Sistemas Operacionais - Aula 22 – Exercícios - Soluções

1) Cite duas diferenças entre endereços lógicos e físicos.

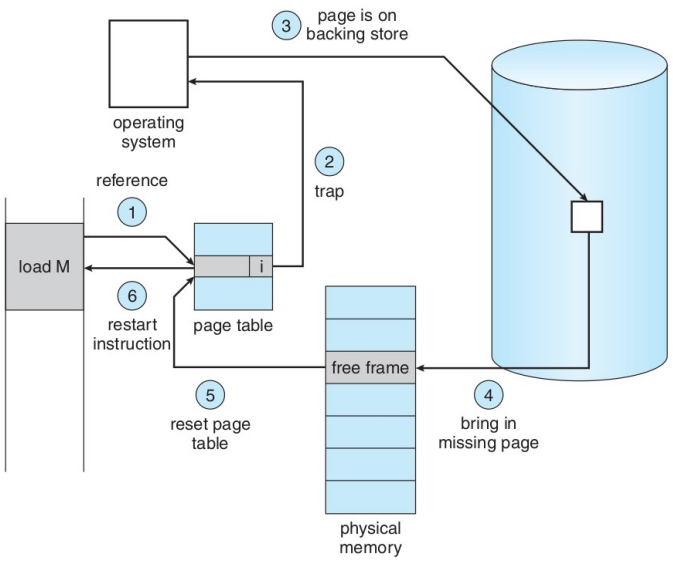
   Endereços lógicos são aqueles gerados pela CPU em função do processo corrente em execução, ou seja, são virtuais. Esses endereços são um mapa virtual de endereços associados à cada processo, ou seja, podem existir múltiplos espaços com endereços de 0 a 1000, por exemplo, sendo esses espaços independentes. Endereços físicos são aqueles relacionados à memória física (hardware), sendo esses compartilhados por múltiplos processos. Esses endereços são compreendidos em um único espaço de endereços unificado.

2) O que é fragmentação de memória externa e interna? Explique o tipo de fragmentação existente nos sistemas de gerência de memória por paginação e por segmentação.

   Fragmentação externa acontece quando temos alocação de regiões de memória de forma contígua, que eventualmente podem ser liberadas, gerando pequenos espaços entre essas regiões que podem não ser aproveitados facilmente. A fragmentação externa ocorre em sistemas que utilizam gerência de memória real (sem paginação) com partições de tamanho variado ou em sistemas que utilizam segmentação pura. Fragmentação interna ocorre quando o espaço alocado em uma página (no caso de um sistema com paginação) ou partição (sistemas sem paginação e com partições de tamanho fixo) não é utilizado em sua totalidade, não podendo ser aproveitado para outras alocações.

3) Em que situações ocorre uma falha de página (page fault)? Descreva as ações tomadas pelo SO quando uma falha de página ocorre.

   A figura abaixo (retirada do livro Silberschatz A, Operating Systems Concepts 9th. Ed.) apresenta o fluxo de ações na ocorrência de um page fault:



   Um page fault ocorre quando o processo corrente gera um endereço (referência) que não está mapeado na tabela de páginas (1), ou seja, é gerado um endereço virtual para uma página que pode estar no disco, porém não está na memória física e portanto não possui uma tradução de endereço virtual para físico armazenada na tabela de páginas. As ações

tomadas são: (2) o page fault gera uma trap (interrupção) e o SO é chamado; (3) se o endereço virtual referencia uma página que está armazenada em disco (é um endereço válido) o OS escolhe uma posição livre na memória física para alocação. Pode acontecer de não existir memória livre, portanto o SO deve colocar o conteúdo de uma moldura no disco antes de trazer a página do disco (4); o SO atualiza a tabela de páginas com a referência da moldura onde reside a página trazida do disco (5); no final (6), a instrução que gerou o page fault é executada novamente.

4) Considere um sistema com páginas de 4K, endereçamento lógico de 16 páginas, e endereçamento físico de 8 frames. Considere a seguinte tabela de páginas do processo em execução:

	bit validade	moldura
0	1	2
1	1	1
2	1	6
3	1	0
4	1	4
5	1	3
6	0	-
7	0	-
8	0	-
9	1	5
10	0	-
11	1	7
12	0	-
13	0	-
14	0	-
15	0	-

a) Mostre em quais endereços físicos a MMU traduz cada uma das seguintes referências à memória feitas pelo processo corrente: 8292 e 4094.

b) Mostre como fica uma Tabela Invertida para este sistema, com o mesmo mapeamento, e como os endereços do item anterior são traduzidos com esta tabela.

c) Mostre como fica uma Tabela em dois níveis para este sistema com o mesmo mapeamento, se o primeiro e segundo níveis suportarem 4 entradas cada. Além disso, mostre como os endereços do item a) são traduzidos com esta tabela.

Páginas de 4kB (12 bits), 16 páginas (4 bits) e 8 frames/quadros/molduras (3 bits).
a)
Endereço 8292, página 2, moldura 6 (deslocamento: 100);
`[0010 000001100100] (2*4k+100) → [110 000001100100] (6*4k+100 = 24676)`

Endereço 4094, página 0, moldura 2 (deslocamento: 4094);

`[0000 111111111110] (0*4k+4094) → [010 111111111110] (2*4k+4094 = 12286)`

*Pode-se multiplicar o número da moldura pelo tamanho da página e somar o deslocamento para calcular o endereço virtual. Exemplo: 6 * 4096 + 100 = 24676.*

b)

endereço 8292 → page 2, frame 6 (endereços 24576 a 28671) → 24676

endereço 4094 → page 0, frame 2 (endereços 8192 a 12287) → 12286

tabela de páginas invertida:

endereço	*frame*	*page*
0k-4k	0	3
4k-8k	1	1
8k-12k	2	0
12k-16k	3	5
16k-20k	4	4
20k-24k	5	9
24k-28k	6	2
28k-32k	7	11

c)

Considerando 4 entradas no primeiro nível (2 bits) e 4 entradas por página de segundo nível (2 bits) o formato do endereço fica: [PT1 | PT2 | offset] → [2 bits | 2 bits | 12 bits]. Para realizar as traduções virtual para físico, deve-se usar os 2 bits mais significativos para acessar a entrada em PT1, os próximos 2 bits para a entrada em PT2 e com o número da moldura, multiplicar esse pelo tamanho da página e somar o deslocamento.

`8292 → [00 10 000001100100], PT1: 0, PT2: 2 → moldura 6`

`4094 → [00 00 111111111110], PT1: 0, PT2: 0 → moldura 2`

PT1:

ent.	table
0	t0
1	t1
2	t2
3	t3

PT2:

t0

ent.	frame
0	2
1	1
2	6
3	0

t1

ent.	frame
0	4
1	3
2	-
3	-

t2

ent.	frame
0	-
1	5
2	-
3	7

t3

ent.	frame
0	-
1	-
2	-
3	-

5) Considere um espaço de endereçamento lógico de oito páginas de 1024 palavras de 2 bytes cada, mapeado em uma memória física de 32 quadros. Quantos bits existem no endereço lógico? E no endereço físico?

- *Cada página possui 1024 palavras com 2 bytes cada, portanto o tamanho da página é 2kB (2048 bytes);*
- *Para um tamanho de página de 2kB, são necessários 11 bits (2^{11} bytes);*
- *Para representar o espaço de endereços lógicos de 8 páginas, são necessários 3 bits (2^3 páginas);*
- *Para representar o espaço de endereços físicos de 32 molduras, são necessários 5 bits (2^5 molduras);*
- *Os respectivos formatos de endereços (lógico e físico) são compostos pela concatenação do número de bits para páginas ou molduras com o número de bits para o deslocamento, ou seja, [páginas | deslocamento] e [molduras | deslocamento], portanto:*
  - *Endereço lógico: [3 bits | 11 bits] = 000 00000000000 (14 bits)*
  - *Endereço físico: [5 bits | 11 bits] = 00000 00000000000 (16 bits)*

6) Considere um sistema com páginas de 4K, endereçamento de 16 bits e memória física de 16K, cujo sistema operacional trabalhe com memória virtual paginada. Considere que nenhum programa esteja em execução e que todas as molduras estejam inicialmente livres. Por simplicidade, suponha que todas as molduras possam ser ocupadas pelo programa (embora isto seja irreal, já que pelo menos o Sistema Operacional deve estar em execução e ocupando espaço de memória). Suponha que um programa entre em execução e faça referência aos seguintes endereços virtuais, nesta ordem: 200, 8191, 4196, 16383, 8192, 65535.

   a) Construa uma Tabela de Páginas (Convencional, ou Invertida ou Multinível) para este sistema, indicando os campos existentes na Tabela de Páginas.

End (virtual)	*Página*	*Moldura*
0k-4k	0	0
4k-8k	1	1
8k-12k	2	3
12k-16k	3	2
16k-20k	4	-
20k-24k	5	-
24k-28k	6	-
28k-32k	7	-
32k-36k	8	-
36k-40k	9	-
40k-44k	10	-
44k-48k	11	-
48k-52k	12	-
52k-56k	13	-
56k-60k	14	-
60k-64k	15	4

Tabela de páginas (1 nível)

End (físico)	*Moldura*	*Página*
0k-4k	0	0
4k-8k	1	1
8k-12k	2	3
12k-16k	3	2
16k-20k	4	15
20k-24k	5	-
24k-28k	6	-
28k-32k	7	-

Tabela de páginas (invertida). Não foi apresentado o pid do processo associado à cada página, pois no exercício é considerado apenas um programa. A tabela foi construída para uma memória física de 32K, diferentemente do exercício (16K). Isso foi feito para evitar a execução de uma política de substituição, que seria necessária no caso de uma memória de 16K (a memória física estaria lotada, e o acesso à página 15 ocasionaria em uma substituição). Aumentando o tamanho da memória, evitamos isso nesse exercício.

b) Mostre para cada endereço de memória virtual referenciado pelo programa, em qual endereço físico ele é traduzido, e como este processo é feito pela MMU e pelo sistema operacional. Atualize a tabela de páginas, quando necessário.

Considerando a tabela de páginas em um nível e que a TLB não esteja sendo utilizada (efetivamente, os acessos geram TLB miss, ocasionando um caminhamento pela tabela de páginas), a sequência seria:

deslocamento: endereço virtual % tamanho da página

*endereço virtual: pagina * tamanho da página + deslocamento*

*endereço físico: moldura * tamanho da página + deslocamento*

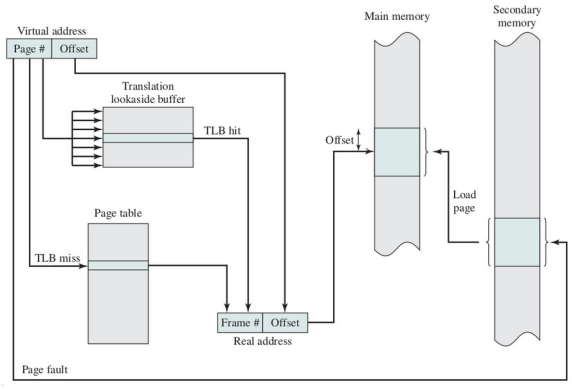
- *end virtual 200 (0*4k+200, page 0) → page fault → próx moldura livre é 0 → end físico 200 (0*4k+200, frame 0)*
- *end virtual 8191 (1*4k+4095, page 1) → page fault → próx moldura livre é 1 → end físico 8191 (1*4k+4095, frame 1)*
- *end virtual 4196 (1*4k+100, page 1) → end físico 4191 (1*4k+100, frame 1)*
- *end virtual 16383 (3*4k+4095, page 3) → page fault → próx moldura livre é 2 → end físico 12287 (2*4k+4095, frame 2)*
- *end virtual 8192 (2*4k+0, page 2) → page fault → próx moldura livre é 3 → end físico 12288 (3*4k+0, frame 3)*
- *end virtual 65535 (15*4k+4095, page 15) → page fault → próx moldura livre é 4 → end físico 20479 (4*4k+4095, frame 4)*

7) Em um sistema com paginação, um processo não pode acessar a memória que ele não possui. Por quê? Como o sistema operacional poderia permitir o acesso à outra área de memória? Por que deveria fazer isso? Faça uma representação onde dois processos compartilham um frame de memória através de paginação.

Em um sistema com paginação cada processo possui um conjunto de endereços virtuais, onde parte desses endereços (páginas) podem estar alocados na memória física (molduras). Uma referência a qualquer endereço fora do espaço de endereços virtuais gerará uma falha, pois não é possível traduzir tal endereço (não existe uma página com tal endereço). O sistema operacional poderia permitir o acesso à outra área de memória (por exemplo, de uma biblioteca dinâmica) compartilhando uma moldura da memória física entre múltiplos processos. Nesse caso, bastaria que as tabelas de páginas de diferentes processos apontassem para a mesma moldura e isso poderia ser feito para reduzir o uso de memória.

8) Explique o que é TLB e como ela é usada pelo SO. Faça uma figura que relaciona corretamente: endereço virtual gerado pela CPU, TLB, tabela de páginas e endereço físico.

TLB (translation lookaside-buffer) é uma estrutura de memória existente na MMU (memory management unit), que tem como objetivo servir de cache de traduções de endereços virtuais para físicos. No início, a TLB possui suas entradas vazias, e dessa forma o SO não encontrará traduções nela, realizando o acesso à tabela de páginas. Essas traduções acessadas na tabela de páginas são armazenadas na TLB, para reduzir o custo nos próximos acessos. (Figura retirada do livro Stallings W., Operating Systems Internals and Design Principles, 7th. Ed.)



9) Assuma que um programa está para acessar um endereço na memória virtual. Explique quando cada uma das situações abaixo pode acontecer.

a) TLB miss sem page fault.

b) TLB miss e page fault.

c) TLB hit sem page fault.

a) O endereço gerado pelo processo não é encontrado na TLB (ocorre um TLB miss). A tabela de páginas é percorrida (page walk), e uma tradução de endereço virtual para físico é encontrado na mesma.

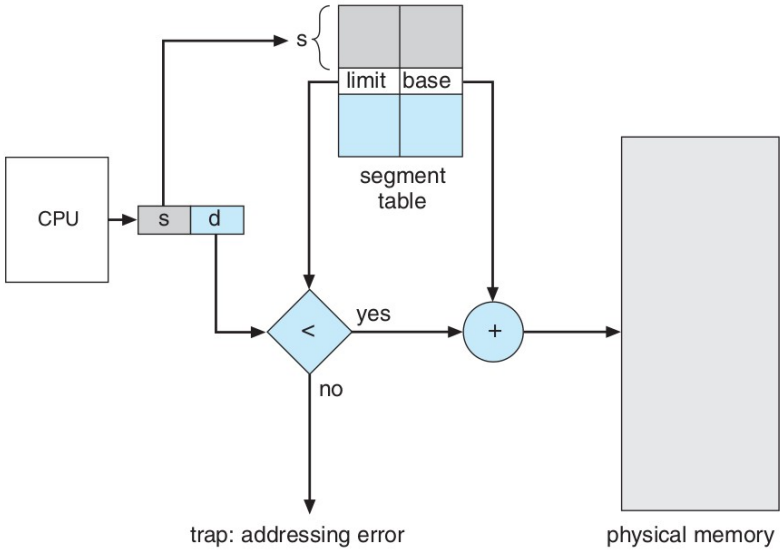
b) O endereço gerado não é encontrado na TLB (TLB miss). A tabela de páginas é percorrida, e como uma tradução para o endereço virtual não é encontrada, ocorre um page fault.

c) O endereço gerado é encontrado na TLB. Dessa forma, a tabela de páginas não precisa ser percorrida.

10) Descreva o funcionamento de um sistema de memória que utiliza segmentação. Apresente um exemplo mostrando como o endereço físico é calculado a partir de um endereço virtual e de uma tabela de segmentos.

Um sistema gerência de memória que utiliza segmentação separa um processo em partes independentes (segmentos) que podem ser gerenciados de forma independente pelo sistema operacional. Cada segmento possui um espaço de endereços virtuais lineares (contíguos) independentes, além de permissões independendes (leitura, escrita e execução).

A figura abaixo (retirada do livro Silberschatz A, Operating Systems Concepts 9th. Ed.) apresenta o hardware necessário para implementar segmentação e ilustra como é feita a tradução de endereços. Para o cálculo de um endereço físico, o endereço virtual (gerado pela CPU) é decomposto em dois campos (segmento e deslocamento). O campo segmento é utilizado como índice, para acessar uma entrada na tabela de segmentos. Na tabela de segmentos, existem informações sobre a entrada selecionada (base e limite). Caso o deslocamento do endereço virtual seja maior que o limite, é gerada uma exceção (trap). O cálculo do endereço físico é realizado somando-se o endereço base (início do segmento na memória física) ao deslocamento, caso esse seja menor que o limite.



11) Apresente como funciona um mecanismo pelo qual um segmento pode pertencer ao espaço de endereçamento de dois processos diferentes. Em que situações isso seria útil?

O mecanismo utilizado para que um segmento possa pertencer ao espaço de endereços de múltiplos processos consiste em manter uma referência à um endereço base e limite de um segmento na memória física nas tabelas de segmento de cada processo, ou seja, ambas tabelas contendo uma referência para o mesmo segmento. Esse mecanismo é útil quando múltiplos processos que são instâncias de um mesmo programa, referenciam o mesmo segmento comum (por exemplo, ambos processos executam o mesmo código) ou quando existe um compartilhamento de dados entre diferentes processos.

12) Qual a diferença entre a gerência de memória com segmentação e gerência de memória real utilizando um esquema de partições com tamanho variável?

A diferença é que no segundo caso as partições são definidas e os processos são alocados à essas partições, mas possuem seus endereços relocados para que funcionem em uma determinada partição. Já no primeiro caso (segmentação) o processo referencia endereços virtuais que serão traduzidos por meio de uma tabela de segmentos dinamicamente, o que simplifica o processo de carga e realocação de segmentos na memória física, uma vez que os endereços virtuais e físicos não são os mesmos.