

"Estou fazendo um sistema operacional gratuito (apenas um hobby, não será grande e profissional como GNU) para 386/486 AT" (Linus Torvalds referindo-se ao Linux; 1991).

Memória Virtual

Paulo Lisboa de Almeida



Memória virtual

Considere que escrevemos dois programas.

Os dois foram escritos seguindo as convenções da memória do RISC-V.

Problema:

Desejamos executar ambos programas em uma máquina com um sistema operacional que permite que os programas sejam executados “ao mesmo tempo” – sistemas multitarefa.

SP = 0x0000 003F FFFF FFF0

0x0000 0000 1000 0000

PC = 0x0000 0000 0040 0000



Memória virtual

Pior ainda.

Considere que temos, por exemplo, 1 GiB de memória principal, mas cada programa precisa de 2 GiB.

SP = 0x0000 003F FFFF FFF0

Pilha

Dados Dinâmicos

Dados Estáticos

0x0000 0000 1000 0000

Texto

PC = 0x0000 0000 0040 0000

Reservado

0

Memória virtual

Precisamos de um sistema que “engane os programas”.

O programa pensa que está sendo executado em uma região da memória, mas está em outra.

Precisamos ainda usar a memória principal como uma cache.

O que não couber na memória principal, pode ir para o HD ou SSD.

Assim como na cache que colocamos próxima da CPU, tudo isso precisa ser transparente para o programador.

Dessa vez vamos precisar de uma combinação de hardware + software (Sistema Operacional).

Páginas

Vamos dividir a memória principal em blocos de tamanho fixo, chamados de **páginas**.

Se a memória principal tem 1 GB, e cada página tem 64 KB, a memória principal comporta 16.384 páginas.

Podemos ter um endereçamento de memória virtual com, por exemplo, 4 GB.

Suporta 65.536 páginas.

Páginas

Vamos dividir a memória principal em blocos de tamanho fixo, chamados de **páginas**.

Se a memória principal tem 1 GB, e cada página tem 64 KB, a memória principal comporta 16.384 páginas.

Podemos ter um endereçamento de memória virtual com, por exemplo, 4 GB.

Suporta 65.536 páginas.

temos mais memória virtual do que física.

As páginas da memória virtual podem ficar, por exemplo, no SSD.

As páginas são carregadas sob demanda para a memória principal, em um esquema parecido com a cache.

Criamos a ilusão de que temos mais memória do que realmente temos.

Falta de Página

Quando um acesso a uma página que não está na memória principal é feito, temos uma **falta de página** – *page fault*.

Necessário carregar a página para a memória principal.

Páginas

Problema: o disco rígido ou SSD é muito mais lento que a memória principal.

A memória principal chega a ter uma latência 100.000 menor que um disco.

Vai demorar milhões de ciclos de clock para carregar uma página faltante para a memória principal.

Mitigando os atrasos do disco

Usar páginas grandes.

Localidade espacial → aproveitar para carregar a maior vizinhança o possível quando uma falta de página acontece.

Discos e SSDs demoram muito para servir o dado, mas podem servir uma grande quantidade de dados de uma vez.

Tamanhos comuns de páginas em sistemas atuais variam de 1 a 64 KiB.

Faça

Use o comando `getconf PAGESIZE` para exibir o tamanho das páginas no seu sistema.

O tamanho é exibido em Bytes.

Mitigando os atrasos do disco

Considerar que a memória principal é totalmente associativa.

Podemos encaixar uma página virtual em qualquer página física da memória.

Mitigando os atrasos do disco

Utilizar algoritmos (via software) sofisticados para resolver **page faults**.

Vai custar tempo (overhead), mas o Disco/SSD são tão lentos, que esse preço é pequeno perto do preço da carga.

Mitigando os atrasos do disco

Esquemas write-back são mandatórios.

Mandar os dados da memória principal para o disco a cada transação (em um esquema write-through) é inviável.

Tabela de Páginas

Uma página virtual pode ser mapeada para qualquer página física.

Uma página virtual é chamada de **página**.

Uma página física é chamada de **quadro**.

O sistema operacional escolhe qualquer quadro na memória para substituir.

Na esperança de remover uma página que não será usada num futuro próximo.

Tabela de Páginas

O sistema operacional mantém uma estrutura chamada de **Tabela de Páginas - Page Table**.

Uma **page table para cada programa**.

Contém o mapeamento da página virtual do programa para quadros da memória.

Podemos ter um registrador no hardware que aponta para o início da page table do programa sendo executado atualmente.

Campo de validade indica se a página virtual está ou não mapeada para uma página física.

Val?	Página	Quadro
1	0	27
1	1	15
0	2	
0	3	
...

Como a Tradução Funciona

Considere uma memória física de 256 bytes.

8 bits para endereçar.

Memória virtual de 4096 bytes.

12 bits para endereçar.

Páginas de 4 bytes.



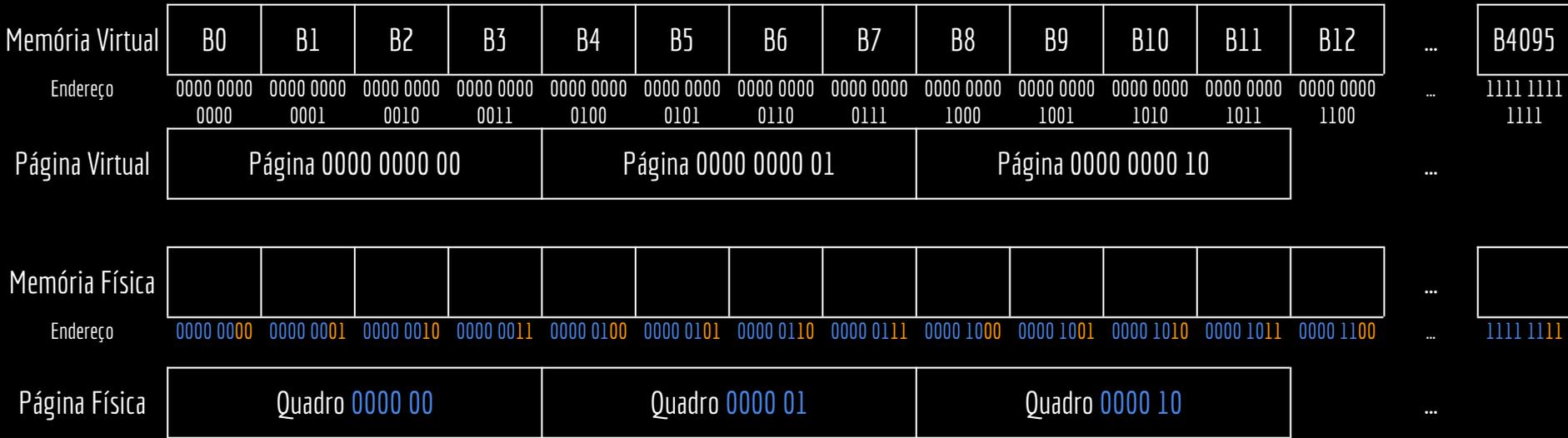
Como a Tradução Funciona

Número (endereço) da página.

Offset (deslocamento) dentro da página.



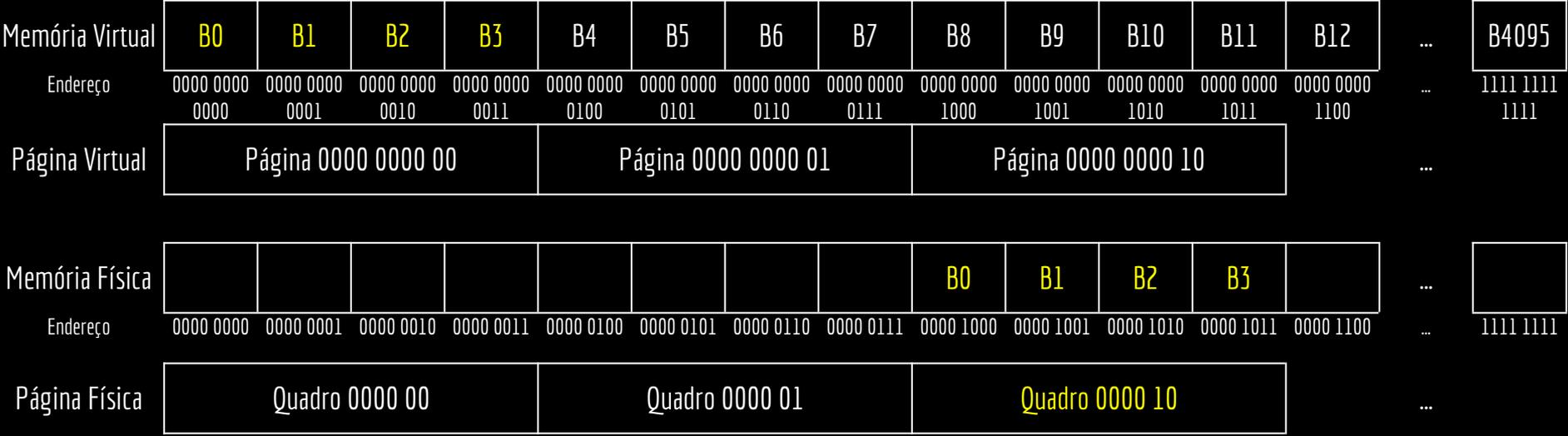
Como a Tradução Funciona



Como a Tradução Funciona

Tabela de Páginas

	Vál?	Quadro
0000 0000 00	1	0000 10
0000 0000 01	0	
0000 0000 10	0	
...		



Para facilitar, vamos colocar o endereço a ser carregado na instrução como imediato (o raciocínio é o mesmo usando registrador base + deslocamento).

A instrução lb carrega um byte - Load Byte.

lb s0, 0000 0000 0010

Tabela de Páginas

	Vál?	Quadro
0000 0000 00	1	0000 10
0000 0000 01	0	
0000 0000 10	0	
...		

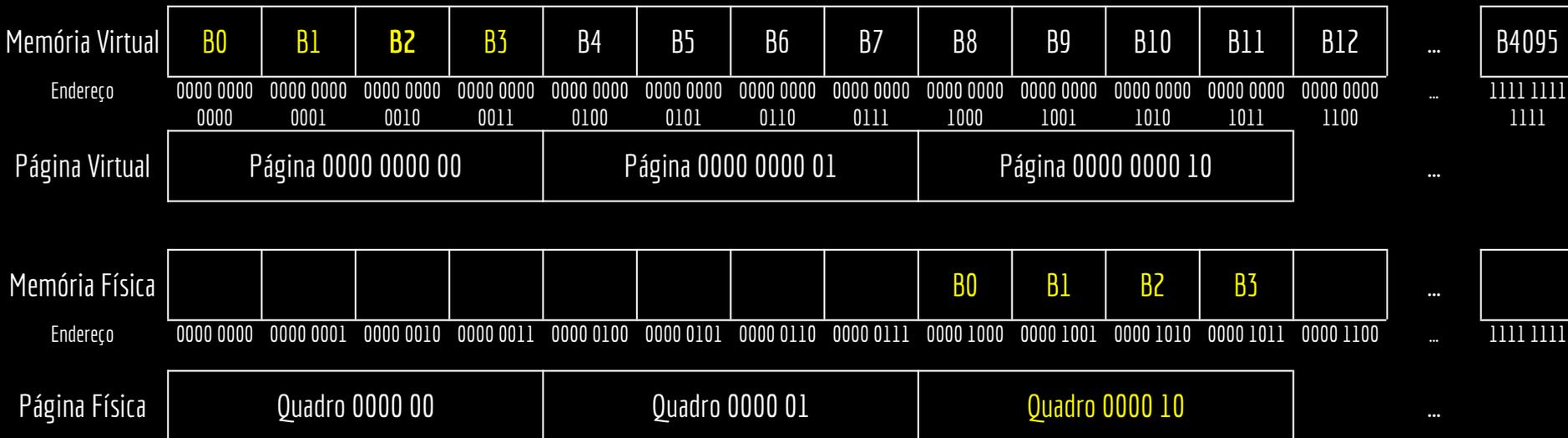


Tabela de Páginas

lb s0, 0000 0000 0010

0000 10

0000 0000 00

0000 0000 01

0000 0000 10

Vál?

Quadro

Vál?	Quadro
1	0000 10
0	
0	
...	

Memória Virtual

B0

B1

B2

B3

B4

B5

B6

B7

B8

B9

B10

B11

B12

...

B4095

Endereço

0000 0000
0000

0000 0000
0001

0000 0000
0010

0000 0000
0011

0000 0000
0100

0000 0000
0101

0000 0000
0110

0000 0000
0111

0000 0000
1000

0000 0000
1001

0000 0000
1010

0000 0000
1011

0000 0000
1100

...

1111 1111
1111

Página Virtual

Página 0000 0000 00

Página 0000 0000 01

Página 0000 0000 10

...

Memória Física

B0

B1

B2

B3

...

Endereço

0000 0000

0000 0001

0000 0010

0000 0011

0000 0100

0000 0101

0000 0110

0000 0111

0000 1000

0000 1001

0000 1010

0000 1011

0000 1100

...

1111 1111

Página Física

Quadro 0000 00

Quadro 0000 01

Quadro 0000 10

...

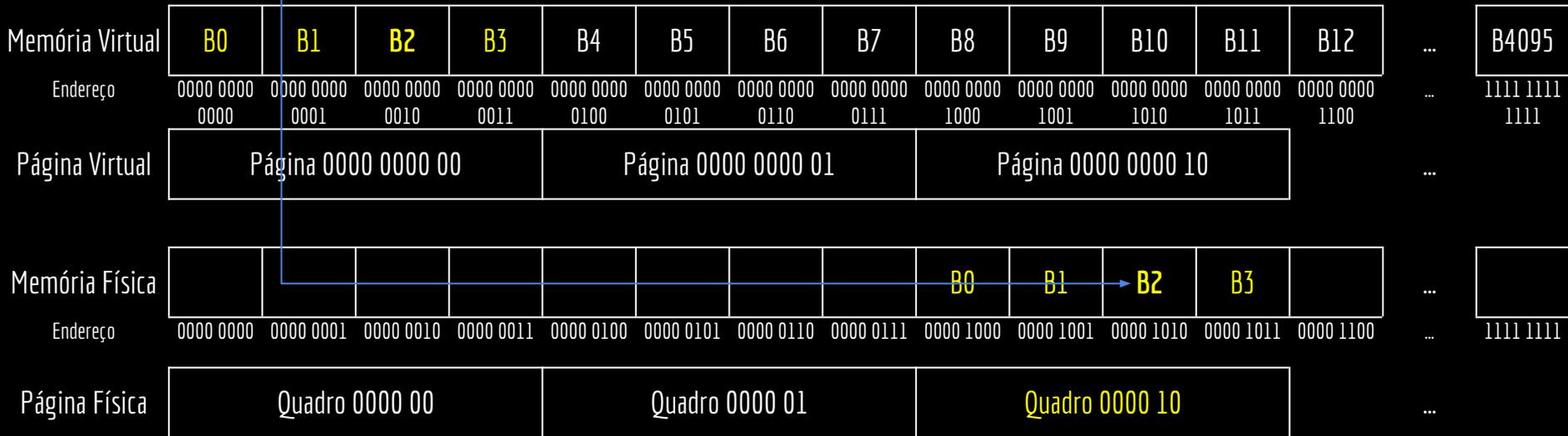
Tabela de Páginas

lb s0, 0000 0000 0010

Offset.

	Vál?	Quadro
0000 0000 00	1	0000 10
0000 0000 01	0	
0000 0000 10	0	
...		

0000 1010



Falta de Página

Quando uma página não mapeada para a memória física é solicitada (seu bit de validade estava inativo), ocorre uma **falta de página**.

O controle deve ser transferido para o Sistema Operacional.

Decide em qual quadro carregar a página.

Sabe onde no disco/SSD a página começa – Essa informação precisa constar na tabela de páginas.

Se não houver mais quadros disponíveis, decidir qual página substituir.

Página	Val?	Quadro	Bloco do Disco
0	1	27	2048
1	1	15	2049
2	0		2050
3	0		2051
...

Falta de Página

Quando uma página não mapeada para a memória física é solicitada (seu bit de validade estava inativo), ocorre uma **falta de página**.

O controle deve ser transferido para o Sistema Operacional.

Decide em qual quadro carregar a página.

Sabe onde no disco/SSD a página começa – Essa informação precisa constar na tabela de páginas.

Se não houver mais quadros disponíveis, decidir qual página substituir.

Exemplo: usando alguma aproximação para o LRU.

Leia sobre os algoritmos em livros de S.O.

O S.O. geralmente escolhe uma região do disco para manter as páginas, chamada de **espaço de swap**.

Página	Val?	Quadro	Bloco do Disco
0	1	27	2048
1	1	15	2049
2	0		2050
3	0		2051
...

Falta de Página

Se S.O. decidir substituir uma página da memória.

A página que vai sair da memória é gravada no disco.

Write-back.

A página que precisa ir para a memória principal é carregada do disco.

A tabela de páginas é atualizada.

Write-Back

Para tornar o write-back mais eficiente, podemos adicionar um *dirty bit* para cada entrada na tabela de páginas.

Indicar se a página foi alterada enquanto estava na memória.

Se não foi alterada, ela pode ser substituída por outra quando necessário, sem precisar salvar no disco.

A cópia que existe no disco ainda é válida.

Dirty?	Val?	Página	Quadro	Bloco do Disco
1	1	0	27	2048
0	1	1	15	2049
0	0	2		2050
0	0	3		2051
...

Problema

A tabela de páginas está na memória principal.

Quando um programa faz uma referência um endereço de memória, são necessários dois acessos:

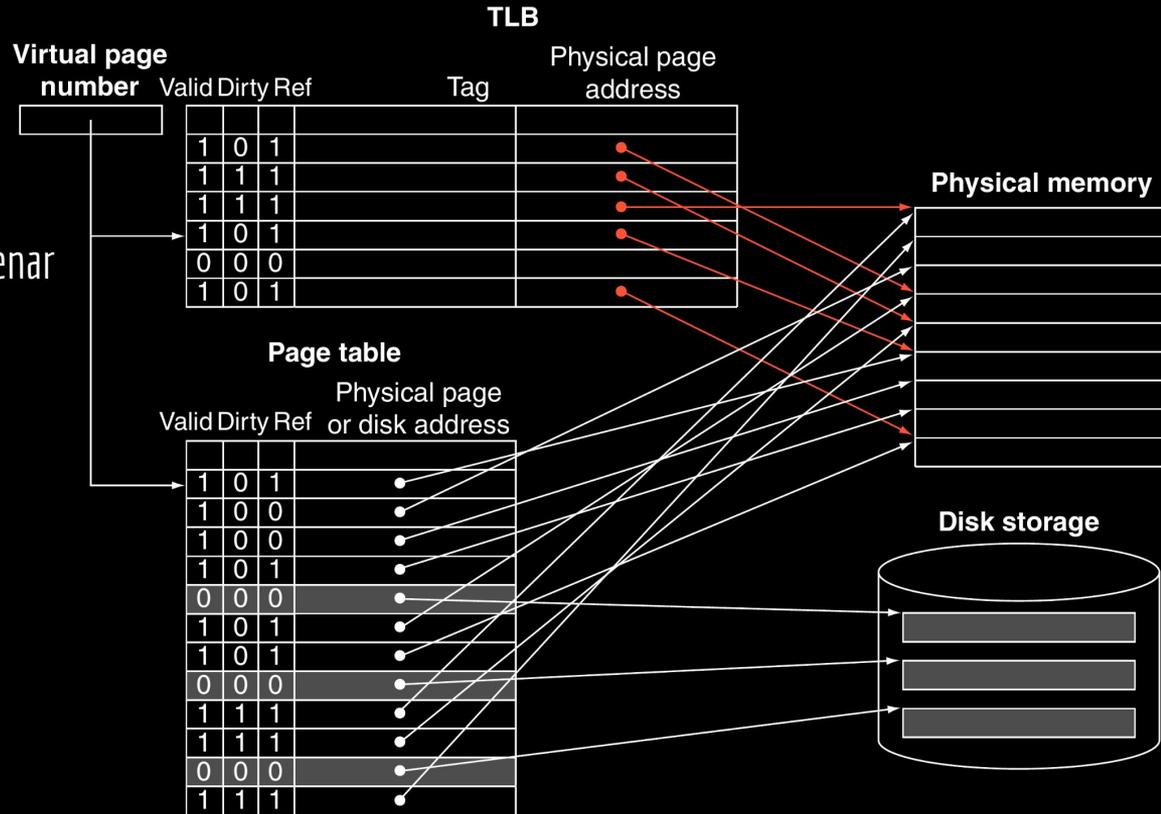
- Um para obter a tradução do endereço virtual para um endereço físico (através da tabela de páginas).

- Um para obter o dado.

TLB

Para tratar o problema, usamos uma **memória cache especial** na CPU, específica para armazenar ao menos uma porção da tabela de páginas.

TLB - Translation-lookaside buffer



TLB

Valores típicos para uma TLB.

Tamanho da TLB: 16–512 registros (linhas da TLB).

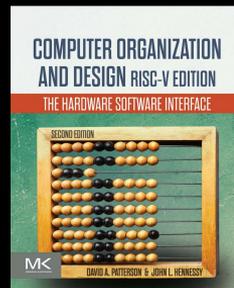
Tamanho do Bloco: 1–2 registros (tipicamente 4–8 bytes).

Hit time: 0.5–1 ciclos de clock.

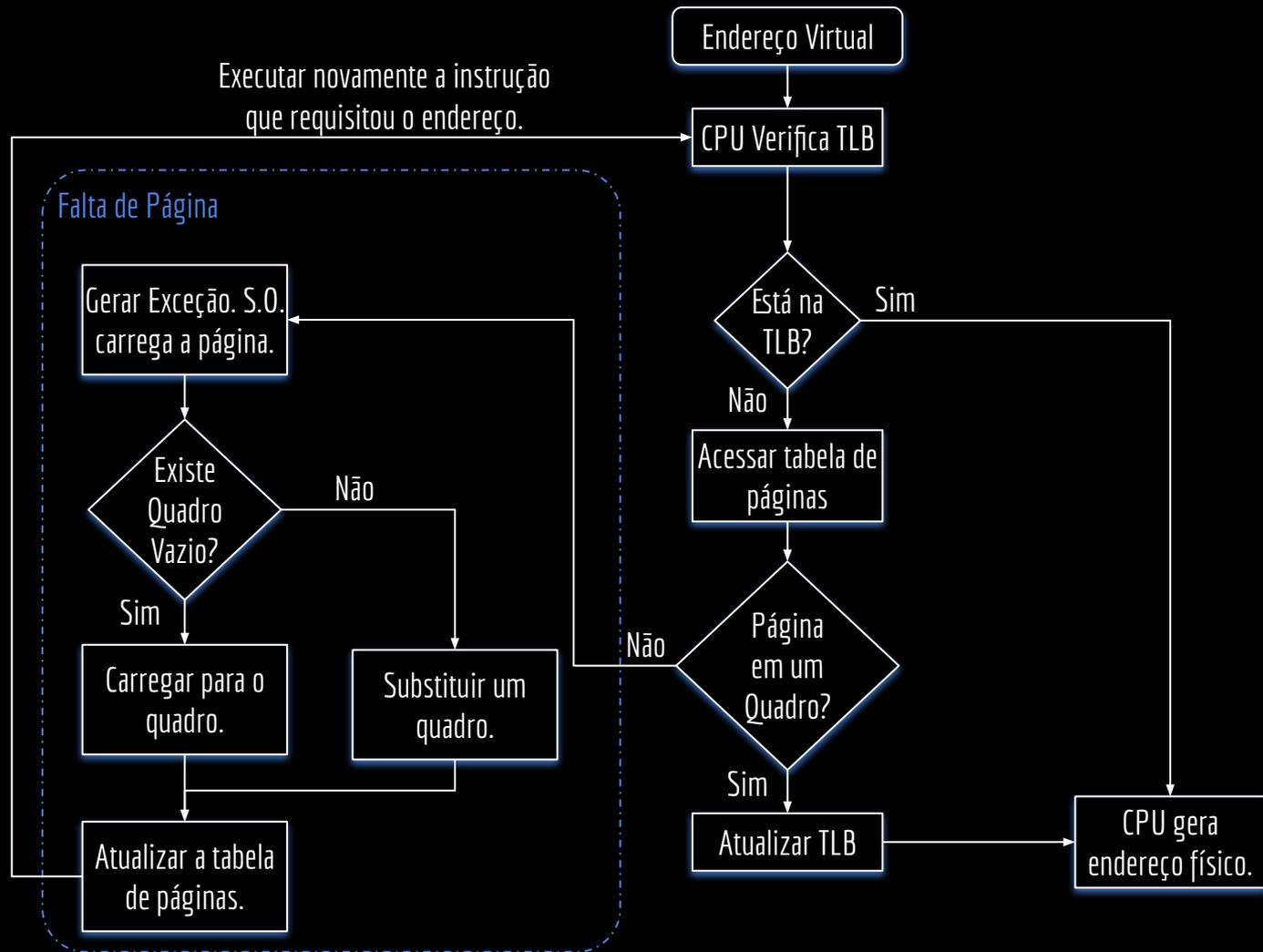
Miss penalty: 10–100 ciclos de clock.

Miss rate: 0.01%–1%.

Patterson, Hennessy.
Computer Organization and
Design RISC-V Edition: The
Hardware Software
Interface. 2020.



Fluxograma



MMU

A tradução de endereços virtuais para físicos pode ser feita automaticamente por um controlador de memória específico.

MMU – Memory Management Unit.

Contém a TLB, e traduz endereços virtuais para físicos.

Thrashing

Quando precisamos trocar constantemente as páginas da memória (fazendo transações com o disco/SSD), devido a, por exemplo, pouca memória física, ou um algoritmo ineficiente de substituição de páginas, temos uma condição chamada de **thrashing**.

O processador vai passar a maior parte do tempo parado esperando as substituições de páginas.

Troca de contexto

Quando há uma troca de contexto (o S.O. manda outro processo ser executado na CPU).

- O registrador que aponta para a tabela de páginas é atualizado para apontar para a tabela do novo processo.
- A TLB é invalidada.

Exemplo do Mundo Real

Intrinsity FastMATH.

Páginas de 4KB e Endereços de 32 bits.

Endereço virtual da página com 20 bits, e offset de 12 bits.

TLB com 16 registros, totalmente associativa.

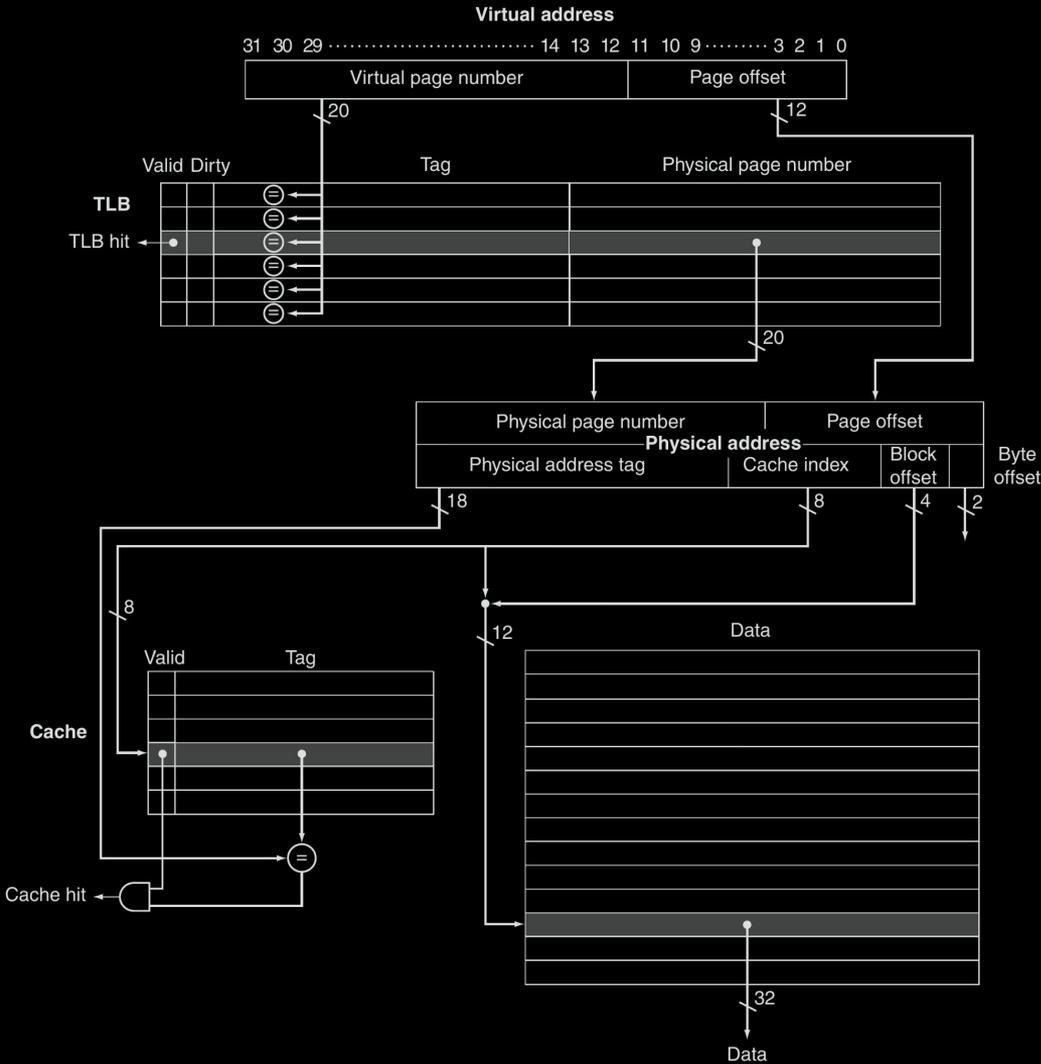
Cada registro tem 64 bits.

20 bits de tag (endereço da pág. virtual), 20 bits da pág. física, bit de validade, *dirty* bit, e outros bits de gerência.

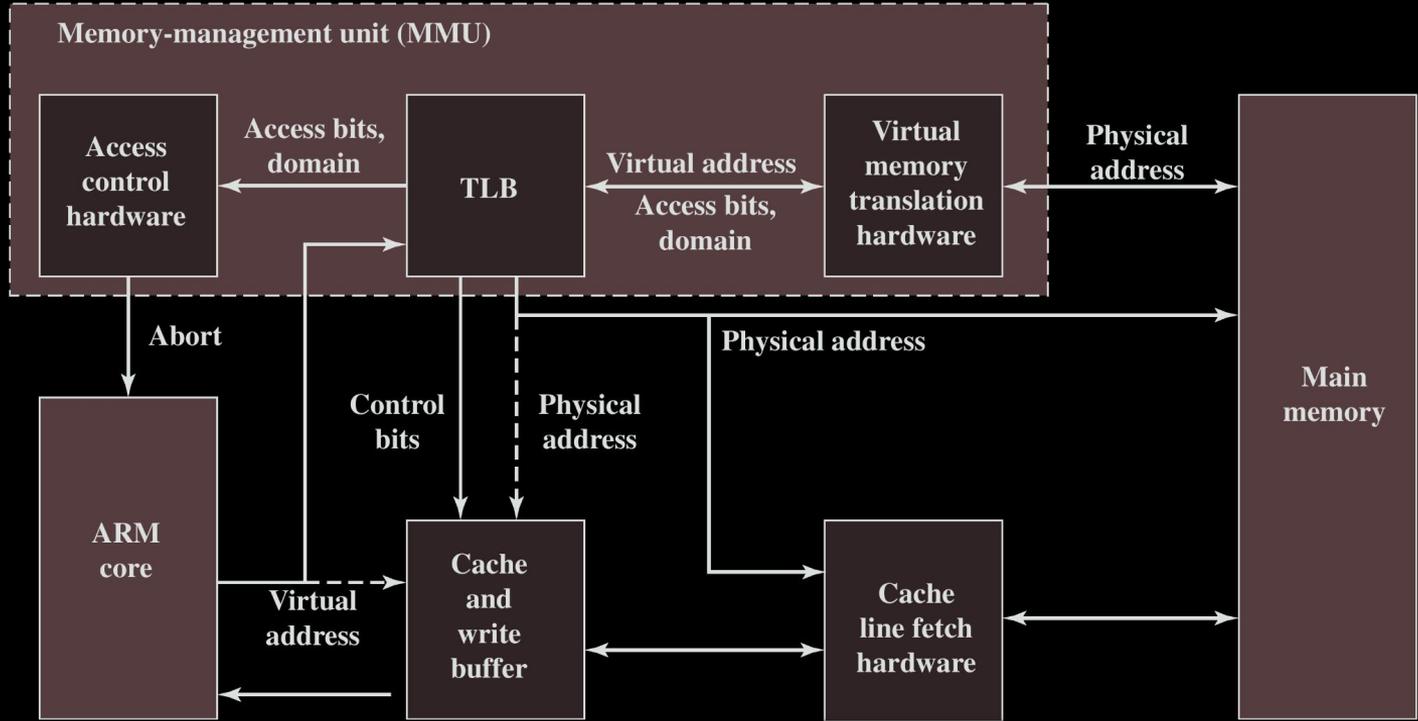
Misses na TLB são tratados por software.

Chama o Sistema Operacional.

Intrinsity FastMATH



Exemplo do Mundo Real

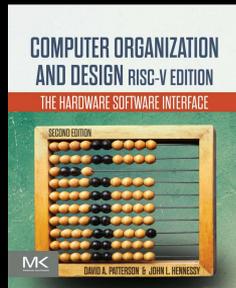


Exercícios

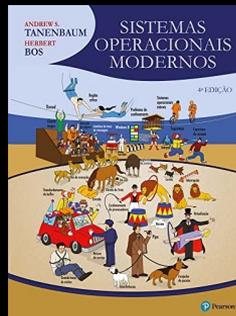
1. Considere um sistema que suporta 32 GB de memória física, e 128 GB de memória virtual (por programa). Considerando páginas de 4 KB, e 4 MB, responda.
 - a. Quantos quadros existem.
 - b. Quantas páginas existem.
 - c. Quantos e quais bits são offset? Quantos e quais bits formam a TAG para a TLB?
 - d. Considerando que cada registro na tabela de páginas precisa armazenar o endereço do quadro, um *dirty bit*, um bit de validade, e um endereço de bloco no disco rígido de 32 bits, qual o tamanho da tabela de páginas?
2. Considere um sistema que suporta 32 KB de memória física, e 64 KB de memória virtual. Considere páginas de 256 bytes. Preencha a tabela disponibilizada na UFPR Virtual com os endereços físicos de memória para cada requisição. Em caso de miss na TLB, substitua aleatoriamente um registro para atualizar a TLB. Em caso de falta de página, apenas marque isso (não precisa “carregar da memória” e atualizar a TLB).

Referências

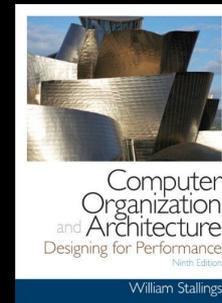
Patterson, Hennessy.
Computer Organization and Design RISC-V Edition: The Hardware Software Interface. 2020.



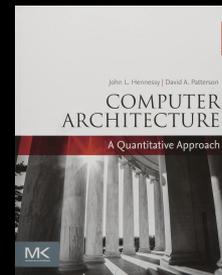
Tanenbaum, Bos. Sistemas operacionais modernos. 4a ed. 2016.



Stallings, W. Organização de Arquitetura de Computadores. 10a Ed. 2016.



Hennessy, Patterson.
Arquitetura de Computadores: uma abordagem quantitativa. 2019.



Licença

Esta obra está licenciada com uma Licença [Creative Commons Atribuição 4.0 Internacional](https://creativecommons.org/licenses/by/4.0/).