Capítulo 7

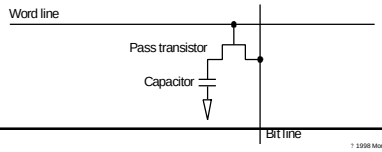
Lecture slides created by Michael Wahl in English

Tradução: Christian Lyra Gomes

Revisão: Wagner M. N. Zola

Memórias: Revisão

- **SRAM:**
  - **valor é armazenado em um par de portas inversoras**
  - **muito rápida mas toma mais espaço que a DRAM (4 a 6 transistores)**
- **DRAM:**
  - **valor é armazenando como uma carga em um capacitor (deve ser renovada)**
  - **muito pequena mas é mais lenta que a SRAM (fator de 5 a 10)**



Construindo a Hierarquia de Memória

- **Usuários querem memórias grandes e rápidas!**
  - **mas a que custo ?**

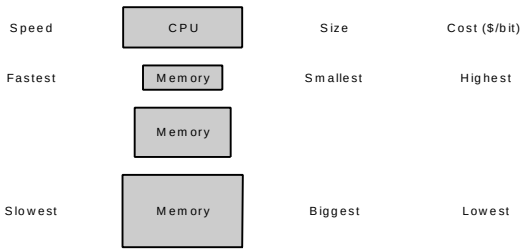
1997

SRAM tempo de acesso entre 2 - 25ns a um custo de $100 a $250 por Mbyte.
DRAM tempo de acesso entre 60-120ns a um custo de $5 a $10 por Mbyte.
Disk tempo de acesso entre 10 to 20 milhões de ns a um custo de $.10 a $.20 por Mbyte.

- **Memoria muito grande e rapida seria muito cara!**
  - **Seria possivel construir alguma coisa intermediària ?**
    - R: Construa uma hierarquia de memória

Construindo a Hierarquia de Memória

- **Ou: Como construir uma memória grande e rápida (na maioria dos casos) e barata?**
  - R: Construa uma hierarquia de memória

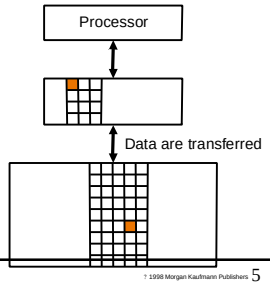


Construindo a Hierarquia de Memória

- **Ou: Como construir uma memória grande e rápida (na maioria dos casos) e barata?**
  - R: Construa uma hierarquia de memória

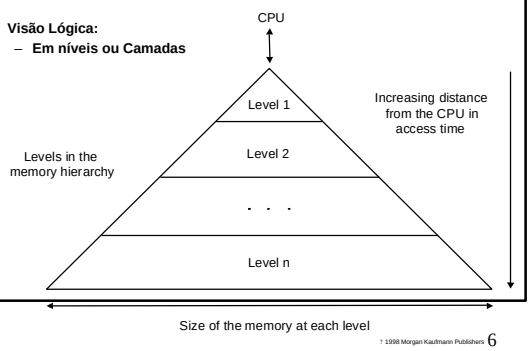
– **CPU acessa palavras ou bytes (no primeiro nível)**

– **Transferência entre níveis:**
  - Em blocos



Hierarquia de Memória

- **Visão Lógica:**
  – **Em níveis ou Camadas**



Localidade

- Um princípio que torna a hierarquia de memória uma boa idéia
- se um item é referenciado,

  localidade temporal: ele tende a ser referenciado novamente logo
  localidade espacial: itens próximos tentem a ser referenciados logo.

Por que o código tem localidade?

- Nosso foco inicial: dois níveis (superior, inferior)
  - bloco: unidade mínima de dados
  - Acerto (hit): dado requisitado está no nível superior
  - falta: dado requisitado não está no nível superior

Cache

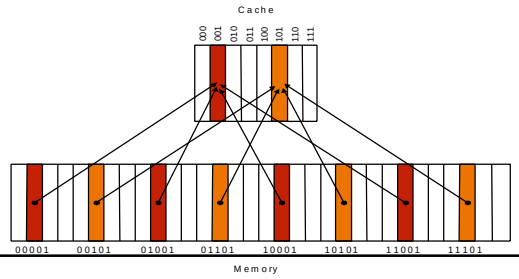
- Dois pontos:
  - Como nós sabemos se um item de dado está no cache?
  - Se ele estiver, como nós o achamos?
- Nosso primeiro exemplo:
  - tamanho do bloco é uma palavra de dado
  - "mapeado diretamente"

Para cada item de dado no nível inferior,
existe exatamente um local no cache onde ele deve estar.

e.g., muitos itens no nível inferior compartilham o mesmo local no nível superior

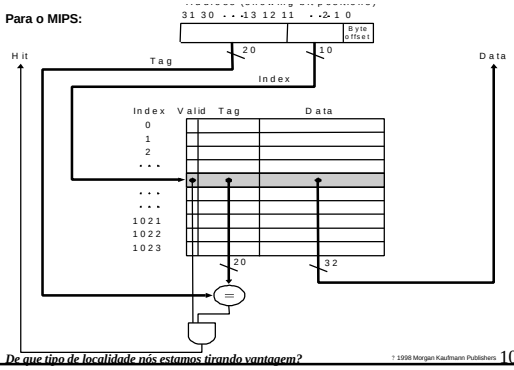
Cache Mapeado Diretamente

- Mapeamento: endereço é o módulo do número de blocos no cache



Cache Mapeado Diretamente

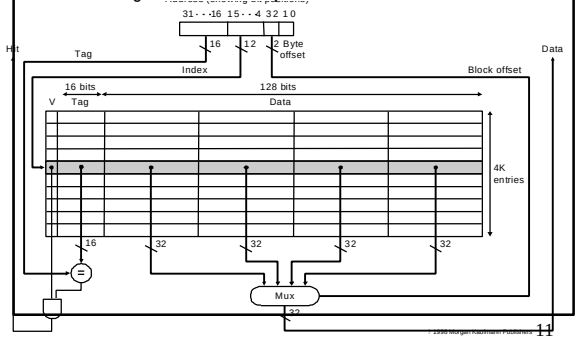
- **Para o MIPS:**



De que tipo de localidade nós estamos tirando vantagem?

Cache Mapeado Diretamente

- **Tirando vantagem da localidade espacial:**

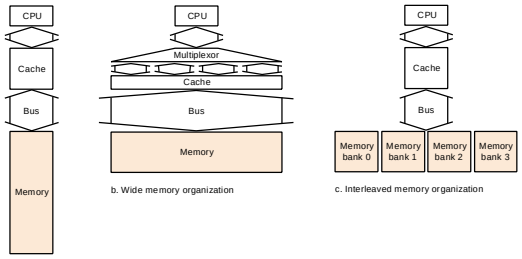


Acertos vs. Faltas

- **Acertos de leitura**
  - **isto é o que queremos!**
- **Faltas de leitura**
  - **paralisa a CPU, busca bloco da memória, entrega ao cache, reinicia**
- **Acertos de escrita:**
  - **pode trocar dados no cache e na memória (write-through)**
  - **escrever dados apenas no cache (write-back no cache depois)**
- **Faltas de escrita:**
  - **ler o bloco inteiro no cache, e depois disso escreve a palavra**

Pontos: Hardware

- **Tornar a leitura de múltiplas palavras mais fácil através do uso de bancos de memória**



a. One-word-wide memory organization

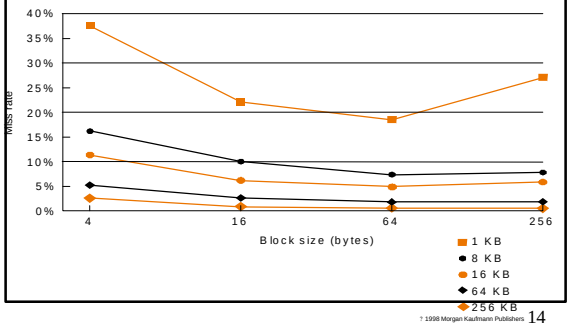
b. Wide memory organization

c. Interleaved memory organization

- **Isso pode se tornar muito mais complicado...**

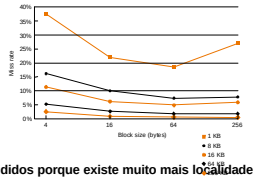
Desempenho

- **Aumento do tamanho do bloco tende a diminuir a taxa de falta:**



Desempenho

- **Aumento do tamanho do bloco tende a diminuir a taxa de falta:**



- **Use caches divididos porque existe muito mais localidade espacial no código:**

programa	tamanho do bloco em palavras	Taxa de Falta de Instruções	Taxa de Falta de dados	Taxa de Faltas efetiva combinada
gcc	1	6.1%	2.1%	5.4%
	4	2.0%	1.7%	1.9%
spice	1	1.2%	1.3%	1.2%
	4	0.3%	0.6%	0.4%

Desempenho

- **Modelo simplificado:**

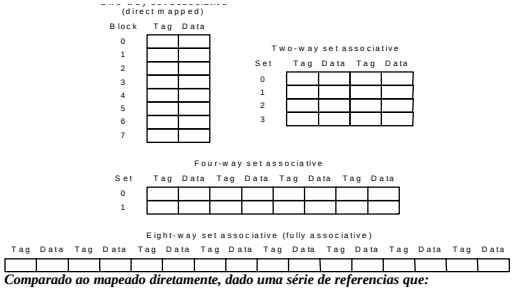
  tempo de execução = (ciclos de execução + ciclos paralisados) ? tempo de ciclo

  ciclos paralisados = # de instruções ? taxa de Faltas ? penalidade da Falta

- **Duas maneiras de melhorar o desempenho:**
  - **diminuir a taxa de faltas**
  - **diminuir a penalidade da falta**

O que acontece se aumentarmos o tamanho do bloco?

Diminuido a taxa de Faltas com Associatividade

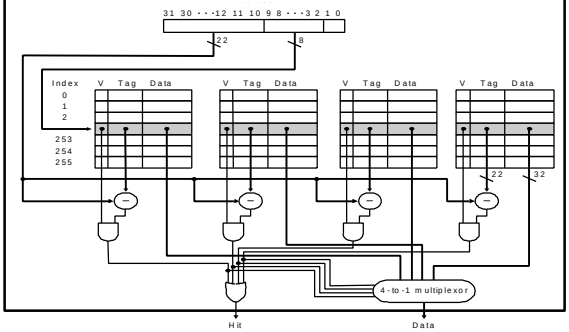


Comparado ao mapeado diretamente, dado uma série de referencias que:

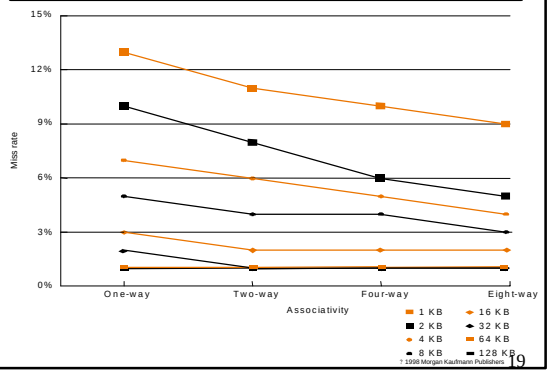
- ***resulta em uma taxa de Falta menor usando um cache associativo de grau 2***
- ***resulta em uma taxa de Falta maior usando um cache associativo de grau 2***

assumindo que usamos a estratégia de reposição do "menos recentemente usado"

Uma implementação



Desempenho

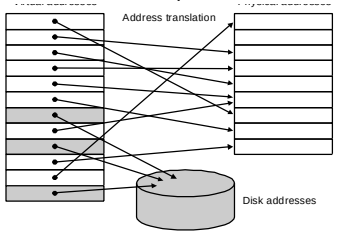


Diminuindo a penalidade de Faltas com caches Multiníveis

- **Adicione um cache de segundo nível:**
  - **freqüentemente o cache primário está no mesmo chip que o processador**
  - **use SRAMs para adicionar outro cache acima da memória primária (DRAM)**
  - **penalidade de falta diminui se o dado estiver no cache de segundo nível**
- **Exemplo:**
  - **CPI de 1.0 em uma máquina de 500Mhz com uma taxa de falta de 5%, DRAM de 200ns de acesso**
  - **Adicionando um cache de segundo nível com tempode acesso de 20ns diminui a taxa de erro para 2%**
- **Usando caches multiníveis:**
  - **tente e otimize o tempo de acerto no cache primário**
  - **tente e otimize a taxa de falta no cache secundário**

Memória Virtual

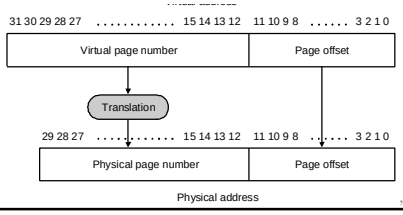
- **Memória principal pode atuar como um cache para o armazenamento secundário (disco)**



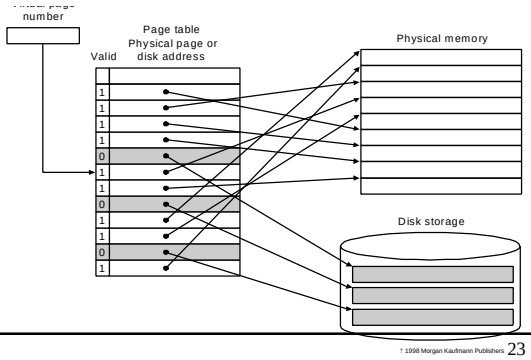
- **Vantagens:**
  - **ilusão de ter mais memória física**
  - **re-alocação de programa**
  - **proteção**

Páginas: blocos de memória virtual

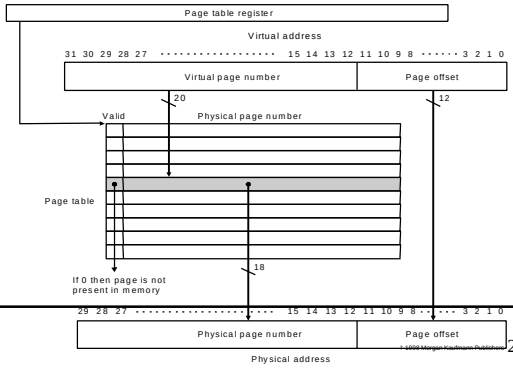
- **Faltas de páginas: o dado não está em memória, retire ele do disco**
  - **penalidade de falta enorme, logo as páginas devem ser razoavelmente grandes (ex., 4KB)**
  - **reduzir as faltas de páginas é importante (LRU vale o preço)**
  - **podemos lidar com as faltas no software ao invés do hardware**
  - **usar write-through é muito caro por isso usamos writeback**



Tabelas de Páginas

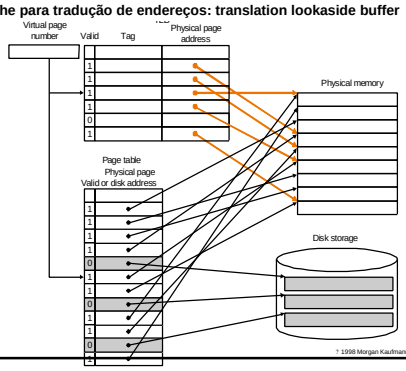


Tabelas de Páginas

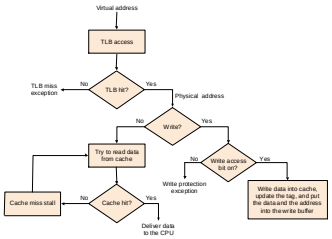


Tornando a Tradução de Endereço Rápida

- **Um cache para tradução de endereços: translation lookaside buffer (TLB)**



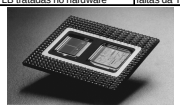
TLBs e caches



Sistemas Modernos

- **Sistemas de memórias muito complicado:**

Característica	Intel Pentium Pro	PowerPC 604
Ender. Virtual	32 bits	52 bits
Ender. Físico	32 bits	32 bits
Tam. Página	4 KB, 4 MB	4 KB, selecionável, e 256 MB
organização TLB	1 TLB p/ instruções e 1 TLB p/ dados	1 TLB p/ instruções e 1 TLB p/ dado
	ambas associativas grau 4	ambas associativas grau 2
	substituição Pseudo-LRU	substituição LRU
	TLB Instruções: 32 entradas	TLB Instruções: 128 entradas
	TLB Dados: 64 entradas	TLB Dados: 128 entradas
	faltas da TLB tratadas no hardware	faltas da TLB tratadas no hardware



Característica	Intel Pentium Pro	PowerPC 604
Organização do Cache	Cache de dados e instruções	Cache de dados e instruções
Tamanho do Cache	8 KB para cada um	16 KB para cada um
Associatividade do Cache	associativo grau 4	associativo grau 4
Substituição	aproximação LRU	LRU
Tamanho do Bloco	32 bytes	32 bytes
Política de Escrita	Write-back	Write-back ou write-through

Alguns pontos

- **Velocidade do processador continua aumentando muito rapidamente — muito mais rápido que a da DRAM ou tempo de acesso de disco**
- **Desafio de projeto: lidar com essa crescente disparidade**
- **Tendências:**
  - **SRAMs síncronas (provem rajadas de dados)**
  - **re-projetar chips DRAM para prover maior largura de banda ou processamento**
  - **re-estruturar o código para aumentar a localidade**
  - **usar pré-busca (fazendo o cache visível pela ISA)**

Capítulos 8 & 9

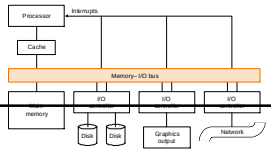
Lecture slides created by Michael Wahl in English

Tradução: Christian Lyra Gomes

Revisão: Wagner M. N. Zola

Interfaceando Processadores e Periféricos

- **Projeto de I/O afetado por muitos fatores (expansibilidade, elasticidade)**
- **Desempenho:**
  - **— latência de acesso**
  - **— vazão**
  - **— conexão entre os dispositivos e o sistema**
  - **— a hierarquia da memória**
  - **— o sistema operacional**
- **Uma variedade de diferentes usuários (ex., bancos, supercomputadores, engenheiros)**



I/O

- **Importante mas negligenciado**

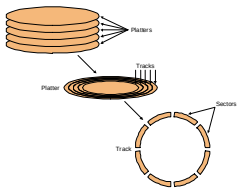
  "As dificuldades em avaliar e projetar sistemas de I/O tem frequentemente relegado o I/O a um status de segunda classe"

Dispositivos de I/O

- **Dispositivos muito diversos**
  - **— comportamento (i.e., entrada vs. saída)**
  - **— parceiro (quem está na outra ponta?)**
  - **— taxa de dados**

Dispositivo	Comportamento	Parceiro	Taxa de Dados (Kb/s)
Teclado	entrada	humano	0.01
Mouse	entrada	humano	0.02
Entrada de Voz	entrada	humano	0.02
Scanner	entrada	humano	400.00
Saída de voz	saída	humano	0.60
Impressora de linha	saída	humano	1.00
Impressora Laser	saída	humano	200.00
Display Gráfico	saída	humano	60,000.00
Modem	Entrada ou Saída	máquina	2.00-8.00
Rede/LAN	Entrada ou Saída	máquina	500.00-6000.00
Disquete	armazenamento	máquina	100.00
Disco óptico	armazenamento	máquina	1000.00
Fita Magnética	armazenamento	máquina	2000.00
Disco Magnético	armazenamento	máquina	2000.00-10,000.00

Exemplo de I/O: Discos Rígidos

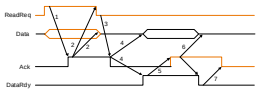


- **Para acessar dados:**
  - **— procura: posicionar a cabeça sobre a trilha (8 a 20 ms. avg.)**
  - **— latência de rotação: esperar pelo setor desejado (.5 / RPM)**
  - **— transferência: pegar os dados (um ou mais setores) de 2 a 15 MB/sec**

Exemplo de I/O: Barramentos

- Link de comunicação compartilhado (um ou mais fios)
- Projeto difícil:
  - — pode ser um gargalo
  - — comprimento do barramento
  - — número de dispositivos
  - — compromissos (buffers para larguras de banda altas aumentam a latência)
  - — suporte para muitos dispositivos diferentes
  - — custo
- Tipos de barramentos:
  - — processador-memória (pequeno e de alta velocidade, projeto personalizado)
  - — backplane (alta velocidade, freqüentemente padrão, ex. PCI)
  - — I/O (comprido, diferentes dispositivos, padronizado, ex. SCSI)
- Síncrono vs. Assíncrono
  - — use um clock e um protocolo síncrono, rápido e pequeno, mas cada dispositivo deve operar na mesma freqüência e o barramento deve ser pequeno por causa do "escorregamento" do clock
  - — não usa um clock, no entanto usa um handshaking

Alguns exemplos de problemas



- Vamos olhar alguns exemplos do texto

  "Análise de Desempenho do Síncrono vs. Assíncrono"
  "Análise de Desempenho de Dois Esquemas de Barramento"

Outros Pontos Importantes

- Arbitragem do Barramento:
  - — arbitragem daisy chain (não muito justo)
  - — arbitragem centralizada (requer um árbitro), ex., PCI
  - — seleção própria, ex., NuBus usado no Macintosh
  - — detecção de colisão, ex., Ethernet
- Sistema Operacional:
  - — polling
  - — interrupções
  - — DMA
- Técnicas de análise de desempenho:
  - — Teoria de enfileiramento
  - — simulação
  - — análise, i.e., achar o elo mais fraco (veja "Projeto de Sistema de Entrada/Saída")
- Muitos desenvolvimentos novos

Multiprocessadores

- **Idéia: criar computadores poderosos através da conexão de muitos computadores menores**

  Notícias boas: funciona para compartilhamento de tempo (melhor que um supercomputador) processamento vetorial pode estar voltando

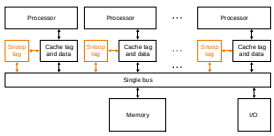
  Notícias ruins: é realmente difícil escrever bons programas concorrentes muitos fracassos comerciais

Questões

- **Como processadores paralelos compartilham dados?**
  - **— espaço de endereçamento único (SMP vs. NUMA)**
  - **— passagem de mensagens**
- **Como os processadores paralelos se coordenam?**
  - **— sincronização (locks, semáforos)**
  - **— construído nas primitivas envia / recebe**
  - **— protocolos de sistemas operacionais**
- **Como eles são implementados?**
  - **— conectados por um barramento único**
  - **— conectados por uma rede**

Alguns problemas interessantes

- **Coerência de Cache**

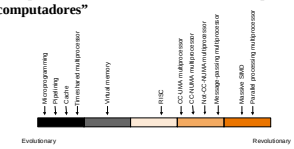


- **Sincronização**
  - **— provê instruções atômicas especiais (test-and-set, swap, etc.)**
- **Topologia da rede**

Observações conclusivas

- **Evolução vs. Revolução**

"Muito freqüentemente o custo da inovação é atrapalhar muito os usuários de computadores"



"Aceitação de ideías no hardware requer aceitação pelo pessoal do software; por isso o pessoal do hardware deve aprender sobre software. E se o pessoal do software quer boas máquinas, eles devem aprender mais sobre hardware para estarem aptos a se comunicar com ele, e portanto influenciar engenheiros de hardware"