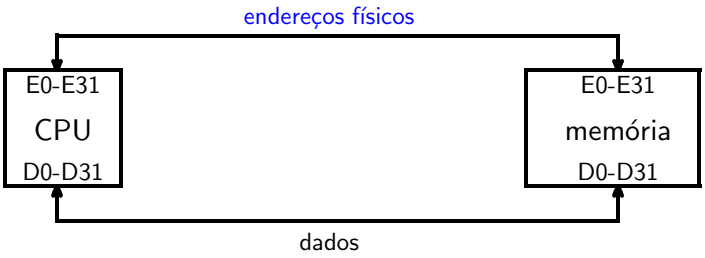
Espaço de Endereçamento – três modelos

- Primeiro Modelo (até ≈1960)
  - Programador supõe que espaço de endereçamento é contínuo
    espaço se extende de 0x0000 0000 a 0xffff ffff
  - programa deve ser carregado sempre no endereço físico inicial
  - memória pode conter somente um programa em execução
- Segundo Modelo (até 1972 - IBM 370, idéia de 1962)
  - EdE contínuo, > que memória física ⟶ overlays
  - Registrador de tradução para fazer relocação do programa
    ⟶ programa pode ser carregado em qualquer endereço físico
  - mais de um programa carregado em memória para execução
- Terceiro Modelo (após 1972 - IBM 370)
  - EdE > que memória física, tradução de ender transparente
  - muitos programas carregados na memória

Limitações do Endereçamento Físico

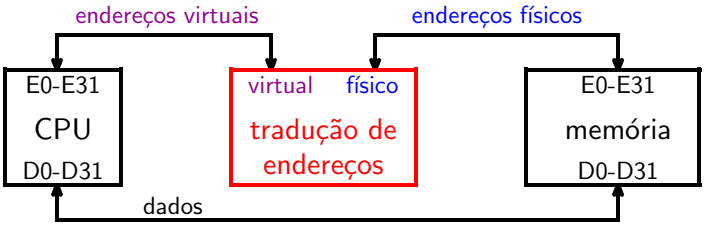


Todos programas compartilham um espaço de endereçamento físico

Programas em linguagem de máquina "conhecem"
organização do computador

Não há maneira de evitar que
um programa acesse qualquer recurso da máquina

Nível de indireção para obter proteção



Programas de usuário executam em espaço de endereçamento virtual

Lógica de tradução de endereços (LTE),
gerenciada pelo Sistema Operacional (SO),
mapeia endereços virtuais em endereços físicos em memória

LTE suporta funcionalidades "modernas" de SOs: Kilburn, 1962
proteção, tradução (de endereços), compartilhamento

Memória Virtual

Processador "enxerga" espaço de endereçamento de 4 Gbytes
⇌ pointers de 32 bits

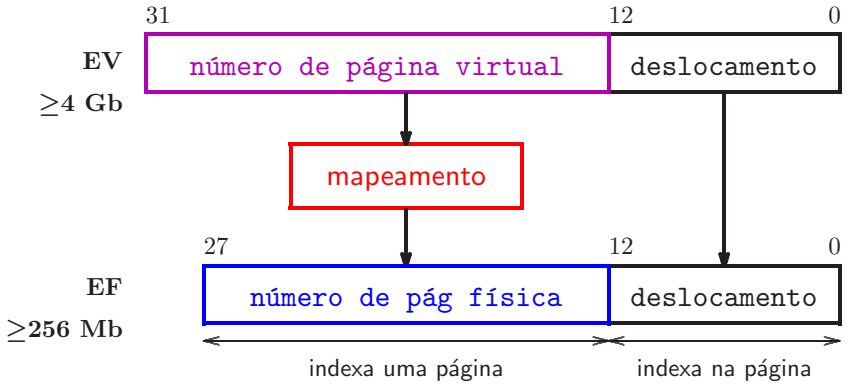
Memória física (RAM) tem de 256Mbytes a 16Gbytes
endereços físicos com 28 a 34 bits

Sistema de memória virtual mapeia
endereços virtuais (4 Gbytes)
em endereços físicos (256M-16Gbytes)

Tradução de endereço virtual ⟶ endereço físico
tem como efeito principal a separação de espaços de endereçamento
e como efeito colateral importante a proteção entre EdEs ≠s

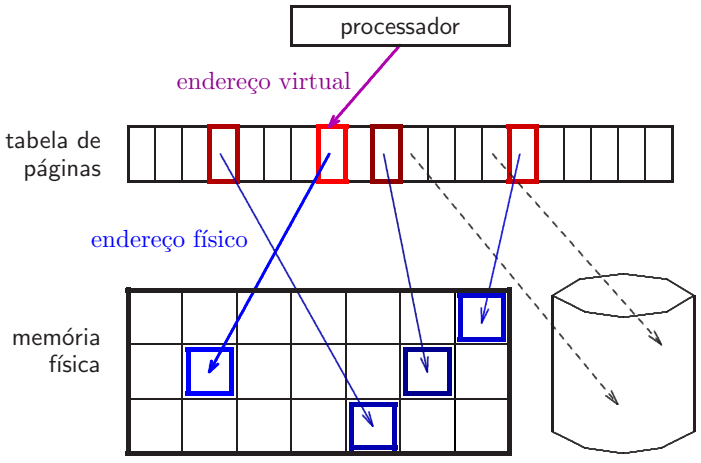
Paginação (i)

Espaço de endereçamento dividido em páginas de 4-8Kbytes
Tabela de Páginas mantém mapeamento entre
endereços virtuais e endereços físicos

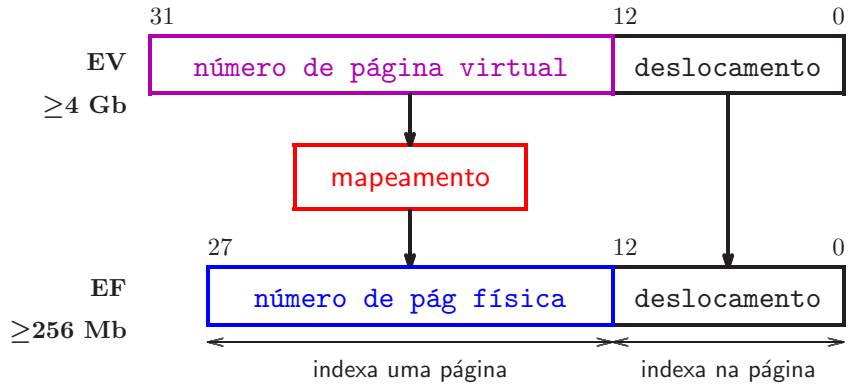


Paginação (ii)

Tabela de Páginas faz mapeamento associativo entre
páginas virtuais e páginas físicas



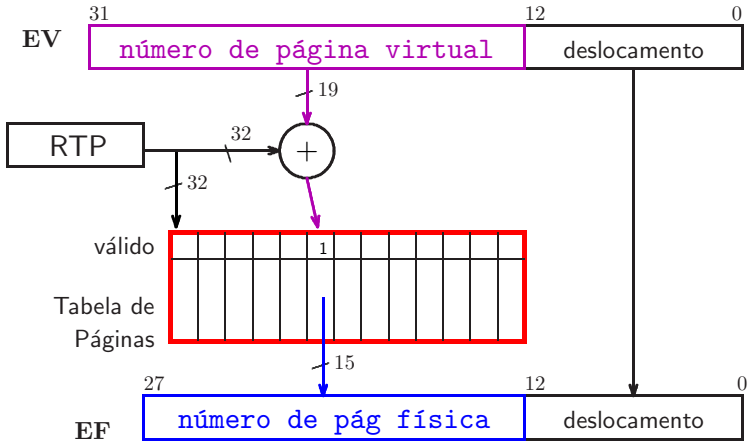
Paginação (iii)



Qual o tamanho da tabela de páginas?

Quantas tabelas de páginas são necessárias?

Paginação (iv)



Registrador de Tabela de Páginas (RTP) aponta
início da Tabela de Páginas do processo

Hierarquia de Memória

Tamanho dos componentes da hierarquia de memória

nível	capacidade [bytes]	$\mathcal{T}_{\text{acesso}}$ [ciclos]
cache primária L1	16–64 K	1
cache secundária L2	64–1024 K	10
memória DRAM	256-4096 M	100
disco	$\gg$ 4 G	100.000

Memória física funciona como cache para o disco

Custo de uma falta de página condiciona projeto da hierarquia:
- páginas grandes para amortizar penalidade (custo da carga)
- reduzir taxa de faltas é importante⟶ mapeamento associativo
- tratamento de faltas em software ⟶ algoritmos melhores
- escrita preguiçosa

DRAM como Cache de Disco I

Se página virtual não está em memória física,
⟶ deve ser copiada do disco
⟶ alguma página deve ser ejetada para abrir espaço
localidade recomenda:
vítima deve ser "usada no passado mais distante" (LRU)

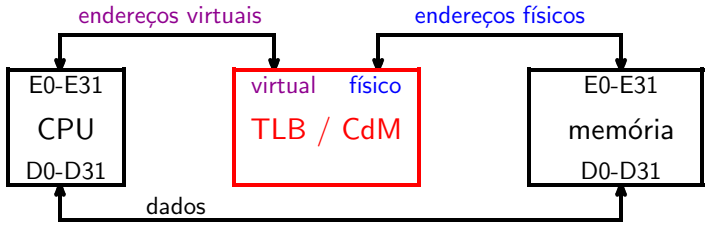
Exemplo: 11, 10, 12, 9, 11, 7, 11, 13 LRU c.r.a 13? 11?

Escrita é preguiçosa:
páginas somente de leitura (código) são substituídas
páginas com atualizações (escritas) são marcadas `sujas` e,
antes de sobre-escritas/substituídas, disco deve ser atualizado

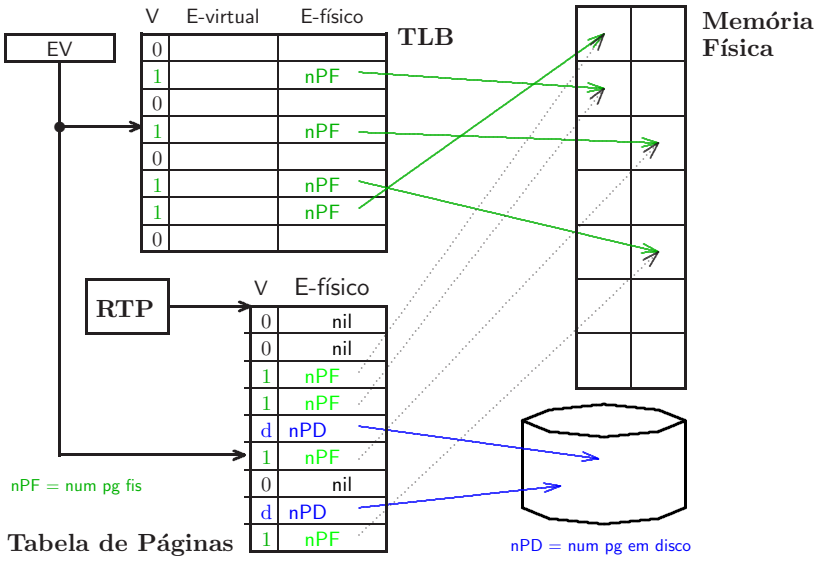
Cache de Mapeamentos (i)

A cada referência, processador consulta TP
para descobrir endereço físico do objeto
⟶ faz uma referência à Tabela de Páginas para obter endereço
e então faz referência ao objeto...
⟶ para cada referência, DOIS acessos à memória...

Solução:
Cache de Mapeamentos (CdM) ou Translation Lookaside Buffer (TLB)



Cache de Mapeamentos (ii)



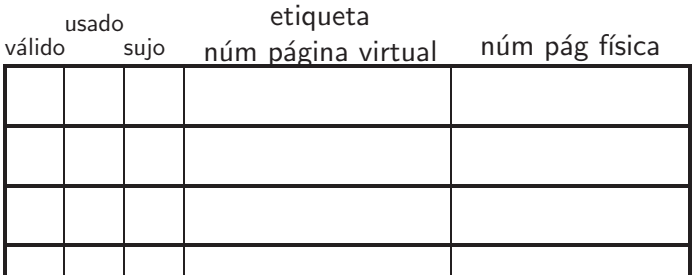
Cache de Mapeamentos (iii)

Processador procura mapeamento na TLB
se encontra, completa referência;
senão, busca mapeamento da Tabela de Páginas (em mem física)
e guarda na TLB para uso futuro;

Parâmetros de projeto:

- tamanho de bloco: 1 ou 2 mapeamentos
- tempo de acerto: 1/2 ciclo
- penalidade por falta: 10 a 30 ciclos
- taxa de faltas: 0.01% a 1%
- tamanho: 32 a 128 blocos
- assocatividade: alta (4,8,total)

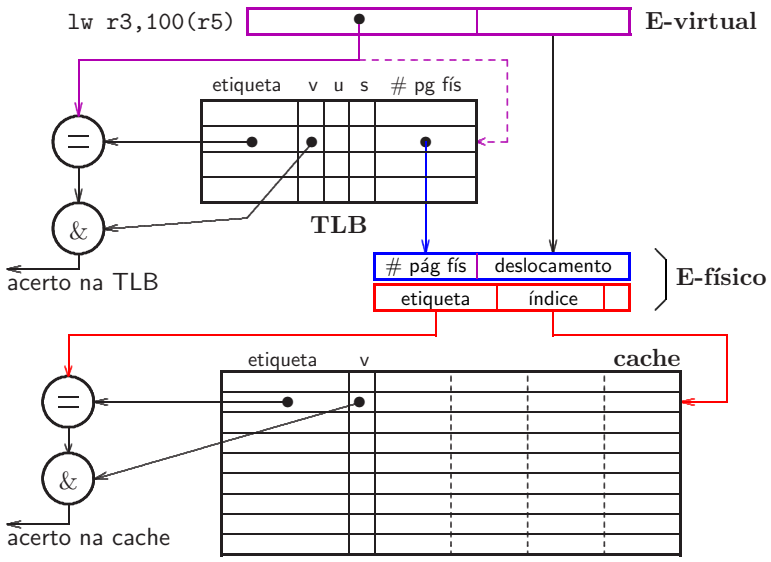
Campos da Cache de Mapeamentos



`válido == 1` se mapeamento é válido
`usado == 1` se página foi referenciada recentemente
`sujo == 1` se ocorreu uma ou mais escritas na página

Se página virtual não está em memória,
então não pode haver um mapeamento da página na TLB

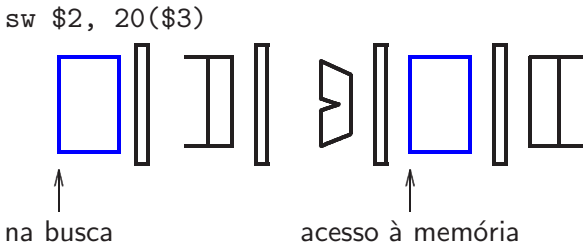
TLB & Cache (i)



TLB & Cache (ii)

```
CPUemiteReferencia(endEfetivo);
if ( endEfetivo na TLB ) {
    endFisico = TLBtrad(endEfetivo);
} else {                    /* busca na Tabela de Páginas */
    TLB[vitima] = TP[endEfetivo];
    endFisico = TLBtrad(endEfetivo);
}
if ( ! endFisico na cache ) {  /* busca em memória */
    Cache[endFisico] = Mem[endFisico];
}
entregaParaProcessador;
```


TLB & Cache (iii)



Faltas & Excessões

Causas: endereço ilegal — fora do espaço de endereçamento
ou desalinhado — end%4≠0
ou falta na TLB
ou falta na tabela de páginas

TLB & Cache (iv)

Falta na TLB pode ocorrer:

- na busca de instrução
  ⟶ processador esvazia "pipeline"
  re-carrega TLB
  e busca instrução novamente
  custo: número de segmentos + carga da TLB
- referência a dados (`lw sw`)
  ⟶ instruções à frente completam
  instrução não pode alterar estado — `lw $1, 0($1)`
  processador re-carrega TLB
  e busca instrução novamente
  custo: 1-2 segmentos + carga da TLB + 4-3 segmentos

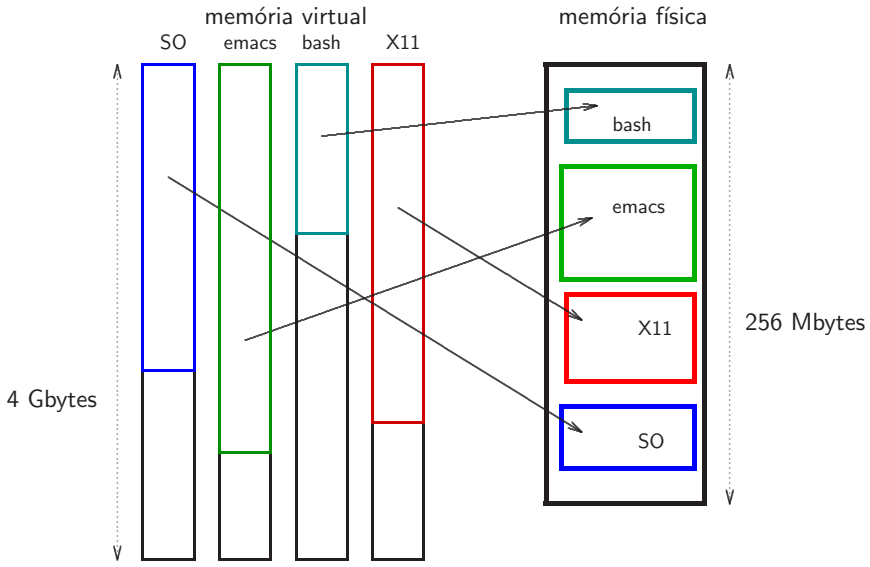
Reposição de Páginas

Falta na Tabela de Páginas:
Se página virtual não está em memória *page fault*
SO assume controle e requisita cópia ao controlador de disco
SO escolhe vítima para ser substituída pela nova página;
$\longrightarrow$ se vítima estiver suja, SO deve atualizar disco

Localidade implica em vítima ser
página usada no passado mais distante (LRU)

Tempo de carga é da ordem de $\mathbf{10^5}$ a $\mathbf{10^6}$ ciclos...

Processos e Proteção



Processos e Proteção (i)

Definição: **processo é um programa em execução**

Estado de um processo:

- conteúdo dos registradores do processador + PC, RTP, status
- conteúdo da TLB
- conteúdo da Tabela de Páginas
- conteúdo da memória (variáveis e pilha)

Processos e Proteção (ii)

A cada processo corresponde um
espaço de endereçamento

A cada espaço de endereçamento corresponde um
domínio de proteção

Proteção:
um processo não pode interferir no
espaço de endereçamento de outro processo

Processador detecta quebra de proteção ao acessar TLB
⟶ excessão de violação de proteção como detecta?

Processos e Proteção (iii)

Mecanismos para implementar proteção:

- dois modos de operação: supervisor e usuário
  ⟶ instruções reservadas ao modo supervisor
- parte do estado do processador não pode ser alterada
  em modo usuário
- instruções especiais permitem mudança no estado
  ⟶ chamadas de sistema (syscall)

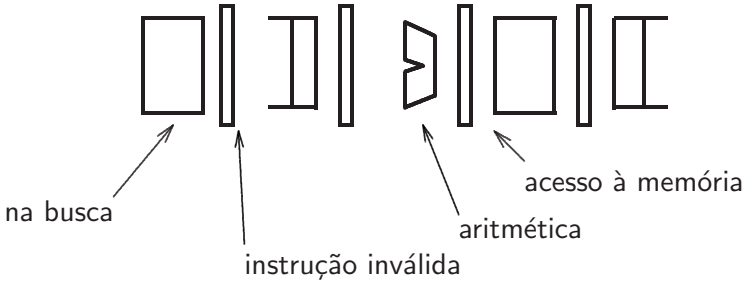
Aluno de primeiro ano não pode formatar os discos... }Bv/

Processos e Proteção (iv)

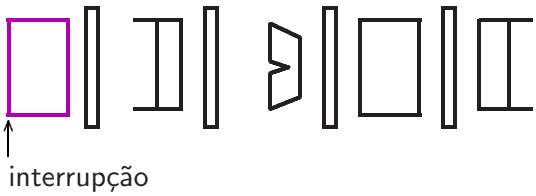
Tratamento de exceções:

- desabilita interrupções
- salva EPC (Error PC)
- salva registrador de status (Cause Register)
- espera drenar pipeline
- salva registradores
- habilita interrupções
- trata excessão ⟶ depende do tipo de evento
- SO faz troca de contexto e escolhe outro processo para executar
- recompõe estado do novo processo
- carrega PC com instrução após última interrupção do novo processo
- excessões ocorrem durante ('no meio') a execução das instruções

Excessões em processador segmentado



Interrupções em processador segmentado



Interrupções sinalizam eventos externos ao processador

Atendimento implica em processamento de código para tratar evento

Vetor de Interrupções (e de excessões)
contém endereços das funções associadas aos eventos

São detectadas entre duas instruções

Tratamento de Interrupções I

Vetor de Interrupções (hipotético)

ender. físico	ender. tratador	evento
0x0000	*hardReset()	reset a frio
0x0004	*softReset()	reset a quente
0x0008	*TLBinstr()	falta na TLB de código
0x000c	*TLBdado()	falta na TLB de dados
0x0010	*underflow()	underflow em PF
0x0014	*overflow()	overflow em PF
0x0018	*interrDisco()	tratador interr disco
0x001c	*interrRede()	tratador interr rede
0x0020	*interrDMA()	tratador interr DMA
0x0024	*divZero()	divisão por zero
0x0028	*instrInval()	instrução inválida
0x002c	...	...

Tratamento de interrupções II

- desabilita interrupções
- salva registrador de status (Cause Register)
- espera drenar pipeline
- salva registradores
- habilita interrupções
- salta para endereço do código que trata da interrupção
- desabilita interrupções
- recompõe registradores
- habilita interrupções
- retorna para instrução onde interrupção detectada
- detectadas entre instruções consecutivas