PCS 3446 - Sistemas Operacionais

Prof. João José Neto

AULA 05

Administração de Memória (4)

Memória Virtual Paginada

Motivação

- Nos esquemas de alocação já vistos, de gerenciamento de memória física, o job deveria **caber inteiramente na memória** para poder ser executado.
- Vários motivos, porém, levam a perceber que a presença física na memória de todo o código de um programa não é condição necessária para que um job possa ser **executado.**
- **Mesmo nas máquinas antigas, como já foi dito, a técnica do *overlay* já viabilizava, ainda que não automaticamente, a execução de programas maiores que a memória física, sem que todo o seu espaço lógico de endereçamento estivesse fisicamente alocado** na memória principal.

- As principais razões dessa possibilidade são:
  - Alguns **trechos do código** só são **executados raramente**, como é o caso de rotinas de erro, em diversos programas.
  - A lógica do programa exibe **porções do programa cujas atividades são mutuamente exclusivas durante a execução**, por nunca serem utilizadas ao mesmo tempo.
  - Numa região do programa, uma **área de dados raramente é utilizada em toda a sua extensão**, sendo frequente o uso de apenas (pequenos) trechos da mesma, por vez.
  - Alguns **trechos do programa são usados só uma vez**, e nunca mais são solicitados, como aqueles responsáveis pela atribuição de valores iniciais às variáveis.

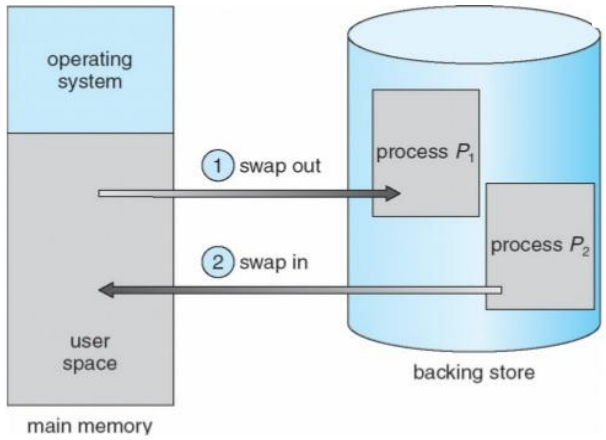
Vantagens da virtualização de memória

- A principal é que **não restringe o espaço de endereçamento do job** ao tamanho da memória física.
- Com isso, dá mais **compatibilidade** a computadores que, embora apresentem conjuntos idênticos de instruções, têm memórias de dimensões diferentes.
- Assim, um job que é normalmente executado em máquinas com um dado tamanho de memória **pode ser também executado**, se necessário (geralmente com redução de desempenho), em outras cuja configuração de **memória** possa ser **bem menor**.

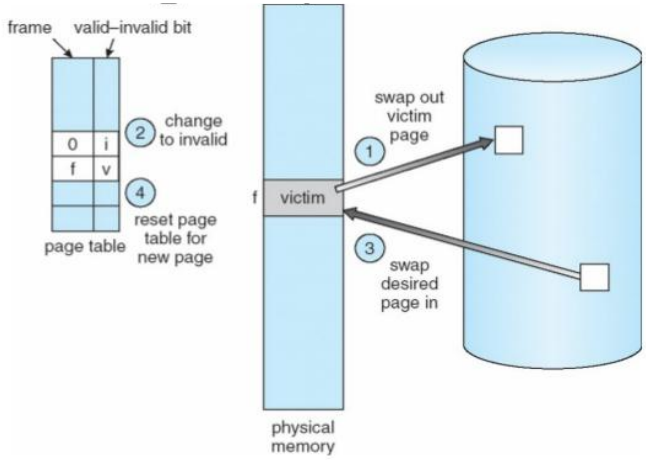
Swapping

- ***Page swapping***: é o ato de substituir páginas dinamicamente entre a memória principal e a memória secundária. **Algoritmos** utilizados para esta finalidade em geral são **complexos** e costumam apresentar **eficácia questionável**.

Swapping de processos

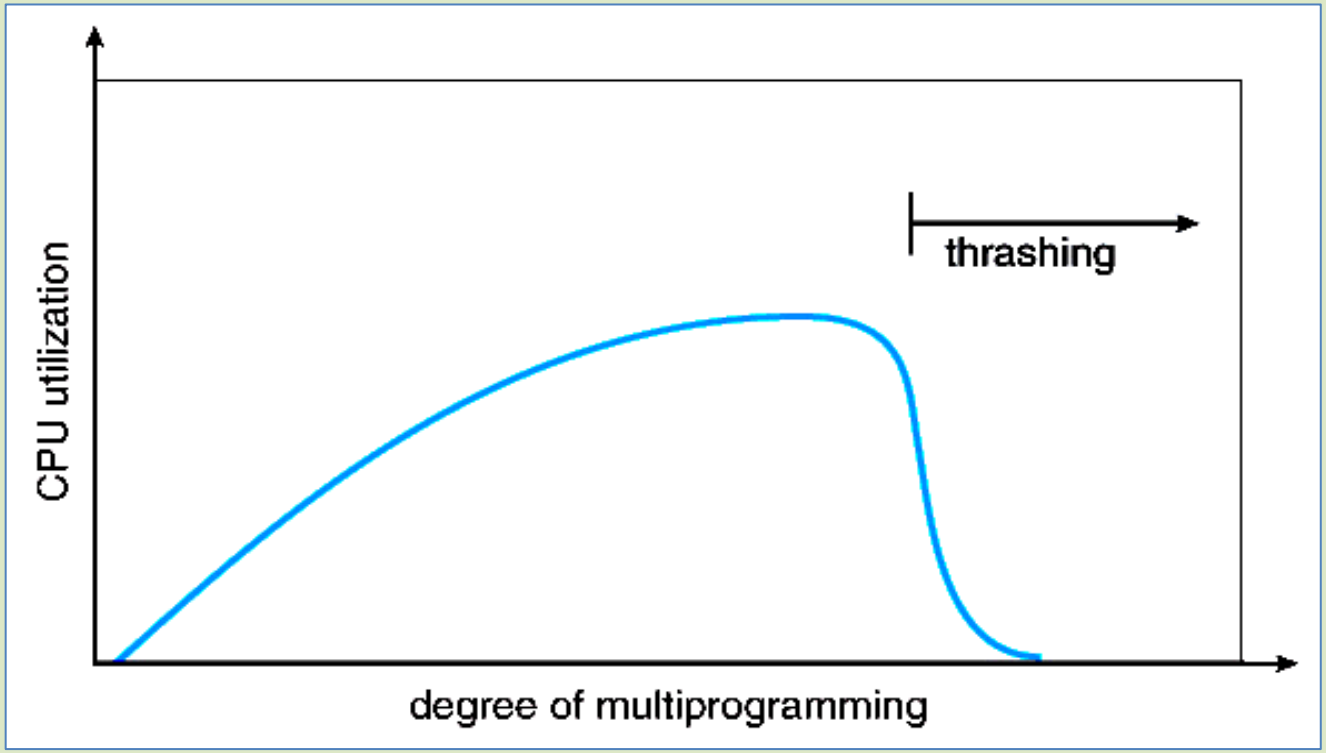


Swapping de páginas



O fenômeno do *Thrashing*

- ***Thrashing***: é um *overhead* (sobrecarga) do sistema, devido ao excesso de leituras e gravações, causadas pelo **vaivém frequente de páginas entre memória e disco,** sem avançar significativamente na execução do programa.



Motivação inicial da Virtualização

- **A virtualização de memória** tem como uma das motivações os ganhos trazidos pela multiprogramação ao uso compartilhado dos recursos da máquina, especialmente os da memória principal.

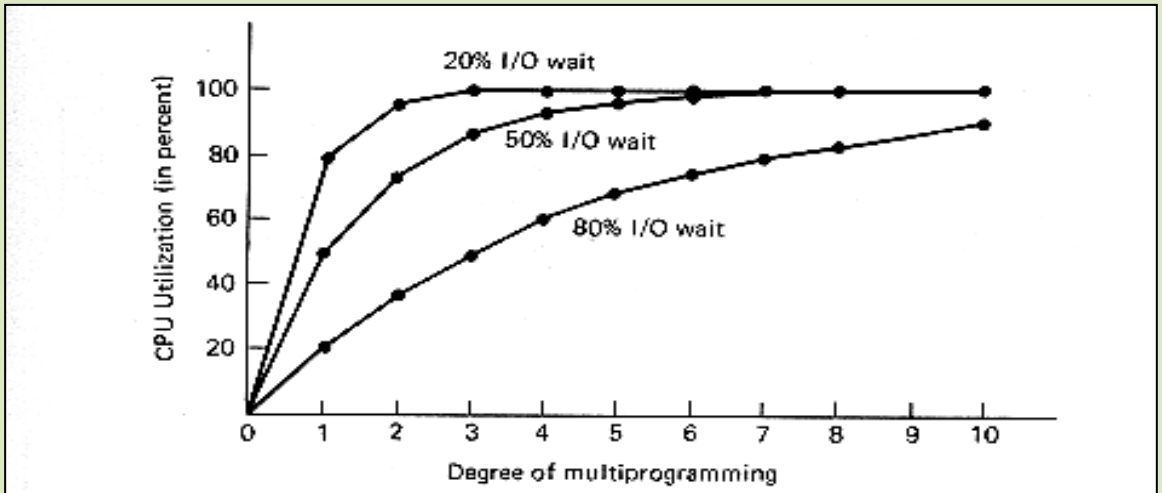
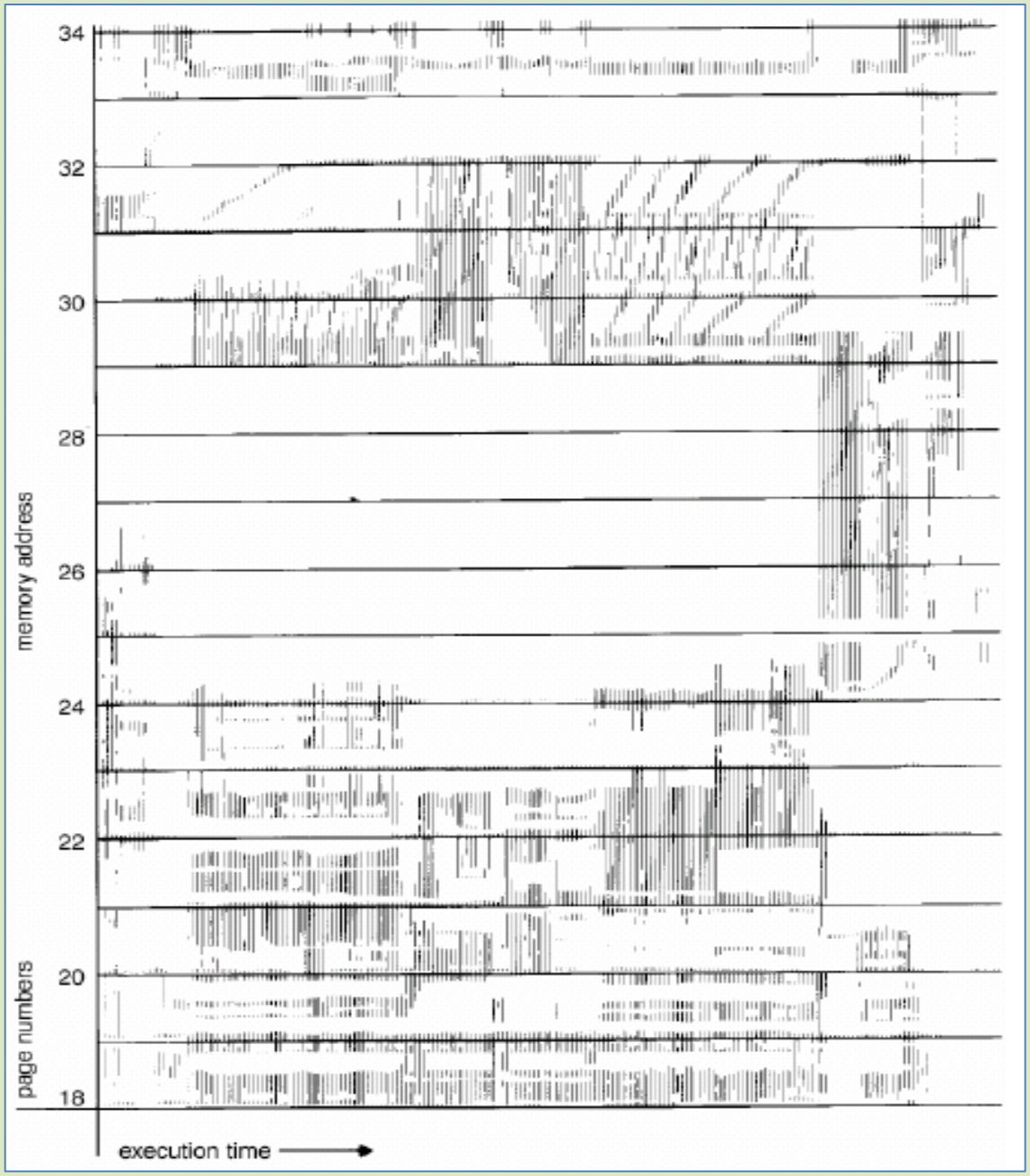


Fig. 3-2. CPU utilization as a function of the number of processes in memory.

Princípio da Localidade

- Observando-se um grande número de programas e seu comportamento durante a execução, é possível identificar um padrão muito interessante, que pode ser usado, em um sistema, como justificativa para a adoção de métodos de virtualização de memória.
- É o chamado **Princípio da Localidade**, fenômeno que se observa em quase todos os tipos de programas, e que permite que tais programas sejam executados mesmo sem que estejam presentes integralmente na memória do computador.

- Segundo o **princípio da localidade**, para cada programa ocorre, a intervalos irregulares, a seguinte cadeia de fenômenos durante o seu processamento:
  - Utilização intensa de um ***working set*** (subconjunto limitado e estável do espaço lógico de endereçamento), por um período significativo de tempo.
  - Acessos esparsos, durante esse período, a endereços de memória externos a esse *working set*.
  - Passagem da situação acima para outra, do mesmo tipo mas envolvendo outro *working set*, após um período relativamente curto de transição.



O Princípio da Localidade e o conceito de *Working Set*

Nesta figura clássica está representado o **rastro da execução de um programa** típico, exemplificando o **princípio da localidade**. Notar que em cada intervalo de tempo de execução, um conjunto relativamente bem definido de endereços de memória (***working set***) é referenciado intensamente, e esse conjunto **varia bruscamente** entre dois intervalos adjacentes. Nesses intervalos, os demais endereços de memória praticamente não são referenciados ou o são esporadicamente.

Virtualização de Memória

- A virtualização dá suporte para o programa fazer acessos ao seu espaço lógico de endereçamento, de alguma forma que independa do local físico de memória no qual essas informações tenham sido alocadas.
- O tamanho total do programa pode ser inclusive **maior que o da memória física**, sem prejuízo à sua execução.
- Em máquinas com vias de endereçamento de **n** bits, em geral são utilizados espaços de $\mathbf{2^n}$ endereços virtuais
- Tipicamente pode-se promover a virtualização de memória:
  - Dividindo-se o programa em partes lógicas de diferentes tamanhos, e a memória física, nas correspondentes partições (técnica de **segmentação**)
  - Dividindo-se tanto o programa quanto a memória física em fatias, todas do mesmo tamanho (técnica de **paginação**)

Virtualização usando Paginação

- A **paginação** pode ser usada para obter esse efeito (dividindo a memória em **blocos**, e o programa, em **páginas** do mesmo tamanho).
- A alocação (virtual ou não) de memória paginada **dispensa** que os **endereços** alocados na memória física sejam **contíguos**.
- A virtualização de memória é **compatível** também com o método da alocação **segmentada**, a ser estudado em outra aula.

Memória Paginada com Virtualização

- A memória física paginada já permitia **eliminar a fragmentação** e evitar a obrigatoriedade de alocação contígua dos espaços de endereçamento dos programas.
- No entanto, ainda não resolvia o problema da exigência de que, para ser executado, o programa estivesse inteiramente presente na memória física.

- Exatamente com a finalidade de livrar-se do problema de alocar fisicamente na memória todo o espaço de endereçamento lógico do programa, foram desenvolvidos o conceito e a técnica da **virtualização** de memória.
- Consiste em criar, para o programa, a **ilusão** de que existe **memória física suficiente para que todo o seu espaço lógico** de endereçamento seja alocado e utilizado para efeito de execução.

- Para promover essa virtualização de memória, é preciso que o programa disponha de uma área suficiente, em um meio de **armazenamento secundário**, para comportar na íntegra o espaço lógico de endereçamento.
- Para viabilizar essa manobra, o *loader* deve ser modificado para que, em cada instante, estejam presentes **na memória física as páginas essenciais** à execução do programa naquele instante, mantendo as demais no armazenamento secundário do sistema.

- Se e quando houver referências a endereços lógicos que não estejam ainda presentes na memória física, o hardware deverá gerar um **pedido de interrupção (“falta de página”)**.
- O tratamento desta interrupção deve promover o acionamento do novo ***loader,*** para que a página faltante seja por ele carregada na memória física, e o endereço correspondente, devidamente mapeado.

- À medida que as páginas do programa vão sendo alocadas e carregadas na memória física, as **tabelas de mapeamento devem ir sendo correspondentemente atualizadas**.
- Um **registrador especial do hardware deve apontar para a tabela de mapeamento** de páginas, de forma que a conversão de endereços lógicos em endereços físicos possa ser devidamente promovida, em tempo de execução.

- Na tabela de mapeamento, **páginas não alocadas** devem ser marcadas como **ausentes**, e associadas ao **endereço da correspondente imagem no disco**.
- No momento em que a alocação acontece (em resposta à interrupção de requisição da página), a página deve ser **mapeada**, ou seja, deve ser marcada como **presente**, e a correspondente informação de mapeamento deve ser registrada na tabela.

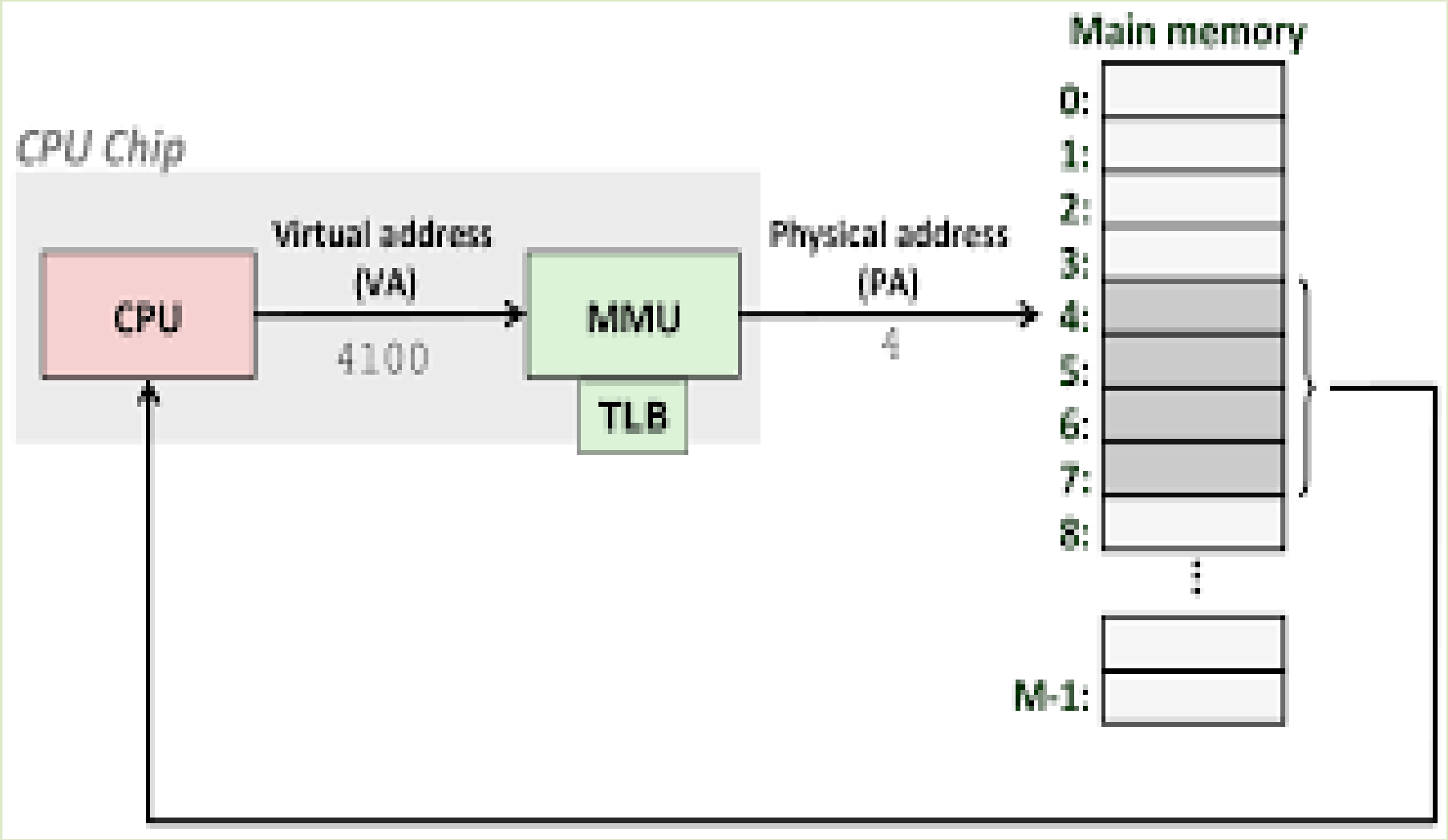
- O mecanismo de paginação usual é mantido, a menos do acréscimo das **interrupções de requisição de páginas**, e do correspondente tratamento.

UNIDADE DE GERENCIAMENTO DE MEMÓRIA VIRTUAL (MMU)

MMU – *Memory Management Unit*

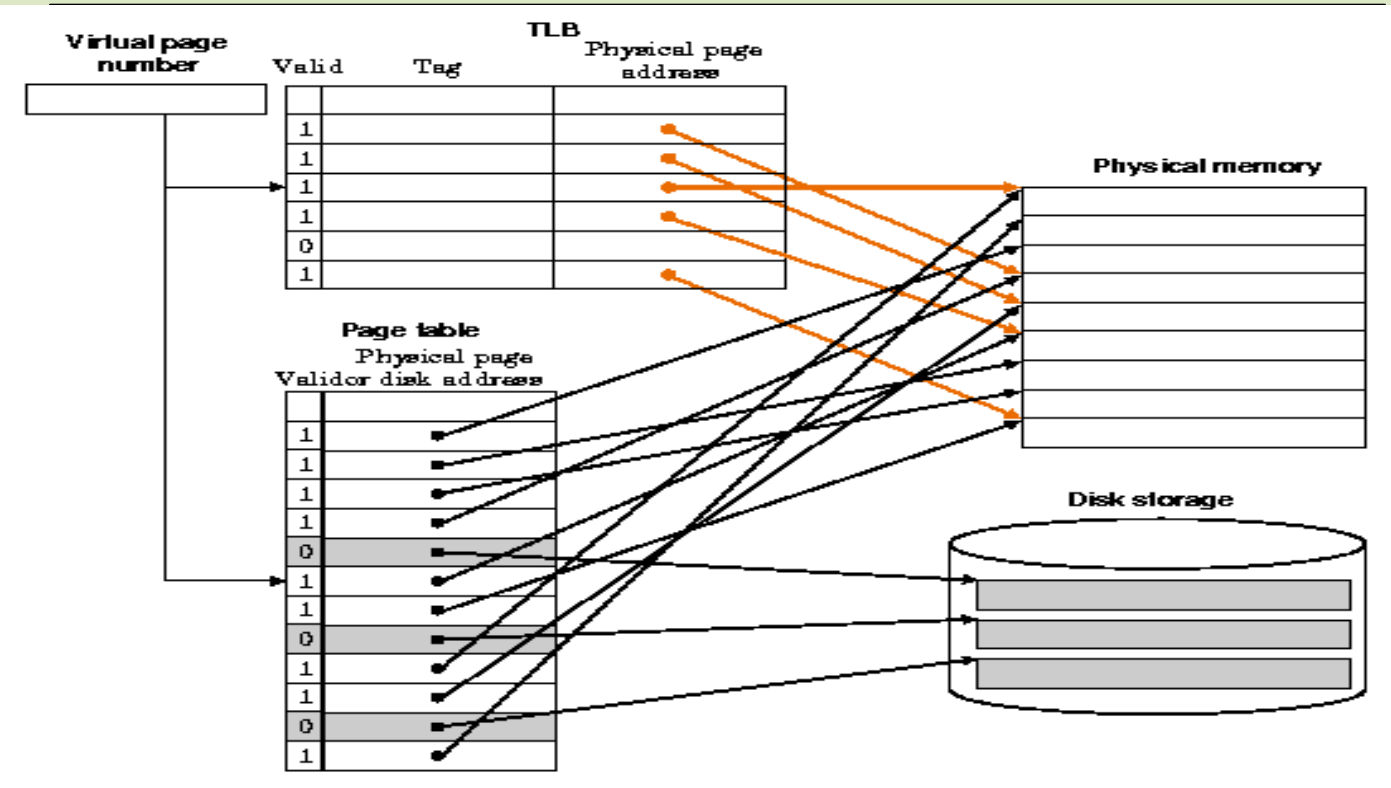
- É um componente do hardware do computador que, entre outras funções, efetua o **mapeamento dos endereços virtuais de memória (neste caso, paginada) em endereços físicos.**
- Traduz para a forma de endereço físico o endereço virtual a ser utilizado por um processo do usuário, **antes que tal endereço seja utilizado em acessos à memória.**
- O programa do usuário trabalha de forma direta e transparente, portanto, com endereços lógicos, desconhecendo completamente os endereços físicos utilizados de fato pelo computador.

Uma arquitetura com MMU



TLB – *Translation Lookaside Buffer*

- É uma **memória associativa** que reside na MMU e permite minimizar acessos desnecessários à memória física.
- Pelo **princípio da localidade**, em um dado intervalo de tempo, a maior parte dos acessos à memória ocorre em um número reduzido de páginas, com alta taxa de uso.



TLB (*translation lookaside buffer*)

- Um TLB (***translation lookaside buffer***) é uma memória associativa, tipicamente implementada como parte da MMU
- Com taxas de utilização suficientemente altas, os acessos extra às tabelas de páginas tornam-se raros.
- Conta-se como acessos efetivos apenas endereços virtuais completamente resolvidos.
- Havendo multiprocessamento, o TLB precisa ser desocupado para haver o chaveamento de contexto, mas isso é oneroso.
- Uma possível solução é adicionar um campo à memória associativa para conter um identificador do processo, para que seja modificado na ocasião de chavear o contexto.
- A gestão do TLB pode ser feito tanto por hardware como pelo sistema operacional.

Tabelas de página invertidas

- É inviável o uso de tabelas de páginas convencionais para espaços de endereçamento grandes (de 64 bits)
- As **tabelas invertidas de páginas** são ordenadas pelo número dos blocos e não pelo das páginas virtuais.
- É usada **uma só tabela invertida de páginas** para todos os processos que compartilham a memória.
- Cada item dessa tabela **indica o processo/página** a que se refere
- Uma tabela de **hash** é usada para evitar que seja feita uma busca linear a cada página virtual endereçada
- Em adição, registros no TLB são usados para indicar os itens da tabela associados às **páginas que foram usadas mais recentemente**

Páginas compartilhadas

- Processos diferentes podem, implícita ou explicitamente, fazer uso compartilhado de páginas comuns (memória compartilhada).
- Naturalmente, isso não costuma ser viável em áreas de dados, que em geral são privativas do processo a que pertencem.
- Áreas de código executável ou áreas públicas de dados podem ser compartilhadas, evitando a criação desnecessária de múltiplas cópias.

Algoritmos de substituição de páginas

- **Algoritmo de substituição ótima** – teórico, só seria viável em sistemas não causais
- **Algoritmo FIFO (*first in, first out*)** – páginas mais antigas vão sendo substituídas conforme ocorrem novas referências. Páginas muito usadas tendem a ficar antigas e portanto tendem a ser indevidamente substituídas por este algoritmo. A anomalia de Belady é uma possibilidade concreta com este algoritmo.

Algoritmos de substituição de páginas

- **Anomalia FIFO (= Anomalia de Belady)** – É de se esperar, à primeira vista, que se for ampliada a área disponível de memória física, deva em consequência ocorrer uma redução do número de substituições de páginas. Todavia, usando-se a política FIFO de substituição de páginas, isso não acontece. Esse fenômeno, relativamente inesperado, se denomina **Anomalia FIFO**, ou **Anomalia de Belady**, e desqualifica o uso da política FIFO em uma série de aplicações.

Ilustração da Anomalia de Belady

- Observe-se, no exemplo abaixo, que com 3 blocos de memória disponíveis, e com substituição FIFO, resultaram da sequência de requisições abaixo menos faltas (F) de página que com 4 blocos disponíveis:

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	4	4	4	5	5	5	5	5	5
	2	2	2	1	1	1	1	1	3	3	3
		3	3	3	2	2	2	2	2	4	4
F	F	F	F	F	F	F			F	F	

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	1	1	5	5	5	5	4	4
	2	2	2	2	2	2	1	1	1	1	5
		3	3	3	3	3	3	2	2	2	2
			4	4	4	4	4	4	3	3	3
F	F	F	F			F	F	F	F	F	F

Algoritmos de substituição de páginas

- **Algoritmo LRU (*least recently used*)** – as páginas **menos recentemente usadas** são preferidas para serem substituídas (aquelas que estão ocupando memória há mais tempo, sem estarem em uso).
- É **relativamente eficaz**, pois utiliza uma **pilha** como memória aproximada do uso das páginas, preserva as páginas situadas mais próximo do topo da pilha, e substitui as mais antigas, se for necessário.

Algoritmos de substituição de páginas

- **Algoritmo NFU (*not frequently used*)** – escolhem-se para serem substituídas as páginas **mais raramente utilizadas**, esperando-se continuarem apresentando uma baixa probabilidade de uso.
- **Algoritmo NRU (*not recently used*)** – substituem-se preferencialmente as páginas que **não tenham sido referenciadas** recentemente, esperando-se que em futuro próximo continuem não sendo referenciadas.

Aproximação do algoritmo LRU

- Usando um **bit de referência** (para indicar se a página foi referenciada) e um **bit de modificação** (indicando se foi alterada), consegue-se uma forma econômica de implementação de um algoritmo de substituição com desempenho que exibe **resultados aproximados, porém com tendências similares** aos do algoritmo LRU.