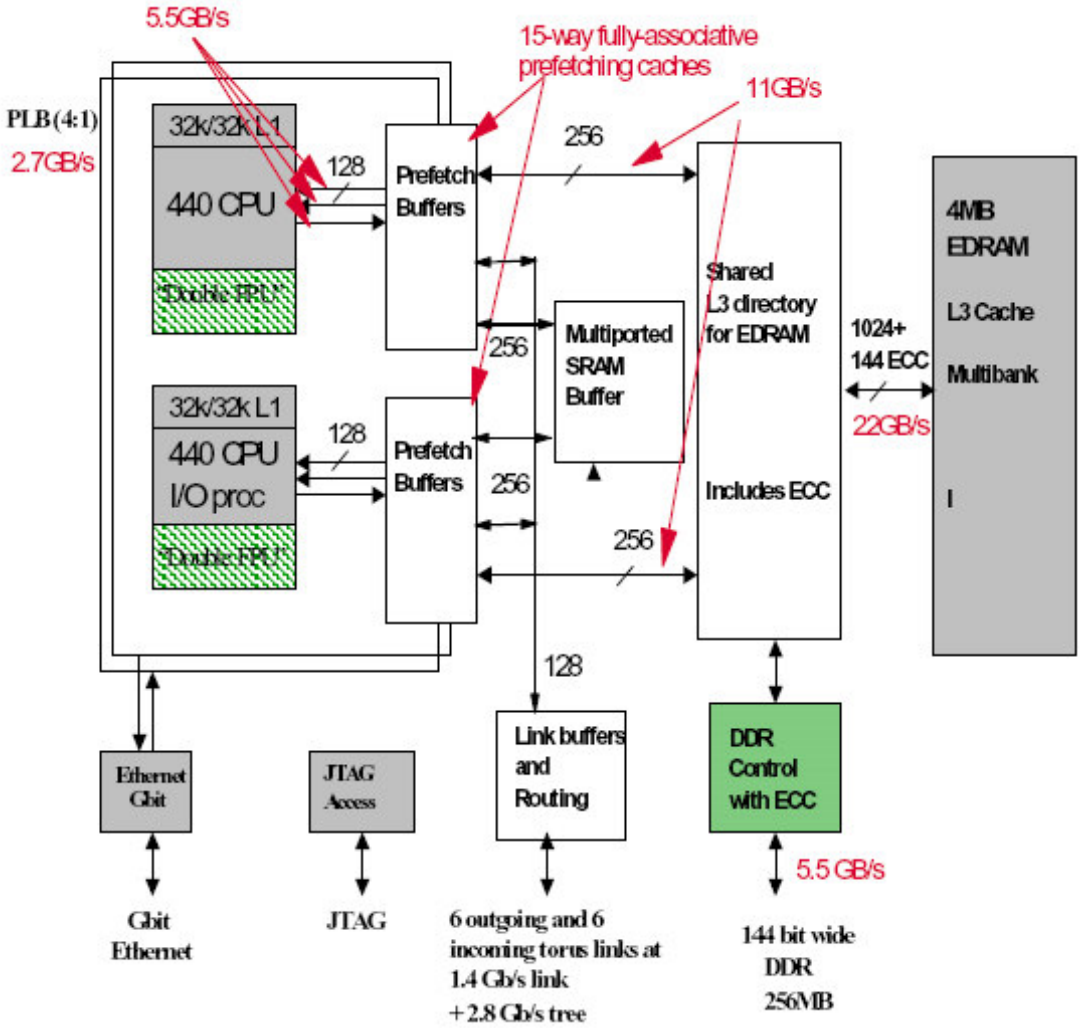

System
(64 cabinets, 64x32x32)
Cabinet
(32 Node boards,
8x8x16)
Node Board
(32 chips, 4x4x2)
16 Compute Cards
Compute Card
(2 chips,
1x2x1)
Chip
(2 processors)
2.8/5.6 GF/s
4 MB
5.6/11.2 GF/s
0.5 GB DDR
90/180 GF/s
8 GB DDR
2.9/5.7 TF/s
256 GB DDR
180/360 TF/s
16 TB DDR

Nós

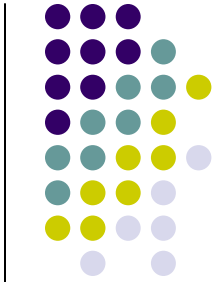


- 2 processadores PowerPC 440
  - Sem coerência de cache L1
- Co-processador FP2, com duas unidades de ponto flutuante, capazes de executar até 8 intruções de ponto flutuante por ciclo
- Cache L2 2KB com prefetch de dados
- Cache L3 4MB
- Interfaces JTAG, ethernet, memória externa
- Geralmente um dois processadores é dedicado ao envio/recepção de mensagens

Nós - Diagrama

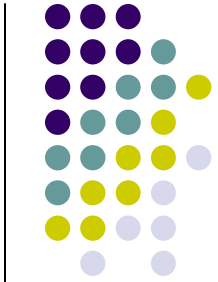


Redes de Interconexão



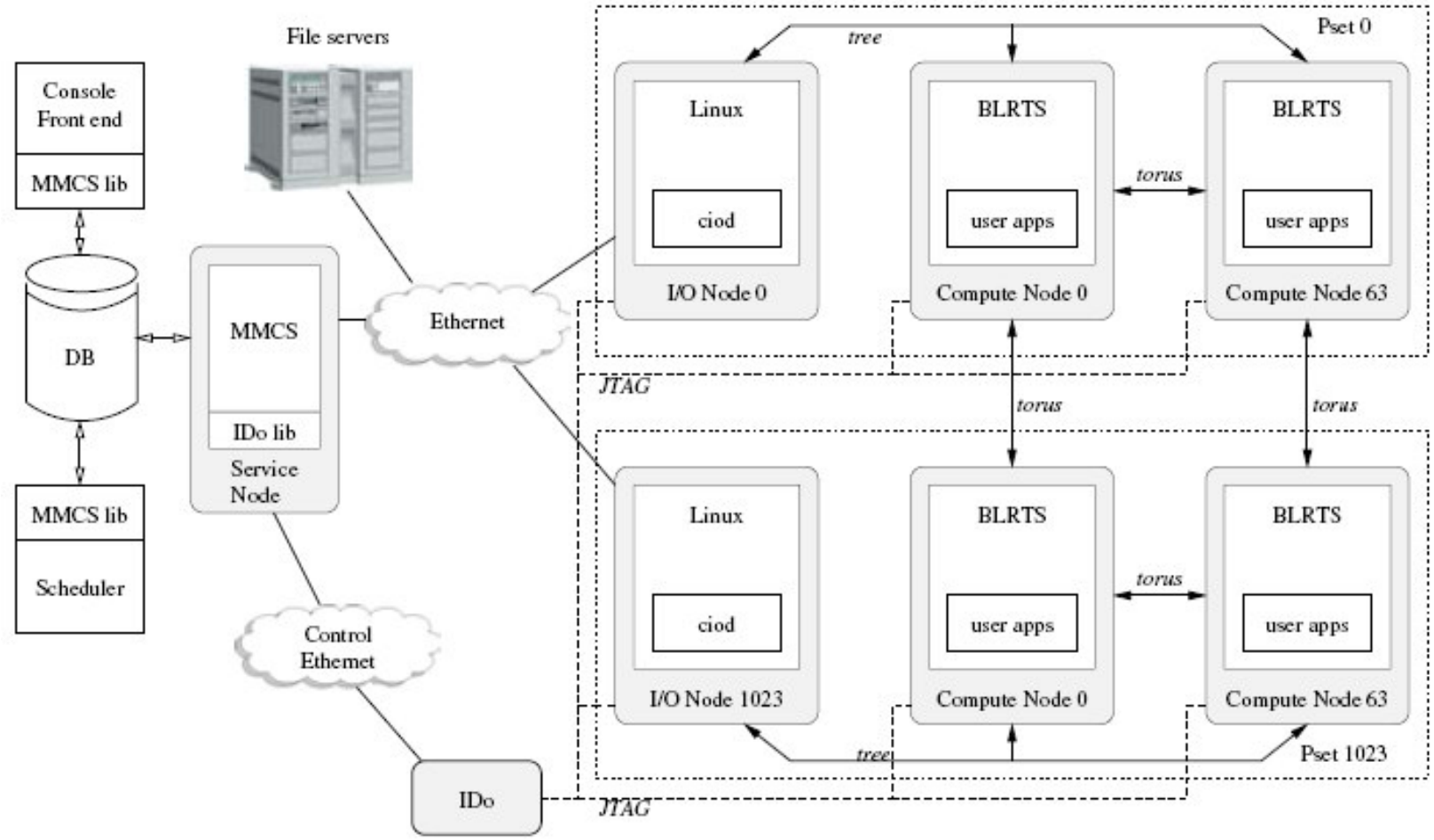
- Toro
  - Links seriais ponto a ponto entre os nós
  - Cada nó ligado com 6 vizinhos
  - Garantia de entrega e checagem de erro
- uma árvore de combinação/broadcast, para otimização de operações do tipo AllReduce();
- uma rede para barrier e interrupt;
- uma rede gigabit para conexão JTAG, para transmitir informações de controle e testes;
- uma rede gigabit para conexão com outros hosts.

Organização

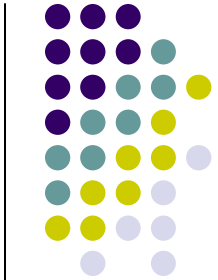


- Nós podem ser de processamento ou I/O
  - Processamento – utilizado para execução dos processos do usuário
    - Kernel monotarefa (BlueGene/L Run Time Supervisor - BLRTS)
  - I/O
    - Execução de serviços relativos a sistemas operacionais
      - Sockets, sistemas de arquivo…
    - Kernel Linux multitarefa
- Em geral 1 nó de I/O para cada 64 de processamento

Organização



Modelo de Programação



- Ferramentas GNU
  - Compilação cruzada
- Padrão MPI para passagens de mensagens
- 3 camadas para envio de mensagens
  - Pacotes
    - Mecanismos básicos para envio e recepção de mensagens
    - API simples (inicializar, enviar, receber)
    - Operações assíncronas
  - Mensagens
    - Envio de mensagens de tamanho arbitrário
    - Divisão da mensagem em pacotes
    - Reodenamento
  - MPI
    - API no nível do usuário

Comparativo entre Diferentes Estratégias de Multithreading

Aluno: Fernando André Kronbauer

e-mail: fernando.kronbauer@students.ic.unicamp.br

Comparativo entre Diferentes Estratégias de Multithreading

Processadores *multithreaded* possuem...

- a habilidade de executar mais de uma *thread* concorrentemente no mesmo *pipeline*
    - mais de um PC
- sistema de rotulação das instruções no *pipeline*
    - diferenciação das instruções de cada *thread*
- capacidade de provocar a troca de contexto entre as *threads* associadas ao *pipeline*

Comparativo entre Diferentes Estratégias de Multithreading

Várias implementações diferentes de *multithreading*

- Cray MTA
- IBM RS64 IV (PowerPC)
- Processadores p/ redes (Intel IXP, IBM PowerNP)
- UltraSPARC V
- Intel Hyper-Threading
- UltraSPARC T1 (Niagara)
- ...

Comparativo entre Diferentes Estratégias de Multithreading

Objetos do nosso estudo:

- Intel Hyper-Threading
- Niagara

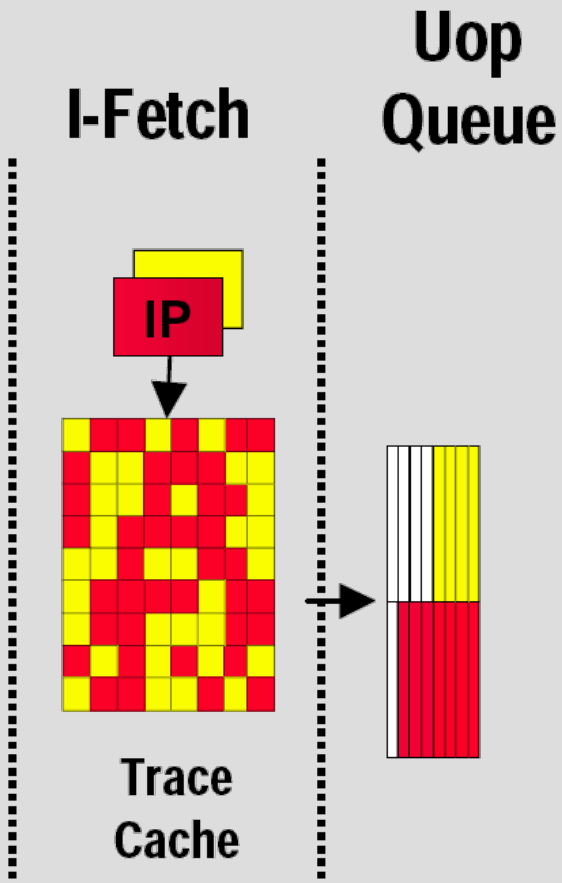
São tecnologias relevantes e atuais, bastante utilizadas (não seria possível tratar de outras implementações em tão pouco tempo!).

Intel Hyper-Threading

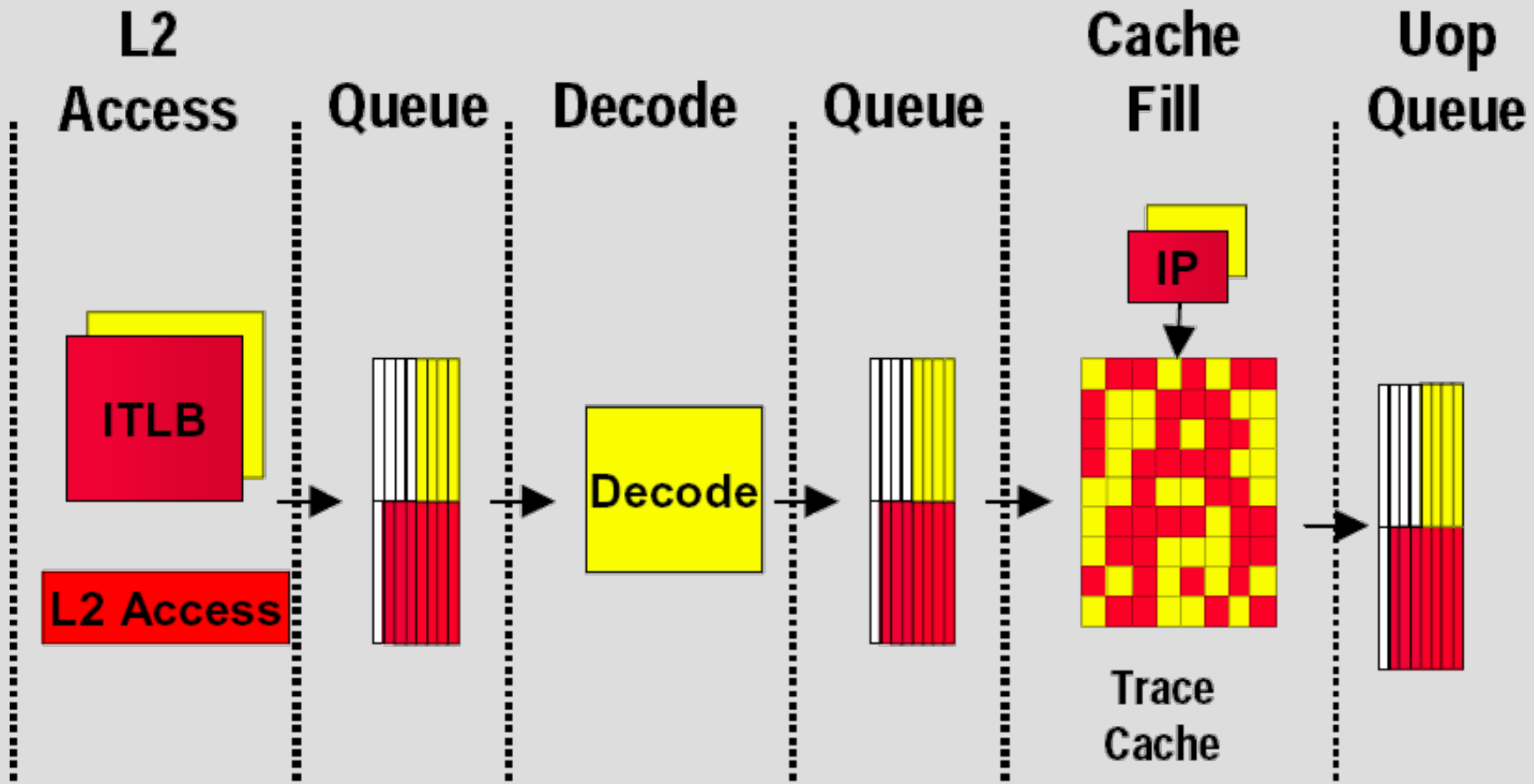
Objetivos da implementação:

- Aumentar a performance utilizando recursos ociosos do processador (unidades de execução)
- Manter a boa performance *single-threaded*
- Baixo custo para a implementação da capacidade *multithreading* (+5%)
- Melhoria de performance (20% ou mais)

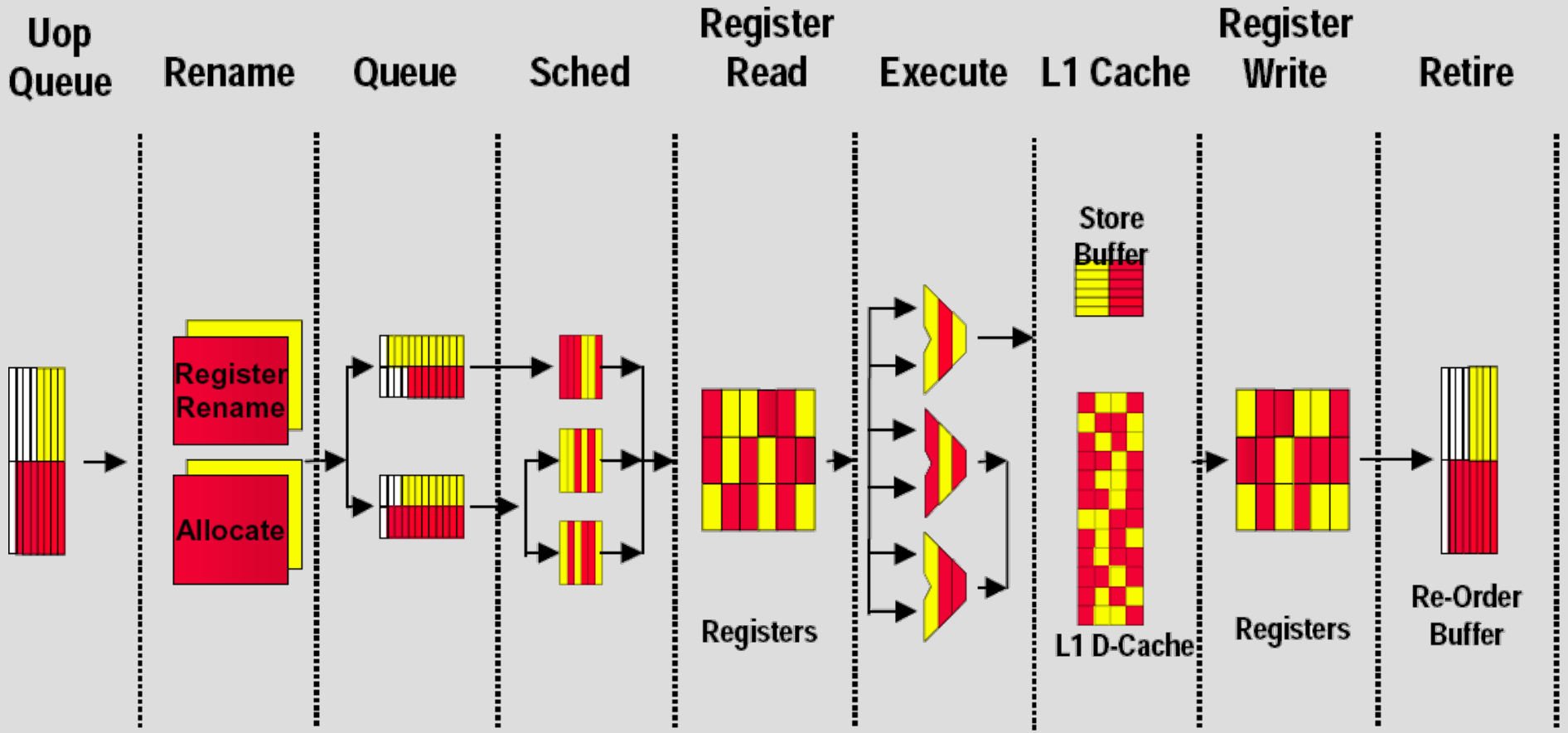
Intel Hyper-Threading



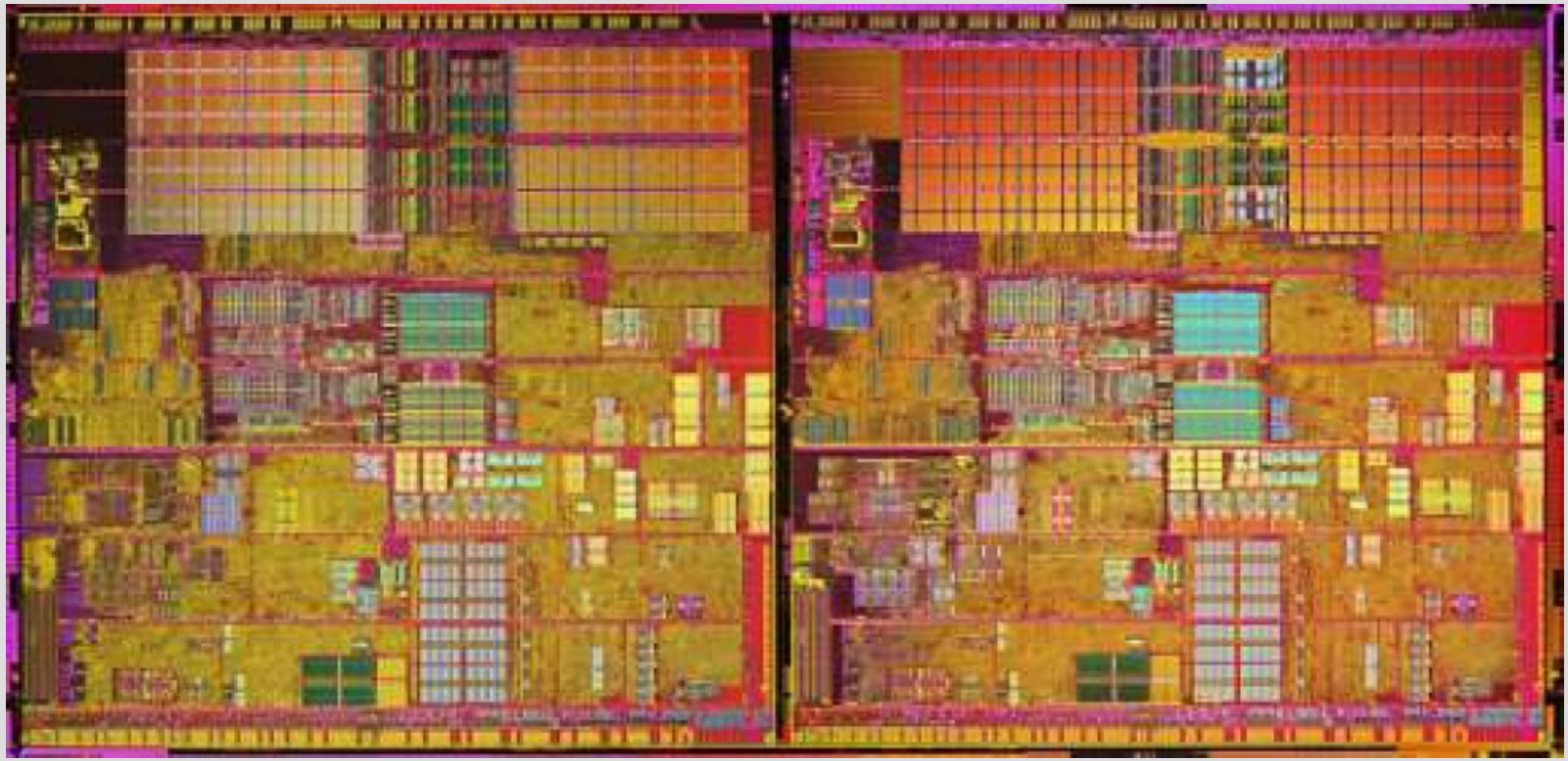
Intel Hyper-Threading



Intel Hyper-Threading



Intel Hyper-Threading

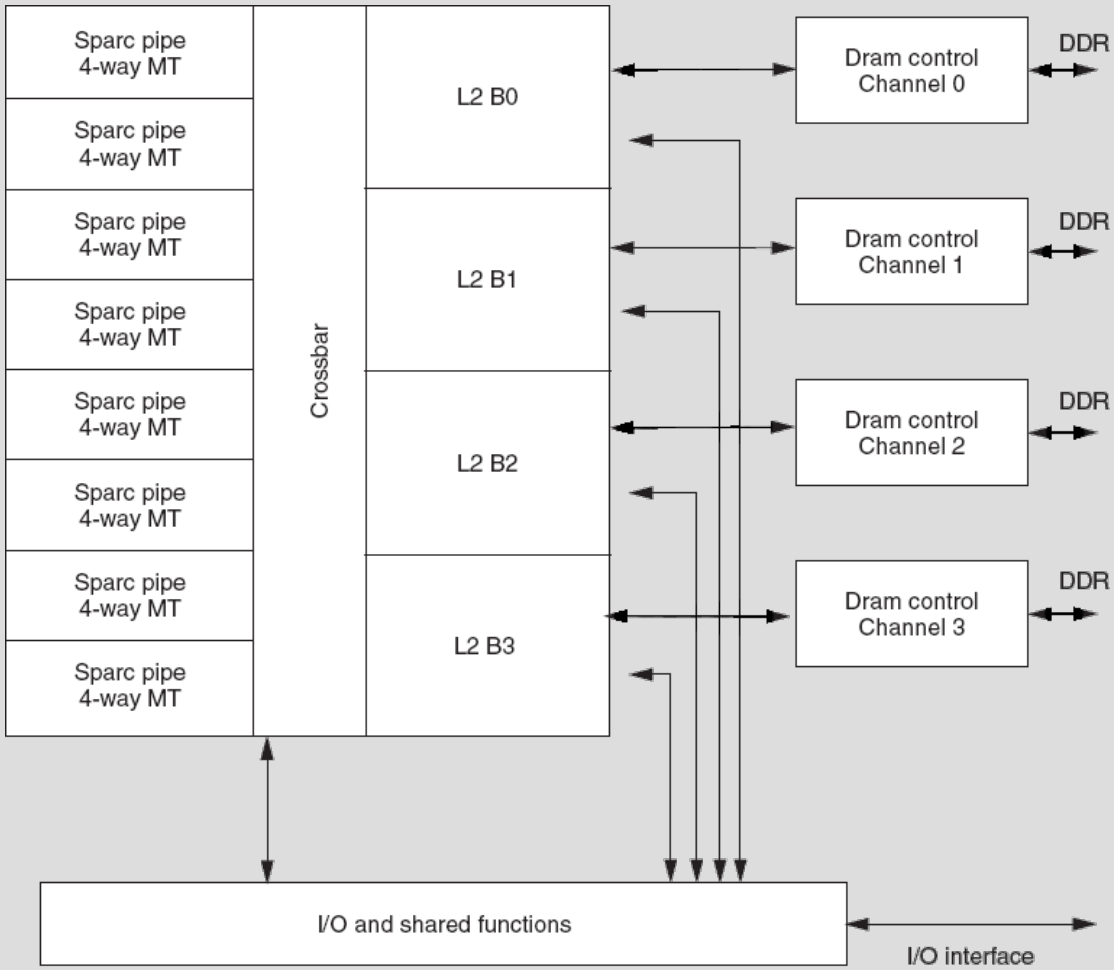


UltraSPARC T1 (Niagara)

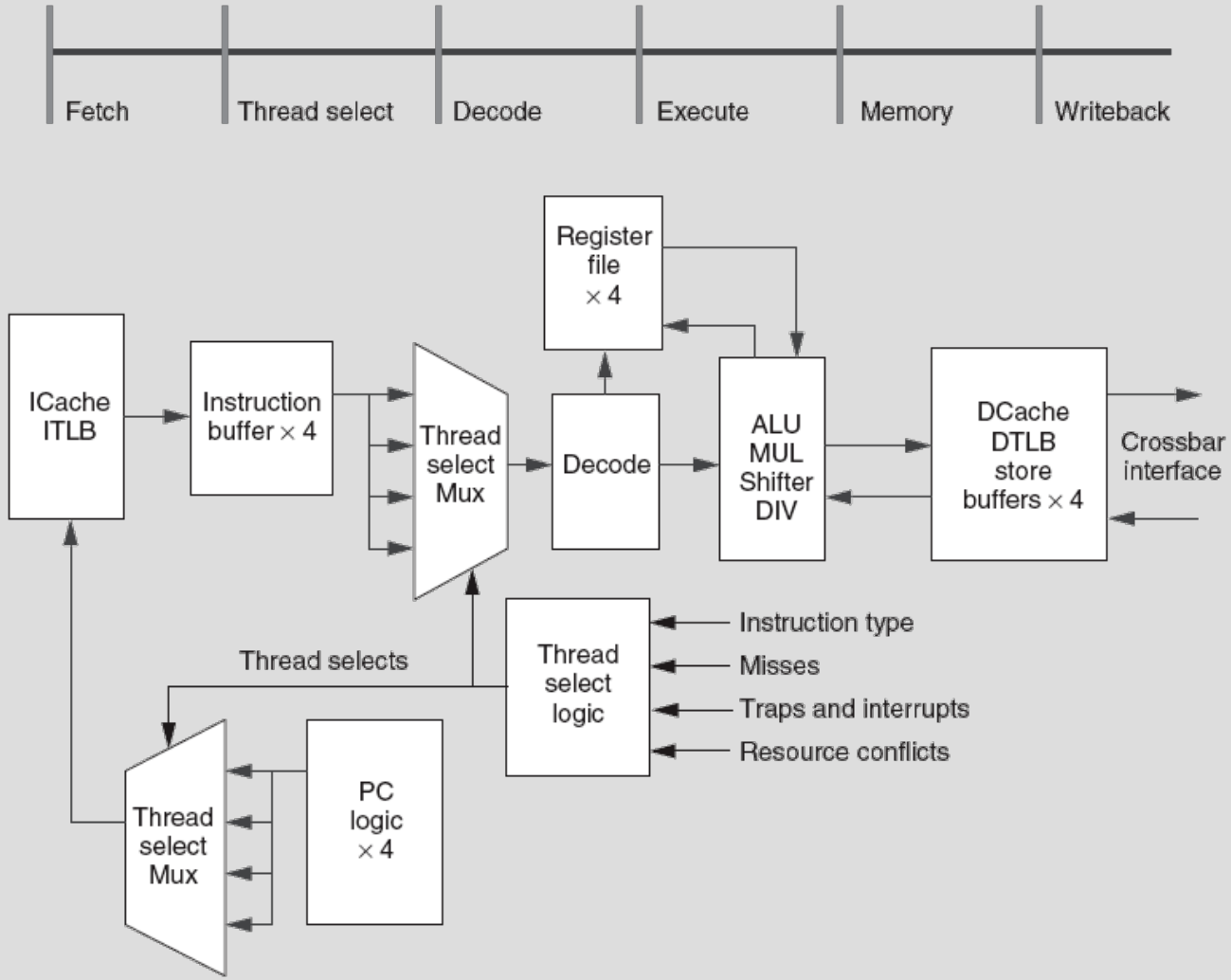
Objetivos da implementação:

- Aumentar o *throughput* de aplicativos de servidores comerciais através do suporte a grande número de *threads* de *hardware* (32)
- Atingir boa relação performance/Watt
- Melhorar a performance agregada do sistema, mesmo se em detrimento do desempenho individual de cada *thread*

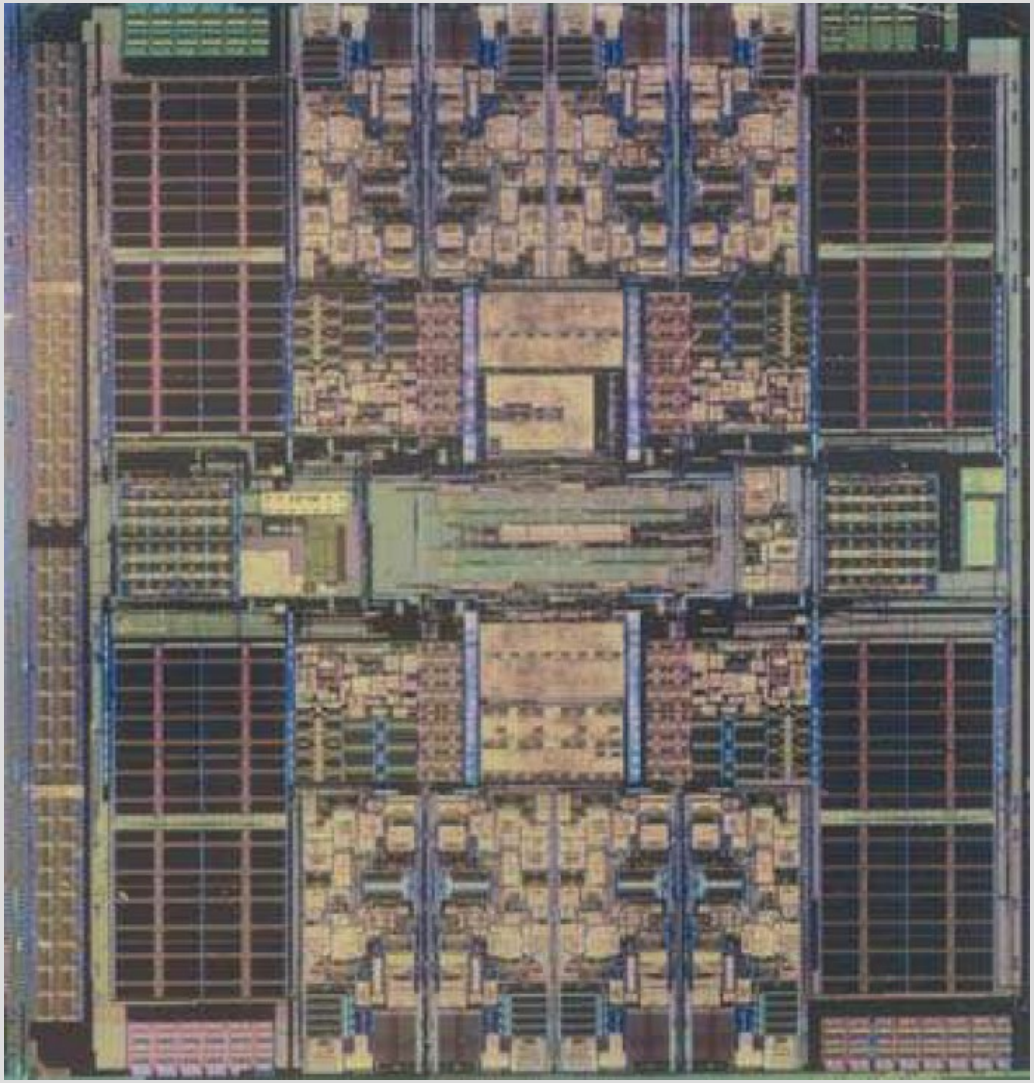
UltraSPARC T1 (Niagara)



UltraSPARC T1 (Niagara)



UltraSPARC T1 (Niagara)



Comparativo

Característica	Pentium Extreme Edition	Niagara
Freqüência de *clock*	3.2 Ghz	1.2 Ghz
Profundidade do *pipeline*	31 estágios	6 estágios
Potência consumida	130 W (@ 1.3 V)	72 W (@ 1.3 V)
Tamanho do *die*	206 mm^2	379 mm^2
Número de transistores	230 milhões	279 milhões
Número de núcleos de processamento	2	8
Número de *threads*	4	32
Cache IL1	12 kμop (8-*way*)	16 kB (4-*way*)
Cache DL1	16 kB (4-*way*)	8 kB (4-*way*)
Latência de instrução *load* (IL1)	1.1 ns	2.5 ns
Cache L2	Duas cópias de 1MB (8-*way*)	3 MB (12-*way*)
Latência de instrução *load* (L2)	7.5 ns	19 ns
Largura de banda da cache L2	~180 GB/s	76.8 GB/s
Latência de instrução *load (Memória Principal)*	80 ns	90 ns
Largura de banda da Memória Principal	6.4 GB/s	25.6 GB/s

Comparativo

Aplicação	Pentium Extreme Edition	Niagara
Código paralelo	1	2-3
Código seqüencial	5-7	1
Performance/Watt Código paralelo	1	4-5
Performance/Watt Código seqüencial	3-4	1

Coerência de Memórias Cache e Modelos de Consistência de Memória

Renato Silva das Neves

Arquitetura de Computadores

Junho 2006

Sumário

- Motivação
- Coerência de Memórias Cache
  - Conceitos iniciais
  - Protocolos de monitoração
  - Protocolos baseados em diretório
- Modelos de Consistência de Memória
- Conclusão

Introdução

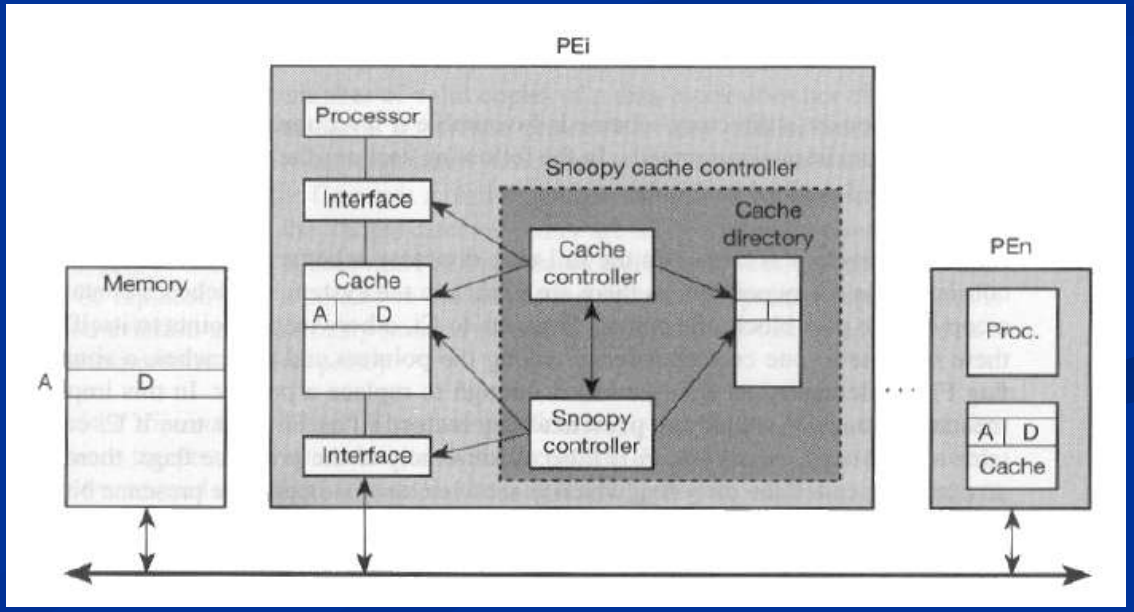
- Caches melhoram desempenho
- Velocidade dos microprocessadores
- Arquiteturas multiprocessadas
- Cópias de blocos distribuídas entre as caches dos processadores
- Coerência de cache
- Consistência de memória

Coerência de Memórias Cache
Conceitos Iniciais

- Software
  - Análise de código pelo compilador
  - Evitam o uso de hardware especial
  - Pode levar a uma utilização ineficiente da cache
- Hardware
  - Mantém coerência em tempo de execução
  - Protocolos de monitoração
  - Protocolos baseados em diretórios

Coerência de Memórias Cache
Protocolos de Monitoração

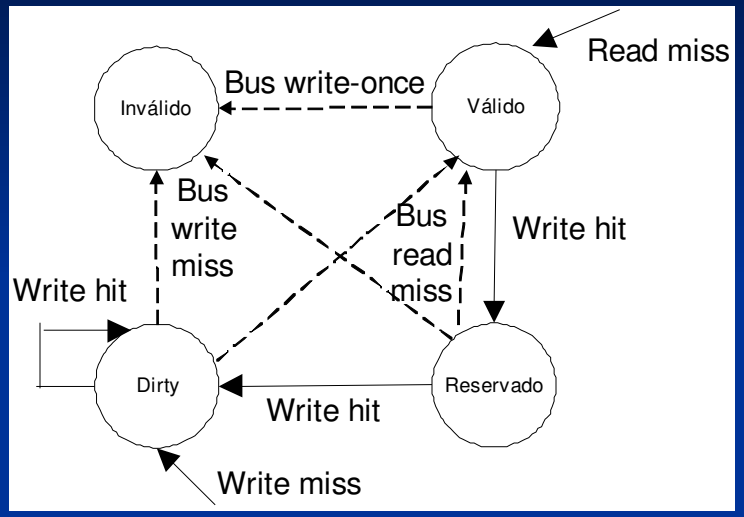
- Barramento compartilhado
- Broadcast
- Protocolos de atualização de gravação
- Protocolos de invalidação de gravação
  - Write-once
  - Berkeley
  - Illinois
  - Firefly
  - MESI
  - ...



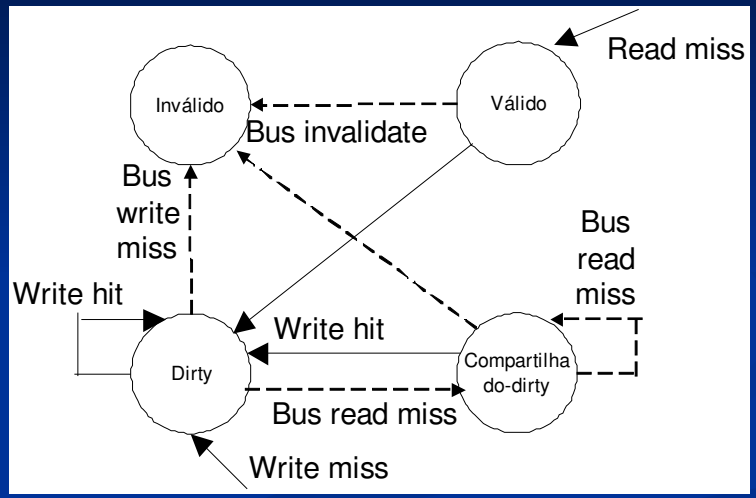
Coerência de Memórias Cache

Protocolos de Monitoração

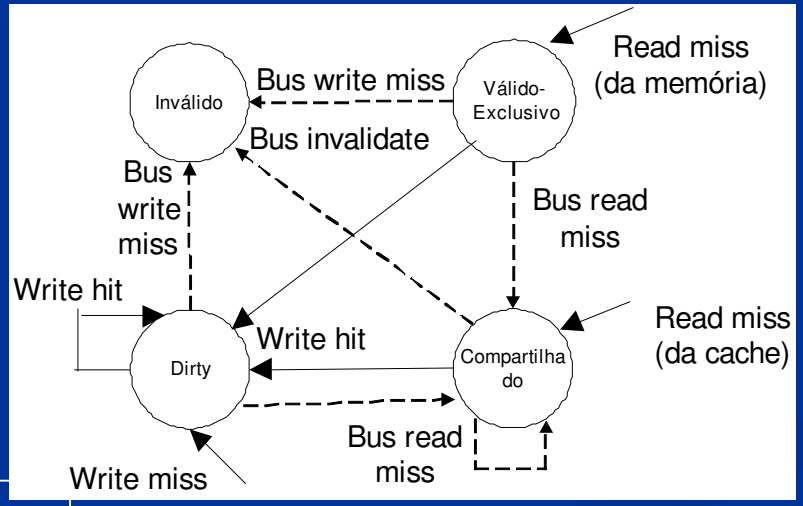
Write Once



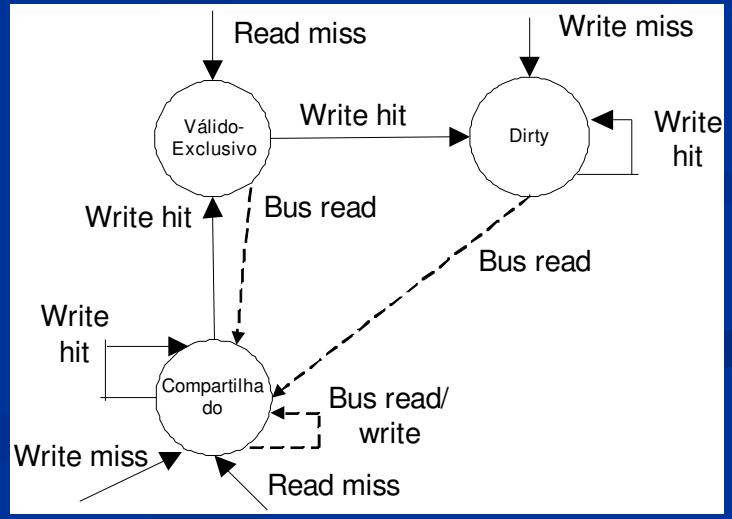
Berkeley



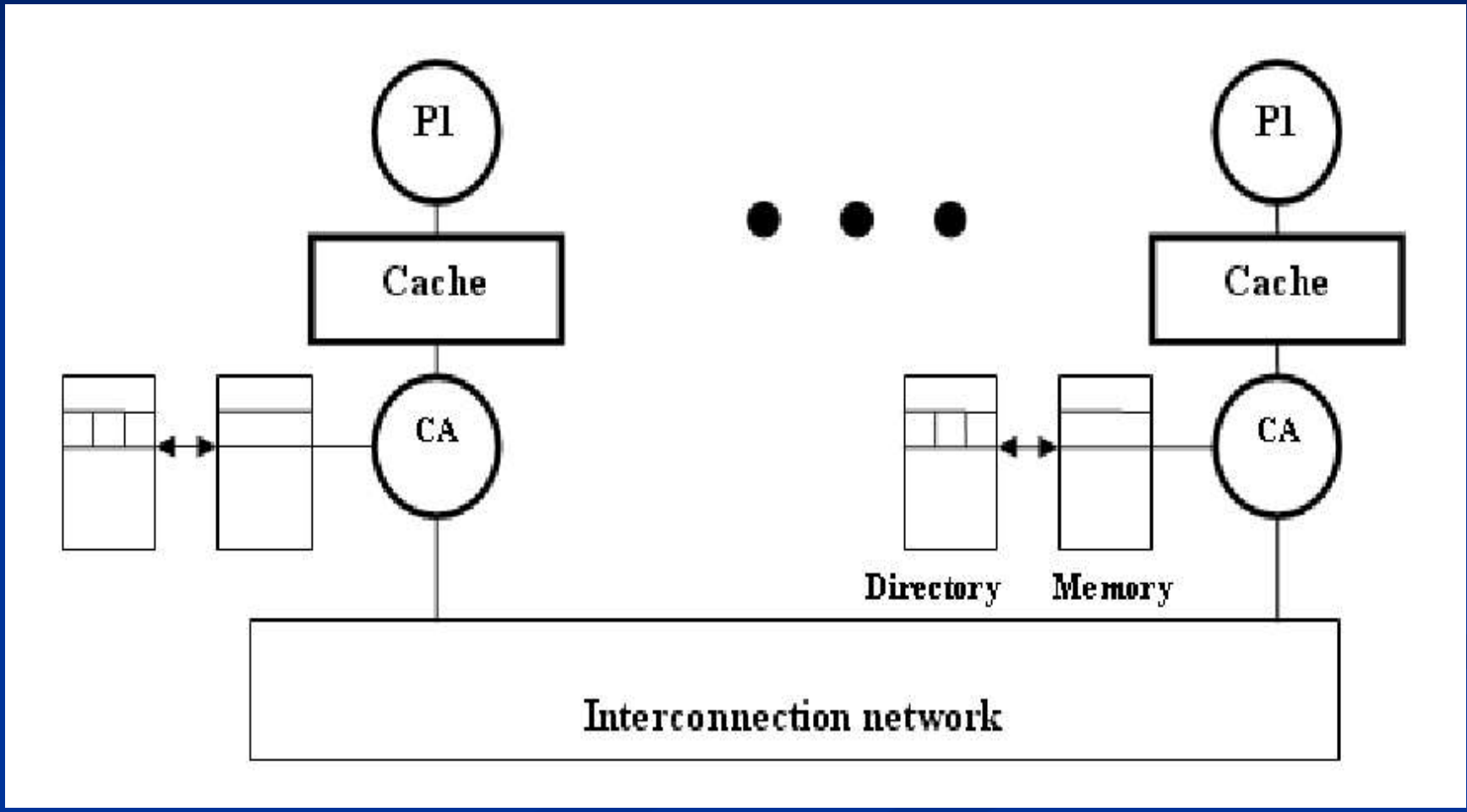
Illinois



Firefly



Coerência de Memórias Cache
Protocolos de Diretório



Coerência de Memórias Cache
Protocolos de Diretório

- Informações de compartilhamento são armazenados em diretórios
  - Centralizado – memória centralizada
  - Distribuído – memória distribuída
- Entrada de Diretório
  - Estado do bloco da memória / cache
  - Lista de compartilhadores
- Escalabilidade – muitos processadores!
- Conexão ponto-a-ponto, sem broadcast
- Classificações: completamente-mapeado, limitado ou encadeado

Coerência de Memórias Cache
Protocolos de Diretório

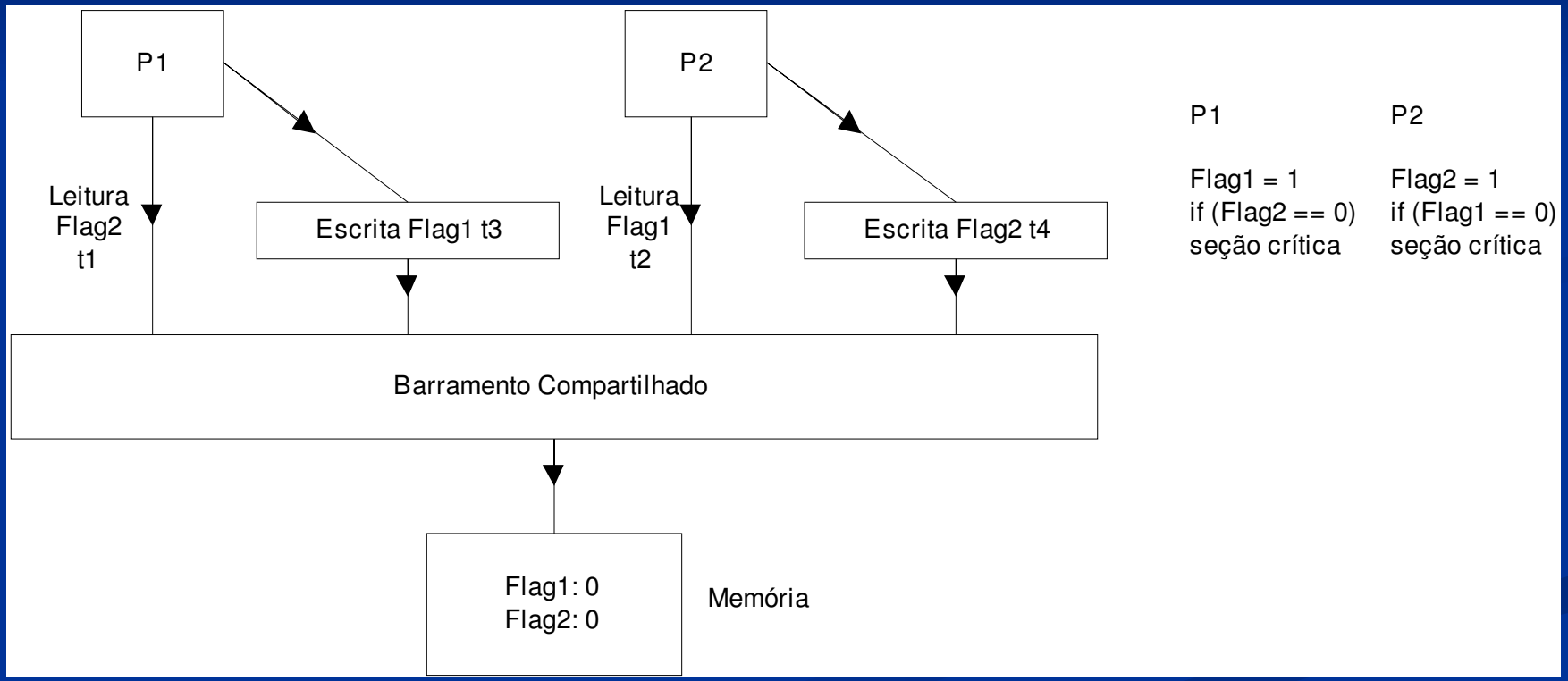
- Estados: bloco compartilhado, não inserido na cache ou exclusivo
- Falha de leitura na cache
  - Bloco exclusivo de outro processador
  - Realiza write-back na memória
  - Estado do bloco passa a ser compartilhado
- Gravação de um bloco na cache
  - Mensagem é enviada para invalidar cópias em caches
  - Confirmações são enviadas pelos processadores
  - O processador requisitante agora tem acesso exclusivo ao bloco

Modelos de Consistência de Memória

- Especifica a ordem vista pelo programador
- Leitura deve retornar o valor da "última" escrita
- Sistemas multiprocessados: execução fora de ordem
- Modelos de consistência
  - Seqüencial
  - Relaxado
    - Escrita para leitura, escrita para escrita ou todas as ordens

Modelos de Consistência de Memória

- Exemplo: relaxamento escrita para leitura



Conclusão

- Coerência de memórias cache
  - Abordagem de monitoração – barramento compartilhado, poucos processadores (até 32)
  - Abordagem de diretórios – conexão ponto-a-ponto, muitos processadores
- Consistência de memória
  - Execução fora de ordem
  - Necessidade de sincronização

Coerência de Memórias Cache e Modelos de Consistência de Memória

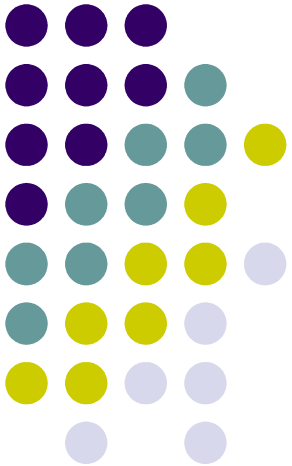
Renato Silva das Neves

Arquitetura de Computadores

Junho 2006

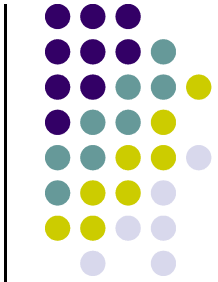
Bluetooth

Características, protocolos e funcionamento



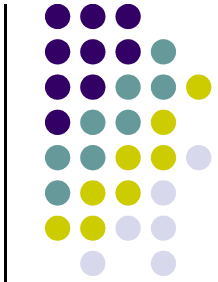
Thiago Senador – RA 057642

Agenda



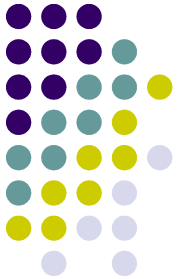
- Introdução
- Motivação
- Conceitos Básicos
- Características
- Arquitetura
  - Software
  - Hardware
- Processo de comunicação
- Considerações finais

Introdução



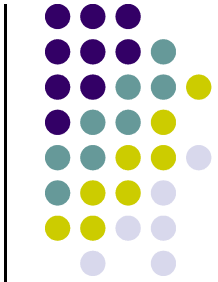
- Por que *Bluetooth*?
- Especificação x Tecnologia
- *Bluetooth* SIG (1994)
  - Ericsson, Intel, IBM, Nokia, Toshiba, 3Com, Lucent/Agere, Microsoft e Motorola
- IEEE 802.15.1
- *Wireless*
  - Curto alcance
  - Baixo consumo de energia
- PDAs, câmeras digitais, componentes automobilísticos, etc...

Motivação



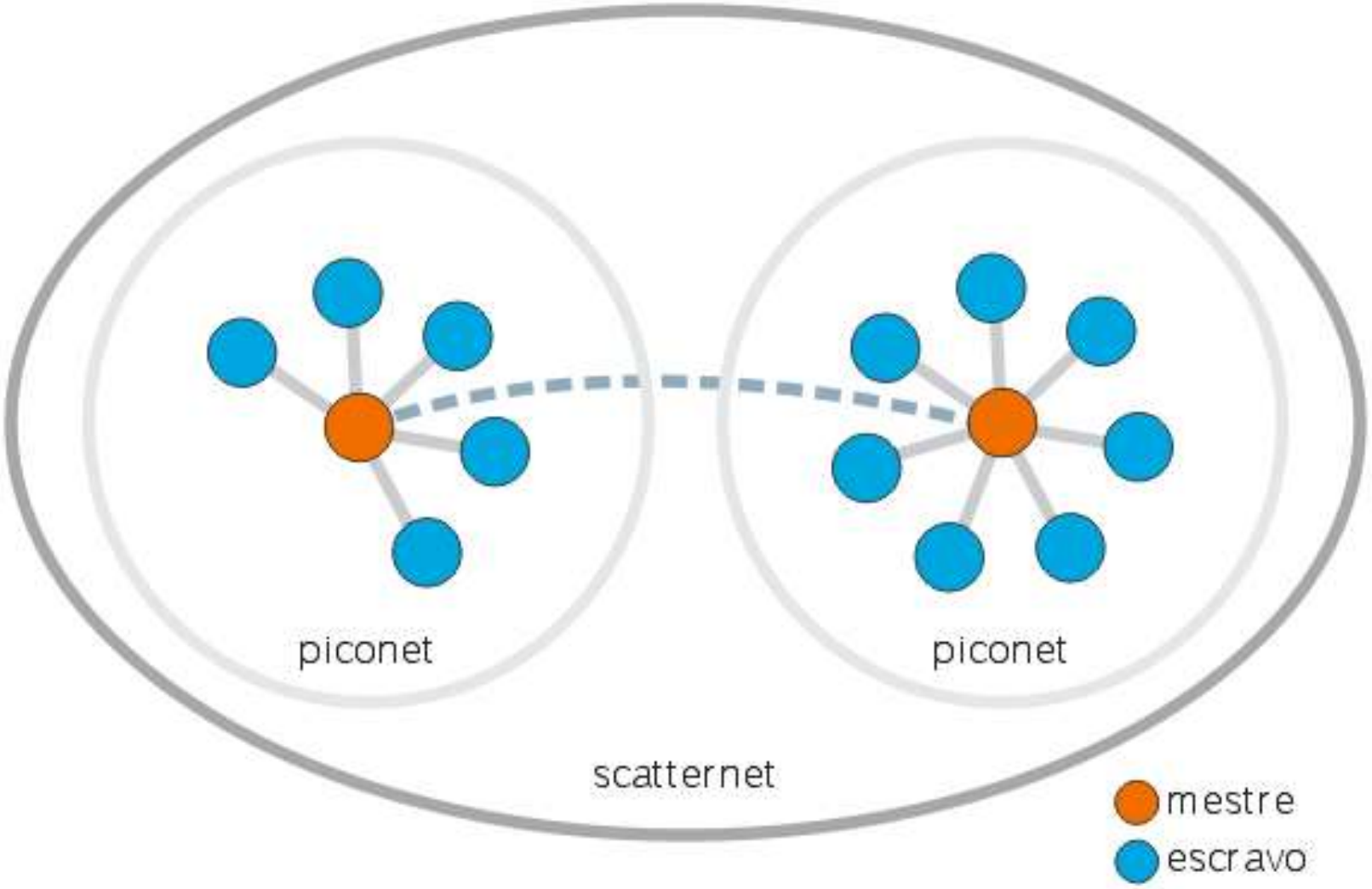
- Tecnologia para substituir cabos
- Tecnologia *wireless* de curto alcance
  - Baixo consumo de energia
  - Baixo custo
- Integração de dispositivos
- Padrões abertos
- Conseqüência
  - Mais de 2100 empresas
  - Mais de 600 milhões de dispositivos

Conceitos Básicos

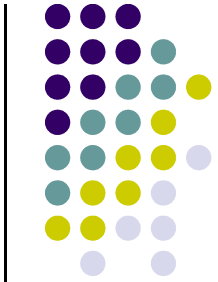


- *Wireless Personal Area Network*
  - *Piconet*
    - Até 8 dispositivos
    - Mestre
    - Escravo
  - *Scatternet*

Conceitos Básicos

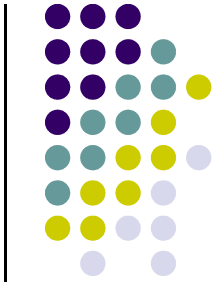


Conceitos Básicos



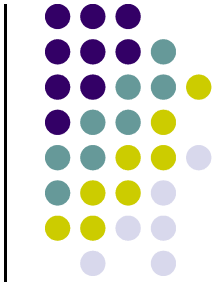
- *Frequency Hopping*
  - 1600 *hops/*s
  - 79 canais de 1 MHz
  - Cada canal é utilizado 625 μ s
- *Time Slots*
- Pacotes

Características



- Faixa não regulamentada ISM
  - 2.4 a 2.485 GHz
- *Frequency Hopping*
  - Isola canais utilizados
  - Impede a interferência
  - Segurança
- Taxa de transferência
  - Normal → 1Mbps
  - *Enhanced Data Rate* → 2 a 3 Mbps

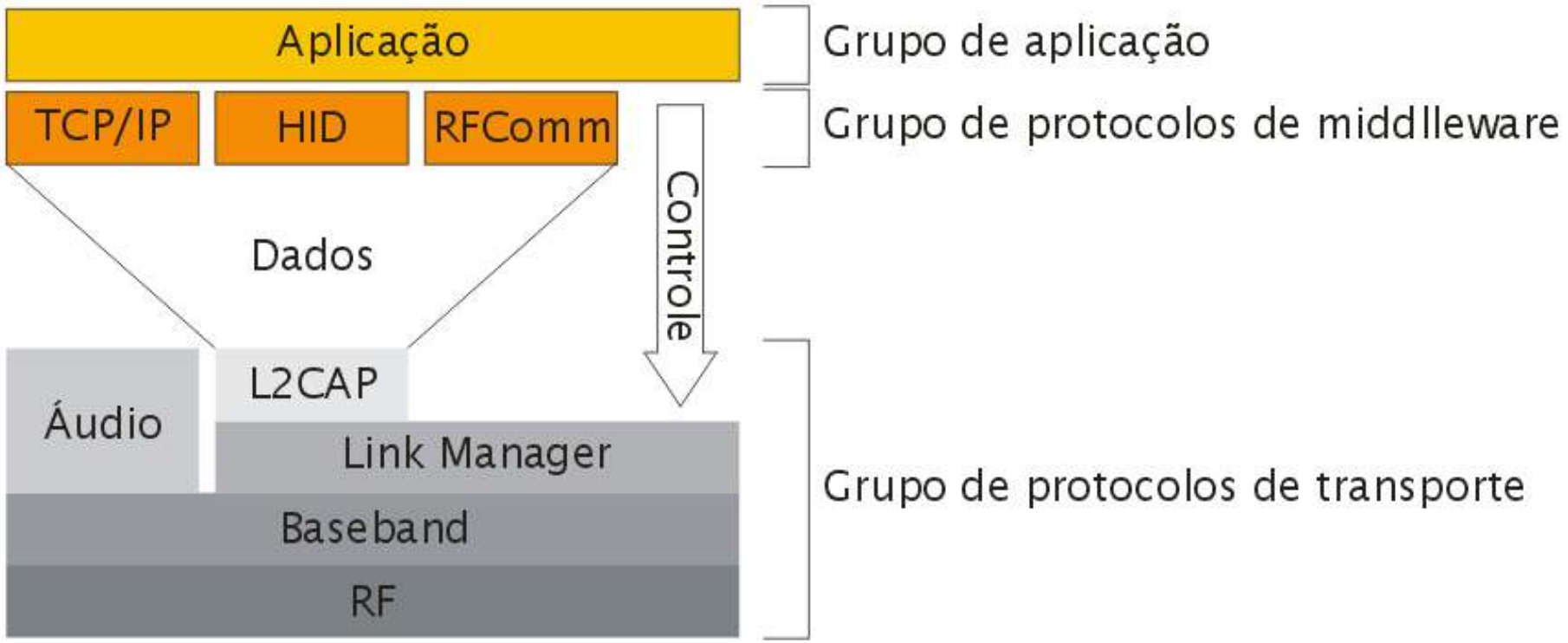
Características



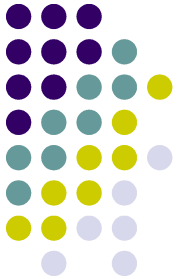
- Segurança
  - Criptografia 128 bits
  - *Frequency Hopping*
  - Autenticação PIN
  - Canais “seguros”
- Distância e consumo de energia
  - Classe 3 → 1 metro
  - Classe 2 → 10 metros
  - Classe 1 → 100 metros

Arquitetura – Software

- Pilha de protocolos

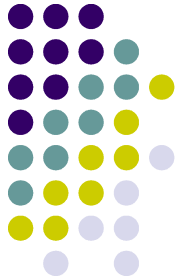


Arquitetura – Software



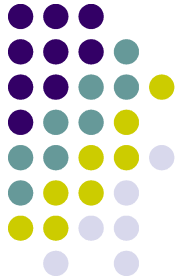
- Protocolos de transporte
  - Localização de dispositivos
  - Gerenciamento de links físicos e lógicos
- Protocolos de *middleware*
  - Protocolos de terceiros e padrões industriais
    - TCP, IP, WAP...
  - SIG
    - RFComm, SDP...
- Aplicação

Arquitetura – Software



- *Baseband*
  - Localização e conexão
  - Papéis Mestre e Escravo
  - Padrões de *frequency hopping*
  - Tipos de pacotes
  - Procedimentos de processamento de pacotes
  - Estratégias de detecção de erros
  - Criptografia
  - Transmissão e retransmissão de pacotes

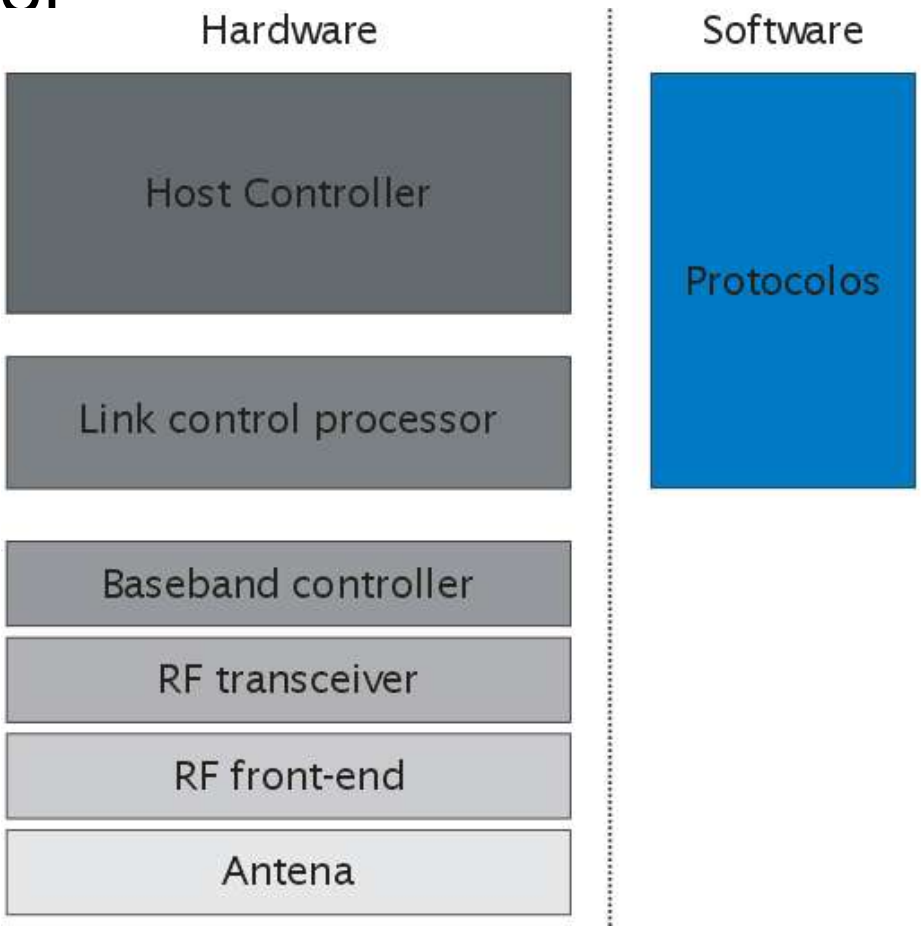
Arquitetura – Software



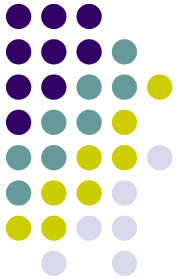
- Link Manager
  - *Link Manager Protocol* (LMP)
    - Propriedades do meio de transmissão (ar)
    - Taxa de transferência de dados
    - Taxa de transferência de áudio
    - Autenticação
    - Níveis de confiança
    - Controle do gasto de energia
  - *Logical Link Control and Adaptation* (L2CAP)
    - Interface entre camadas superiores e inferiores

Arquitetura – Hardware

- Transceiver

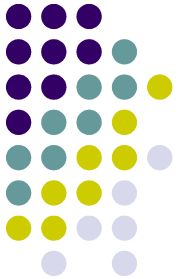


Arquitetura – Hardware



- ***Host Controller***
  - Processamento de código de alto nível
  - Controle de link lógico, L2CAP, RFComm e outras funcionalidades
- ***Link Control Processor***
  - Processamento de camadas mais baixas da pilha de protocolos como *Link Manager* e *Link Controller*
  - Combinado com o *Host Controller* em um único chip
- ***Baseband Controller***
  - Rádio freqüência (RF)

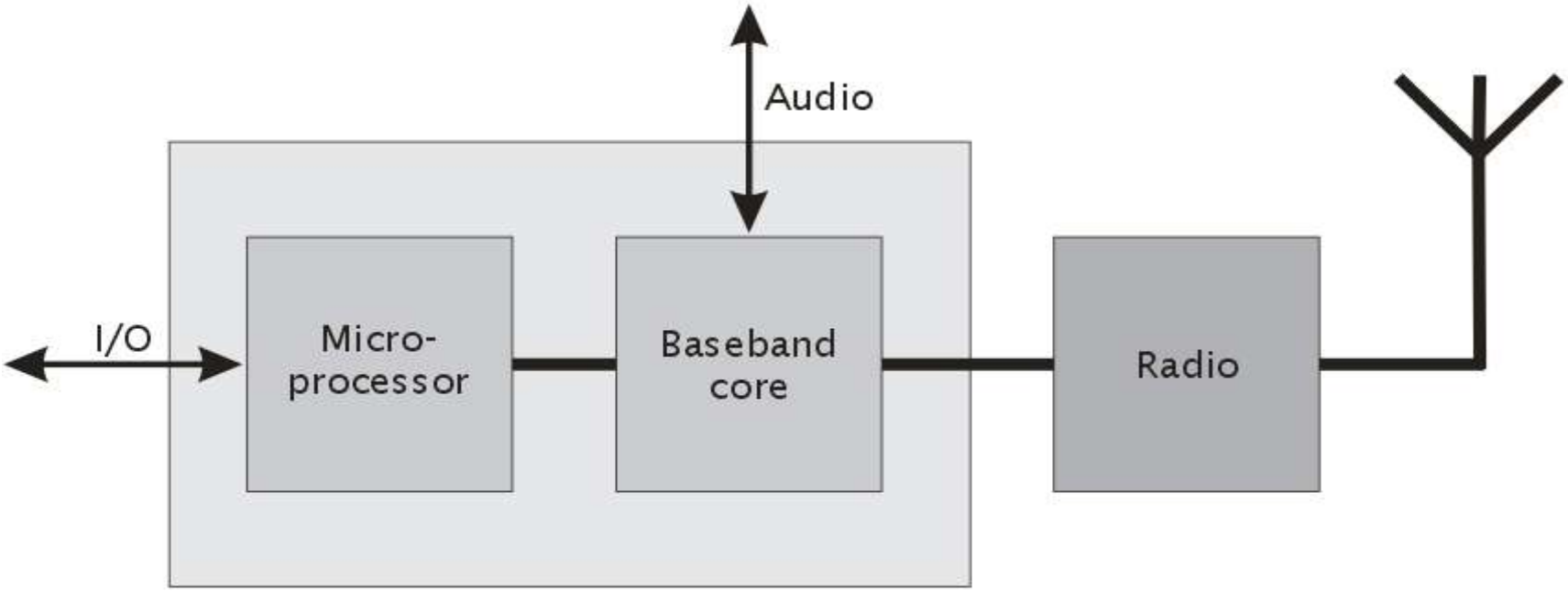
Arquitetura – Hardware



- ***Transceiver* RF**
  - Sintetizador de rádio freqüência
  - Filtros Gaussianos
  - Recuperação de *clock*
  - detecção de dados
- **RF *Front-End***
  - Filtro de banda passante da antena
  - Amplificador de ruídos
  - Amplificador de energia
  - Troca de estados – emissor x receptor
- **Antena**
  - Interna ou externa
  - Integrada em componentes de terceiros

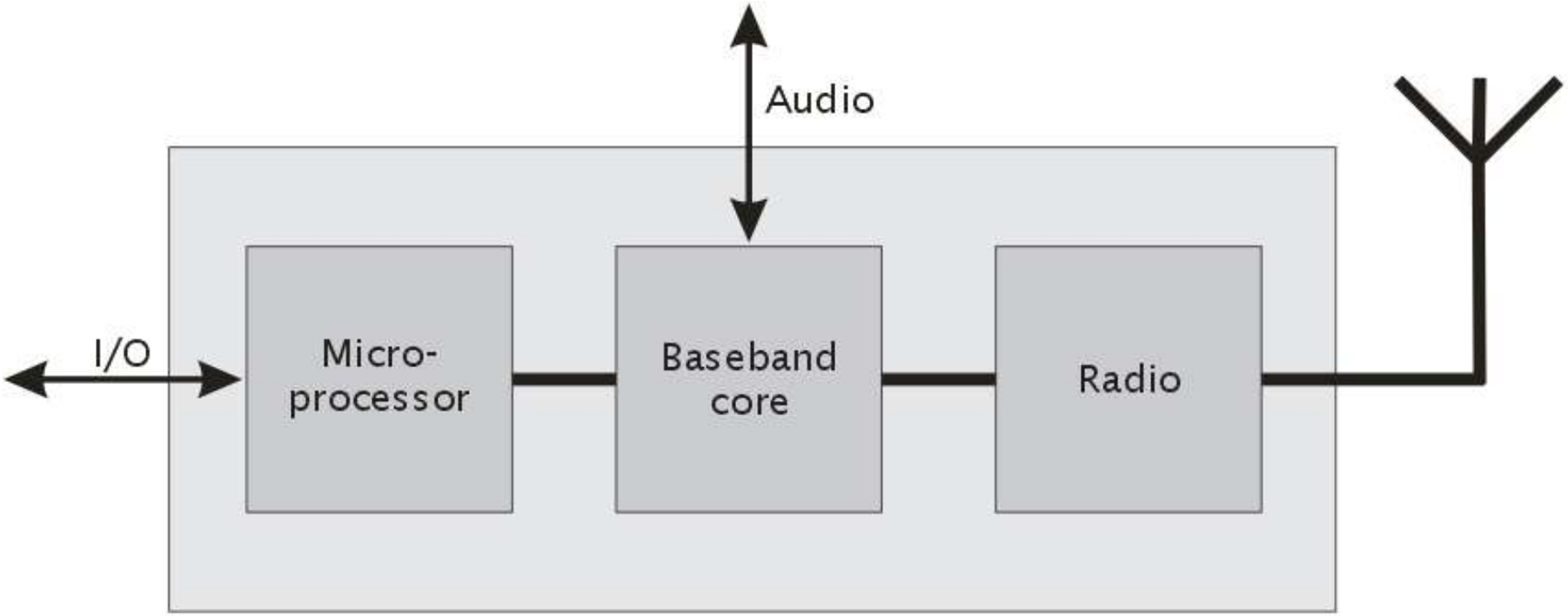
Arquitetura – Hardware

- Abordagens de projetos
  - *Multi chip*



Arquitetura – Hardware

- Abordagens de projetos
  - *Single chip*



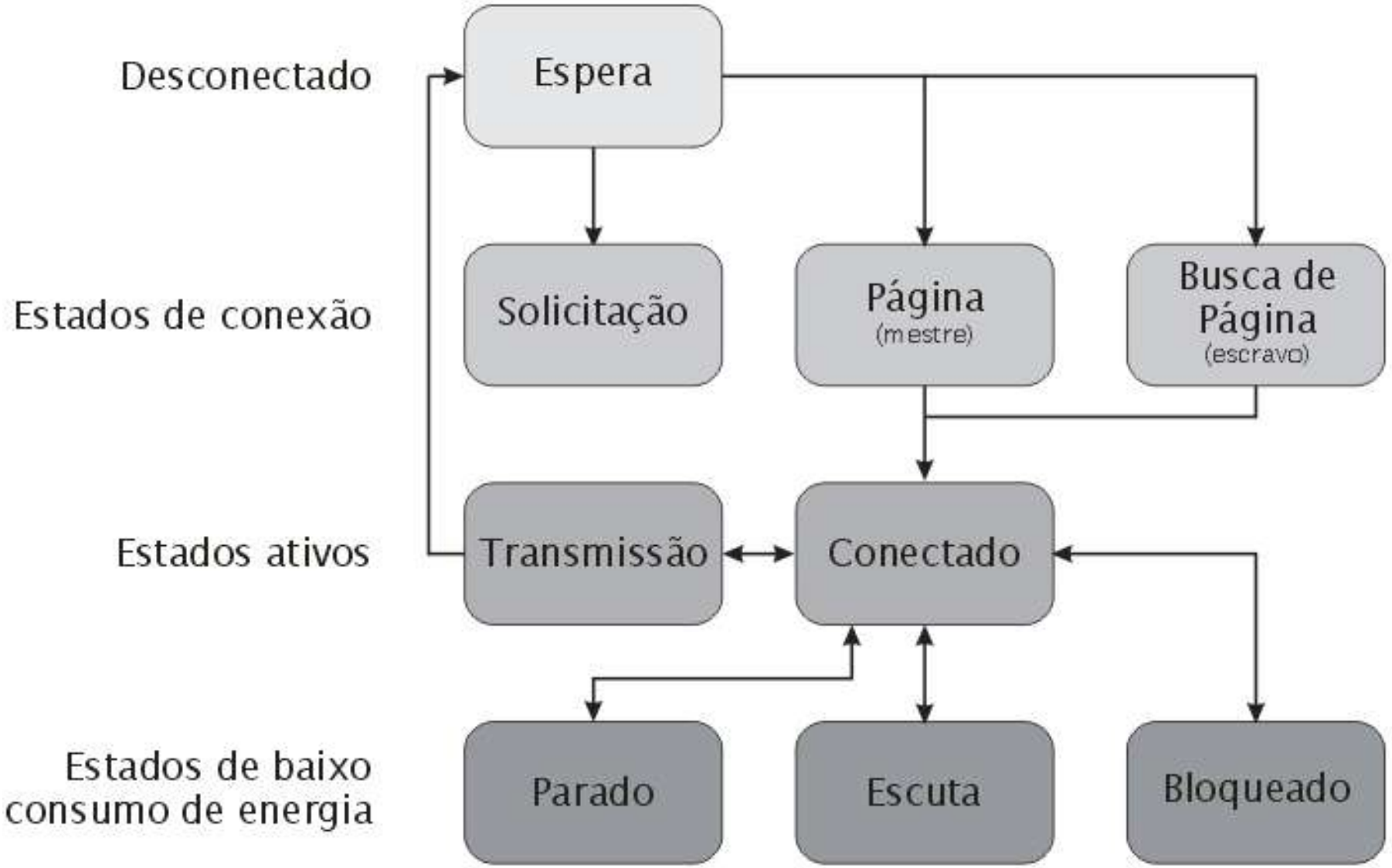
Processo de comunicação

- *Time Division Multiple Access* – TDMA

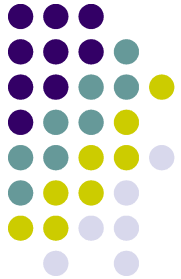
- Transmissões
  - Mestre → slots pares
  - Escravos → slots ímpares

- Pacotes

Processo de comunicação



Considerações finais



- Especificação aberta e publicamente disponível
- Sua tecnologia sem fio de curto alcance permite dispositivos periféricos comunicarem entre si através de uma interface simples, o ar, ao contrário das tecnologias de cabos, que utilizam conectores de uma grande variedade de formas, tamanhos e números de pinos
- Transferências de voz dados, tornando-se uma tecnologia ideal na comunicação de dispositivos heterogêneos
- Utiliza uma faixa de freqüências não regulamentada e vastamente disponível em qualquer lugar do mundo

Tecnologias de Disco

Edmar Welington Oliveira

Introdução

- Preocupação com tecnologias de disco
  - Desempenho de sistemas computacionais dependem:
    - Propriedades do processador
    - Propriedades dos dispositivos de armazenamento
    - Propriedades das tecnologias de interconexão
  - Novas fontes de informação
    - Imagens, sons e vídeos
  - Demanda das empresas

Perspectiva Histórica

- O primeiro projeto de unidade de disco foi o IBM 305 RAMAC (*Random Access Method of Accounting and Control*), lançado em 1956.

- 1ª Evolução: Unidade de disco rígido removível desenvolvida pela IBM em 1962. Ajudou os discos a superar as fitas com meio de armazenamento preferencial

- 2ª Evolução: Projeto *Winchester*, em 1973
  - Utilizava tecnologia HDA

Perspectiva Histórica

- Primeiro disco hermeticamente fechado
  - Dois eixos com um disco de 30MB cada - “30-30”
- *Winchester* – o rifle esportivo mais popular da América. ficou conhecido como “30-30” devido ao calibre de seu cartucho.
- ST-506
  - Capaz de armazenar 40MB; taxa de transferência de 645KB/s
- *Enhanced Small Device Interface* (ESDI)
  - Meados da década de 80
  - Taxas de 20MB/s

Tecnologias de Disco

- Classificação em função da interface
  - ATA
  - SCSI
  - SATA
  - SAS
  - SSA
  - Fibre channel

ATA

- Advanced Technology Attachment

- Características
  - Desenvolvida em 1986
  - Possui uma família de padrões

  - ATA-1, ATA-2, ATA-3, ATA/ATAPI4, ATA/ATAPI5, ATA/ATAPI6, ATA/ATAPI7,

ATA

Generation	Standard	Year	Speed	Key features
IDE		1986		Pre-standard
	ATA	1994		PIO modes 0,1,2 multiword DMA 0
EIDE	ATA-2	1996	16 MB/sec	PIO modes 3-4, multiword DMA modes 1-2, LBAs
	ATA-3	1997	16 MB/sec	SMART
	ATA/ATAPI-4	1998	33 MB/sec	Ultra DMA modes 0-2, CRC, overlap, queuing, 80-wire
Ultra DMA 66	ATA/ATAPI-5	2000	66 MB/sec	Ultra DMA mode 3-4
Ultra DMA 100	ATA/ATAPI-6	2001	100 MB/sec	Ultra DMA mode 5, 48-bit LBA
Ultra DMA 133	ATA/ATAPI-7	2002	133 MB/sec	Ultra DMA mode 6

SCSI

- Small Computer System Interface

- Um barramento de sistema com controladores “inteligentes” no qual os dispositivos trabalham em conjunto para controlar o fluxo de dados no canal de comunicação.

- Características
  - Desenvolvida em 1979
  - Taxa de transmissão de dados de 1.5MB/s
  - Conta com 3 nomes comumente utilizados

  - SCSI-1, SCSI-2, SCSI-3

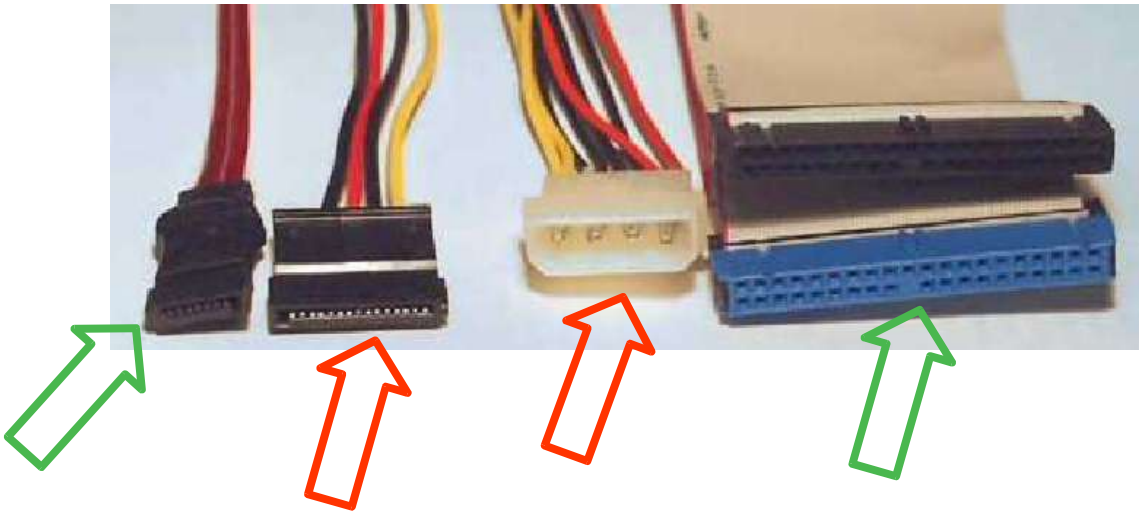
SCSI

Interconnect	Standard	Year	Speed	Key features
SASI		1979		Shugart Associates
SCSI-1	SCSI-1	1986	3 MB/sec 5 MB/sec	Asynchronous Synchronous
SCSI-2	SCSI-2	1990	10 MB/sec	CCS
SCSI-3	Split command sets, transport protocols, and physical interfaces into separate standards			
Fast	SPI	1992	20 MB/sec	
Ultra	SPI-1	1995	40 MB/sec	
Ultra 2	SPI-2	1999	80 MB/sec	
Ultra 3	SPI-3	2001	160 MB/sec	Packetized , QAS
Ultra 320	SPI-4	2002	320 MB/sec	

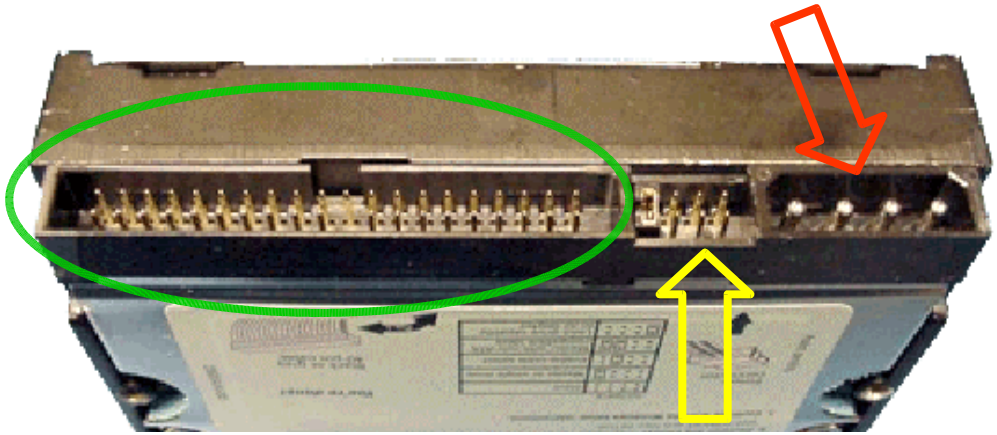
SATA

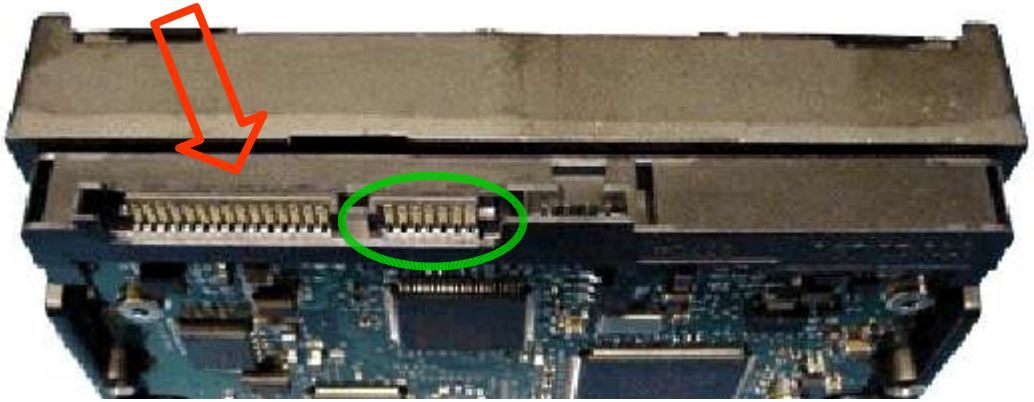
- Serial Advanced Technology Attachment
- Características
  - Desenvolvido em 2001
  - Substituir a tradicional interface PATA
  - Taxas de 150MB/s, 300MB/s
- Velocidade de transmissão
  - Interface Serial x interface paralela

SATA



SATA





SAS

- Trabalhos iniciados em 2001
- Especificação inicial em 2004
- Taxas de transferência
  - 300MB/s, 600MB/s e 1200MB/s

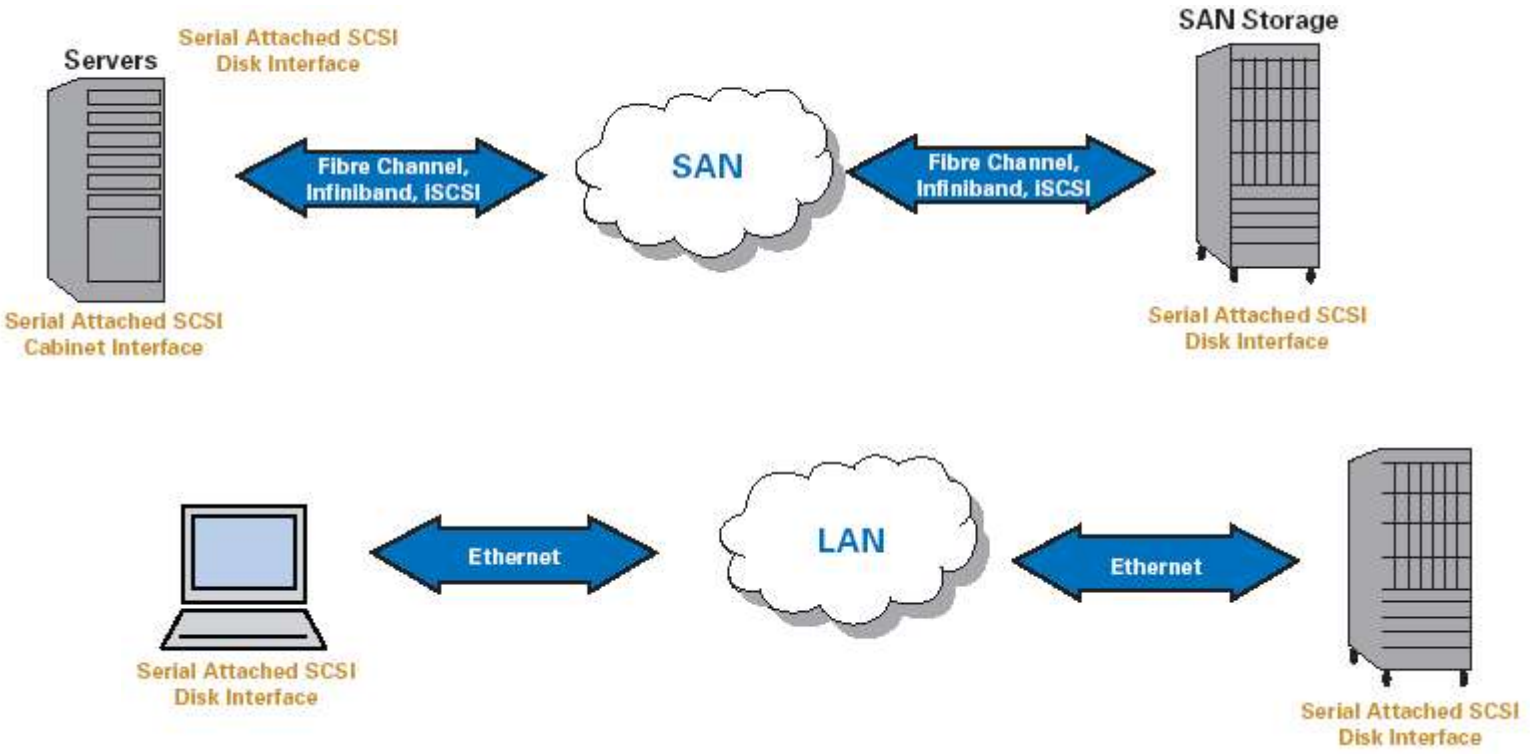
- Destinado ao mercado de servidores
- Compatibilidade com Serial ATA – mercado de desktops

- Uso de discos de diferentes taxas de transmissão
- Permite o uso da taxa de cada disco
- Propriedade Hot swap

SAS


Serial
Attached
SCSI Disks
Serial ATA or
Serial Attached SCSI Disk

SAS


Servers
Serial Attached SCSI
Disk Interface
Fibre Channel,
Infiniband, iSCSI
SAN
Fibre Channel,
Infiniband, iSCSI
SAN Storage
Serial Attached SCSI
Cabinet Interface
Serial Attached SCSI
Disk Interface
Ethernet
LAN
Ethernet
Serial Attached SCSI
Disk Interface
Serial Attached SCSI
Disk Interface

SSA

- *Serial Storage Architecture* (SSA)

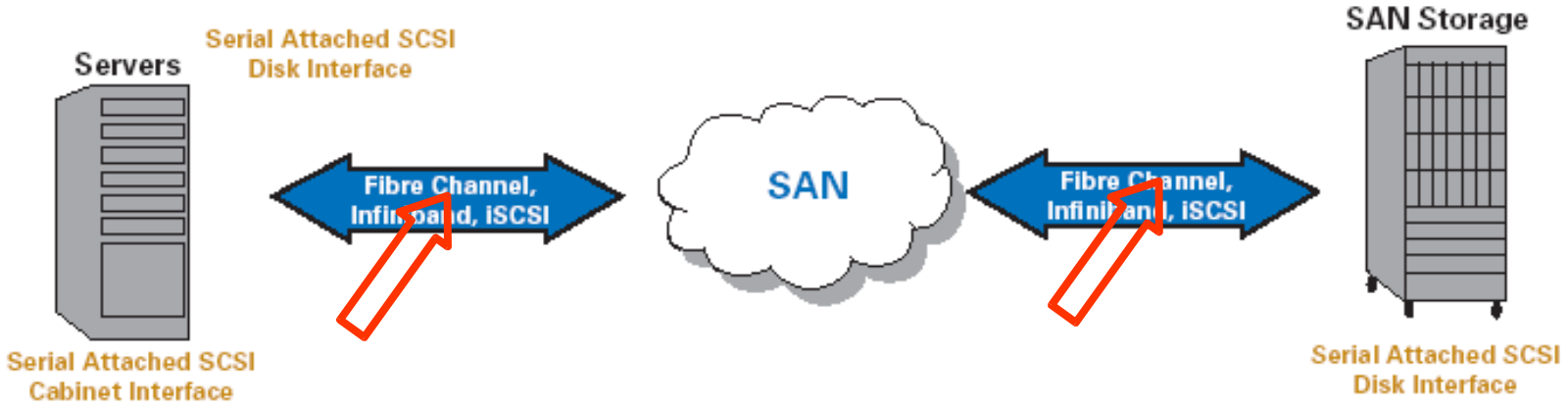
- Protocolo de transmissão serial usado para conectar discos rígidos a servidores
- Capaz de conectar até 192 discos rígidos com taxas de transmissão de até 80MB/s.
- Ela é tolerante a falhas no sentido de que defeitos em um único cabo não interrompem o acesso aos dados.
- SSA caiu em desuso com o surgimento do padrão *Fibre Channel*.

Fibre Channel

- *Fibre Channel* é tanto um padrão de comunicação quanto um protocolo de transporte aberto

- Implementado em fibras óticas ou fiações de cobre, sendo, portanto, um padrão para meios físicos de comunicação.
- Amplamente utilizado na conectividade física do que é conhecido hoje como "*SANs – Storage Area Networks*", ou seja, redes especializadas na interconexão de dispositivos de armazenamento a servidores

Fibre Channel



TRACE CACHE

Mário Luiz Rodrigues Oliveira
mario.oliveira@students.ic.unicamp.br
IC/UNICAMP
JUNHO/2006

MOTIVAÇÃO

- Melhorar desempenho dos processadores superescalares
- Dificuldades no uso de ILP para melhorar desempenho de processadores superescalares
- Necessidade de aumentar bandwidth do mecanismo de busca dos processadores superescalares

MOTIVAÇÃO

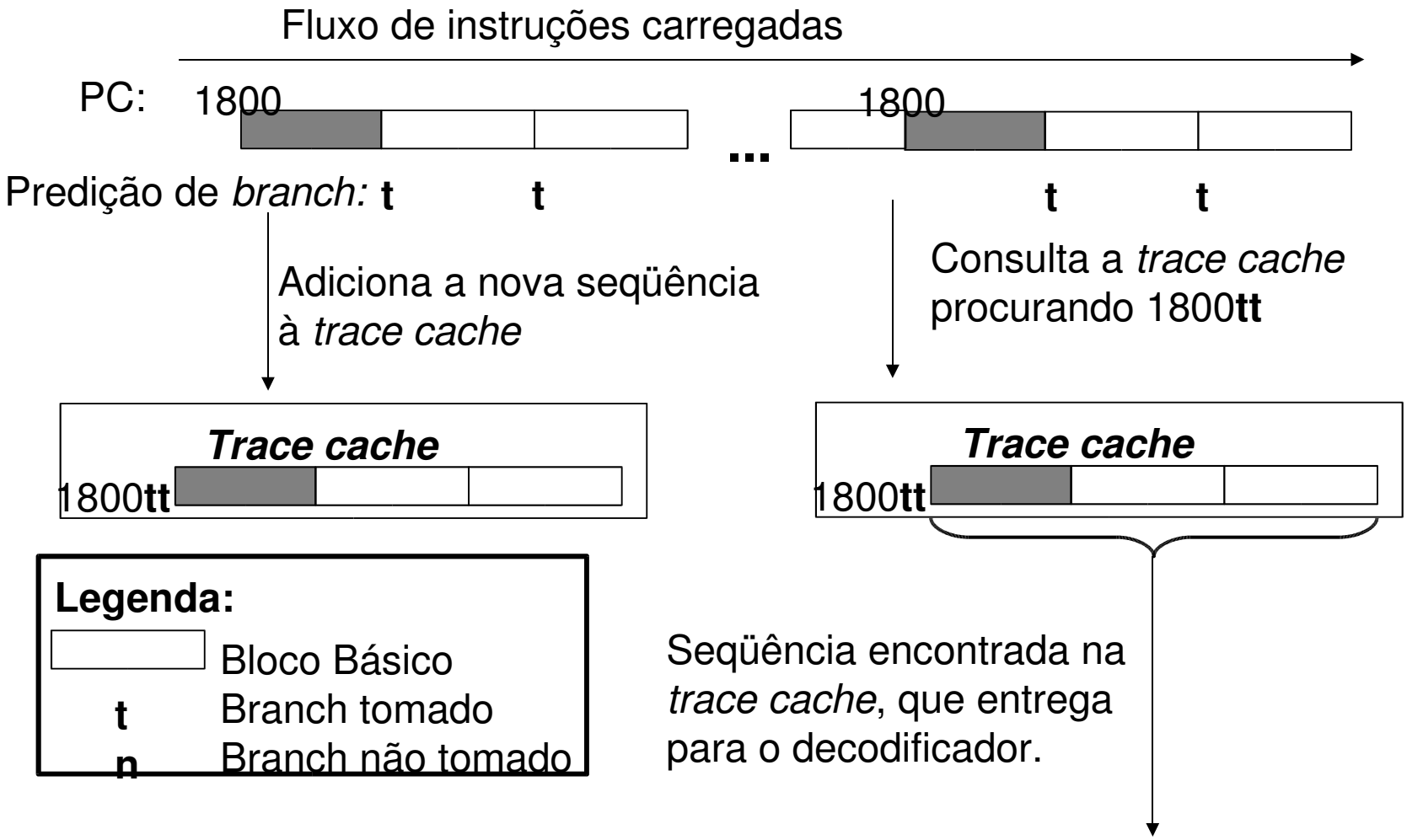
Benchmark SPEC92 para inteiros

Benchmark	***Branch* tomados %**	**Tam. médio do bloco básico**	**Número de instruções entre *branch* tomados**
Eqntott	86.2%	**4.20**	4.87
Espresso	63.8%	**4.24**	6.65
Xlisp	64.7%	**4.34**	6.70
Gcc	67.6%	**4.65**	6.88
Sc	70.2%	**4.71**	6.71
Compress	60.9%	**5.39**	8.85

Um bloco básico é um trecho do programa que não apresenta instruções de desvio, a não ser eventualmente a última instrução, que pode ser uma instrução de desvio.

SOLUÇÃO: TRACE CACHE

• Proposta por Rotenberg em 1996
• Cache adicional à cache de instruções
• Captura e armazena seqüências de instruções (*traces*) durante a busca de instruções
•Uma *trace* é uma seqüência de no máximo **n** instruções e no máximo **m** blocos básicos, iniciando em qualquer ponto da seqüência dinâmica de instruções. O limite **n** é o tamanho da linha da *trace cache* e **m** é o *throughput* do preditor de *branches*
•Uma *trace* é completamente especificada através de um endereço inicial dado pelo *Program Counter* e de uma seqüência de saídas dos **m - 1** preditores de *branches*, os quais indicam o caminho a ser seguido
•Durante a execução se estas seqüências se repetirem a busca da instrução é feito da trace cache


Fluxo de instruções carregadas
PC:
1800
1800
...
Predição de *branch:* t t t t
Adiciona a nova seqüência à *trace cache*
Consulta a *trace cache* procurando 1800**tt**
Trace cache
1800**tt**
Trace cache
1800**tt**
Legenda:
Bloco Básico
t Branch tomado
n Branch não tomado
Seqüência encontrada na *trace cache*, que entrega para o decodificador.

TRACE CACHE NO PENTIUM 4

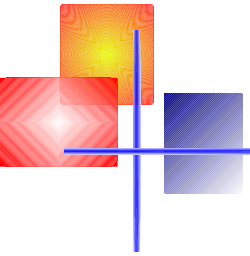
- A arquitetura *NetBurst* da *Intel* substituiu a cache de instrução convencional por uma *trace cache*
- Armazena na *trace cache* micro-operações e não instruções IA-32.
- A *trace cache* armazena até 12000 micro-operações e apresenta desempenho similar à uma cache de instruções convencional de tamanho entre 8 Kbytes e 16 Kbytes

DESEMPENHO

- A implementação original de *Rotenberg* obteve melhorias de 28%, na média, usando *benchmarks* para programas de inteiros
- *Trace cache* seletivas
- *Trace cache* baseada em blocos

CONCLUSÃO

- *Trace cache* é uma solução viável
- *Dependência da saída do preditor de* branches *não é um fator preocupante*

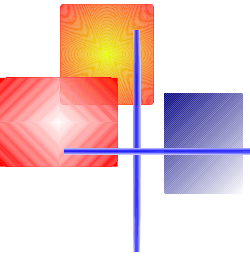


– Software Pipelining –

Uma técnica para paralelização de Loops

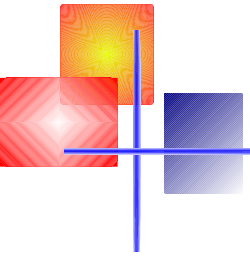
Carla Geovana Macário

Junho, 2006



Roteiro

- Visão Geral
- Suporte necessário
- Algoritmos existentes
- Estado Atual
- Conclusões



O que é?

- Técnica para prover paralelismo à execução de loops
- Busca sobrepor operações de diferentes iterações aumentando o paralelismo

Loop Unrolling X Software Pipeling

- Loop Unrolling
  - Desenrolamento do loop: programa seqüencial

```
for i from 1 to 100 do
    a[i] := a[i] + b[i]
    i := i + 1
```

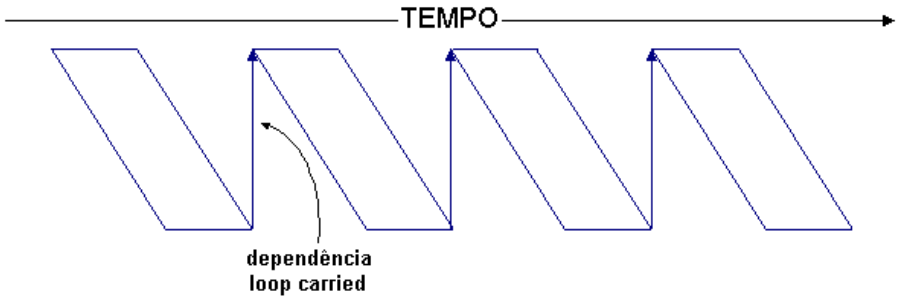
Loop Original

```
for i from 1 to 100 do
    a[i] := a[i] + b[i]
    a[i+1] := a[i+1] + b[i+1]
    a[i+2] := a[i+2] + b[i+2]
    a[i+3] := a[i+3] + b[i+3]
    i := i + 4
```

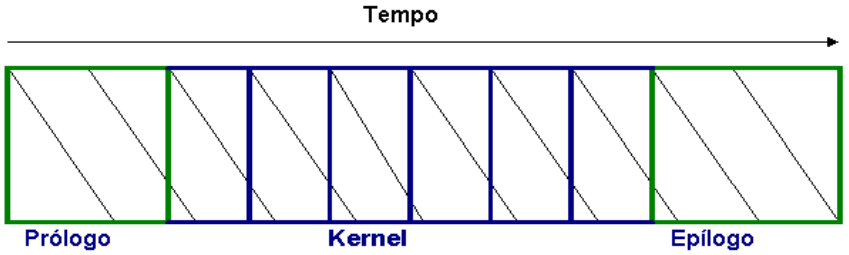
Loop Unrolling

Loop unrolling X Software Pipeling

- Software Pipeling
  - Reforma do loop para agrupar operações que possam ser executadas em paralelo
  - $\{ABC\}_n$ é semanticamente igual a $A\{BCA\}_{n-1}BC$
  - BCA é o novo corpo do loop (kernel)

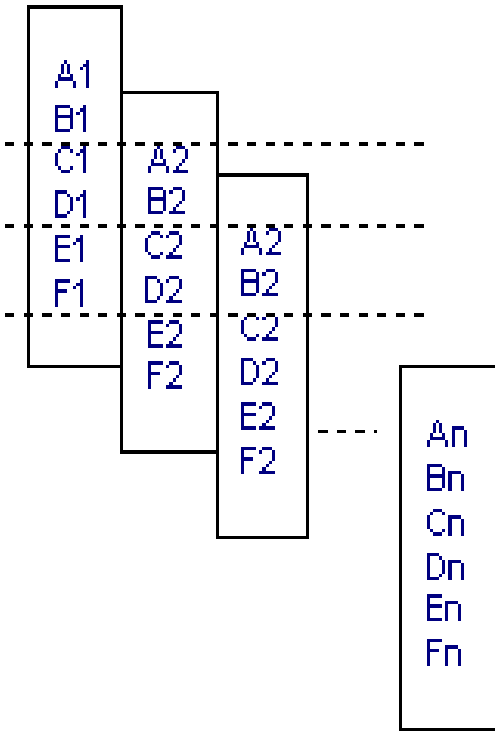


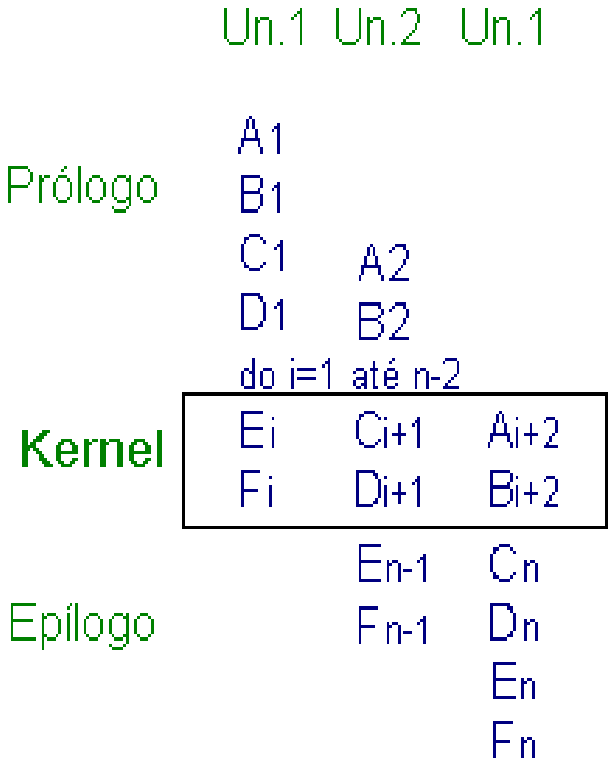
Loop sem software pipelining



Loop com software pipelining

Software Pipeling





Visão Geral

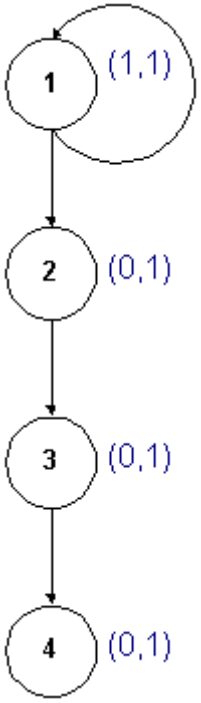
- Derivado do trabalho de [Patel & Davidson 1976]
- Rau e Aiken desenvolveram o 1o. Compilador que tirava proveito das arquiteturas paralelas.

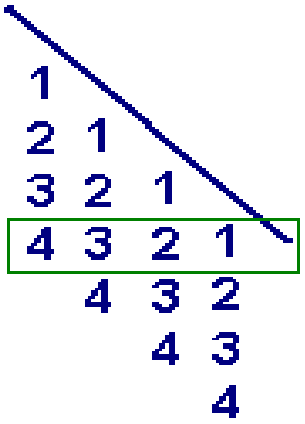
- **Objetivo principal:**
  - Identificar o novo corpo do Loop (kernel)
-
- Restrições a serem respeitadas nesta atividade:
  - Dependência de dados entre iterações
  - Uso de recursos

Suporte ao Sw Pipelining

- Grafo de dependência de dados (DDG)
  - Fornece a dependência entre operações e as latências existentes entre elas

```
for i from 1 to n do
        a[i+1] := a[i] + 1
    b[i] := a[i+1] /2
    c[i] := b[i] +3
    d[i] := c[i]
    i := i + 1
```





Restrição de Recursos

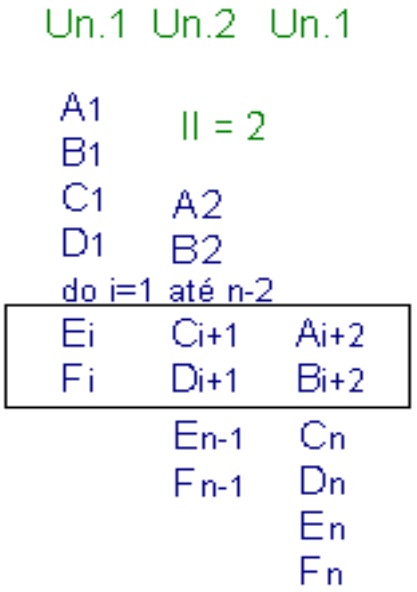
- 2 operações O1 e O2 estão em conflito se requerem o mesmo recurso ao mesmo tempo
- Tabela de reserva de recursos:
  - usada para checar conflitos de recursos durante o

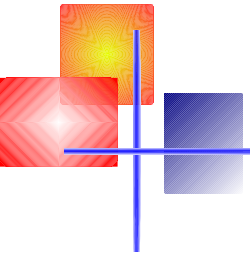
Time	Source 1	Source 2	ALU		Multiplier				Result Bus
			Stage 0	Stage 1	Stage 0	Stage 1	Stage 2	Stage 3	
0	X	X							
1			X						
2				X					
3									X

Time	Source 1	Source 2	ALU		Multiplier				Result Bus
			Stage 0	Stage 1	Stage 0	Stage 1	Stage 2	Stage 3	
0	X	X							
1					X				
2						X			
3							X		
4								X	
5									X

Intervalo de Iniciação (II)

- Intervalo de tempo necessário entre cada início de iteração
- Quanto menor o II, maior o throughput (maior paralelização)
- Dependente da restri ; e da dependência de dados.





Calculando o *II mínimo (MII)*

- Determinar a restrição de recursos é simples
- Determinar o escalonamento considerando a restrição de dependência de dados é um problema NP-Completo
- Técnicas para isso:
  - Enumeração exaustiva de todos os ciclos simples
  - Todos os pares do algoritmo de menor caminho
  - Algoritmo iterativo do menor caminho: fecho transitivo
  - Programação linear

Classificação dos Algoritmos

Escalonamento Modulo:

- faz um escalonamento e usa movimentação de código para melhorar o kernel encontrado

Identificação do Kernel:

- Promove um desenrolamento do loop para identificar o conjunto de instruções que podem compor o kernel

Escalonamento Modulo

- Proposto por [Rau & Glaeser 1981]
- Principais algoritmos desenvolvidos:
  - Redução Hierárquica [Lam 1988]
    - Variável de expansão
    - Bastante difundido, sendo uma referência até hoje
  - Escalonamento Predicado
    - Arquivo rotacional

Identificação do Kernel

- Perfect Pipeling [Aiken & Nicolau 1988]
  - combina movimentação de código com escalonamento para melhorar o paralelismo, buscando adequar o problema à arquitetura em uso
- PetriNet model
  - uma variação do anterior, usando redes de petri para resolver o problema de reconhecimento do kernel
- Vegdahl's
  - um método exaustivo que considera todas as soluções possíveis de escalonamento para escolher a melhor delas.

Identificação do Kernel

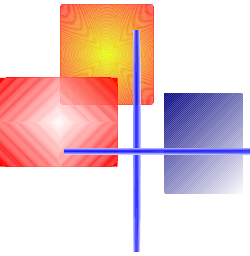
- Enhaced Pipeline
  - extensão do Perfect Pipelining, desenvolvido para tirar proveito de arquiteturas que suportam a execução multi-predicadas
  - usa movimentaçao de código, mas retém o corpo do loop, deixando de ser necessária a identificação do kernel

Comparação dos Algoritmos

Algoritmo	*Classe*	*Contribuição*	*Vantagem*	*Desvantagem*
Reduçao hierárquica.	ME	- aplicável a todos os loops - expansão de variável	Gera escal. próximos de ótimo	Expansão de código
Escalonamento Predicado	ME	- uso de arquivos rotivos para alocação de registradores	Escalonamento após análise de todas as operações	Execução de todas as instruções
Perfect Pipeline	IK	- adequar o escalonamento aos recursos disponíveis	Gera escalonamentos ótimos após movimentação do código	Usa técnicas ad hoc que limitam sua eficiência
Rede de Petri	IK	- usa rede de Petri para identificar as operações que estão prontas para serem executadas	provê maior formalismo ao Perfetc Pipeling	Dificuldade em encontrar o kernel
Técnica de Vergdahl	IK	- fornecer heurísticas para outros algoritmos	Considera todas as soluções possíveis para o escalonamento	Complexidade exponencial para escalonamento
Enhanced Pipeline	IK	- desenvolvido para tirar proveito de arquiteturas que suportam a execução multi-predicada	Retém o corpo do loop, sem necessidade de identificação do kernel	Difícil entendimento

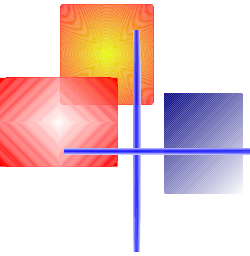
Estado Atual

- Nenhum novo algoritmo de impacto foi proposto.
- Observa-se apenas trabalhos buscando melhorar pontos específicos nos algoritmos existentes
- Exemplo:
  - alocação de registros entre iterações: [Chabin et al. 2005] e [Rong et al. 2005]
  - Redução do tamanho do código: [Zhuge et al. 2003] e o de [Kim et al. 2003]
- Retrospectiva de Lam [Lam 2004]: software pipelining é efetivo em máquinas VLIW sem exigir suporte complicado de hardware .



Conclusão

- Software pipelining é uma técnica importante para melhorar desempenho de programas via paralelismo
- Não é simples promover este paralelismo: problema NP-Completo
- Diversos algoritmos para sw pipelining, poucos de impacto
- Software pipelining dirige o projeto de algumas arquiteturas, mas não exige hardware muito sofisticado
- Área estratégica para investir: Lei Amdahl



Dúvidas?

Título do Trabalho:
Arquitetura Raw

Ricardo M Nishihara

RA 936161

Razões para o sucesso dos ASICs na exploração de paralelismo

- **Especilização de operadores:**
  - ASICs especializam operações da aplicação no nível de gate. Ex.: op. de ponto-flutuante em hardware dedicado pode ser executada em um ou alguns ciclos de relógio.
- **Maior utilização de recursos em paralelo.**
  - Aceleradores gráficos em ASICs: 100s ~ 1000s de ops de baixa granularidade / ciclo
- **Gerenciamento dos *wire delays***
  - *Minimização de latências:*

  placement adequado de operadores, canais de comunicação dedicados
- **Gerenciamento de pinos**
  - Inexistência de gargalos devido a sistema de memória
  - E/S mais eficiente e aderente à aplicação para diferentes dispositivos: conversores A/D, CCD, array de sensores, etc

Objetivo da Arquitetura Raw

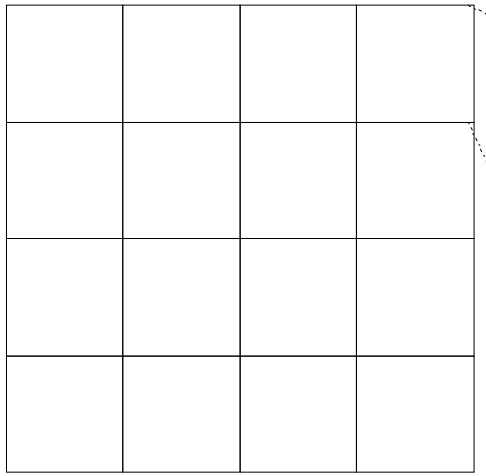
- O objetivo principal do projeto Raw:

  Propor uma arquitetura que viabilizasse novos microprocessadores de propósito geral, capazes de executar com bom desempenho tanto aplicações sequenciais típicas (via exploração de ILP), quanto um maior número de aplicações "streaming" com alto grau de paralelismo, e que até aqui vem sendo implementadas principalmente através de hardware dedicado.

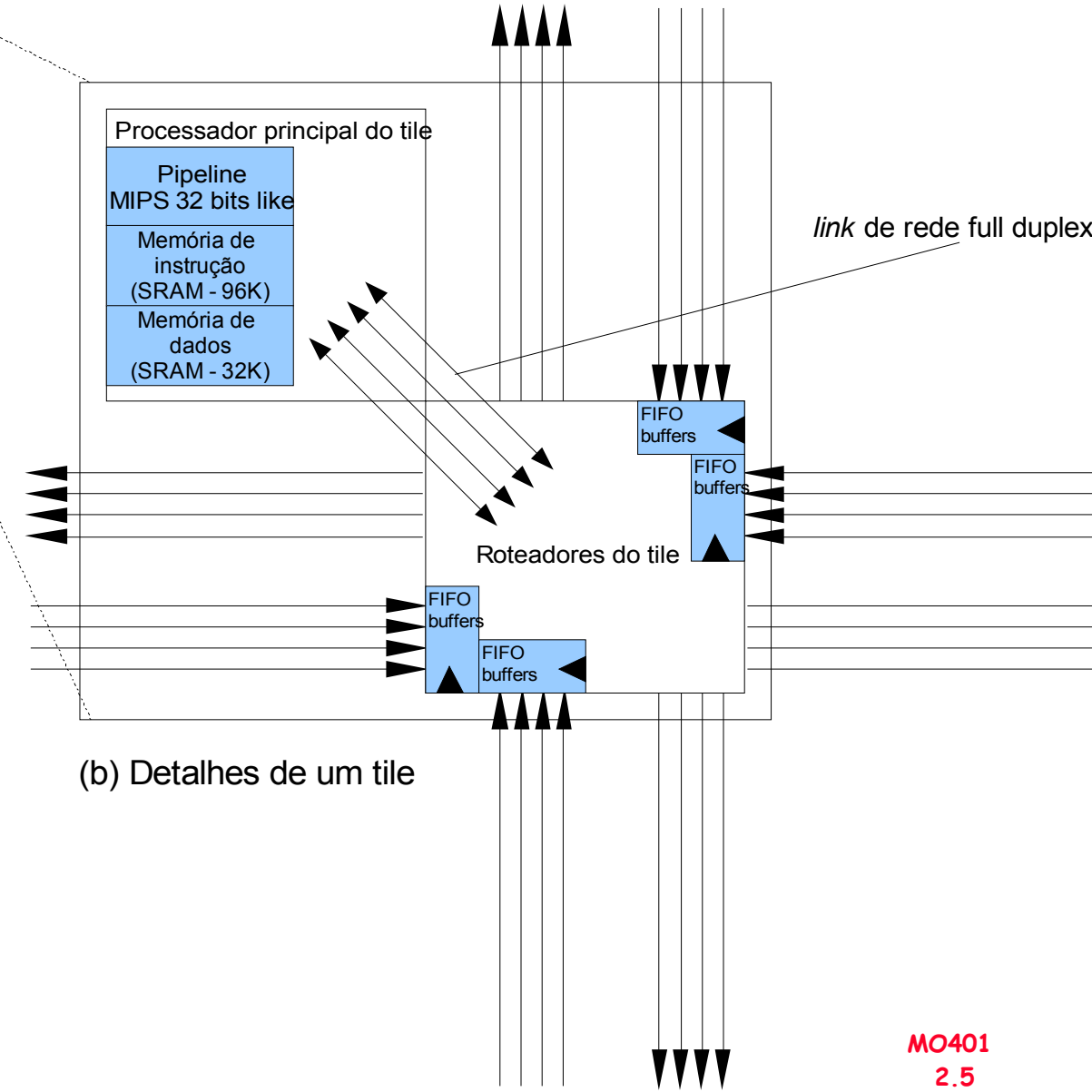
Como Raw atende os quatro fatores ?

- 1. Especialização: Implementação de operadores requeridos para a exploração de ILP em aplicações seqüênciais, e/ou paralelismo "streaming".
- 2. Disponibilidade de recursos em paralelo: replicar em grande número tais operadores, e expô-los ao software através do ISA.
- 3. Gerenciamento do wire delay: expondo ao software operadores relativos aos canais de comunicação interligando estas unidades.
- 4. Gerenciamento de pinos: As abstrações expostas pelo ISA referentes aos "pinos", permitem o gerenciamento via software de sistemas de memória cache e de interfaces E/S de alto desempenho.

Arquitetura Raw
Array Mesh 2D de tiles



(a) Processador Raw – conjunto de tiles idênticos

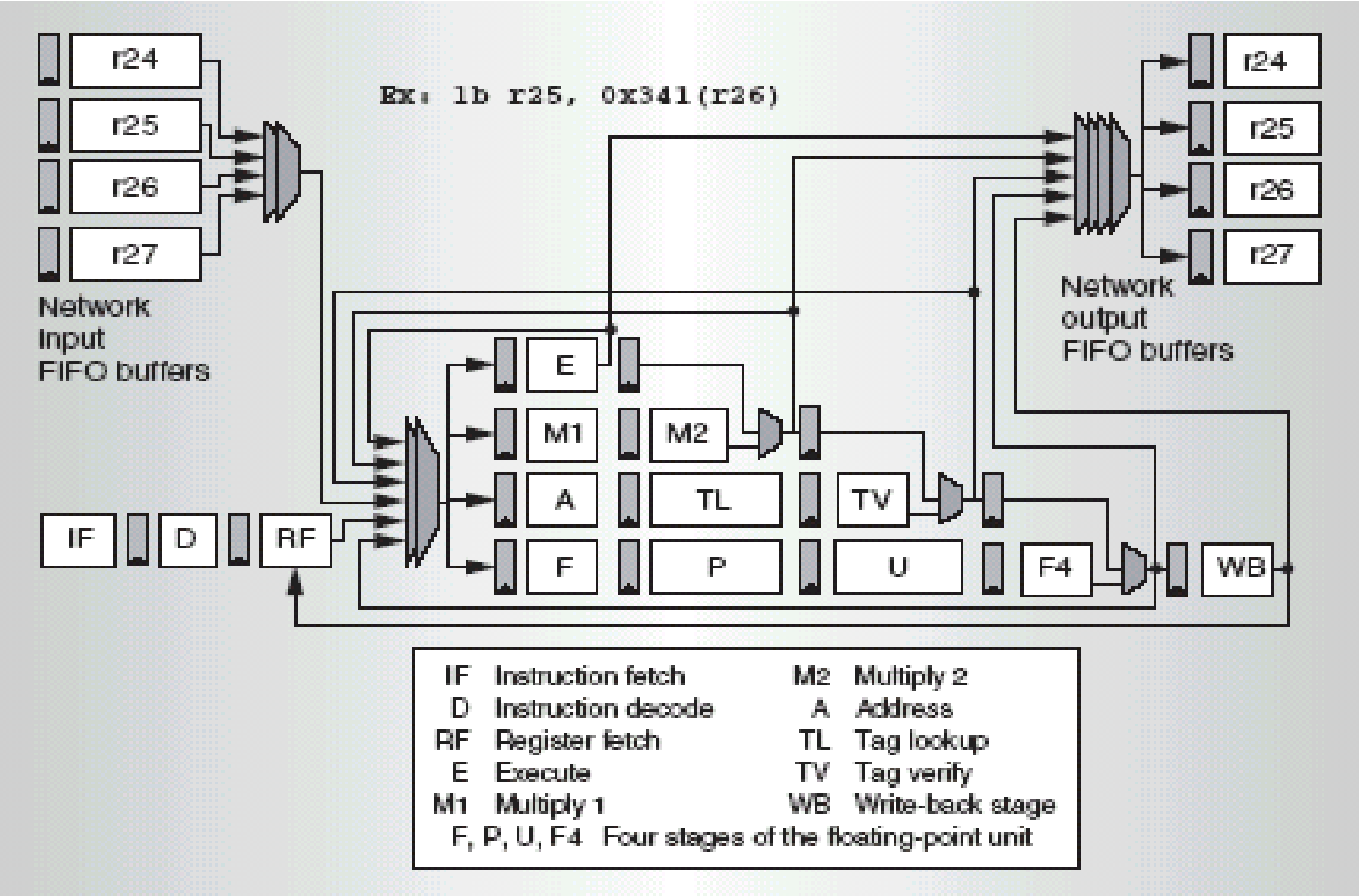


(b) Detalhes de um tile

Estrutura do tile

- Cada tile contém:
  - Um Processador MIPS *like* de 32 bits
    - » Pipeline de 8 estágios, in-order, single issue
    - » Unidade ponto-flutuante com pipeline de quatro estágios
  - 32 Kbytes data cache
  - 96 Kbytesmemória de instruções (cache via software)
  - Um roteador estático programável
  - Dois roteadores dinâmicos programáveis
- Cada tile é conectado apenas a seus quatro tiles vizinhos mais próximos (norte, sul, leste e oeste)

Processador Pipeline do Tile



Exposição em software dos ports da rede de comunicação entre tiles

Registrador	Alias	Descrição
$24	$csti	Port de entrada para a rede estática
$25	$csgn[i/o]	Port de E/S para a rede dinâmica de uso geral
$26	$csti2	Segundo port de entrada para a rede estática
$27	$cmn[i/o]	Port de E/S para a rede dinâmica reservada à memória

Tabela 1 – Mapeamento de registradores aos *ports* das redes estáticas e dinâmicas

```
    # XOR register 2 with 15,
# and put result in register 31
xori $31,$2,15
# get two values from switch,
# add to register 3, and put
# result in register 9
addu $9,$csti2,$csti
# an ! indicates that the result
# of the operation should also
# be written to $csto
and! $0,$3,$2
# load from address at $csti+25
# put value in register 9 AND
# send it through $csto port
# to static switch
ld! $9,25($csti)
# jump through value specified
# by $csti2
jr $csti2
```

Figura 3 – Trecho de código assembly ilustrando o acesso às interfaces com as redes para comunicação entre tiles

Roteador estático

- **Roteador estático:**
  - Processador pipeline de 5-estágios + dois crossbar switchers + duas redes físicas
  - Cada crossbar rotea valores entre sete entidades: o processador pipeline do roteador estático, os quatro tiles vizinhos (norte, sul, leste e oeste), o processador principal do tile, e o outro crossbar.
  - Conjunto de instruções com palavras de largura de 64 bits. Cada instrução codifica um pequeno comando (ex.: branch) e 13 rotas (uma para cada saída de crossbar)
  - Memória de instruções com 8096 palavras e implementa um conjunto de instruções restrito com palavras de largura de 64 bits.
- **Flow control:**
  - O roteador estático prossegue para a próxima instrução somente depois que todas as rotas de uma dada instrução são completadas.

Redes estáticas

- O conjunto dos roteadores estáticos de todos os tiles implementam duas redes estáticas de comunicação entre tiles.
- Estática = rota é estabelecida em tempo de compilação

Para cada palavra de dado transmitida entre *tiles* sobre a rede estática, deve existir uma instrução na memória de instrução de cada roteador estático, que faz parte do caminho percorrido pela palavra de dado.

a memória de instrução do roteador é virtualizada
tem-se portanto uma enorme flexibilidade na criação de novos padrões de comunicação entre tiles

Redes dinâmicas

- Cada tile conta também com 2 roteadores dinâmicos
- Considerados eles em conjunto tem-se um par de redes dinâmicas para comunicação entre tiles.
- Dinâmica = rotas estabelecidas em tempo de execução
- Uso:
  - envio de uma palavra de cabeçalho especificando:
    - » o tile destino (ou port de E/S),
    - » um campo de usuário e o
    - » comprimento da mensagem.
  - O usuário pode enviar até 31 palavras de dados em uma mensagem.

Redes dinâmicas - Deadlock

- Podem ocorrer deadlocks nos acessos aos buffers da rede (roteadores). Como resolver ?
  - Eliminação: requerer dos usuários que restrinjam o modo de uso a um conjunto de práticas comprovadamente livres de deadlock.
  - Recuperação: não restrinje o modo, mas deadlock tem de ser detectado e os dados nos buffers em deadlock tem de ser removidos para memória externa a rede => implementação custosos e talvez não disponíveis
- Solução Raw
  - Uma rede é dedicada a acesso a memória (e E/S), e
    - » Usa eliminação de deadlock
    - » Apenas clientes c/ privilégio podem utilizá-la: SO, Interrupção, cache de software,...
  - A outra rede de uso geral e acesso irrestrito
    - » Usa recuperação de deadlock: contador no processador do tile + interrupções + rede dedicada à acesso a memória

Superescalar x Raw x VLIW

- Superscalares: mecanismos de controle dinâmico são providos por unidades de hardware tais como caches, buffers para predição de desvios, lógica para register renaming, e unidades para escalonamento de instrução.
- Raw: não incorpora tais mecanismos no HW, a orientação é movê-los para o software (similar ao VLIW) => maximizar o número de tiles no chip !
- VLIW x Raw: No Raw cada tile tem seu próprio fluxo de instruções => maior flexibilidade, sem lock steps.

Suporte de software – Run time software

- Em um processador Raw funcionalidades como (como caches, predição de desvios, register renaming, escalonamento de instrução) devem ser providas pelo run-time software do sistema e/ou compilador.
- Os custos adicionais de se implementar tais serviços em software precisam ser reduzidos, e esta redução pode vir tanto através do compilador, a partir da eliminação de operações desnessárias, quanto a partir de algoritmos melhores de suporte.
- O desempenho geral de um sistema Raw depende do compromisso de alguns fatores. Em razão do suporte a comportamentos dinâmicos ter sido movido para o software, um programa executará mais instruções Por outro lado, o hardware mais simples da arquitetura Raw poderá rodar a freqüências de clock maiores, e disporá de mais recursos computacionais para explorar o paralelismo da aplicação.

Suporte de software – Compilador Raw

- Alocação de recursos

(similar ao placement dos ASICs)

  - o programa é dividido em regiões paralelas de maior granularidade.
  - cada região é executada em múltiplos tiles - conjunto de tiles = processador lógico único.
  - o número de tiles alocados em uma região depende do paralelismo de granularidade fina nesta região.
  - tiles dentro de uma dada região comunicam-se via comunicações estáticas; as regiões comunicam-se entre si usando mensagens dinâmicas.

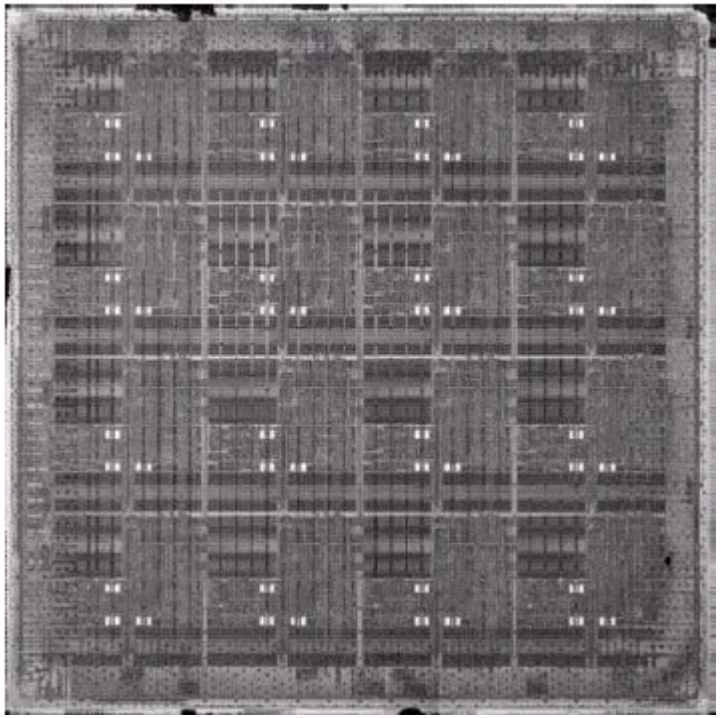
Suporte de software – Compilador Raw

- Exploração de paralelismo de granularidade fina
  - Oportunidade provida pela arquitetura Raw, devido às baixas latências associadas a comunicação entre tiles via rede estática
  - Múltiplos tiles podem trabalhar juntos para explorar o paralelismo no nível de instrução - ILP de um fluxo único de instruções.
  - O compilador toma este fluxo único de instruções como entrada, particiona-o em multiplos fluxos de instruções, mapea cada fluxo particionado para um tile, e finalmente escalona a comunicação estática entre os fluxos particionados.

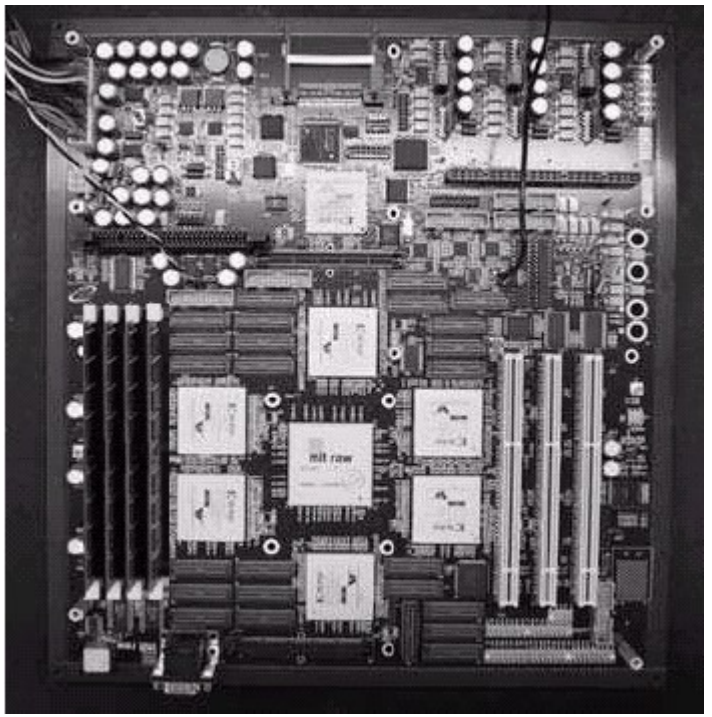
Suporte de software – Compilador Raw

- **Escalonamento de comunicação**
  - Devido a presença da rede estática, um compilador para arquitetura Raw enfrenta um problema mais complexo: ele tem de escalonar tanto instruções quanto eventos de comunicação, sendo que ambos tipos de eventos têm de ser escalonados temporalmente e espacialmente.
  - Correção e desempenho:
    - » Com relação à correção, o código de roteamento tem de ser livre de deadlock.
    - » Com relação ao desempenho, o compilador deve ser capaz de organizar cuidadosamente a comunicação de maneira a minimizar a ocorrência de stalls ao longo da rede.
    - » Estes dois aspectos tem ainda sua complexidade aumentada na presença de eventos dinâmicos como desvios, cash misses, e mensagens dinâmicas.

Raw Chip



- Array de 16 tiles
- Processo ASIC da IBM SA-27E (0.15-micron, 6 níveis, cobre).
- Die: 18.2 x 18.2 mm
- encapsulamento *CCGA* de 1657 pinos, com 1080 pinos de E/S HSTL
- O chip consome em média 18.2 watts 425MHz/500Mhz c/ 1.8V/2.2V.



Avaliação: Comparação Raw x Pentium 3

- Comparação direta com Pentium 3: não esconde ineficiênciais do compilador e da arquitetura.
- Facilita a comparação indireta da arquitetura Raw com outras alternativas desenvolvidas por outros grupos, uma vez que, a comparação de um dada alternativa com o mesmo sistema comercial de referência pode ser extentida para uma comparação com a arquitetura Raw.

Parameter	Raw (IBM ASIC)	P3 (Intel)
Lithography Generation	180 nm	180 nm
Process Name	CMOS 7SF (SA-27E)	P858
Metal Layers	Cu 6	Al 6
Dielectric Material	SiO_2	SiOF
Oxide Thickness (T_{ox})	3.5 nm	3.0 nm
SRAM Cell Size	4.8 μm^2	5.6 μm^2
Dielectric k	4.1	3.55
Ring Oscillator Stage (FO1)	23 ps	11 ps
Dynamic Logic, Custom Macros (SRAMs, RFs)	no	yes
Speedpath Tuning since First Silicon	no	yes
Initial Frequency	425 MHz	500-733 MHz
Die Area[2]	331 mm^2	106 mm^2
Signal Pins	~ 1100	~ 190
Vdd used	1.8 V	1.65 V
Nominal Process Vdd	1.8 V	1.5 V

- Pentium 3 era a implementação de processador Intel mais próxima:
  - > processo com a mesma geração de litografia 180 nm
  - > latências associadas às unidades funcionais muito próximas

Classes de aplicações testadas

- 1. ILP: aplicações sequenciais convencionais - paralelismo disponível nestas aplicações é o ILP.
- 2. Stream: Nesta classe foram colocadas aplicações "streaming" que lidam com grandes conjuntos de dados, ou podem manipular fluxos contínuos de dados em tempo-real.
- 3. Server: Spec2000 em cada um dos tiles em rel. A uma exec. no P3.
- 4. Bit-Level: Inclui duas aplicações manipulação no nível de bit em geral implementadas em FPGA e ASIC: 802.11a ConvEnc, 8b/10b Encoder.

Resultados

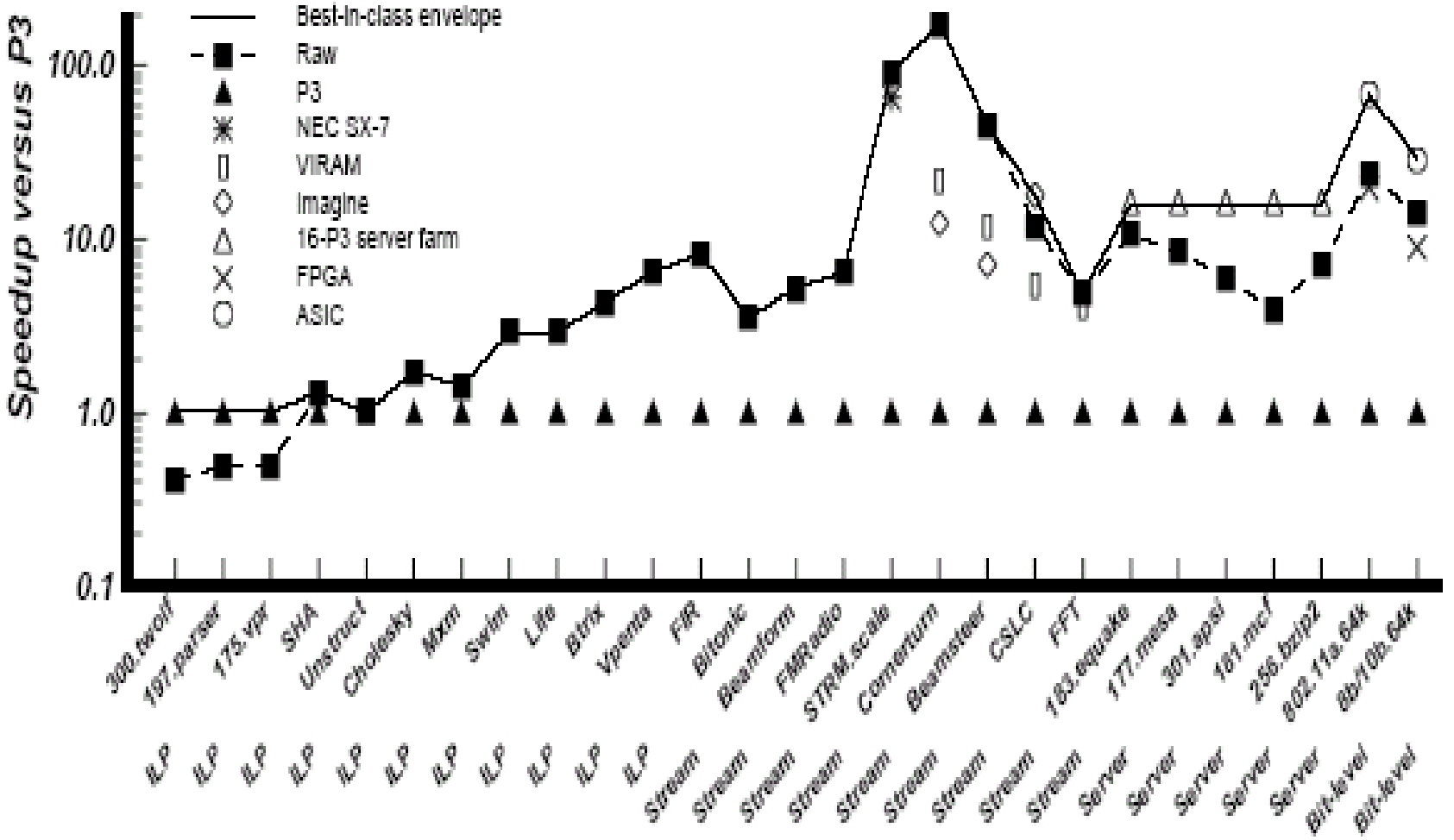


Figura 5 – Desempenho comparativo para várias aplicações

Razões para o speedup

Classe	Benchmark	S	R	W	P
ILP	Swim, Tomcatv, Btrix, Cholesky, Vpenta, Mxm, Life, Jacobi, Fpppp-kernel, SHA, AES, Encode, Unstructured, 172.mgrid, 173.applu, 177.mesa, 183.equake, 188.ammp, 301.apsi, 175.vpr, 181.mcf, 197.parser, 256.bzip2, 300.twolf	X	X	X	
Stream	Beamformer, Bitonic Sort, FFT, Filterbank, FIR, FMRadio, Mxm, LU fact., Triang. solver, QR fact., Conv., Copy, Scale, Add, Scale & Add, Acoustic Beamforming, FIR, FFT, Beam Steering, Corner Turn, CSLC	X	X	X	X
Server	172.mgrid, 173.applu, 177.mesa, 183.equake, 188.ammp, 301.apsi, 175.vpr, 181.mcf, 197.parser, 256.bzip2, 300.twolf		X		X
Bit-level	802.11a ConvEnc, 8b/10b Encoder	X	X	X	

Utilização de funcionalidades Raw:
S = Especialização
R = Utilização de Recursor Paralelos
W = Gerenciamento dos atrasos dos condutores
P = Gerenciamento de Pinos

Conjunto de Instruções Multimídia para Processadores de Propósito Geral

MO401 – Trabalho 2

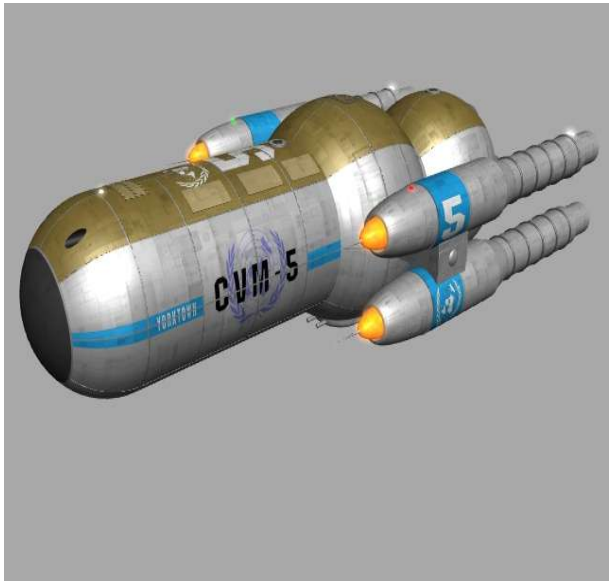
Fábio Augusto Menocci Cappabianco

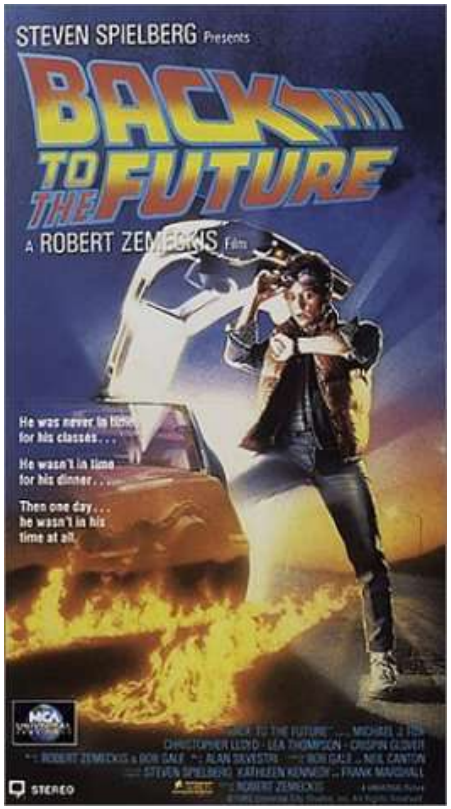
RA:991724

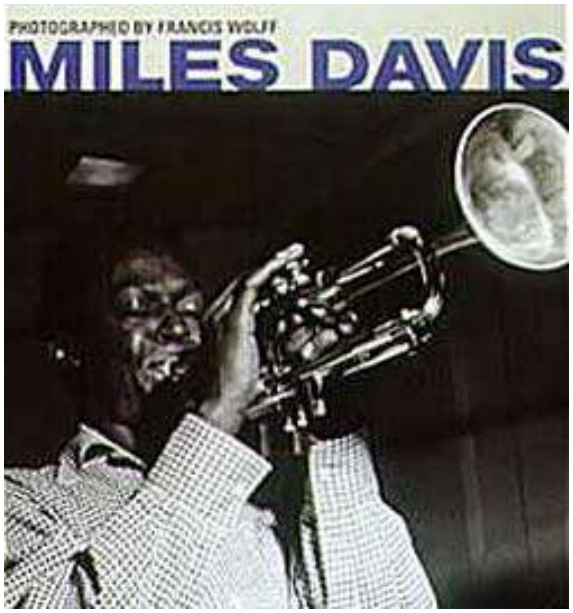
Índice

- Aplicações
- Tipos de Processadores
- Motivação
- Instruções
- Principais Extensões
- Instruções Multimídia

Aplicações Multimídia







Tipos de processadores multimídia

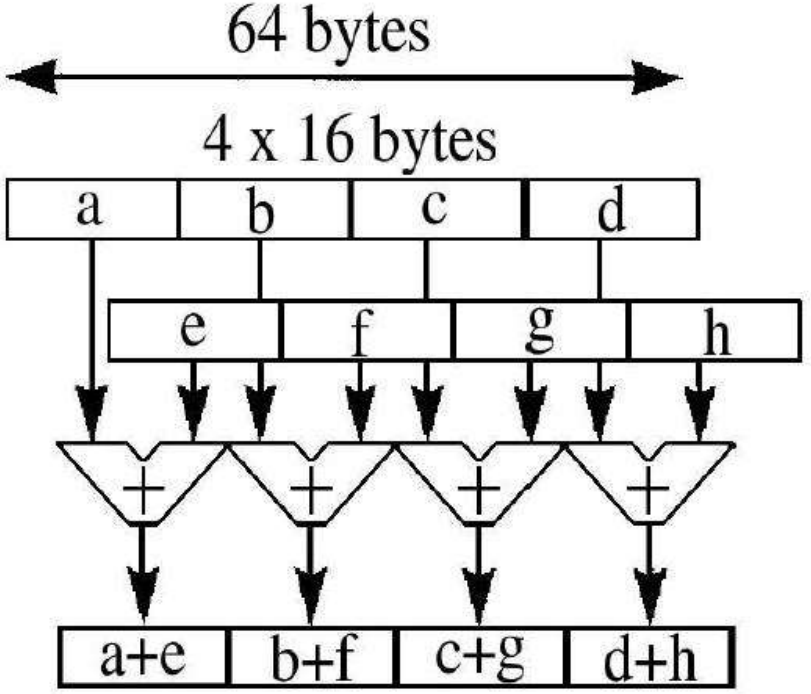
- Digital Signal Processor (DSP).
- Processadores de propósito geral.
- Processadores embarcados e de propósito específico.

Motivação para Conjunto de Instruções em Microprocessadores

- Aumento da demanda de aplicações multimídia.
- Economia de hardware externo.
- Praticidade.

Conjunto de Instruções Multimídia

- Dados de 8 e 16 bits.
- Processamento paralelo SIMD.
- Grande fluxo de dados.



Principais Extensões Multimídia

- MAX-1(1994), MAX-2 (1995) da HP.
- VIS(1995) da Sun.
- MDMX(1996), MIPS-64(1996), MIPS-3D(1999) da MIPS.
- MMX(1997),SSE(1999),SSE2(2000), SSE3(2004) da Intel.
- MVI(1997) DEC.
- Extended MMX(1997) da Cyrix.
- 3DNow!(1998), Enhanced 3DNow!(1999), 3DNow! Professional(2000) da AMD.
- AltiVec(1999) Motorola.

MAX-1, MAX-2, MVI

- Banco de registrador de inteiros.
- Poucas instruções 9, 8 e 13.
- Mais específicos, para áudio, 3D e vídeo

Conjunto de Instruções VIS

- 121 instruções.
- Primeiro a tratar aplicações gráficas sem hardware adicional.
- Foco em 3D e MPEG.
- Banco de registradores de ponto flutuante.
- Utilizado no UltraSPARC – I, II e III.

Conjunto de Instruções MDMX

- 29 instruções.
- Não implementado.
- Banco de registradores de ponto flutuante.
- Uso de acumulador.

Conjuntos de Instruções MIPS64, MIPS3D

- Complementares a MXMD de 74 e 23 instruções.
- Primeiro a tratar dados multimídia de ponto flutuante.
- Foco em geometria 3D.
- Banco de registradores de ponto flutuante.

Conjunto de Instruções MMX

- 57 instruções.
- O mais popular até SSE.
- Implementado no Pentium P55.
- Banco de registradores de ponto flutuante.

Conjunto de Instruções Extended MMX

- 12 instruções.
- A partir da licença MMX da Intel.
- Expansão das instruções para evitar destruir dados de registradores.
- Banco de registradores de ponto flutuante.

Conjunto de Instruções 3DNow!

- 21 instruções.
- A partir da licença MMX da Intel.
- Expandido para tratar dados multimídia de ponto flutuante.
- Utilizado no K6.
- Banco de registradores de ponto flutuante.

Conjunto de Instruções SSE

- 70 instruções.
- Mudança em arquitetura de novos registradores.
- Mudança em SO.
- Expandido para tratar dados multimídia de ponto flutuante.
- Adotado no Pentium III.

Conjunto de Instruções AltiVec

- 162 instruções.
- Primeiro com arquitetura de 128 bits para multimídia.
- Instruções de quatro operandos .
- Adotado no MPC 7400 da estação Apple G4.

Conjuntos de Instruções Enhanced 3DNow! e 3DNow Professional

- Complementares a 3DNow! de 24 e 70 instruções.
- Suporte a instruções SSE.
- Banco de registradores separado.
- Implementados nos processadores Athlon e Athlon MP.

Conjuntos de Instruções SSE2, SSE3

- Complementares a SSE com 74 e 13 novas instruções.
- Implementados nos processadores Pentium 4 e Pentium 4 Prescott.
- Adição de tratamento de ponto flutuante de dupla precisão e sincronismo de threads.

Instruções Multimídia

- Inteiras.
- Ponto flutuante.
- Polimorfas.
- Comparação e Controle de Fluxo.
- Memória

Instruções sobre Inteiros

- Conversão de tipos(pack, unpack).
- Saturação e resto.
- Adição e subtração.
- Soma das diferenças absolutas.
- Shifts.
- Multiplicações.
- Comunicação de dados (merge,align,insert,shuffle).

Resultado de Multiplicações

- Redução
  - Soma dos elementos particionados.
- Par/Impar
  - Elementos pares ou impares.
- Truncado
  - Bits perdidos.
- Registrador mais “largo”.
- Primitivas
  - Divisão em sub-instruções.

Instruções sobre Ponto flutuante

- Aritmética
- Divisão e inverso aproximados.
- Exceções tratadas por:
  - Resultado da operação,
  - Registradores de status/controle.

Instruções Polimorfas

- Independente do particionamento.
- Lógicas(AND,OR,NAND,...)
- Não necessárias para conjuntos de instrução que utilizam banco de registradores inteiro.

Comparação e Controle de Fluxo

- Comparação: =,<=,>,...
- Controle de fluxo: difícil de gerar exceção.
- Soluções:
  - Elementos máscara – não cria dependência.
  - Vetor de bits – um bit para cada elemento.

Instruções de Acesso à Memória

- Load/Store
- Não há operandos indicando endereços de memória.
- Pontuais, seqüenciais e em intervalos.