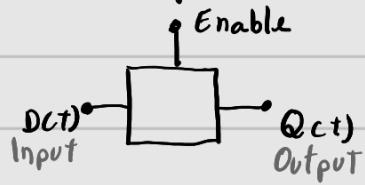


Hierarquia de Memória - Parte 1

Latch

É um elemento lógico que pode acompanhar as variações do dado e transferir estas mudanças para uma linha de Saída.

Círculo binário: O pode valer 0 ou 1.



Hierarquia de Memória



Objetivo: Oferecer ilusão de máximo tamanho de memória, com mínimo custo.

- Máxima velocidade.
- Cada nível contém cópia da parte da informação armazenada ao nível superior segun.

Princípio da localidade

• **Localidade Espacial:** sugere que quando um dado é referenciado, os dados próximos a ele tendem a ser acessados em breve. Por isso, é vantajoso mover blocos de palavras contíguas mais perto do processador na hierarquia de memória.

• **Localidade Temporal:** indica que um dado é frequentemente

~~Cada bloco temporal~~. indica que um bloco referenciado recentemente tem um grande chance de ser referenciado novamente em um futuro próximo. Assim, é útil manter itens de dados que foram acessados recentemente nos níveis mais próximos do processador na hierarquia da memória.

Parte 2

Organizações de Memória Cache

O processador gera um endereço de memória para ler ou escrever dados, ele verifica na memória cache (é uma memória menor e mais rápida), para ver se os dados já estão lá ("hit").

A cache deve:

- Verificar se possui uma cópia da posição de memória correspondente ao endereço gerado pelo processador.
- Se não possuir cópia (miss), a cache deve buscar o conteúdo da memória principal e decidir onde essa informação será armazenada na cache para acessos futuros.
- Se possui a cópia (hit), a cache deve fornecer rapidamente essa informação ao processador.

O mapeamento é o método que define como os dados da memória principal são localizados na cache. Existem três estratégias principais:

- **Mapeamento completamente associativo**: qualquer bloco da memória pode ser carregado em qualquer linha da cache.
- **Mapeamento direto**: cada bloco da memória tem uma linha específica na cache a que é mapeado.
- **Mapeamento set-associativo**: é um meio-termo entre os dois métodos anteriores, onde cada bloco é mapeado para um conjunto de linhas na cache, e qualquer linha dentro desse conjunto pode ser usada.

Características

- Pouco espaço de armazenamento.
- Alto custo financeiro

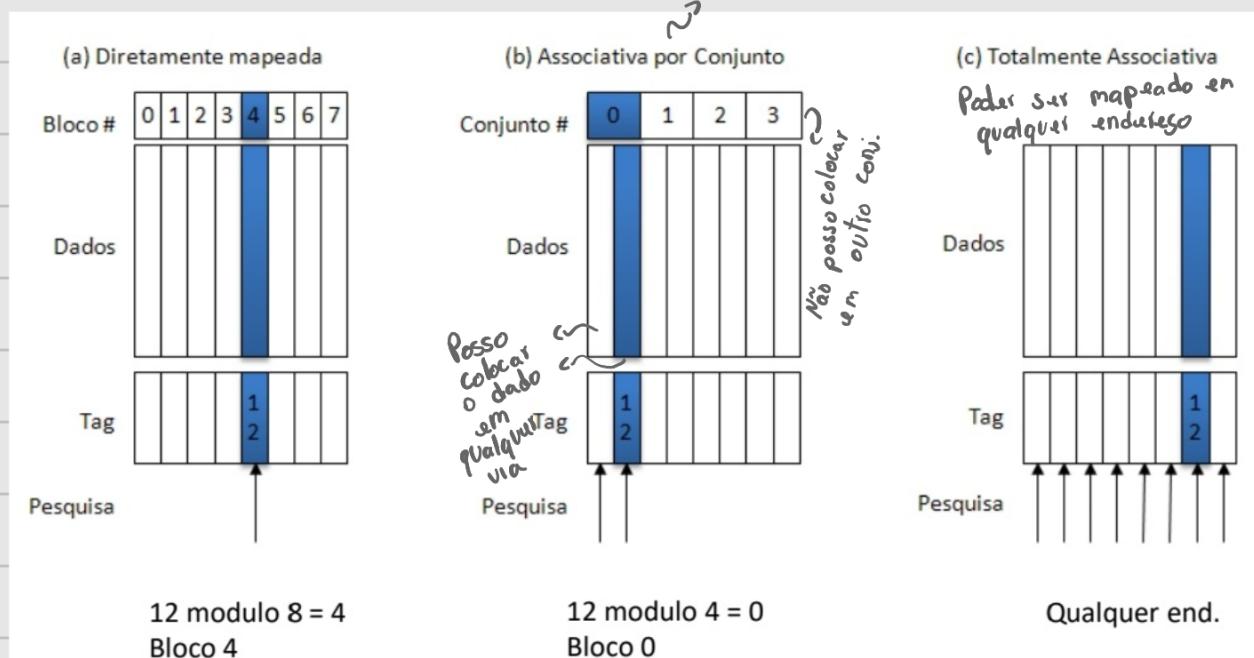
- Baixo tempo de acesso

Conceitos

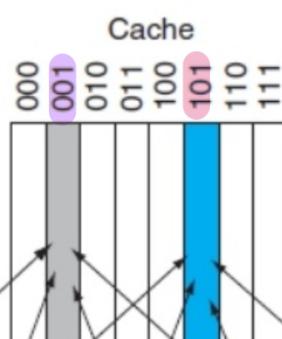
- **Palavra**: conjunto de um ou mais bytes
- **Bloco**: conjunto de uma ou mais palavras (un. da cache)
- **Bit de Válido**: indica se o dado ou bloco está válido.
- **Tag ou rótulo**: parte do endereço de uma palavra na memória principal.
- **Slot**: cada linha de uma cache pode armazenar um ou mais blocos dependendo da organização da cache.
- **Comparador**: compara a tag de um endereço de uma palavra com as tags dos endereços armazenados na cache.

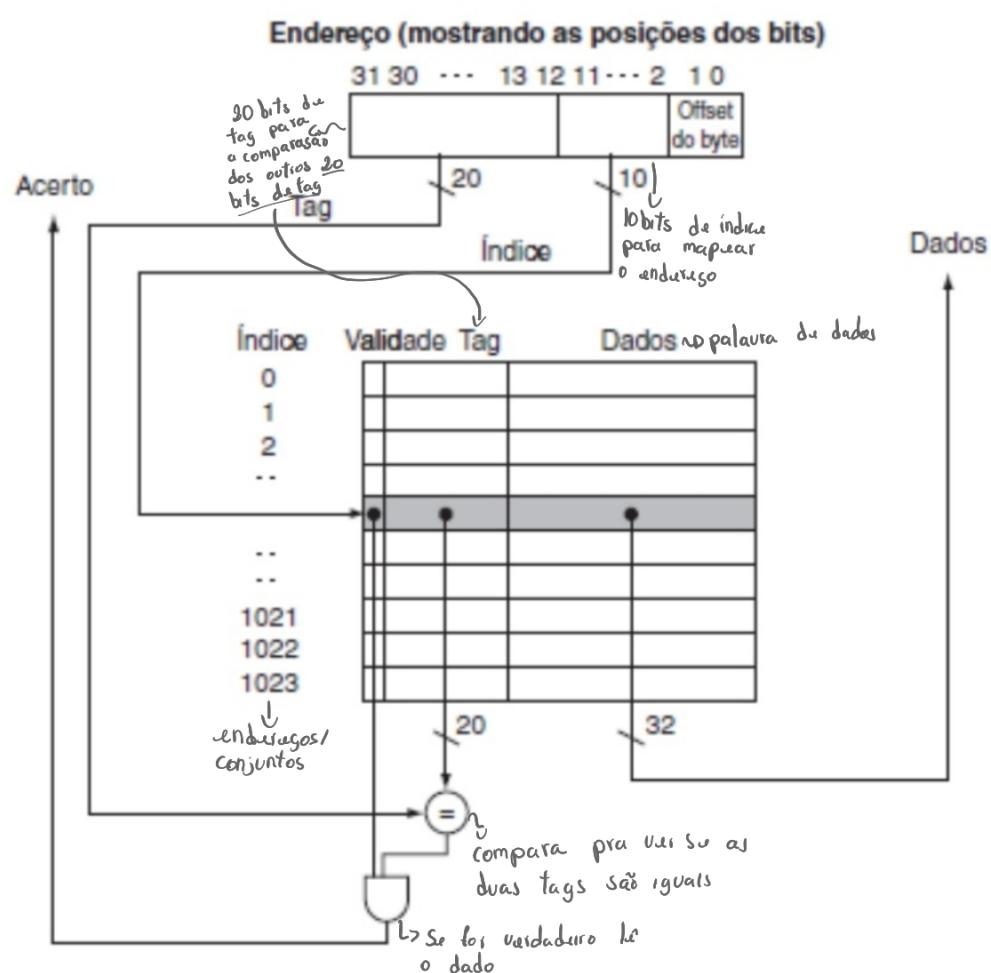
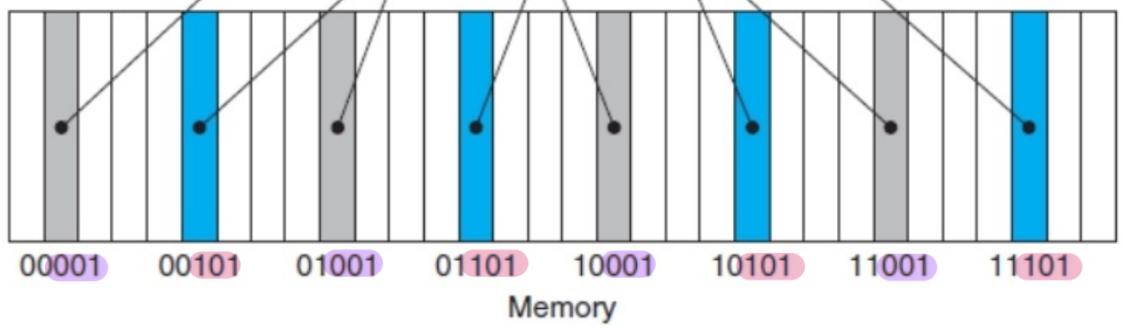
Todos os mapeamentos

Set-associativo

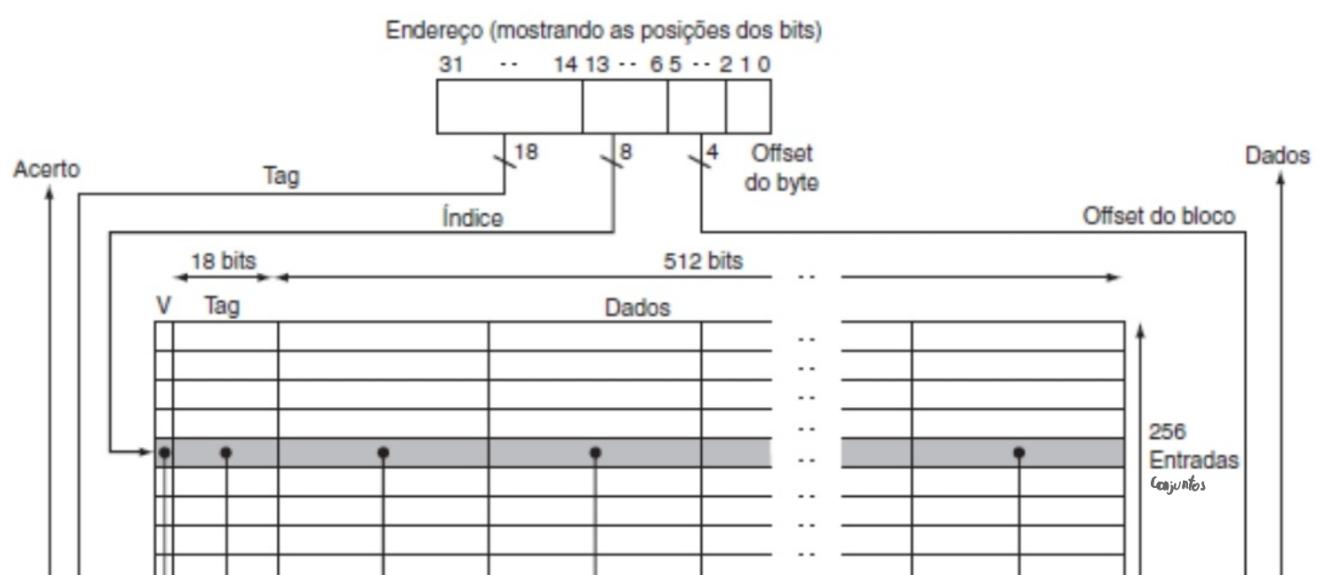


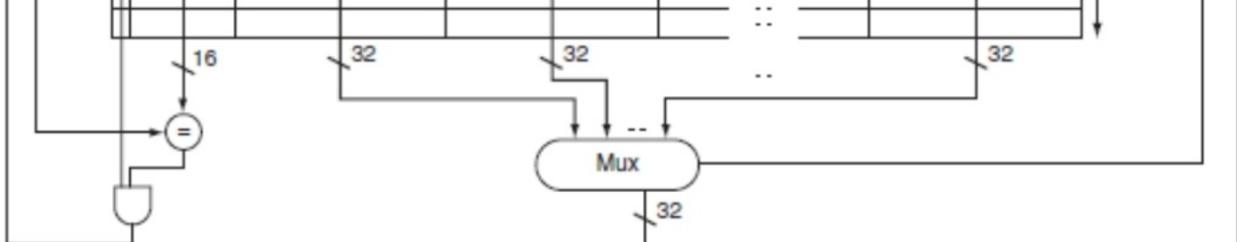
Mapeamento direto





Tamanho da linha tirando vantagem da localidade espacial

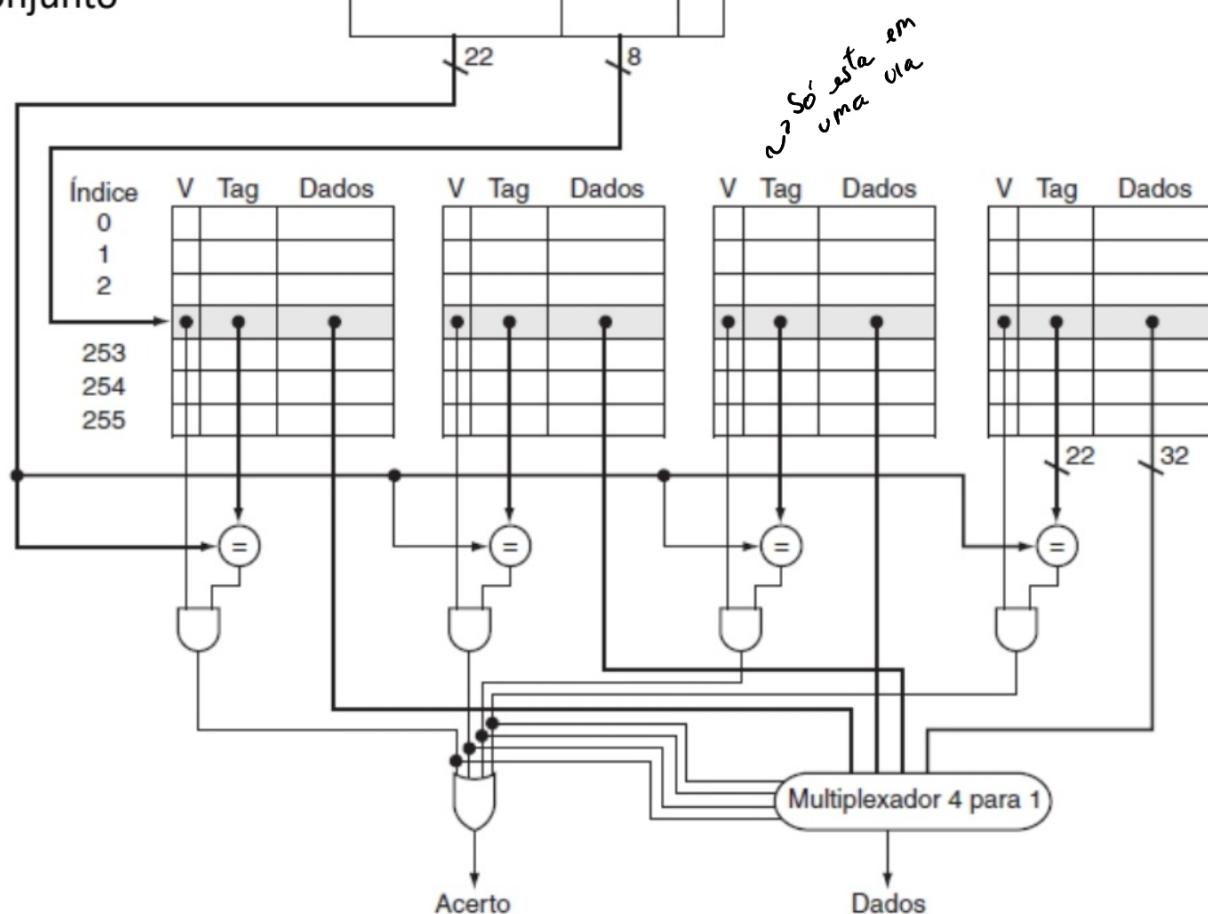
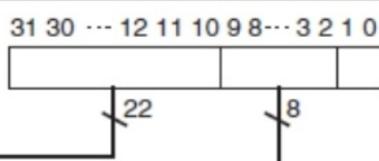




Organização da Cache
Associativo por conjunto
4 vias

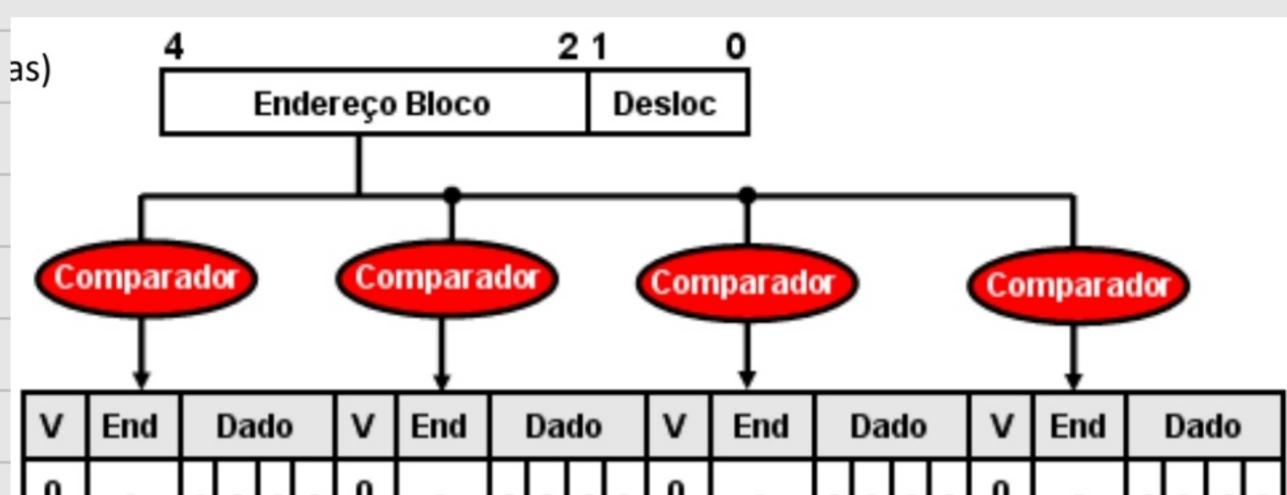
N^o Melhor solução

conjunto



Temos 4 vias porque para cada conjunto eu tenho 4 vias

Completamente Associativo



Um conjunto com a qtd máxima de vias

- Possui um slot com N blocos
- Necessita de N comparadores
- Endereça o bloco diretamente
- Se eu tiver 34 vias eu vou ter 34 comparadores, custo muito alto.

Acesso à Cache

Quando não tiver espaço, qual bloco será substituído?

- **Mapeamento direto:** o bloco que está no slot
- **Associativo por conjunto e Completamente Associativa:** usa uma política de substituição
 - **LRU:** substituir o bloco menos recentemente utilizado.
 - **LFU:** substituir o bloco menos frequentemente utilizado.
 - **FIFO:** substituir o primeiro bloco que entrou na cache.
 - **Aleatório.**

Mapeamento direto

0000	0 miss	0001	1 miss	0010	2 miss	0011	3 miss
00	Mem(0)	00	Mem(0)	00	Mem(0)	00	Mem(0)
		00	Mem(1)	00	Mem(1)	00	Mem(1)
				00	Mem(2)	00	Mem(2)
						00	Mem(3)
0100	4 miss	0011	3 hit	0100	4 hit	1111	15 miss
01	00 Mem(0) 4	01	Mem(4)	01	Mem(4)	01	Mem(4)
	00 Mem(1)	00 Mem(1)	00 Mem(1)	00 Mem(1)	00 Mem(1)	00 Mem(1)	
	00 Mem(2)	00 Mem(2)	00 Mem(2)	00 Mem(2)	00 Mem(2)	00 Mem(2)	
	00 Mem(3)	00 Mem(3)	00 Mem(3)	00 Mem(3)	00 Mem(3)	11	00 Mem(3) 15

8 requests, 2 hits

Quando ocorre uma escrita, como manter a coerência com a memória principal?

- **write-through:** a palavra é escrita tanto no bloco da cache, quanto no bloco da memória principal.
- **write-back:** a palavra é escrita apenas no bloco da cache. Quando o bloco for substituído, então a palavra será escrita na memória principal.

Cache hit, Cache miss

- **Cache hit:** os dados solicitados por um processo estão disponíveis no cache
 - **Hit time:** tempo de acesso ao nível superior, tempo de acesso + tempo para determinar hit/miss.
- **Cache Miss:** os dados que o processo precisa não são encontrados no cache e deve ser procurado na memória principal.
 - **Miss Penalty:** tempo necessário para buscar os dados da memória principal (ou de outro nível inferior da memória) e carregá-los no cache.
- **Métrica** é a taxa de acerto dado um número de acessos a cache, que a porcentagem de cache hit.

$$\text{Taxa de acerto} = \frac{\text{Nº Cache hit}}{\text{Nº de Acessos}}$$



Tipos de cache miss

Compulsórios são os misses que ocorrem no primeiro acesso a uma linha da memória que nunca foi carregada na cache. São inevitáveis, pois quando um programa começa a rodar, o cache está vazio ou contém dados irrelevantes para o programa atual.

De conflito ocorre quando múltiplas linhas de memória tentam ocupar o mesmo espaço na cache. Isso pode acontecer em cache com mapeamento direto ou quando não há associativo suficiente (quando o cache pode armazenar mais de um bloco de memória em um único lugar).

- **Solução 1:** aumentar o tamanho da cache, ela fica mais lenta porque aumentamos o tempo de acesso.

- **Solução 2:** aumentar a associatividade

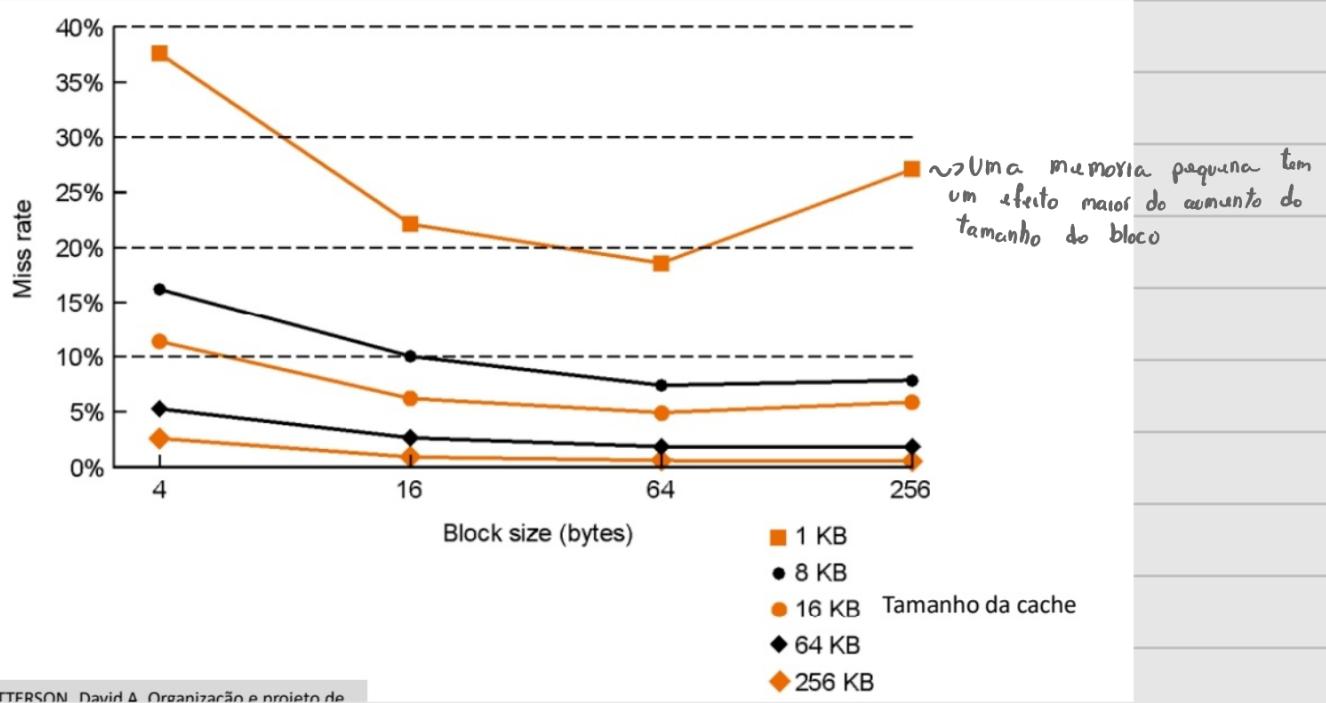
De capacidade acontece quando o cache não tem espaço suficiente para conter todas as linhas de memória que um programa precisa acessar.

- **Solução** é aumentar o tamanho da cache.

Invalidez, outro processo atualiza a memória, o que torna os dados no cache antigo ou incorretos.

Tamanho da linha vs. miss ratio

!!!!!!



Tamanho da linha

Uma linha maior aproveita melhor a localidade espacial mas

- linhas de cache maiores significam que quando ocorre um cache miss, demora mais tempo para preencher essa linha com dados da memória principal, aumentando assim o miss penalty.
- se o tamanho da linha é muito grande em relação ao tamanho total da cache, isso pode levar a um aumento no miss ratio, que é a porcentagem de acessos que resultam em misses, porque há menos linhas de cache disponíveis para armazenar todos os dados.



Tenho mais bytes para trazer da memória principal para memória cache.

$$\text{tempo médio de acesso} = \text{hit time} (1 - \text{miss ratio}) + \text{miss penalty} \times \text{miss ratio}$$

$$N = 2^{10} = 1024$$

Exercício

Quantos bits tem a cache no total?

$$32b = 4B = 2^2$$

$$\text{Bloco} = 4 \text{ palavras} = 2^2$$

• supondo cache com mapeamento direto, com 64 kB de

Supondo cache com mapeamento direto, com 64 kB de dados, linha com uma palavra de 32 bits (4 bytes), e endereços de 32 bits

- 64 kB \rightarrow 16 kpalavras, 2^{14} palavras, neste caso 2^{14} linhas
Cada palavra está em um linha
- cada linha tem 32 bits de dados mais um tag (32-14-2 bits) mais um bit de validade:
 $2^{14} \times (32 + 32 - 14 - 2 + 1) = 2^{14} \times 49 = 784 \times 2^{10} = 784$ kbytes
- 98 kB para 64 kB de dados, ou 50% a mais
 \downarrow tamanho do tag
 \downarrow tamanho real

Índice e offset
Cache tag

- supondo cache com mapeamento direto, com 16 kB de dados, blocos de 4 palavras, sendo cada palavra de 32 bits e endereços de 32 bits

- 16 kB \rightarrow 4 kpalavras, 2^{12} palavras

- Bloco de 4 palavras (2^2), 2^{10} blocos (linhas)

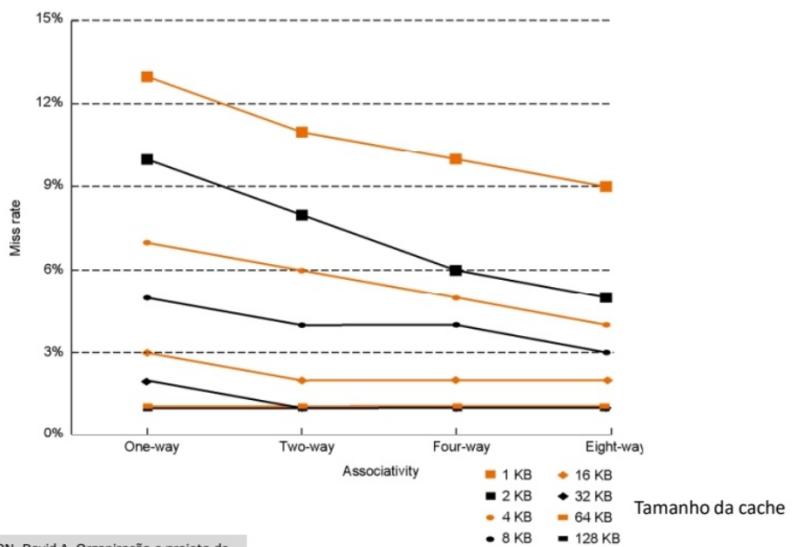
- cada bloco tem 32 bits \times 4 = 128 bits de dados mais um tag (32-10-2-2 bits) mais um bit de validade:

$$2^{10} \times (128 + 32 - 10 - 2 - 2 + 1) = 2^{10} \times 147 = 147 \text{ kbytes}$$

\downarrow blocos
 \uparrow bit válido

- 18.4 kB para 16 kB de dados, ou 15% a mais

Impacto da associatividade



DATTERMSON, David A. Organizações e projeto de sistemas de computadores.

Impacto no desempenho

Medindo o impacto do hit ratio no tempo efetivo de acesso

T_{ce} = tempo de acesso à memória cache

T_m = tempo de acesso à memória principal

T_{ce} = tempo efetivo de acesso à memória cache, considerando efeito dos misses.

O ideal seria prox. ou igual ao T_c $\sim T_{ce} = h T_c + (1-h) T_m$

Ex.

$$T_c = 1 \text{ ns}, T_m = 20 \text{ ns}$$

$$\begin{array}{cccc} b & = & 0,85 & 0,95 & 0,99 & 1,0 \\ & & \downarrow & \downarrow & \downarrow & \downarrow \end{array}$$

$$T_{ac} = 3,85 \text{ ns} \quad 1,95 \text{ ns} \quad 1,19 \text{ ns} \quad 1,0 \text{ ns}$$

↓
Só aumentar muitas
vezes o desempenho
cai.

Impacto no desempenho

Supondo um processador que executa um programa com:

- CPI = 1,1 ~ 1,1 ciclos
- 50% aritm/lógica, 30% load/store, 20% desvio

10% das operações de acesso à memória sejam misses. Cada miss resulta numa penalidade de 50 ciclos.

$$CPI = CPI \text{ ideal} + \text{nº médio de stalls por instrução}$$

- = 1,1 ciclos + 0,3 × 0,1 × 50
- = 1,1 ciclos + 1,5 ciclos ~ em média os dados que não são encontrados na cache.
~ 0,5 ciclos a mais para levar para a memória principal.
- = 2,6 ~ tempo que o processador fica parado

58,1% do tempo o processador está parado esperando pela memória!

Um miss ratio de 1,1 no fetch da instrução resultaria na adição de 0,5 ciclos ao CPI médio.

Hierarquia de caches

Caches integradas dentro de um processador têm limitação de tamanho.

- **Penalidade de Miss:** quando os dados não são encontrados na cache (um miss), dá uma penalidade de tempo significativa porque o acesso à memória principal é muito mais lento.
- **Solução de Múltiplas Níveis:** a solução para o problema da penalidade de miss é ter várias camadas de cache, geralmente 2 ou 3:
 - Cache integrada → cache integrada

- Cache L1: é o cache de primeiro nível, integrado e de tamanho pequeno (por exemplo, 8 Kilobytes), com tempo de acesso muito rápido.
- Cache L2: é o cache de segundo nível, com tamanho maior (por exemplo, 256 Kilobytes) e tempo de acesso mais lento que o L1, mas ainda muito rápido que a memória principal.
- Cache L3: geralmente é um cache de terceiro nível e pode ser localizado fora do chip do processador. É maior que L1 e L2 e tem um tempo de acesso mais lento.
- Misses e Transferência: as falhas podem ocorrer em qualquer nível de cache, e as transferências entre os níveis de cache podem apresentar problemas similares aos já discutidos, como latência e largura de banda.

Caches de dados e Instruções

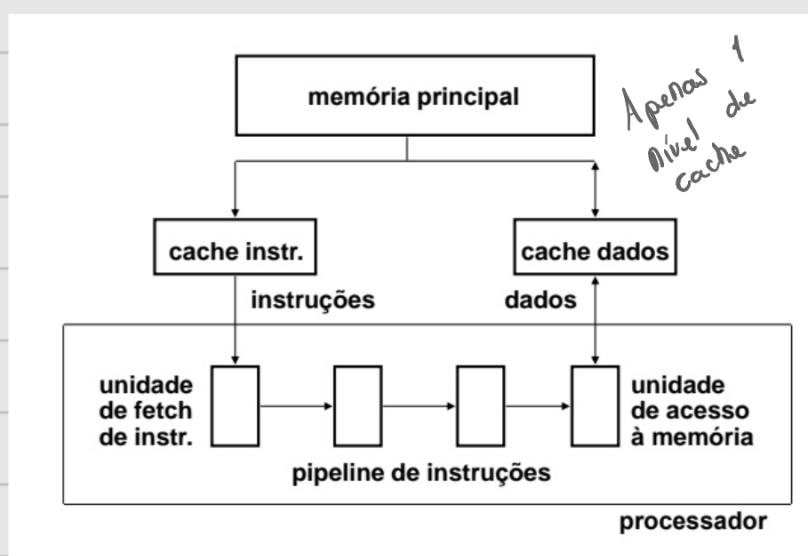
Dados e instruções: cache unificada x cache separada

Vantagem da cache separada

- política de escrita só precisa ser aplicada à cache de dados.
- caminhos separados entre memória principal e cada cache, permitindo transferências simultâneas, ex. pipeline
- estratégias diferentes para cada cache: tamanho total, tamanho de linha, organização.

Caches separadas são usadas na maioria dos processadores, no nível L1.

Caches unificadas nos níveis L2 e L3.



Desempenho em caches multinível

- Suponha que o processador tenha um CPI de 1,0 e que todas as referências acertem na cache primária a uma velocidade de clock de 5GHz (0,2ns). O tempo de acesso à memória principal é de 100ns com todos os tratamentos de faltas. Taxa de falhas por instrução na cache primária é de 2%.

- O quanto mais rápido será o processador se acrescentarmos uma cache secundária com tempo de acesso de 5ns para um acerto ou uma falha e que seja grande o suficiente para que a taxa de falhas na L2 seja de 0,5%?

• Penalidade de falhas para a memória principal:

$$\frac{100\text{ns}}{0,2\text{ ns/ciclos de clock}} = 500 \text{ ciclos de clock}$$

0,2 ns/ciclos de clock

- Para processadores com apenas L1: $\text{CPI} = \text{CPI ideal} + n^{\circ} \text{máximo de stalls por instrução}$

$$\text{CPI total} = 1,0 + 2,5 \times 500 = 11,00$$

- Em relação a L1, penalidade de falha para L2:

$$5\text{ns} / 0,2\text{ns} = 25 \text{ ciclos de clock}$$

- Para cache de dois níveis: $\text{CPI} = \text{CPI ideal} + \text{stall L1} + \text{stall L2}$

$$\text{CPI total} = 1 + 2,5 \times 25 + 0,5 \times 500 = 9,00$$

- Com cache L2:

$$11/9 = 1,22 \text{ vezes mais rápido.}$$

Parte 4

Memória Virtual

Introdução

- memória principal semicondutora
 - capacidade limitada.
 - tempo de acesso entre 10 e 20 ns.
- memória secundária em disco
 - capacidade maior
 - tempo de latência entre 10 e 30 ns.

O problema

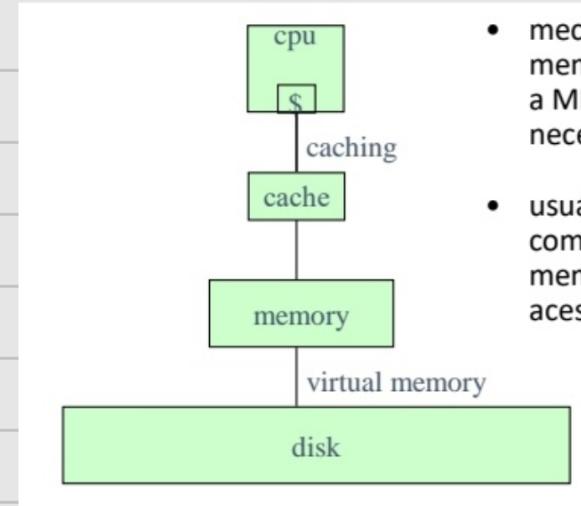
- Nossos computadores têm 32Kbytes de memória principal.

• Como podemos:

- rodar programas que usam mais do que 32 Kbytes?
- permitir que vários usuários usem o computador?
- executar vários programas ao mesmo tempo?

Memória Virtual: a solução!

- **Memória Virtual**: uma técnica que permite que a memória principal seja vista como se tivesse uma capacidade de armazenamento maior do que realmente tem, funcionando como uma cache de grande capacidade.
- É apenas mais um nível na hierarquia de memórias.
- mecanismo automático de gerência de memória, que traz automaticamente para a MP os blocos de informações (do disco) necessárias.
- Usuários têm a impressão de trabalhar com memória única, de tamanho da memória única, do tamanho da memória secundária, mas com tempo de acesso próximo do tempo da MP.



Tempo de acesso

Tempo médio de acesso T_{ma} é dado por $T_{ma} = T_m + (1-h)T_s$

Ex. $T_m = 20\text{ns}$, $T_s = 20\text{ms}$, $h = 0,9999$, $T_{ma} = 2,02\text{us}$ (100x maior do que T_m)

$$T_{ma} = T_m + (1-h)T_s$$

(hit ratio)

$\sim \text{tempo de acesso à MP}$
 $\sim \text{tempo de acesso ao disco}$

Por que MV é diferente das caches?

- Miss Penalty é muito maior do que nas caches, se a informação não está na memória, está no disco!

Observação: miss ratio é sobre a frequência com que os misses ocorrem, o miss penalty é sobre o custo de desempenho de cada miss individual. Elas estão interligadas na medida em que muitos misses com alto

miss ratios combinados com um alto custo por miss (com alto miss penalty) terão um efeito negativo substancial no desempenho do sistema.

- Logo:

- miss ratio precisa ser bem menor do que em cache.
- alta penalidade do miss \rightarrow necessário buscar blocos maiores em disco.
- princípio de localidade opera sobre blocos maiores de dados ou instruções e leva hit ratios bem mais elevadas
- mapeamento totalmente associativo das páginas, usamos mapeamento para reduzir ao máximo o miss
- misses são tratados por software (há tempo disponível) enquanto pegamos as coisas do disco
- técnica de escrita write-through não é uma opção. Usa-se write-back.

write-through significa que todas as escritas de dados são feitas simultaneamente na cache e no armazenamento do nível inferior, write-back armazena as alterações apenas na cache e escreve de volta para o armazenamento inferior apenas quando a página da memória é removida da cache.

Terminologia

Caché

MMU

bloco

página

caché miss

page fault

endereço

endereço virtual (ou lógico)

índice

endereço real (ou físico)

- Endereço virtual (lógico): gerado pelo programa

- deve endereçar todo espaço em disco
- maior número de bits

- Endereço real (físico): endereço na memória principal

- menor número de bits.

Unidade de Gerenciamento de Memória

- MMU (Memory Management Unit)

- gerencia da hierarquia de memória
- proteção de memória
- usualmente integrada dentro do microprocessador

- MMU devo fazer mapeamento do endereço virtual para endereço real.
- SO usa MMU

Endereço virtual é de uma memória que não existe fisicamente, memória de trabalho, um programa trabalha em memória principal só que ele gera um endereço de memória grande que não existe, ele não vai trabalhar nessa memória virtual. MMU vai ser responsável pela gerência do mapeamento desse endereço virtual para endereço real.

Se a memória virtual não existe fisicamente eu preciso de um lugar para armazenar os dados e colocado na memória secundária.

Paginação

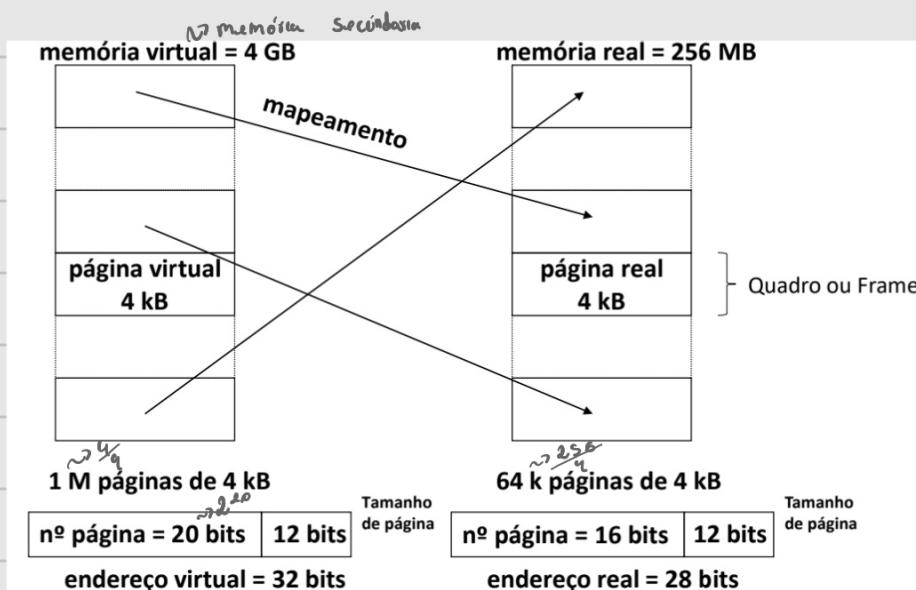
Mecanismo simples para tradução de endereços virtuais em reais e para gerenciamento do espaço de memória.

Espaços de memória real e virtual divididos em blocos chamados de páginas.

- páginas tem tipicamente de 4Kbytes e 16 Kbytes
- páginas para sistemas embarcados são de 1Kbytes

Endereços virtuais e reais divididos em 3 campos

- endereço da página
- endereço da linha (ou palavra), dentro da página



Parte 5

Paginação

- Page fault ocorre quando a página não está na memória principal.
- Mapeamento completamente associativo, mais eficiente, ajuda a diminuir alta penalidade.

de page faults

- **Page tables:** são estruturas de dados usadas pelo sistema operacional para manter o mapeamento de correspondência entre páginas virtuais e as reais. Cada entrada da tabela de páginas contém informações sobre onde uma página virtual específica está localizada na memória física. Elas permitem a tradução do endereço do espaço de endereço virtual para o espaço de endereço físico.

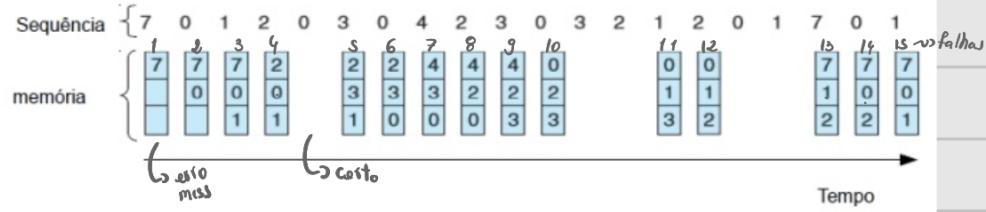
Gerência de processos

- Todo processo tem sua própria tabela de páginas
 - processos são compilados para espaços de endereçamentos virtuais
 - tabela de páginas define toda a utilização do espaço de endereçamento pelo processo
- Sistema operacional é responsável pela alocação de espaço físico para o espaço virtual de cada processo.
 - Só carrega tabela de páginas de cada processo
- Hardware possui registrador que aponta para inicio da tabela de páginas do processo atual
 - { Se a tabela da página estiver em memória e é necessário fazer uso da tabela para descobrir qual é o endereço de página real que o processo tem que acessar, temos um deadlock, existe um registrador que aponta para o inicio da tabela de páginas, fazendo a conversão e gerar endereço real e gerar o processo faz o "acesso à memória principal". }
- Quando novo processo passa a ser ativo, sistema operacional só precisa atualizar valor desse registrador.

Paginação

- Main Memory Translation Table (MMTT)
 - implementada em hardware no inicialmente futuramente pode ir pra software
 - tamanho = nº de páginas na memória principal
- disk Memory translation table (DMTT) tabela maior cabe mais página
 - implementada em software, armazena na memória principal
 - tamanho = nº de páginas em disco.
- Algoritmo de substituição, em software, para selecionar página da memória principal a ser substituída em caso de page fault.
- bom desempenho é garantido pelo princípio da localidade.

- FIFO
- Memória com 3 frames (quadros)
 - Frame tem tamanho igual a uma página virtual



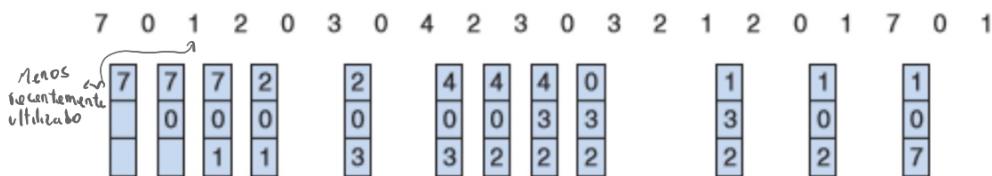
- Total de requisições = 20
- Total de falhas = 15
- Taxa de falha = $15/20 = 0,75$

https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/9_VirtualMemory.html

2022

Arquitetura de Computadores III - Engenharia e Ciéncia da Computação - PUC Minas

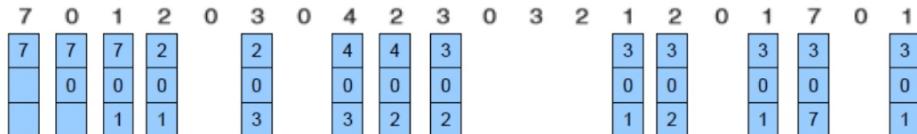
- LRU
- Memória com 3 frames (quadros)
 - Frame tem tamanho igual a uma página virtual



- Total de requisições = 20
- Total de falhas = 12
- Taxa de falha = $12/20 = 0,6$

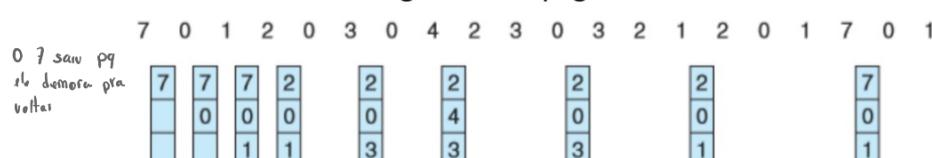
- LFU → Oq tivs mais contados saí, podendo tss contados igual
- FIFO como critério de desempate

- Memória com 3 frames (quadros)
 - Frame tem tamanho igual a uma página virtual



- Total de requisições = 20
- Total de falhas = 13
- Taxa de falha = $13/20 = 0,65$

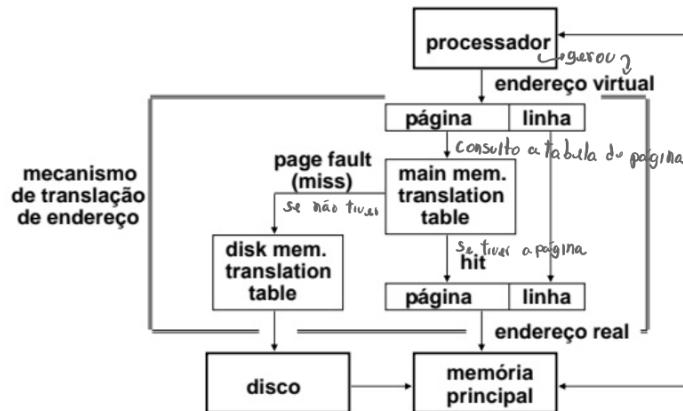
- OPT (solução ótima)
- Substitui a página que não será usada por um tempo mais longo (prevê futuro)
- Memória com 3 frames (quadros)
 - Frame tem tamanho igual a uma página virtual



- Total de requisições = 20
- Total de falhas = 9
- Taxa de falha = $9/20 = 0,45$

https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/9_VirtualMemory.html

Paginação



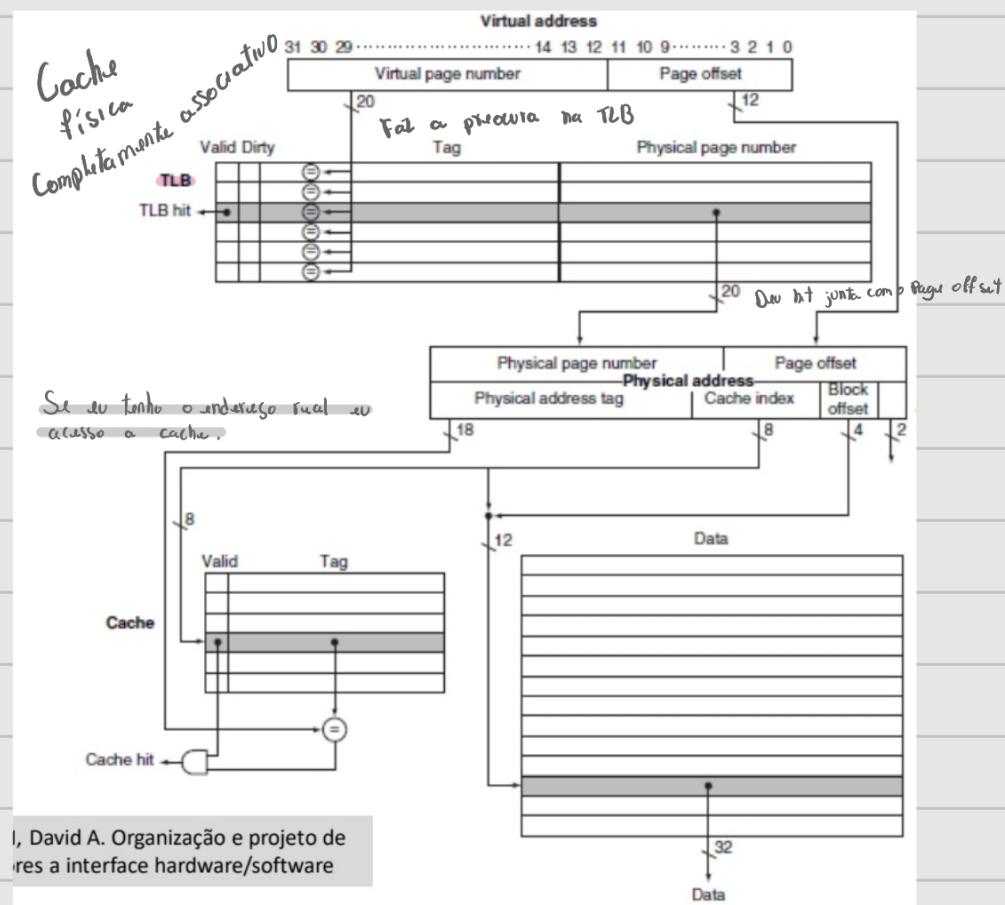
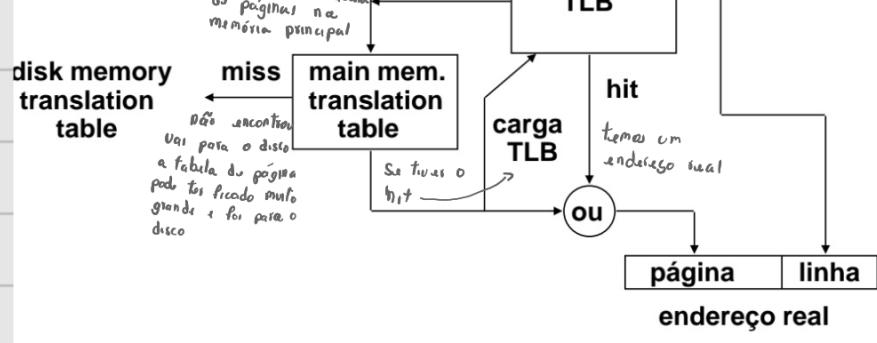
2022

Arquitetura de Computadores III - Engenharia e Ciência da Computação - PUC Minas

Translation Look-Aside Buffer

- Número de páginas na memória secundária é muito grande
 - espaço virtual de 2^{32} bytes, páginas de 4K bytes, 4 bytes por entrada na tabela
 - 4 MBytes apenas para a Tabela de páginas !!!
 - Tamanho excessivo da main memory translation table → mapar endereços virtuais
- Se a tabela ficar na memória principal ⇒ dois acessos à memória a cada cache miss. Porque precisamos fazer a consulta a Tabela de Páginas a conversão, e também quando eu tento pegar o endereço na cache e eu tento miss esses dois processos fazem eu voltar a memória.
- Working set = conjunto de páginas mais prováveis de serem acessadas num dado momento, devido ao princípio da localidade.
- Translation Look-Aside Buffer (TLB) se baseia no working set
 - implementado em hardware
 - Traduz endereços virtuais para endereços reais
 - Só inclui páginas do working set
 - pode ser considerado como uma "cache" da MMU
- MMU
 - implementada em software





Diferença entre cache físico e virtual, a cache física é endereçada por endereços físicos e a virtual por endereços virtuais.

Se eu tiver um projeto com cache virtual o endereço virtual é gerado não passa pela TLB para acessar a cache ele vai direto para a cache.

miss ↗
hit ✕

TLB	Page table	Cache	Possible? If so, under what circumstance?
Hit	Hit	Miss	Possible, although the page table is never really checked if TLB hits.
Miss	Hit	Hit	TLB misses, but entry found in page table; after retry, data is found in cache.
Miss	Hit	Miss	TLB misses, but entry found in page table; after retry, data misses in cache.
Miss	Miss	Miss	TLB misses and is followed by a page fault; after retry, data must miss in cache.
Hit	Miss	Miss	Impossible: cannot have a translation in TLB if page is not present in memory.
Hit	Miss	Hit	Impossible: cannot have a translation in TLB if page is not present in memory.
Miss	Miss	Hit	Impossible: data cannot be allowed in cache if the page is not in memory.

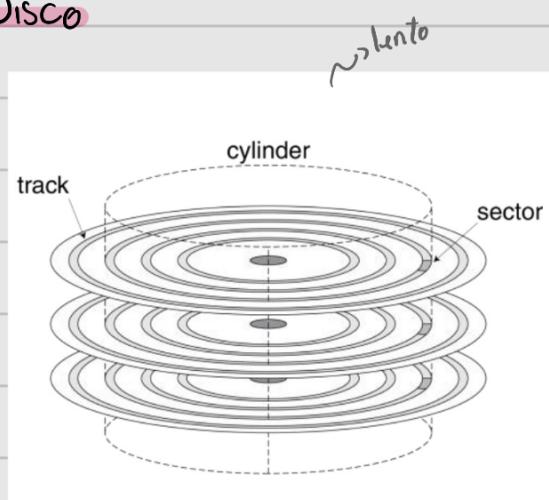
→ não pode porque se eu não tiver o endereço real eu não consigo consultar a cache.

Se eu tiver um 'hit' na TLB eu tiver um 'hit' na tabela de páginas porque a TLB é uma tabela de páginas.

Mecanismos de translagão de endereços mapeamento direto

- endereço da página virtual é utilizado como endereço de uma memória cujo conteúdo é o endereço de página real procurado.
- Tamanho = nº de páginas na memória virtual
- utilizado na MMU (em Software) mas não na TLB

Disco



Para acessar dados:

- **Busca:** posiciona a cabeça sobre a trilha correta (3 a 14 ms em média)
- **Latência rotacional:** espera pelo setor desejado (0,5 pm)
- **Transferência:** recuperar os dados (um ou mais setores; 30 a 80 kB/seg)

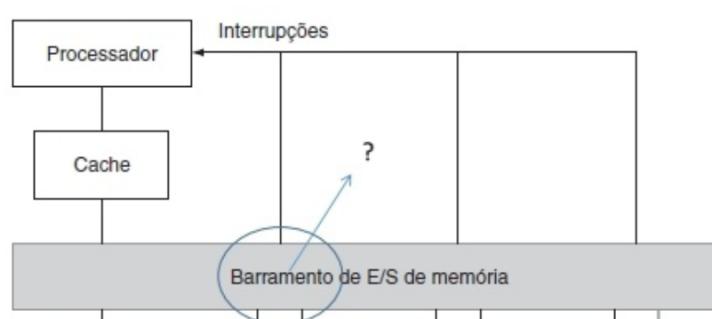
Desempenho do Disco

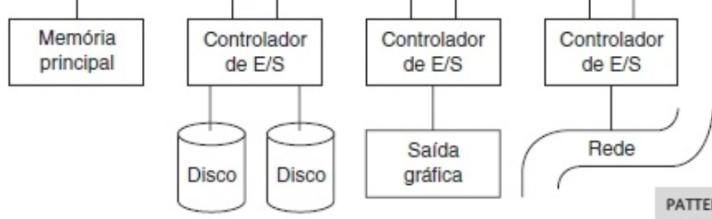
- **Tempo de Disco** = Tempo de busca + Latência rotacional + Tempo de transferência
- A latência rotacional média para a informação desejada está a meio caminho ao redor do disco.
- TB = 1ms
- $$LRM = \frac{0,5 \text{ rotação}}{5400 \text{ RPM}} = \frac{0,5 \text{ rotação}}{5400 \text{ RPM} / 60(\text{seg})} = 0,0056\text{s} = 5,6\text{ms}$$
- $$TT = \frac{1\text{KB}}{50\text{MB/s}} = \frac{1 * 2^{10}}{50 * 10^6} = 20,48\mu\text{s}$$
- TD = 1ms + 5,6ms + 0,02048ms = 6,62048ms

2022

Arquitetura de Computadores III - Engenharia e Ciência da Computação - PUC Minas

Visão geral





PATTERSON, David A. Organização dos computadores a interface hardware.