Sistemas Operacionais I

Profa. Kalinka Regina Lucas Jaquie Castelo Branco
kalinka@icmc.usp.br

Universidade de São Paulo

Outubro de 2020

Gerência de Memória

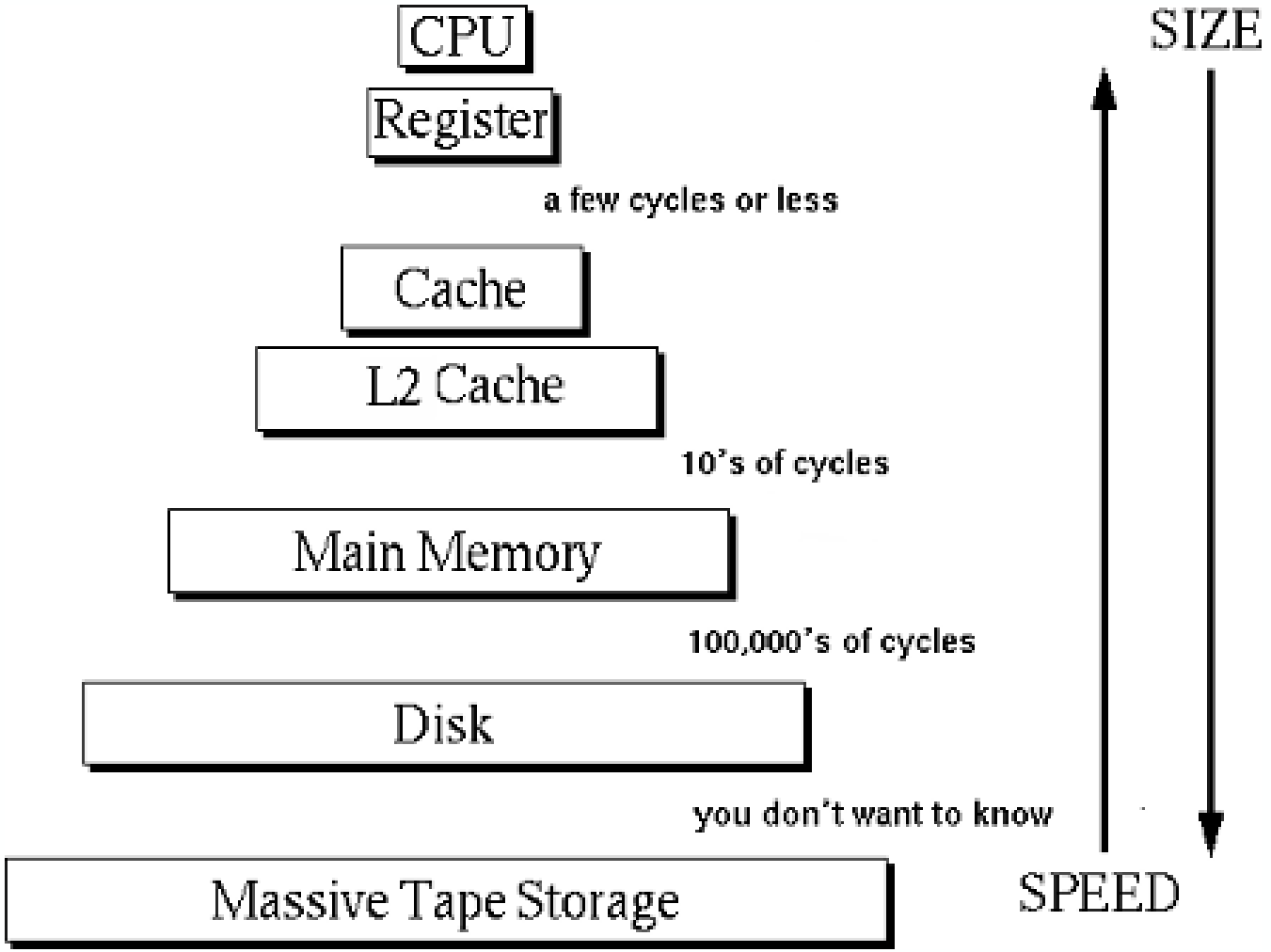
- Recurso importante.
- Tendência atual do software.
  - Lei de Parkinson: "Os programas se expandem para preencher a memória disponível para eles" (adaptação).
- Hierarquia de memória:
  - Cache
  - Principal
  - Disco

- Idealmente os programadores querem uma memória que seja:
  - Grande
  - Rápida
  - Não Volátil
  - Baixo custo
- Infelizmente a tecnologia atual não comporta tais memórias
- A maioria dos computadores utiliza Hierarquia de Memórias que combina:
  - Uma pequena quantidade de memória cache, volátil, muito rápida e de alto custo
  - Uma grande memória principal (RAM), volátil, com centenas de MB ou poucos GB, de velocidade e custo médios
  - Uma memória secundária, não volátil em disco, com gigabytes (ou terabytes), velocidade e custo baixos

Gerência de Memória

- Cache
  - Pequena quantidade
  - k bytes
  - Alto custo por byte
  - Muito rápida
  - Volátil
- Memória Principal
  - Quantidade intermediária
  - M bytes
  - Custo médio por byte
  - Velocidade média
  - Volátil
- Disco
  - Grande quantidade
  - G bytes
  - Baixo custo por byte
  - Lenta
  - Não volátil

Gerência de Memória



Gerência de Memória

- Cabe ao Gerenciador de Memória gerenciar a hierarquia de memória
- Controla as partes da memória que estão em uso e quais não estão, de forma a:
  - alocar memória aos processos, quando estes precisarem;
  - liberar memória quando um processo termina; e
  - tratar do problema do *swapping* (quando a memória é insuficiente).

Gerência de Memória

- Para cada tipo de memória:
  - Gerenciar espaços livres/ocupados;
  - Alocar processos/dados na memória;
  - Localizar dado.
- Entre os níveis de memória, deve-se:
  - Gerenciar as trocas
- Gerenciador de memória:
  - Responsável por alocar e liberar espaços na memória para os processos em execução;
  - Responsável por gerenciar chaveamento entre a memória principal e o disco, e memória principal e memória cache

Gerência de Memória

- Sistemas de Gerenciamento de Memória, podem ser divididos em 2 classes:
  - Sistemas que, durante a execução levam e trazem processos entre a memória principal e o disco (troca de processos e paginação)
  - Sistemas mais simples, que não fazem troca de processos e nem paginação

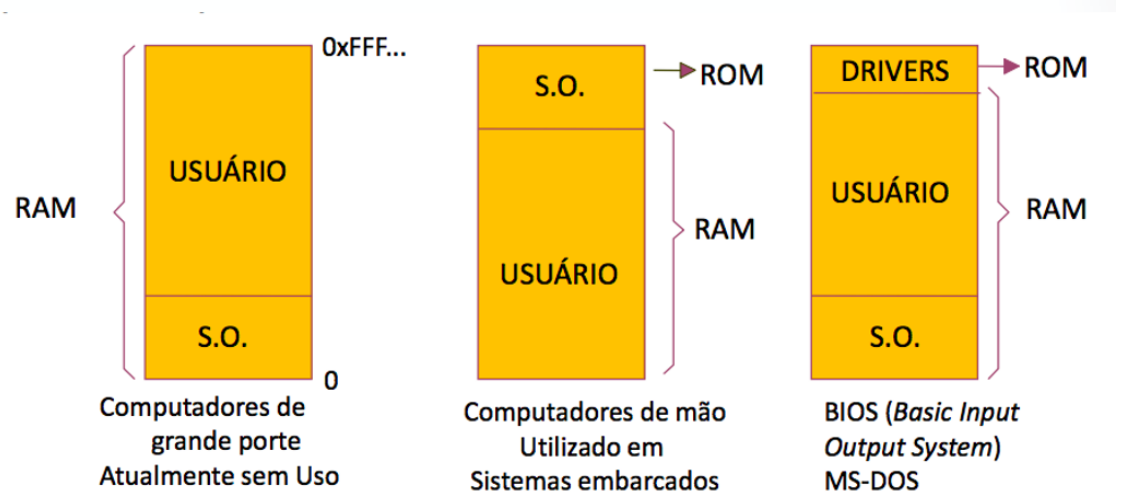
Gerência de Memória

- Monoprogramação sem trocas de processos ou Paginação
- Sistemas Mono-usuários: gerência de memória é bem simples, pois toda a memória é alocada à próxima tarefa, incluindo a área do S.O.
- Erros de execução podem vir a danificar o S.O.
- Neste caso, a destruição do S.O. é um pequeno inconveniente, resolvido pelo recarregamento do mesmo.

Gerência de Memória

Monoprogramação sem trocas de processos ou Paginação

- Gerenciamento mais simples
- Um sistema operacional com um processo de usuário



Três esquemas simples de organização de memória

Gerência de Memória

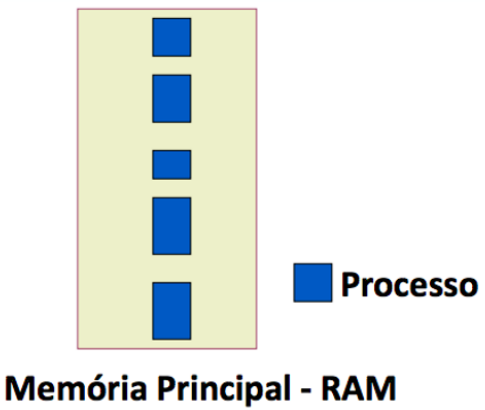
Multiprogramação com partições Fixas

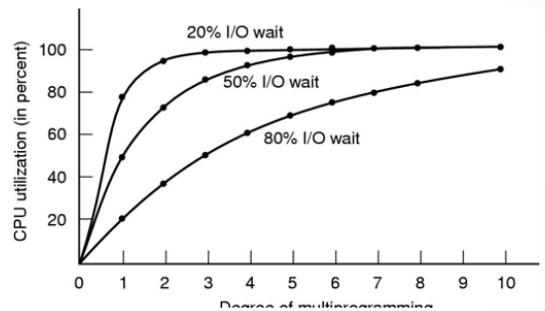
- Sistemas Monoprogramados: raramente usados atualmente.
- Sistemas modernos: permitem multiprogramação
- A maneira mais comum de realizar a multiprogramação é dividir simplesmente a memória em n partições (provavelmente de tamanhos diferentes).
- Esta divisão pode ser feita de maneira manual, quando o sistema é inicializado
- Ao chegar, um *job*, pode ser colocado em uma fila de entrada associada à menor partição, grande o suficiente para armazená-lo

Gerência de Memória

Modelagem da Multiprogramação

- Múltiplos processos sendo executados.
- Eficiência de CPU.





A utilização da CPU como uma função do número de processos na memória.

Necessidade de Particionamento da Memória Principal.

Gerência de Memória

Modelagem da Multiprogramação

- O uso da Multiprogramação pode melhorar a utilização da CPU
- Considerando a utilização da CPU de modo probabilístico:
  - Suponha que um processo gaste uma fração p de seu tempo esperando pela finalização de sua solicitação de E/S
  - Com n processos simultâneos na memória, a probabilidade de todos os n processos estarem esperando por E/S (situação em que a CPU está ociosa) é p^n
  - A utilização da CPU é dada pela fórmula: utilização da $CPU = 1 - p^n$

Gerência de Memória

- Multiprogramação - Vários processos na memória:
  - Proteção: Como proteger os processos uns dos outros e o *kernel* de todos os processos?
  - Necessidade de realocação: Como tratar a realocação? O processo pode estar em diferentes posições da memória.
- Todas as soluções envolvem equipar a CPU com um hardware especial - MMU (*memory management unit*).

Gerência de Memória

- MMU (do inglês *Memory Management Unit*) é um dispositivo de **hardware** que transforma endereços virtuais em endereços físicos.
- Na MMU, o valor no registro de relocação é adicionado a todo o endereço lógico gerado por um processo do utilizador na altura de ser enviado para a **memória**. O programa do utilizador manipula endereços lógicos; ele nunca vê endereços físicos reais.

Vamos voltar a falar disso!!!

Gerência de Memória

Realocação e Proteção

- Pode não ter certeza de onde o programa será carregado na memória
  - As localizações de endereços de localização das variáveis e do código das rotinas não podem ser absolutos
  - Deve-se manter um programa fora das partições de outros processos
- Uso de valores de base e limite
  - Os endereços das localizações são somados a um valor de base para mapear um endereço físico
  - Valores de localizações maiores que um valor limite são considerados erro

Gerência de Memória

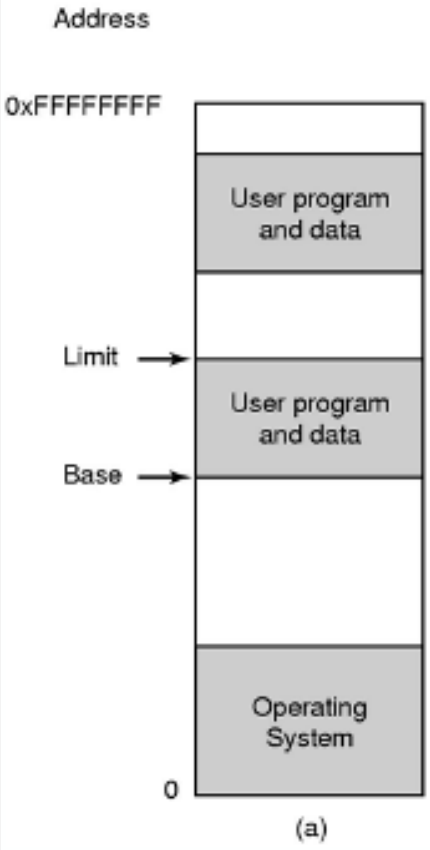
- Realocação:
  - Quando um programa é *linkado* (programa principal + rotinas do usuário + rotinas da biblioteca - executável) o *linker* deve saber em que endereço o programa irá iniciar na memória;
  - Nesse caso, para que o *linker* não escreva em um local indevido, como por exemplo na área do SO (100 primeiros endereços), é preciso de realocação: $\#100 + Delta$ - que depende da partição!!!
- Proteção:
  - Com várias partições e programas ocupando diferentes espaços da memória é possível acontecer um acesso indevido.

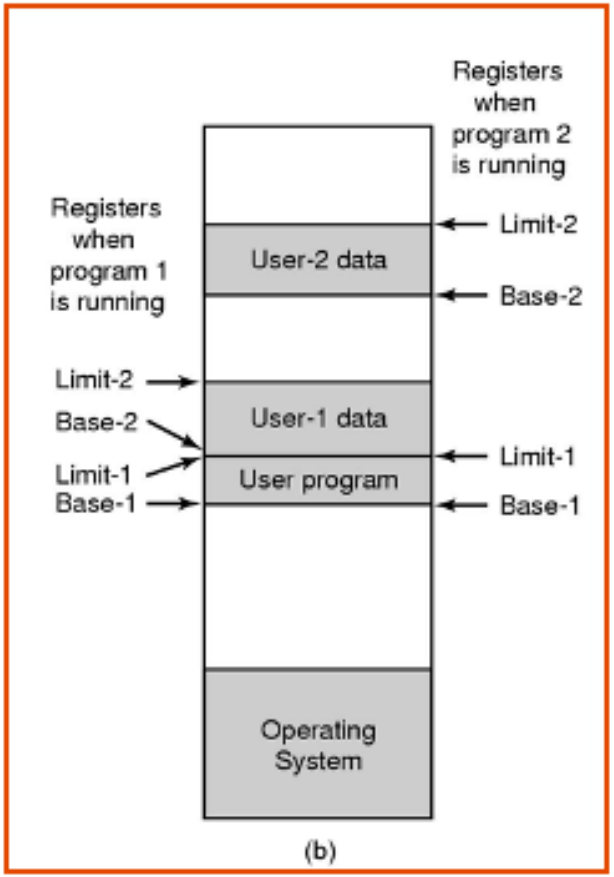
Gerência de Memória

Solução para ambos os problemas

- 2 registradores - base e limite
  - Quando um processo é escalonado o **registrador-base** é carregado com o endereço de início da partição e o **registrador-limite** com o tamanho da partição.
  - O registrador-base torna **impossível** a um processo uma remissão a qualquer parte de memória abaixo de si mesmo.
  - Automaticamente, a MMU adiciona o conteúdo do **registrador-base** a cada endereço de memória gerado.
  - Endereços são comparados com o **registrador-limite** para prevenir acessos indevidos.

Registradores Base e Limite





b) MMU mais sofisticada → dois pares de registradores: segmento de dados usa um par separado;

- MMU modernas têm mais pares de registradores.

Gerência de Memória

Partição/Alocação

- Particionamento da memória pode ser realizado de duas maneiras:
  - Partições fixas (ou alocação estática)
  - Partições variáveis (ou alocação dinâmica)
- **Partições Fixas:**
  - Tamanho e número de partições são fixos (estáticos);
  - Não é atrativo, porque partições fixas tendem a desperdiçar memória (Qualquer espaço não utilizado é literalmente perdido)
  - Mais simples.

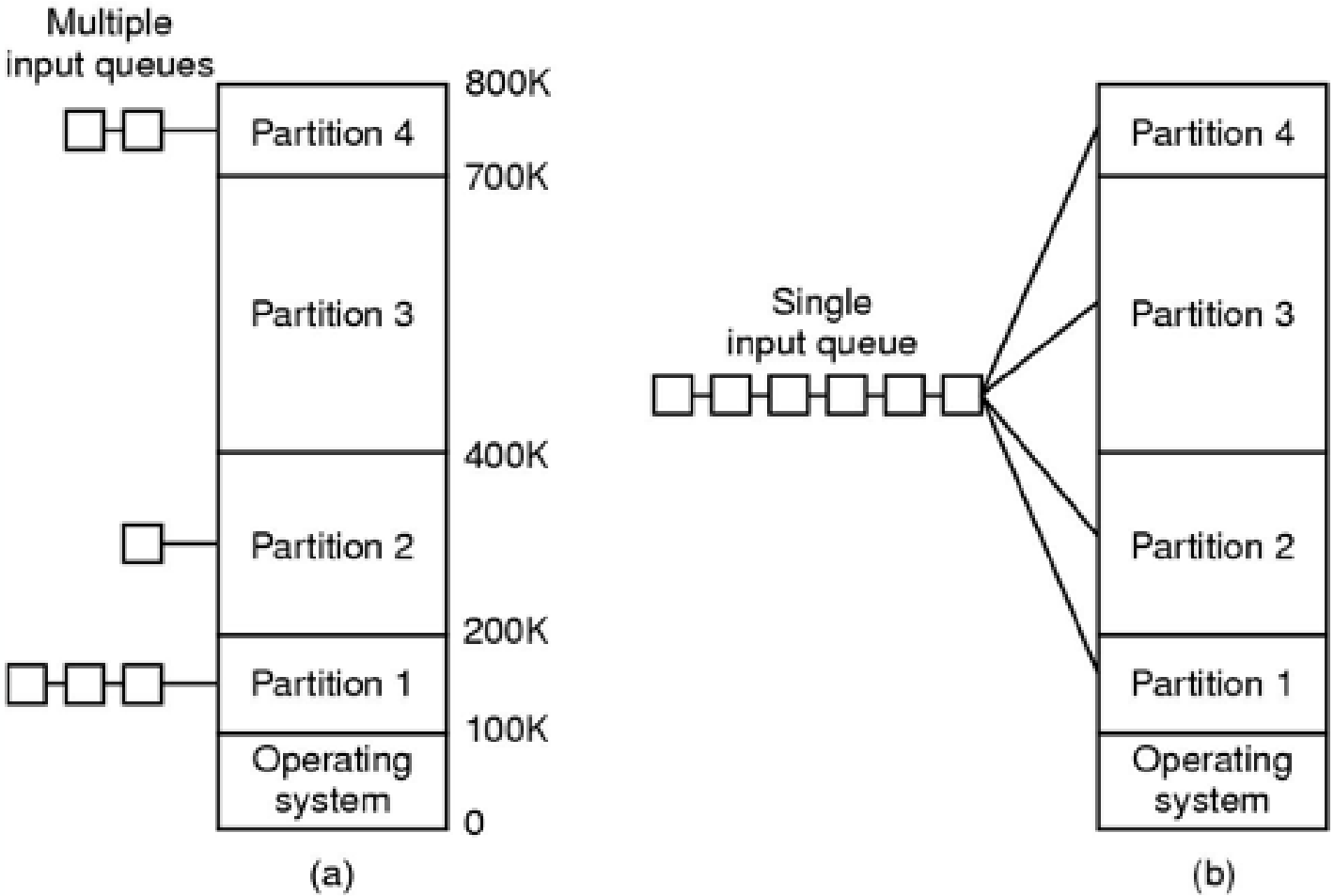
Gerência de Memória

Partições Fixas

- Filas múltiplas:
  - Problema: filas não balanceadas
- Fila única:
  - Facilita gerenciamento;
  - Implementação com Lista:
  - Melhor utilização da memória, pois procura o melhor processo para a partição considerada;
  - Diferentes algoritmos podem ser considerados para alocar os processos.

Partições Fixas

- Fila separada para cada partição
- Única fila de entrada



Gerência de Memória

Partições Fixas

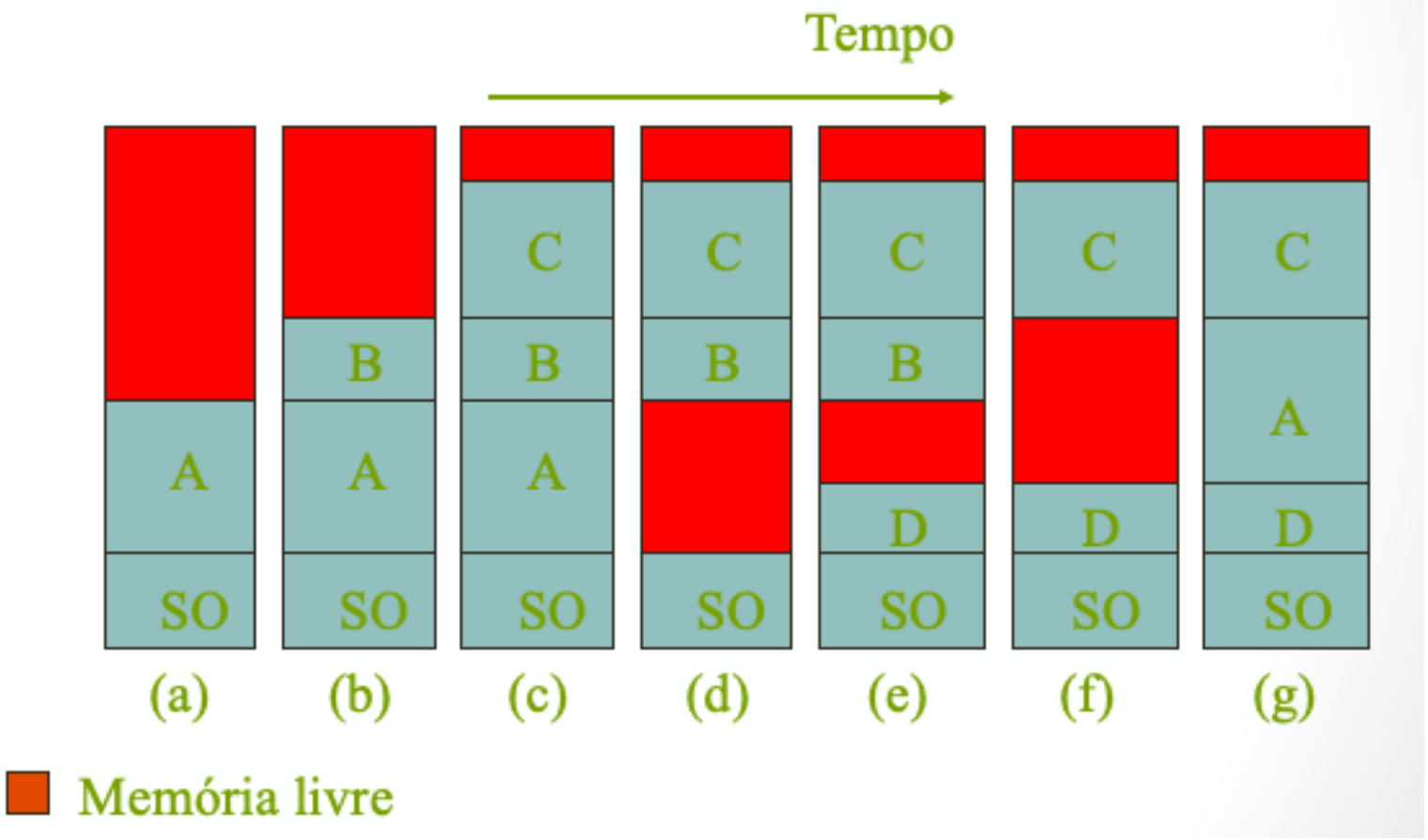
- Partições Fixas: problemas com fragmentação:
  - Interna: desperdício dentro da área alocada para um processo. Ex.: processo de tamanho 40K ocupando uma partição de 50k;
  - Externa: desperdício fora da área alocada para um processo. Duas partições livres: PL1 com 25k e PL2 com 100k, e um processo de tamanho 110K para ser executado. Livre: 125K, mas o processo não pode ser executado.

Gerência de Memória

Partições Variáveis

- Tamanho e número de partição variam.
- Otimiza a utilização da memória, mas complica a alocação e liberação da memória.
- Partições são alocadas dinamicamente.
- SO mantém na memória uma lista com os espaços livres.
- Menor fragmentação interna e grande fragmentação externa. Solução: Compactação.

Partições Variáveis



Gerência de Memória

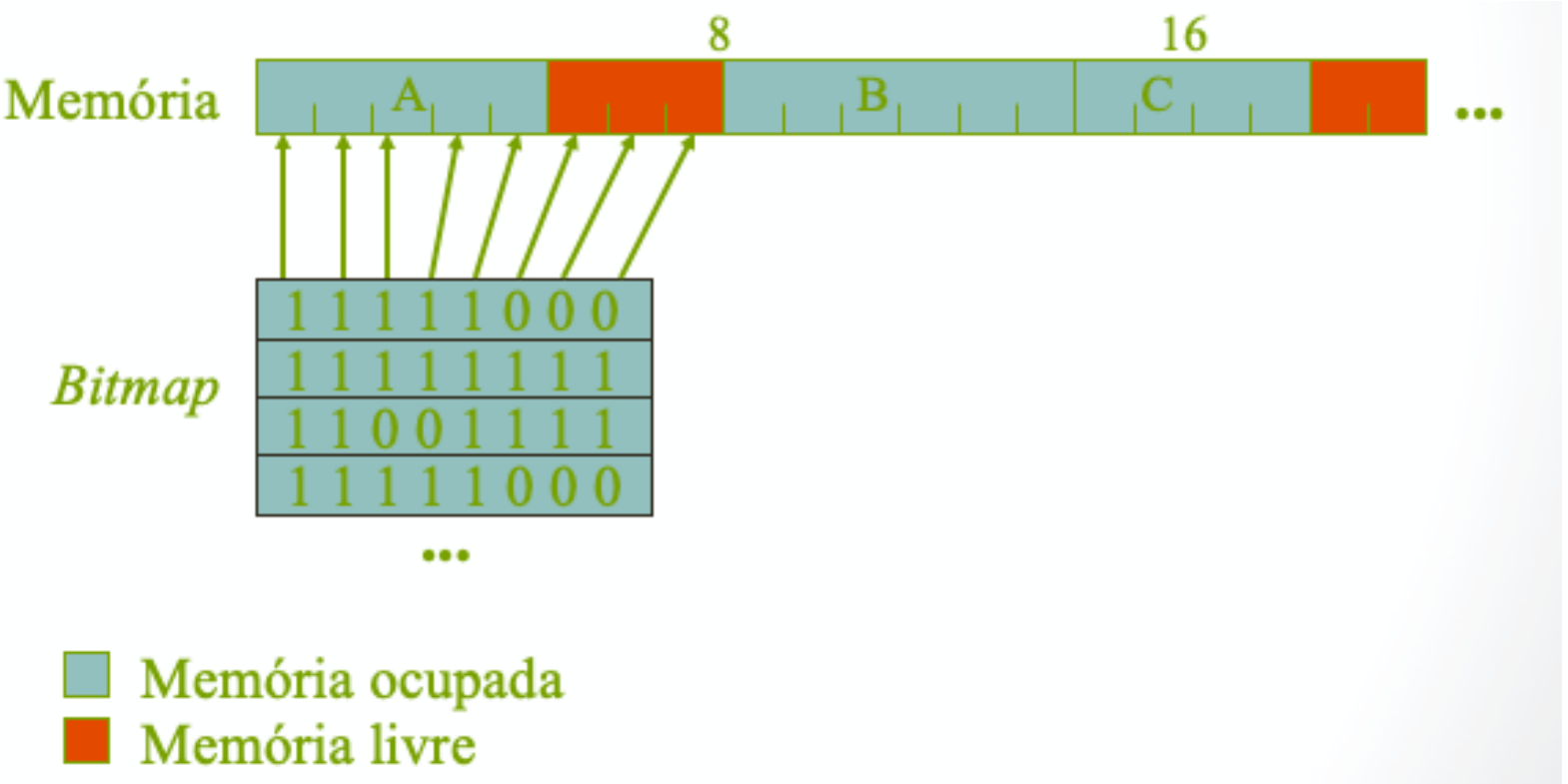
- Minimizar espaço de memória inutilizados:
  - Compactação: necessária para recuperar os espaços perdidos por fragmentação; no entanto, muito custosa para a CPU.
- Técnicas para alocação dinâmica de memória:
  - *Bitmaps*.
  - Listas Encadeadas.

Gerência de Memória

Técnica com *Bitmaps*

- Memória é dividida em unidades de alocação em kbytes;
- Cada unidade corresponde a um bit no *bitmap*:
  - 0 - livre
  - 1 - ocupado
- Tamanho do *bitmap* depende do tamanho da unidade e do tamanho da memória;
- Ex.:
  - unidades de alocação pequenas - *bitmap* grande;
  - unidades de alocação grandes - perda de espaço.

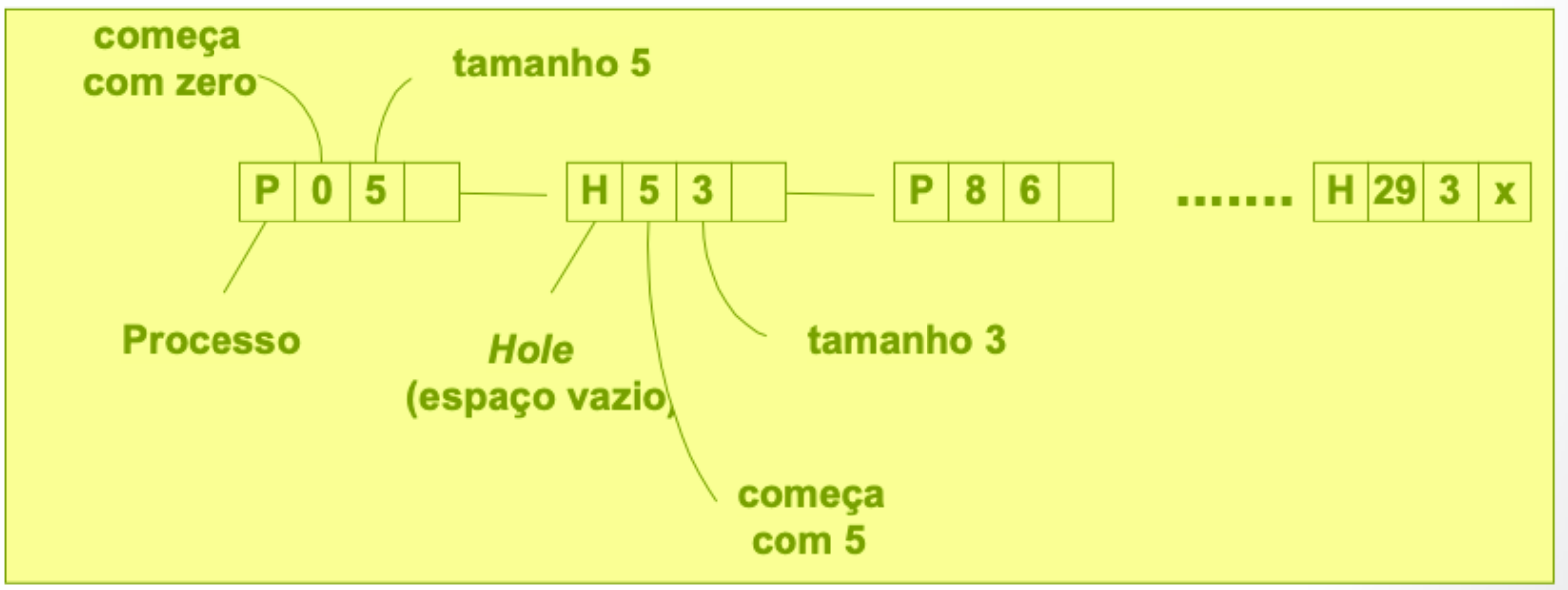
Técnica com *Bitmap*



Gerência de Memória

Técnica com Listas Encadeadas

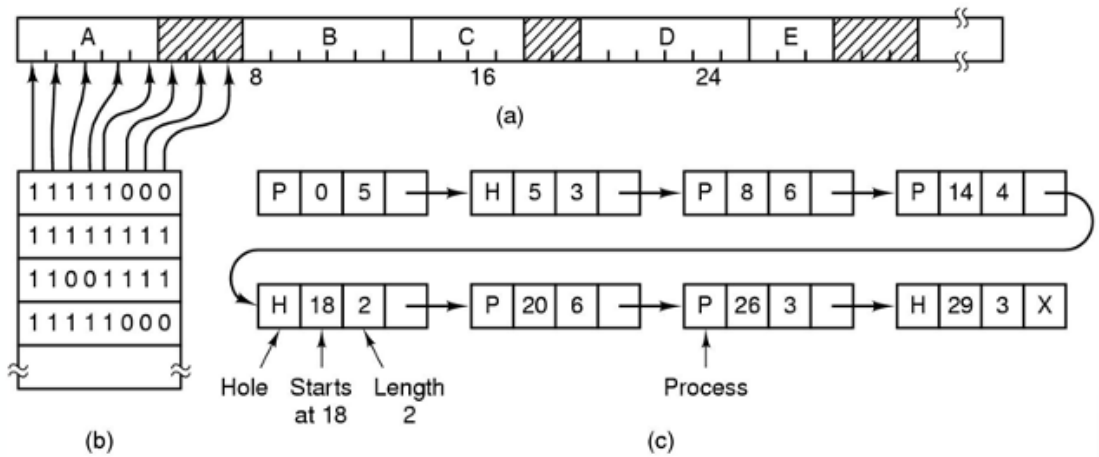
- Uma lista para os espaços vazios e outra para os espaços cheios, ou uma lista para ambos!
- "espaço - semelhante - segmento"



Gerência de Memória

Técnica com Listas Encadeadas

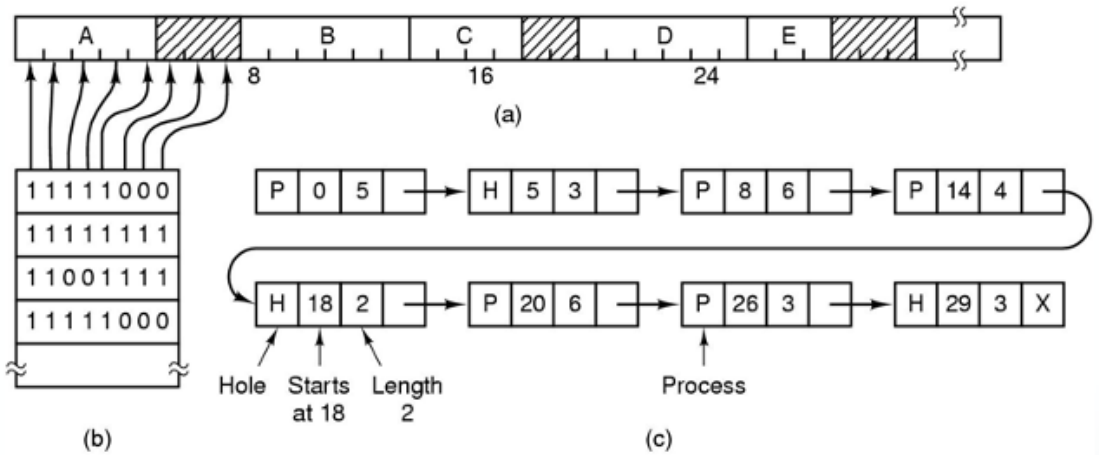
- Outra maneira de gerenciar memória é manter uma lista encadeada de segmentos de memória alocados e segmentos disponíveis.
- Cada elemento desta lista especifica:
  - um segmento disponível (H), ou alocado a um processo (P).
  - o endereço onde se inicia este segmento.
  - e um ponteiro para o próximo elemento.



Gerência de Memória

Técnica com Listas Encadeadas

- (a) Uma parte da memória com 5 processos e 3 buracos
  - As regiões em branco (1 no *bitmap*) marcam as unidades já alocadas.
  - As regiões sombreadas (0 no *bitmap*) marcam unidades desocupadas.
- (b) O *bitmap* correspondente.
- (c) A mesma informação como uma lista encadeada.



Gerência de Memória

- **Algoritmos de Alocação** - quando um novo processo é criado
  - *First Fit*
    - 1o. segmento é usado
    - Rápido, mas pode desperdiçar memória por fragmentação.
  - *Next Fit*
    - 1o. segmento é usado
    - Mas na próxima alocação inicia busca do ponto que parou anteriormente
    - Possui desempenho inferior

Gerência de Memória

- **Algoritmos de Alocação** - quando um novo processo é criado
  - *Best Fit*
    - Procura na lista toda e aloca o espaço que mais convém;
    - Menor fragmentação;
    - Mais lento.
  - *Worst Fit*
    - Aloca o maior espaço disponível.
  - *Quick Fit*
    - Mantém listas separadas para os espaços mais requisitados.

Gerência de Memória

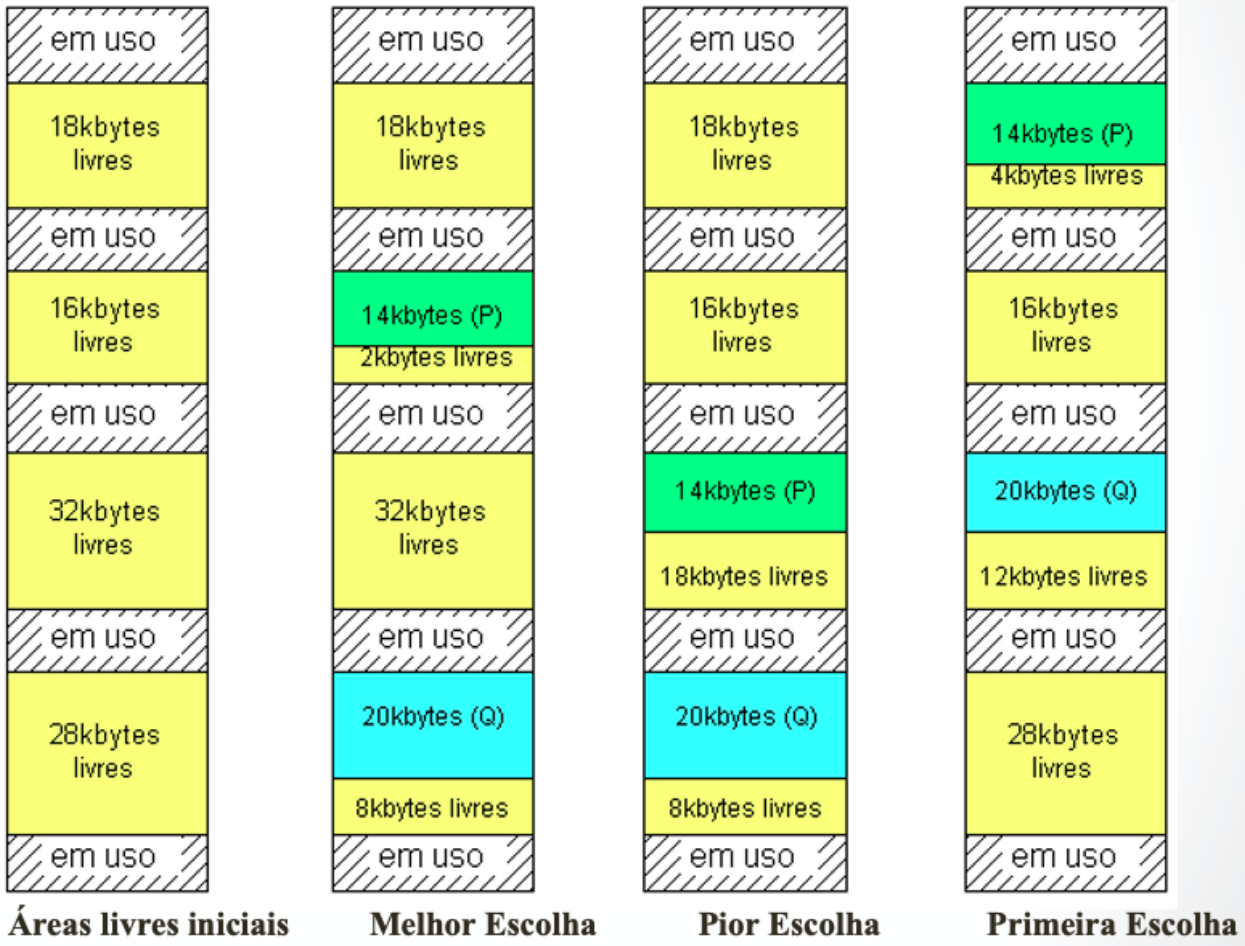
- Cada algoritmo pode manter listas separadas para processos e para espaços livres:
  - Vantagem
    - Aumenta desempenho.
  - Desvantagem
    - Aumenta complexidade quando espaço de memória é liberado – gerenciamento das listas.
    - Fragmentação.

Gerência de Memória

Alocação de segmentos livres

- Principais Consequências
- **A melhor escolha:**deixa o menor resto, porém após um longo processamento poderá deixar "buracos" muito pequenos para serem úteis.
- **A pior escolha:** deixa o maior espaço após cada alocação, mas tende a espalhar as porções não utilizadas sobre áreas não contínuas de memória e, portanto, pode tornar difícil alocar grandes *jobs*.
- **A primeira escolha:** tende a ser um meio termo entre a melhor e a pior escolha, com a característica adicional de fazer com que os espaços vazios migrem para o final da memória.

Alocação de segmentos livres



Gerência de Memória

O que fazer quando não existe espaço suficiente para todos os processos ativos?

- *Swapping*
  - Chaveamento de processos inteiros entre a memória principal e o disco.
- *Overlays* - Memória Virtual
  - Programa de são divididos em pedaços menores.
  - Pedaços são chaveados entre a memória principal e o disco.

Gerência de Memória

Swapping

- Chaveamento de processos inteiros entre a memória principal e o disco.
- Transferência do processo da memória principal para a memória secundária (normalmente o disco): *Swap-out*.
- Transferência do processo da memória secundária para a memória principal: *Swap-in*.
- Pode ser utilizado tanto com partições fixas quanto variáveis.

Gerência de Memória

- Programas maiores que a memória eram divididos em pedaços menores chamados de *overlays* – programador:
  - Desvantagem: custo muito alto.
- Memória Virtual
  - Sistema operacional é responsável por dividir o programa em *overlays*;
  - Sistema operacional realiza o chaveamento desses pedaços entre a memória e o disco.

Gerência de Memória

- Programas maiores que a memória eram divididos em pedaços menores chamados de *overlays*.
  - Programador define áreas de *overlay*;
  - **Vantagem**: expansão da memória principal;
  - **Desvantagem**: custo muito alto.

Gerência de Memória

- Sistema operacional é responsável por dividir o programa em *overlays*;
- Sistema operacional realiza o chaveamento desses pedaços entre a memória principal e o disco;
- Década de 60: ATLAS - primeiro sistema com MV (Universidade Manchester - Reino Unido);
- 1972: sistema comercial: IBM System/370.

Gerência de Memória

- Com MV existe a sensação de se ter mais memória principal do que realmente se tem;
- O hardware muitas vezes implementa funções da gerência de memória virtual:
  - SO deve considerar características da arquitetura.

Gerência de Memória

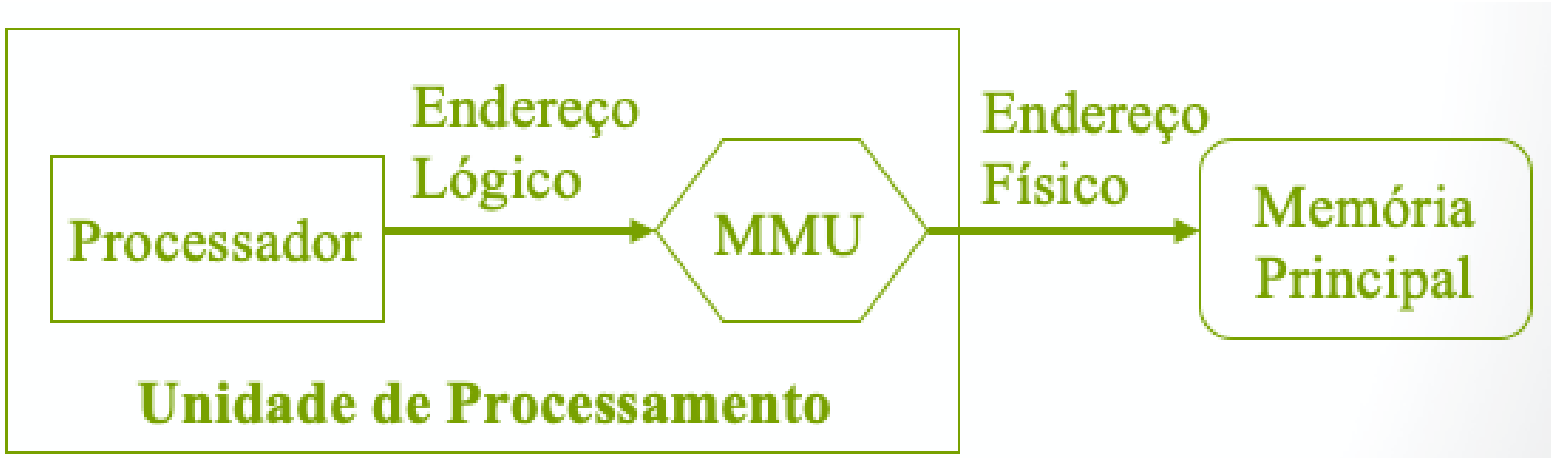
- **Espaço de Endereçamento Virtual** de um processo é formado por todos os endereços virtuais que esse processo pode gerar;
- **Espaço de Endereçamento Físico** de um processo é formado por todos os endereços físicos/reais aceitos pela memória principal (RAM).

Gerência de Memória

- Um processo em Memória Virtual faz referência a endereços virtuais e não a endereço reais de memória RAM;
- No momento da execução de uma instrução, o endereço virtual é traduzido para um endereço real, pois a CPU manipula apenas endereços reais da memória RAM - **MAPEAMENTO**.

Gerência de Memória

- MMU: Realiza **mapeamento** dos endereços lógicos (usados pelos processos) para endereços físicos;



Gerência de Memória

- Endereços virtuais formam um espaço de endereçamento virtual;
- Mapeamento entre endereços reais e virtuais é realizado pela MMU;
  - Técnicas de MV:
  - Paginação;
  - Segmentação.

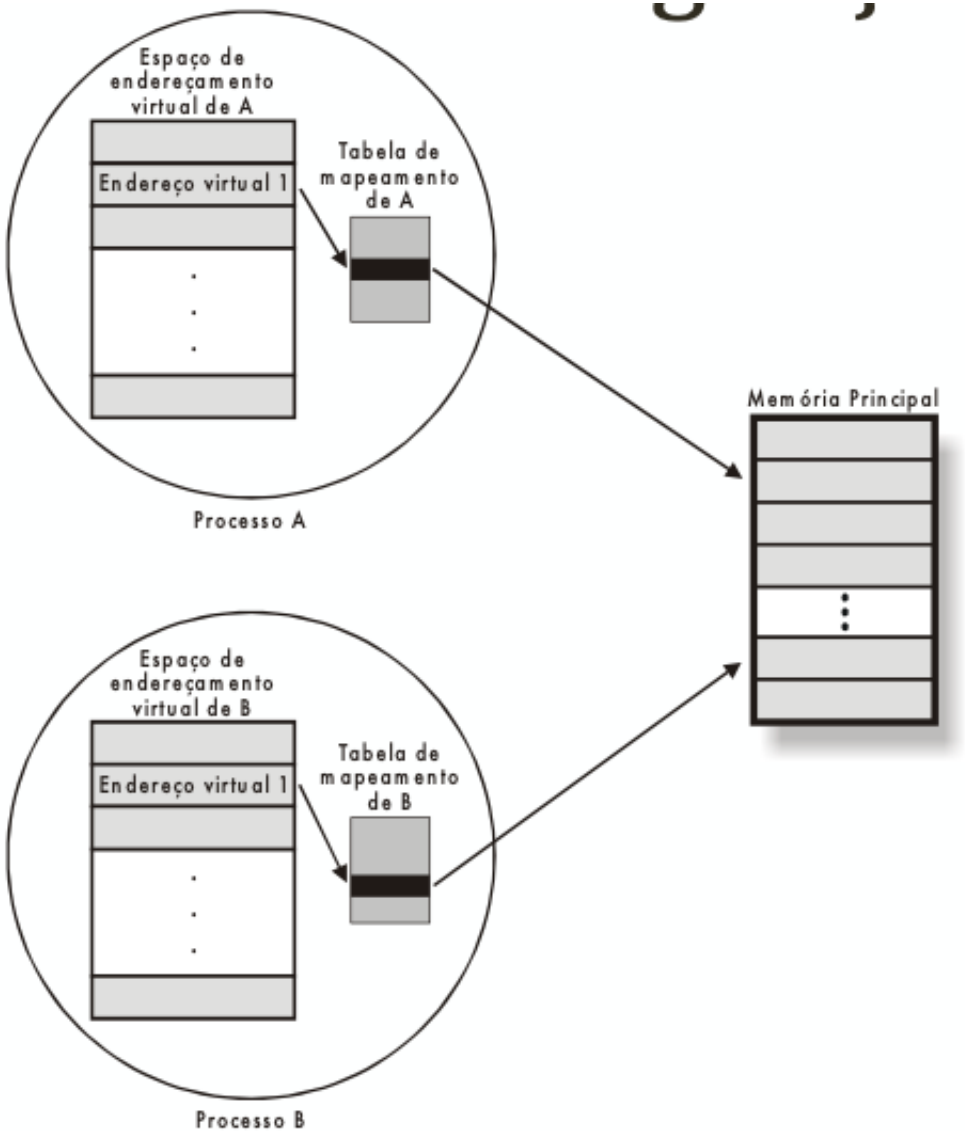
Gerência de Memória

- Paginação
  - Blocos de tamanho fixo chamados de páginas;
  - SO mantém uma lista de todas as **páginas**;
  - O espaço de endereçamento virtual é dividido em páginas virtuais.
- Segmentação
  - Blocos de tamanho arbitrário chamados **segmentos**;
  - SO mantém uma lista de todos os segmentos;
  - O espaço de endereçamento virtual é dividido em segmentos virtuais.

Gerência de Memória

- Memória Principal e Memória Secundária são organizadas em páginas de mesmo tamanho;
- Página é a unidade básica para transferência de informação;
- **Tabela de páginas:** responsável por armazenar informações sobre as páginas virtuais:
  - argumento de entrada - número da página virtual;
  - argumento de saída (resultado) - número da página real (ou moldura de página - *page frame*).

Gerência de Memória



Gerência de Memória

Exemplo:

- Páginas de 4Kb
  - 4096 bytes/endereços (0-4095).
- 64Kb de espaço virtual;
- 32Kb de espaço real;
- Temos:
  - 16 páginas virtuais;
  - 8 páginas reais.

Gerência de Memória

Espaço Virtual X Tamanho da Página

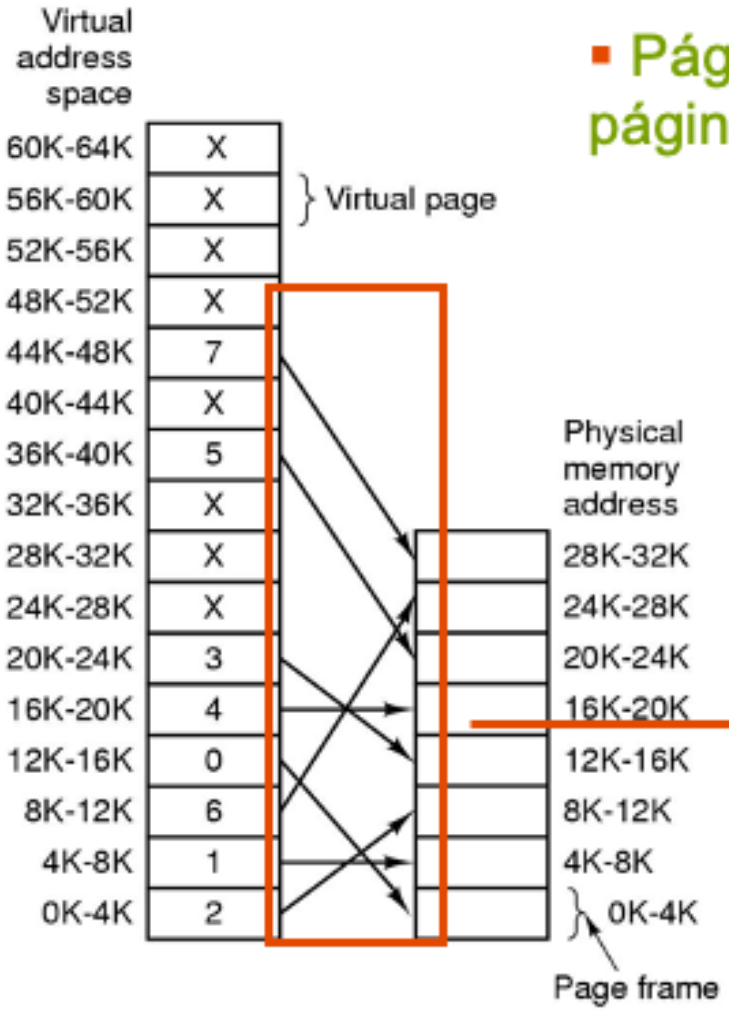
Espaço de Endereçamento Virtual	Tamanho da página	Número de páginas	Número de entradas nas tabela de páginas
2^{32} endereços	512 bytes	2^{23}	2^{23}
2^{32} endereços	4 kbytes	2^{20}	2^{20}
2^{64} endereços	4 kbytes	2^{52}	2^{52}
2^{64} endereços	64 kbytes	2^{48}	2^{48}

Gerência de Memória

Problemas:

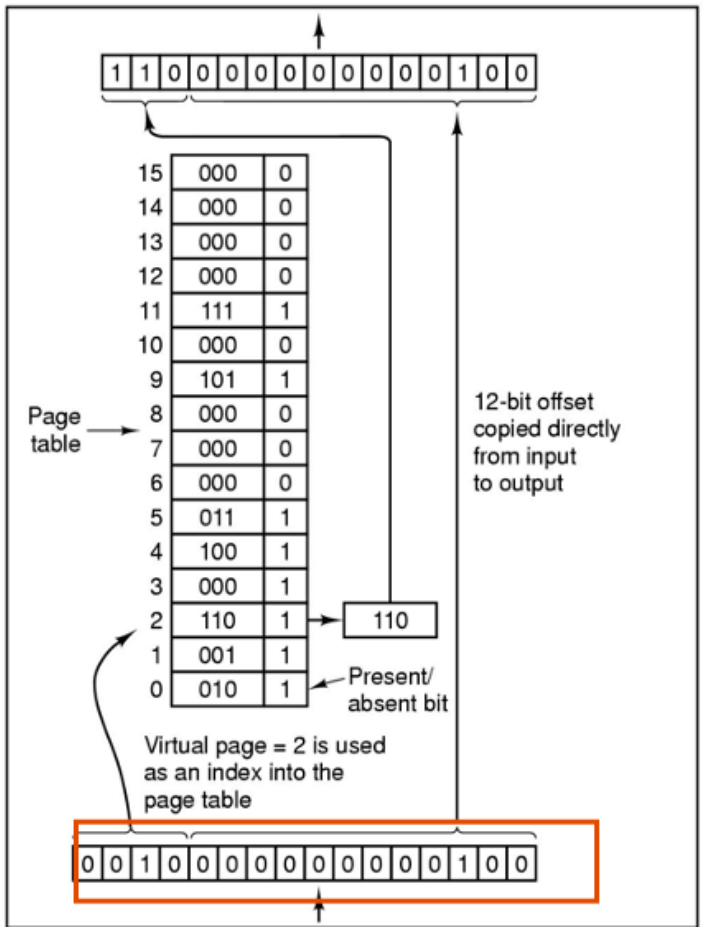
- Fragmentação interna;
- Definição do tamanho das páginas;
  - Geralmente a **MMU** que define e não o SO;
  - Páginas maiores: leitura mais eficiente, tabela menor, mas maior fragmentação interna;
  - Páginas menores: leitura menos eficiente, tabela maior, mas menor fragmentação interna;
  - Sugestão: 1k a 8k;
- Mapa de bits ou uma lista encadeada com as páginas livres.

Endereço Virtual **mapeado para** Endereço Real



- Página virtual **mapeada** para página real;
- MMU realiza o mapeamento

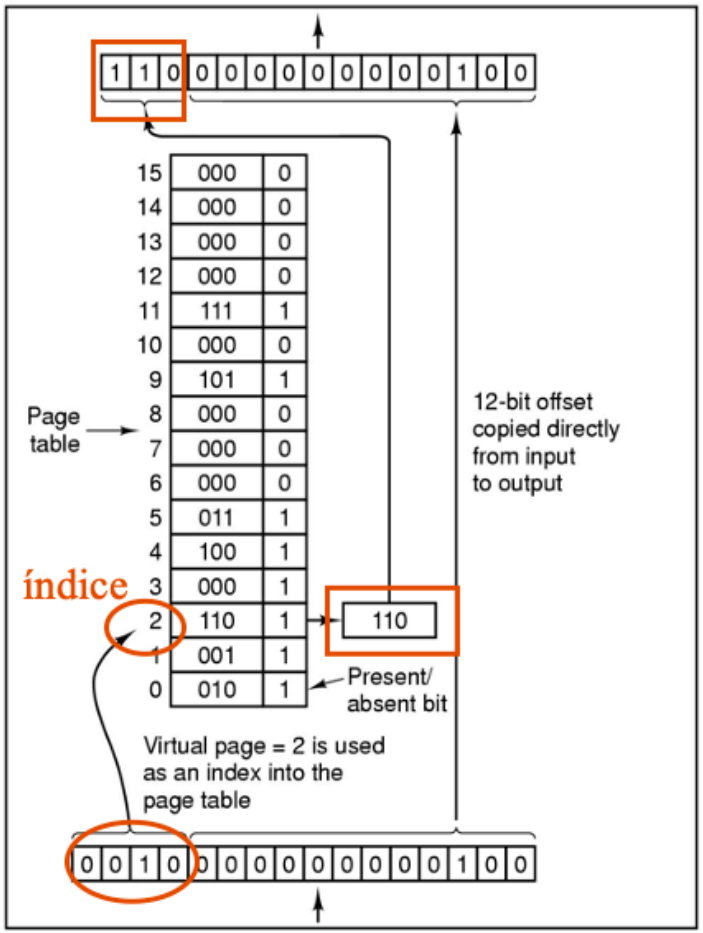
Endereço Virtual **mapeado para** Endereço Real



Mapeamento da MMU

- Operação interna de uma MMU com 16 páginas de 4Kb;
- Endereço virtual de **16 bits**: 4 bits para nº de páginas e 12 bits para deslocamento;
- Com 4 bits é possível ter 16 páginas virtuais (2^4);
- 12 bits para deslocamento é possível endereçar os 4096 bytes.

Endereço Virtual **mapeado para** Endereço Real



Mapeamento da MMU

- Número da página virtual é usado como índice;
- Se página está na memória RAM, então o nº da página real (110) é copiado para os três bits mais significativos do endereço de saída (real), juntamente com o deslocamento sem alteração;
- Endereço real com 15 bits é enviado à memória.

Gerência de Memória

A Tabela de páginas pode ser armazenada de três diferentes maneiras:

- Em conjunto de registradores, se a memória for pequena
  - Vantagem: rápido
  - Desvantagem: precisa carregar toda a tabela nos registradores a cada chaveamento de contexto.
- Na própria memória RAM – MMU gerencia utilizando dois registradores:
  - Registrador Base da tabela de páginas (PTBR – *page table base register*): indica o endereço físico de memória onde a tabela está alocada
  - Registrador Limite da tabela de páginas (PTBR – *page table limit register*): indica o número de entradas da tabela (número de páginas)
  - Precisa de dois acessos à memória: um para acessar a tabela de páginas e outro para acessar a posição de memória.

Gerência de Memória

A Tabela de páginas pode ser armazenada de três diferentes maneiras:

- Em memória cache da MMU
  - Também conhecida como TLB (*Translation Lookside Buffer* – *buffer* de tradução dinâmica);
  - *Hardware* especial para mapear endereços virtuais para endereços reais sem ter que passar pela tabela de páginas na memória principal;
  - Melhora o desempenho.

Gerência de Memória

- Algumas questões que surgem com relação à Paginação:
  - Onde armazenar a tabela de páginas?

Gerência de Memória

Projeto mais simples

- Uma única tabela de páginas que consista em um vetor de registradores rápidos em hardware (um registrador para cada entrada);
- Quando o processo estiver para ser executado, o SO carregará esses registradores a partir de uma cópia da tabela de páginas desse processo mantida na memória;
- Vantagem: Não requer nenhum acesso à memória durante a tradução
- Desvantagens:
  - Caro!
  - Ter que carregar toda a tabela de páginas em cada troca de contexto.

Gerência de Memória

- Tabela de páginas totalmente na memória.
- O Hardware necessário resume-se a um único registrador (que aponta para o início da tabela de páginas).
- Desvantagem:
  - A execução de uma instrução implicará em pelo menos dois acessos à memória
    - O primeiro para acessar a tabela de páginas (e descobrir o endereço físico desta instrução).
    - O segundo para buscar a respectiva instrução na memória. Isso sem falar nos operandos da instrução que podem estar em memória.

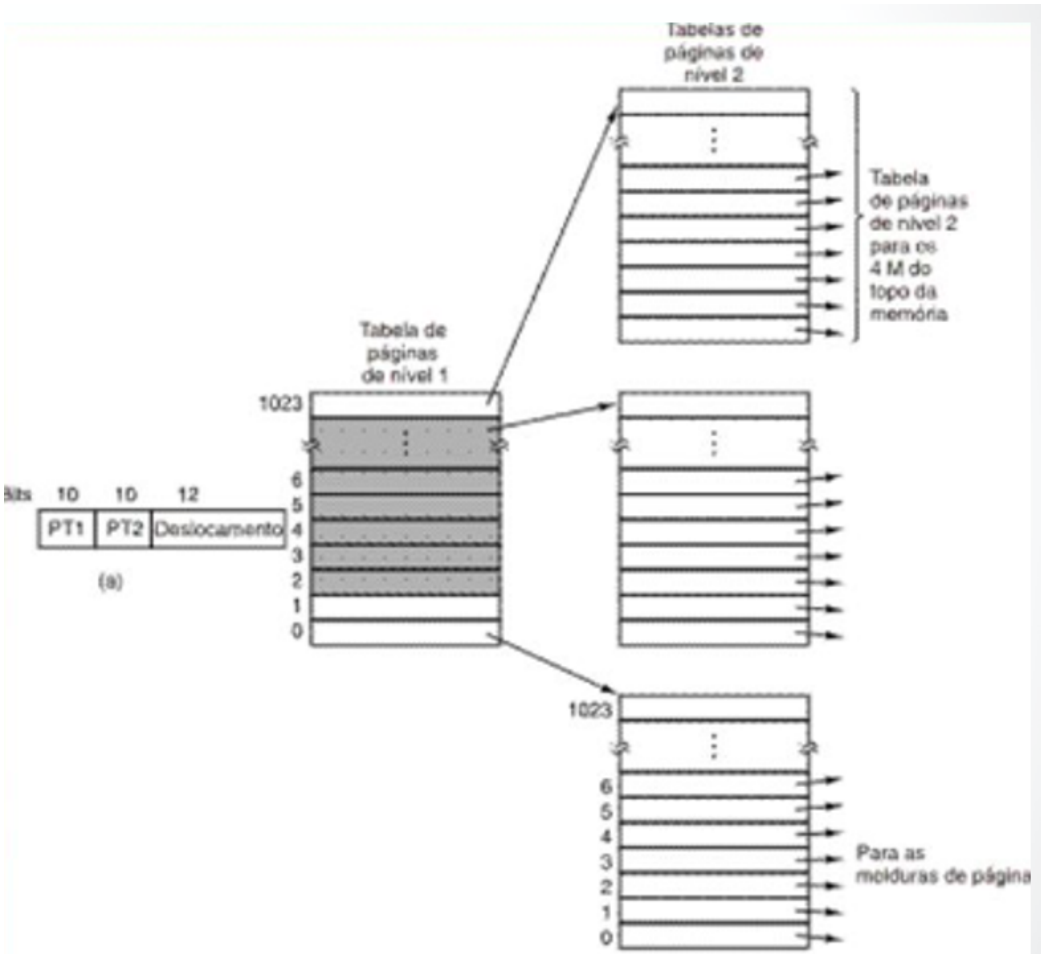
Gerência de Memória

Tabela de Página Multinível

- O objetivo é evitar manter toda a tabela de páginas na memória durante todo o tempo
- Apresenta-se como uma solução para o dimensionamento da tabela de páginas
- Uso de dois apontadores e um deslocamento
- Exemplo: Tabela de dois níveis
  - O endereço de 32 bits de endereço divido em 3 campos
  - PT1 [10 bits] : indexa o primeiro nível da tabela
  - PT2 [10 bits] : indexa o segundo nível da tabela
  - Deslocamento [12 bits]: - páginas de 4 KB

Gerência de Memória

- 1o. nível com 1024 entradas
- Cada uma dessas entradas representa 4 MB
  - 4 GB / 1024



Gerência de Memória

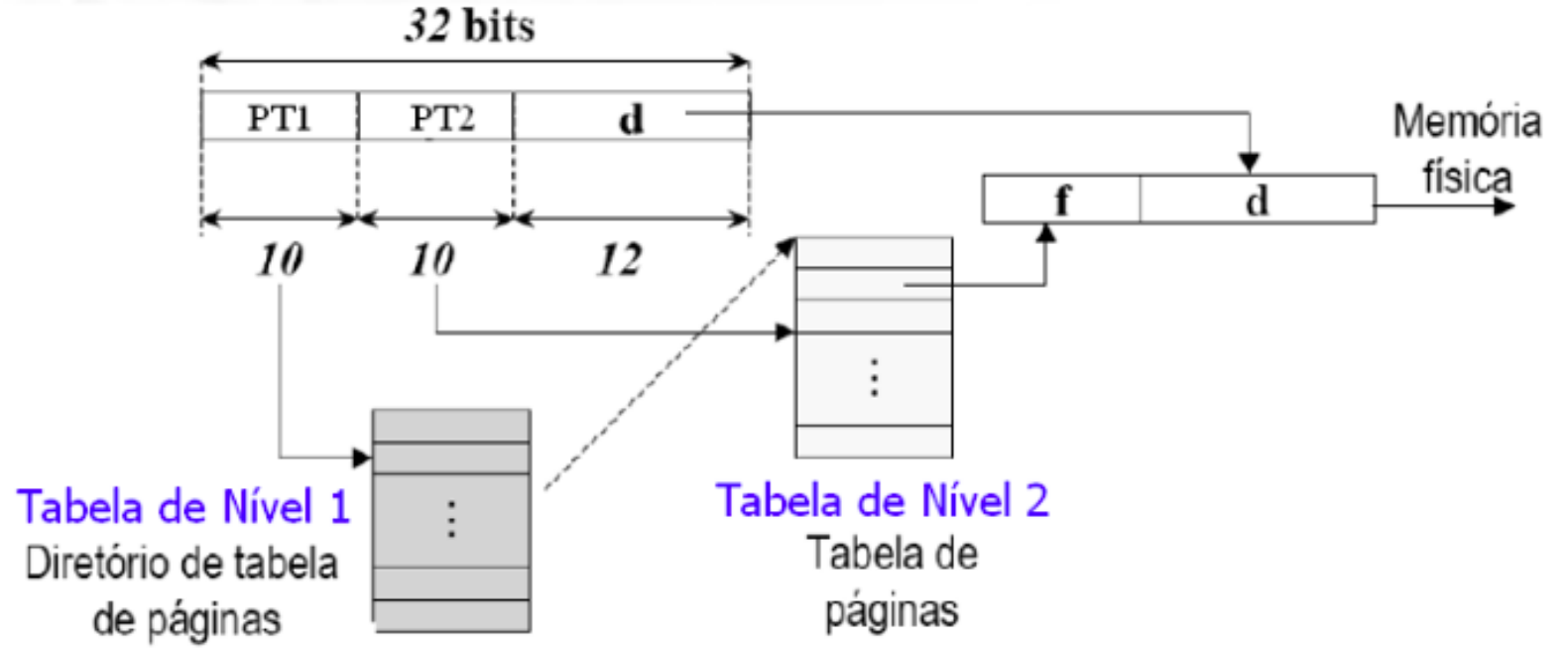
- No exemplo anterior:
- Suponha que um processo utilize apenas 12 MB do seu espaço de endereços virtuais
  - 4MB da base da memória para código de programa
  - Outros 4 MB para dados
  - 4 MB do topo da memória para pilha
- Portanto:
  - A entrada 0 da tabela de nível 1 aponta para a tabela de páginas de nível 2 relativa ao código do programa
  - A entrada 1 da tabela de nível 1 aponta para a tabela de páginas de nível 2 relativa aos dados do processo
  - A entrada 1023 da tabela de nível 1 aponta para a tabela de páginas de nível 2 relativa à pilha do processo

Gerência de Memória

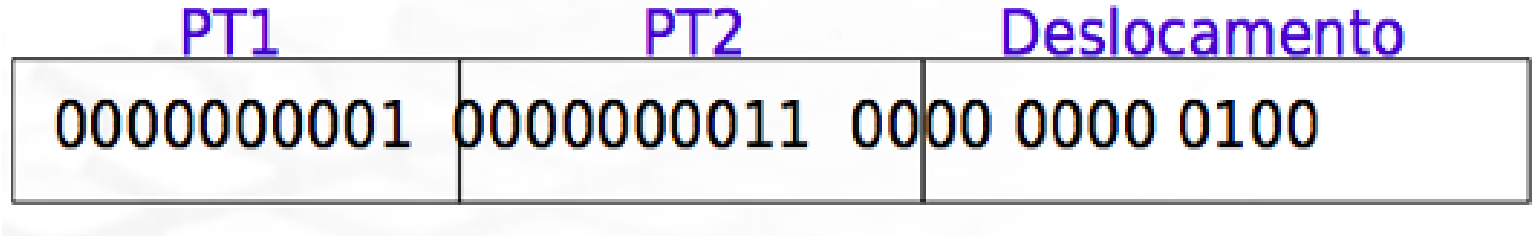
- Quando um endereço virtual chega à **MMU**, ela primeiro extrai o campo PT1 e o utiliza como índice da tabela de páginas do nível 1
- A entrada da tabela de páginas de nível 1 aponta para a tabela de páginas do nível 2.
- Então PT2 é usado como índice nesta segunda tabela para localizar a entrada correspondente à pagina virtual
- Esta entrada indicará em qual moldura física encontra-se o endereço a ser acessado
- No exemplo:
  - Suponha que um processo utilize apenas 12 MB do seu espaço de endereços virtuais
  - A entrada 0 da tabela de nível 1 aponta para a tabela de páginas de nível 2 relativa ao código do programa

Gerência de Memória

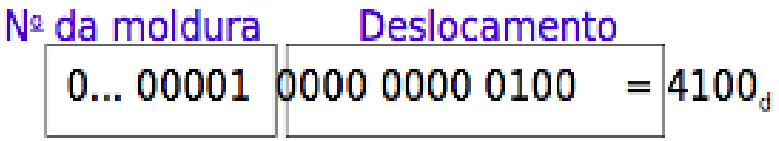
- Considere o endereço virtual 0x00403004 (4206596)
  - Qual será o endereço físico correspondente?



Gerência de Memória



- PT1: Entrada 1 da tabela do 1o nível
  - 2o bloco de 4M (4M a 8M de memória virtual)
- PT2: Entrada 3 da tabela do 2o nível
  - Esta entrada indica em qual moldura encontra-se esta página
  - O endereço físico do primeiro byte dessa moldura é somado ao deslocamento
    - Supondo a página encontre-se na moldura 1 (4k a 8k-1), o endereço físico correspondente será $4096 + 4 = 4100$
  - OU

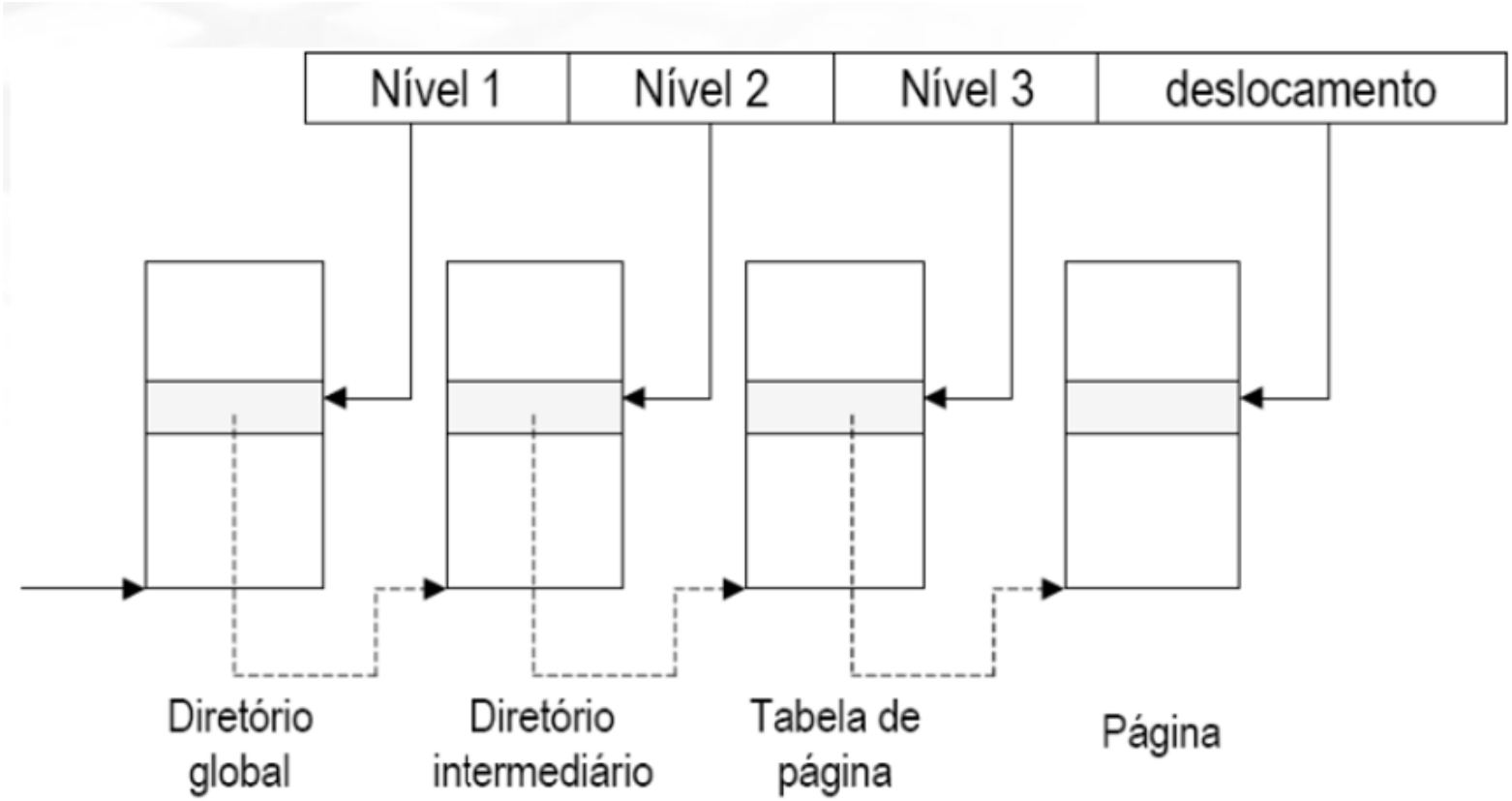


Gerência de Memória

- Para entender as vantagens, considere o exemplo anterior (endereço virtual de 32 bits – página de 4kB)
- Usando tabela de páginas tradicional:
  - 1 tabela de 2^{20} entradas (1 M entradas)
- Usando tabela de páginas em 2 níveis
  - 4 tabelas de 2^{10} entradas cada (1 K entradas)
- Se cada entrada da tabela de páginas ocupa 16 bits
  - primeiro caso: 2^{20} x $2^4 = 16$ Mbits p/ armazenar a tabela de páginas
  - segundo caso: 4 x 2^{10} x 24 $=$ 64 Kbits p/ armazenar a tabela de 2 níveis

Paginação de três níveis

- Típicos de arquiteturas de processadores de 64 bits



Gerência de Memória

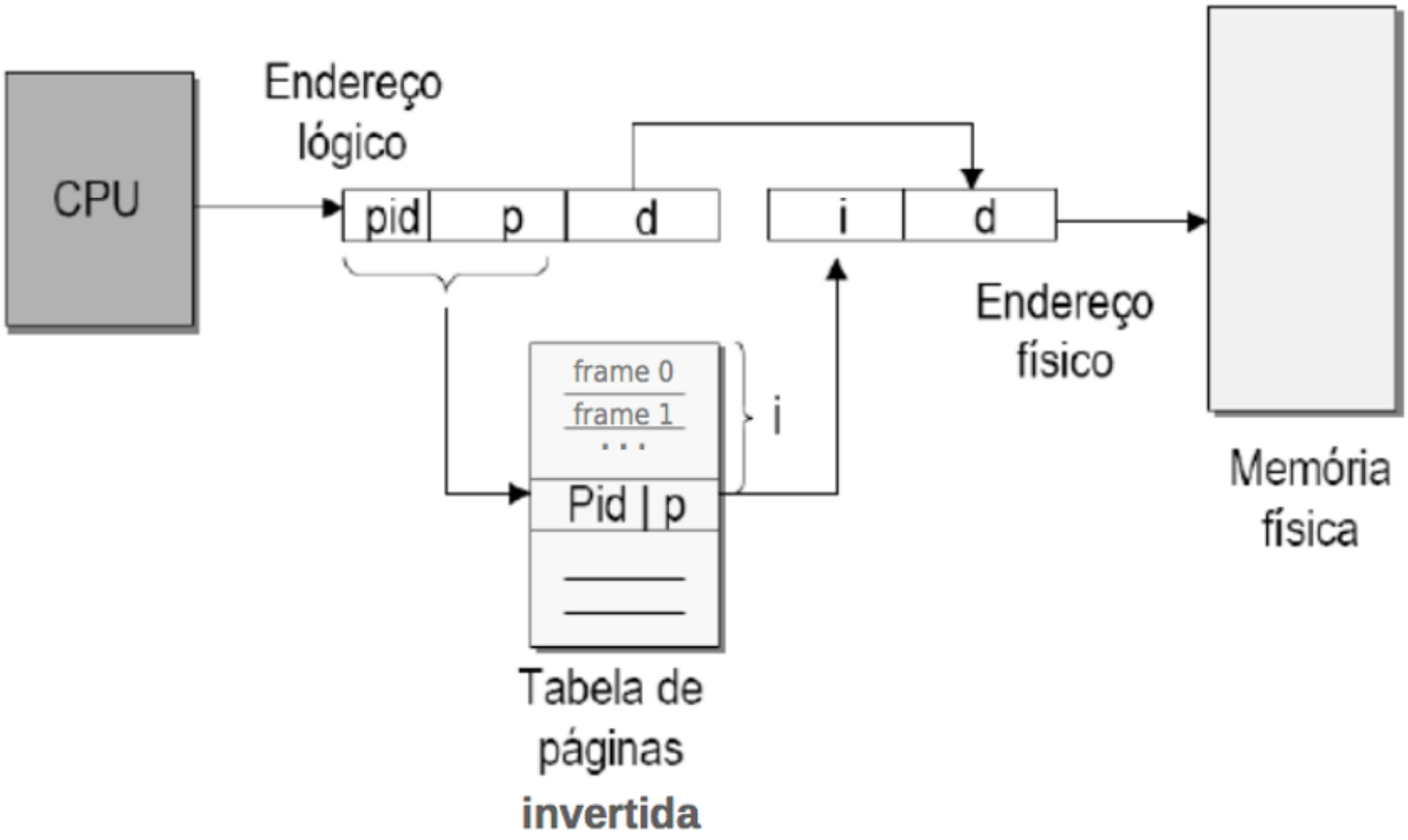
- Espaço de endereçamento virtual pode ser exageradamente grande em máquinas de **64 bits**.
  - Páginas de 4KB
  - 2^{52} entradas na tabela - Se cada entrada ocupa 8 B - tabela de aprox. 30.000.000 GB
- O armazenamento da tabela torna-se viável se a mesma for invertida, isto é, ter o tamanho da quantidade de molduras (memória real) e não da quantidade de páginas (memória virtual)
- Se memória real é de 256 Mbytes, e páginas de 4 KB:
  - Tem-se 65536 entradas

Gerência de Memória

Tabela de Páginas Invertidas

- Uma entrada por moldura de memória real
- Cada entrada na tabela informa
  - Par: (PID, # página virtual) alocado naquela moldura
- Entretanto
  - Tradução de virtual/físico mais complicada
  - Quando o processo n endereça a página p
    - p não serve de índice da tabela
    - Toda a tabela deve ser pesquisada em busca de uma entrada (p,n)
- Solução muito lenta
  - A busca é feita para toda referência à memória

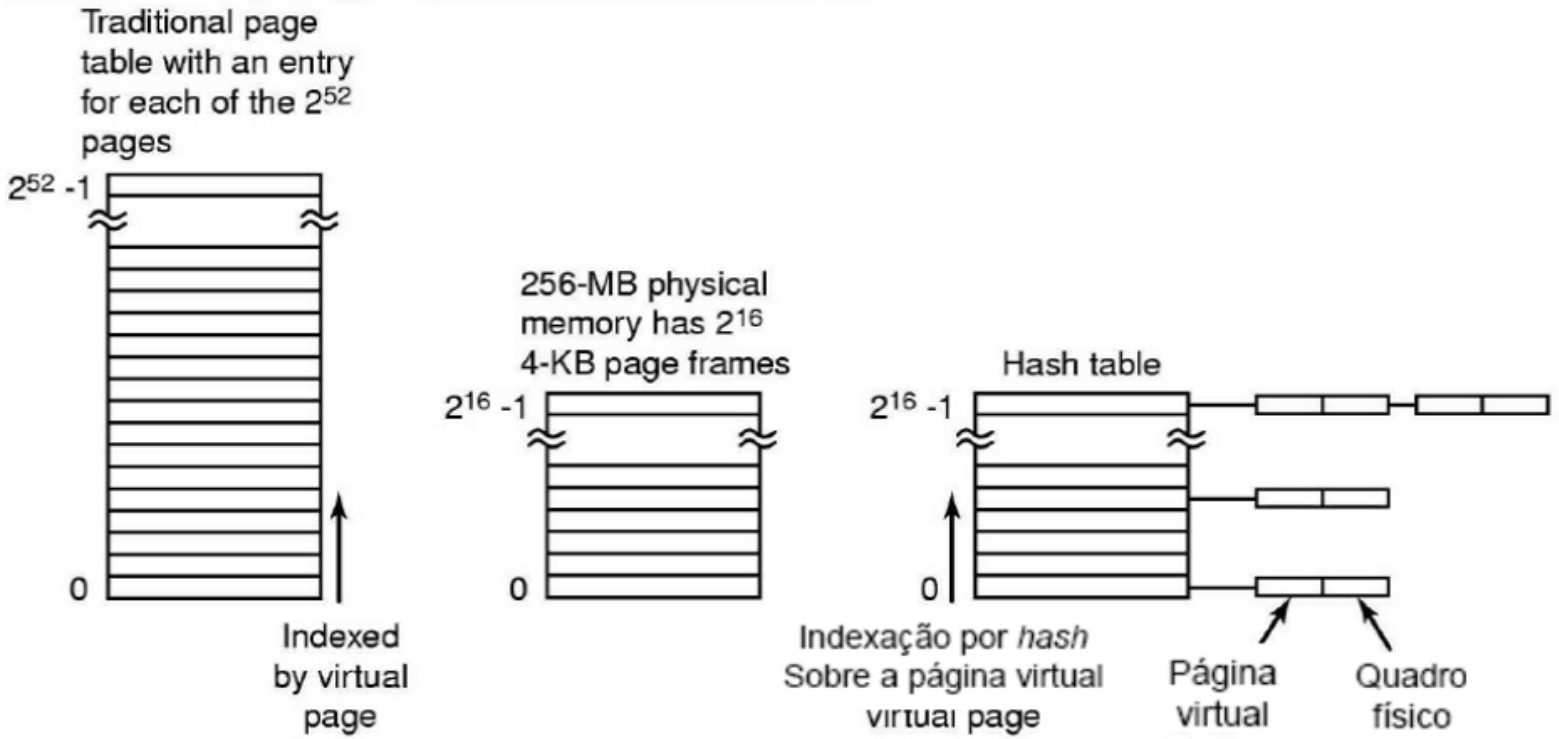
Gerência de Memória



Gerência de Memória

Aceleração pode ser obtida

- TLB para páginas mais referenciadas
- Indexar a tabela por *hash*
  - Uma função *hash* que recebe o número da página e retorna um entre N valores possíveis, onde N é a quantidade de molduras (memória instalada).
  - Páginas com mesmo *hash* serão encadeadas em uma lista
  - Cada entrada da tabela armazena um par (página/quadro)



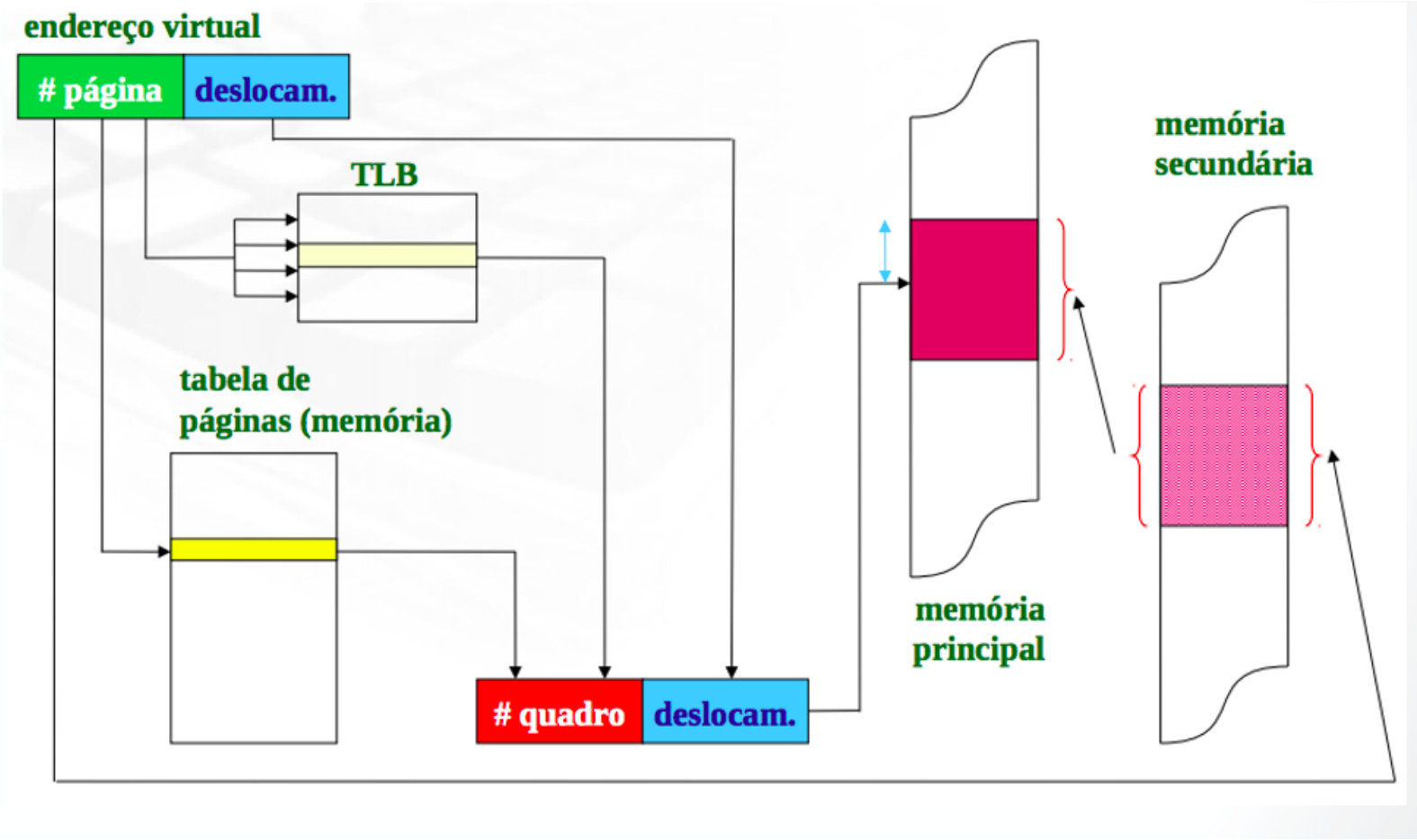
Comparação de uma *page table* tradicional com uma *page table invertida*

Gerência de Memória

TLB – *Translation Lookaside Buffer*

- Como diminuir o número de referências à MP introduzido pelo mecanismo de paginação?
- Os programas tendem a fazer um grande número de referências a um mesmo pequeno conjunto de páginas virtuais
  - Princípio da localidade temporal e espacial
- Solução: equipar a MMU com uma TLB
  - Também chamada de Memória Associativa
  - Dispositivo de hardware implementado com um reduzido número de entradas
- Contém algumas entradas (linhas) da tabela de páginas do processo em execução

Gerência de Memória



Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

Exemplo de TLB

- Loop acessando pag. 19, 20, 21
- Dados principais: pag. 129, 130, 141
- Pilha: 860, 861

Memória associativa (TLB) - De 64 a 4096 entradas

HIT Ratio (Taxa de Sucesso)

- Razão de referências à memória que podem ser satisfeitas a partir da TLB
- ↑Hit Ratio => ↑ performance
- Tempo de acesso com HIT (sucesso) à memória via TLB

$$T_{Hit} = T_{TLB} + T_{MEM}$$

- Tempo de acesso com MISS (insucesso) à memória via TLB

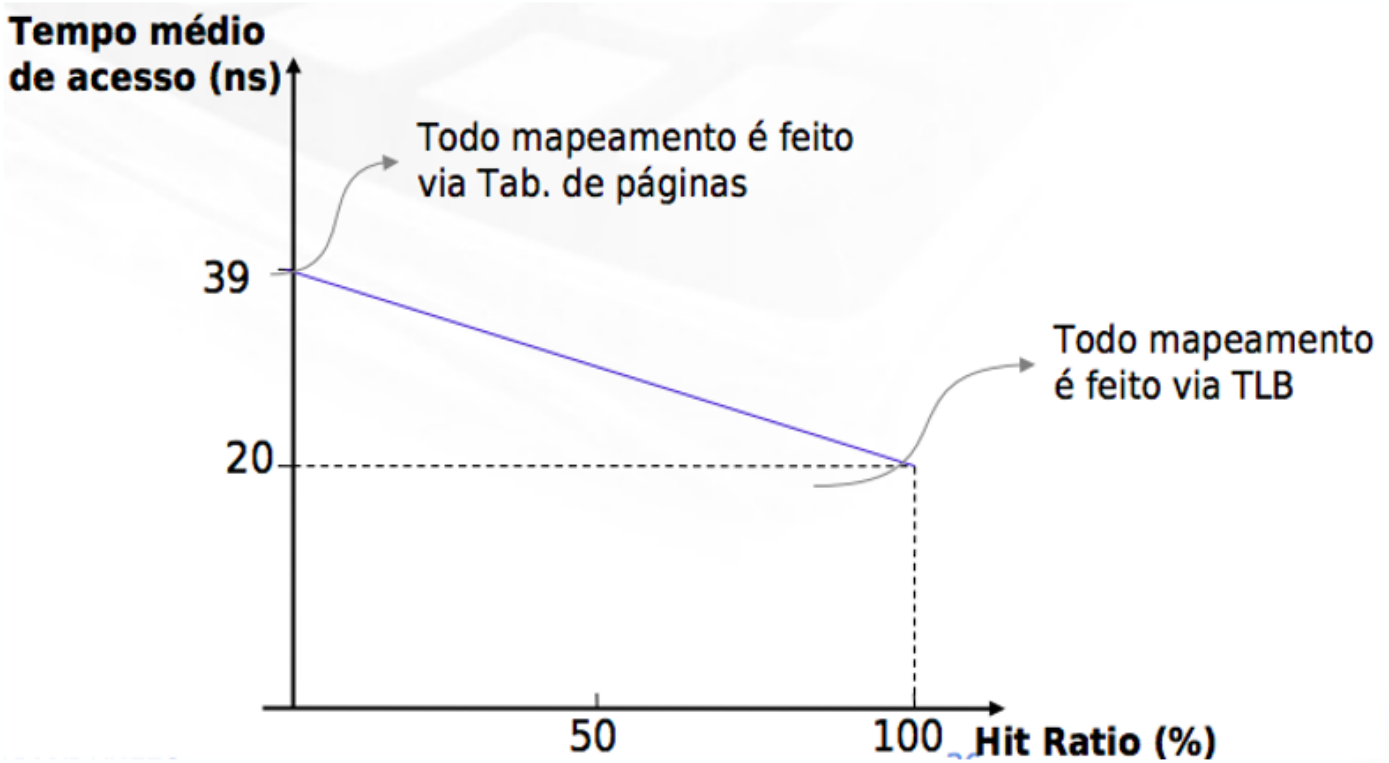
$$T_{Miss} = T_{TLB} + T_{MEM} + T_{MEM}$$

- Tempo médio de acesso= $hr \cdot T_{Hit} + (1\text{-}hr) \cdot T_{miss}$
  - hr é o Hit Ratio

HIT Ratio (Taxa de Sucesso)

- Suponha: T_{Hit} = 20ns ; T_{Miss}= 39 ns; H.R. = 90%

Tempo médio de acesso = 0,9x20+0,1x39 = 21,9ns



Continuemos com **GERENCIA DE MEMÓRIA** ....