Nome: Davi Augusto Neves Leite
RA: 191027383

Resolução – Lista 1 de Sistemas Operacionais II – Paginação

1) O endereço virtual está relacionado aos endereços que um programa pode referenciar, aglomerando toda a capacidade disponível de memória do sistema (ou seja, ao máximo que o processador comporta). Esse tipo de endereçamento está relacionado ao programa na memória secundária (disco rígido) e é ele quem será utilizado durante a execução de instruções na memória principal. Já o endereço físico está relacionado aos endereços que de fato existem no sistema, aglomerando toda a capacidade **real** disponível de memória física (RAM). Esse tipo de endereçamento está relacionado ao programa na memória primária (RAM).

2) Primeiramente, é importante ressaltar que a conversão de um endereço virtual (EV) em endereço físico (EF), durante a paginação, é realizado por um hardware denominado **MMU** (Unidade de Gerenciamento de Memória), e não pelo Sistema Operacional.

Para realizar a conversão de um EV em EF são necessários os seguintes passos:

1) Suponha a ocorrência de uma instrução do tipo: MOVE REG, [EV]; em que EV é um endereço virtual referenciado pela página presente na memória primária.
2) O endereço EV é mandado a MMU que identifica a página específica desse endereço. Para fins didáticos, vamos supor que EV = 5 e que este endereço estejá contido na **sexta posição** da **primeira** página virtual (PV0) de tamanho 4K (abrange intervalo de 0K até 4K). A instrução a ser levada para a MMU é: MOVE REG, [5]
3) Então, a MMU verifica a existência dessa página na memória principal, utilizando a chamada **tabela de páginas** (estrutura de dados que contém, em cada linha, informações a respeito de cada página virtual do programa). Caso a página esteja, então vai para o passo abaixo; do contrário, ocorre um evento denominado **falta de página**.
4) Agora, vamos supor que a PV0 está na terceira página física (PF2), ou seja, a página está alocada na memória principal (intervalos de 8K até 12K para

páginas de tamanho 4K). Dessa forma, a MMU transforma o EV da seguinte maneira: é identificado a distância de “EV” para o endereço de início da PV0 (neste caso, a distância seria: EV - Endereço_Virtual_Inicial = 5 - 0 = 5; uma vez que a PV0 tem endereço associados de 0K até 4K), tendo essa distância somada ao endereço físico onde a página está alocada (neste caso, Endereço_Físico_Inicial + Distância = 8K + 5 = 8197).

5) Dessa forma, a instrução final para o exemplo acima é dada por: MOVE REG, [8197].

É importante ressaltar que o **deslocamento** do endereço virtual e do endereço físico são sempre os mesmos!

3) Supondo uma página de tamanho P, então um endereço virtual “EV” produz um número de página virtual do tipo “NPV” e um deslocamento “D” relacionados da seguinte forma: **NPV = EV//P (divisão inteira)** e **D = EV % P (resto da divisão)**. Dessa forma, ao utilizar-se tamanhos de página em potência de 2 (ou seja, 2^n), então sempre os **“n” bits menos significativos** de um EV designarão o **deslocamento** “D” dentro de uma página, enquanto que os **bits restantes mais significativos** designarão o **número da página virtual “NPV”**. Ou seja, ao utilizar o tamanho da página como sendo um múltiplo de 2 (2^n), pode-se evitar o processo de divisão inteira da fórmula citada acima.

4) Calcular NPV e D: página de 4KB (P4K) e página de 8KB (P8K).

A)

P4K: NPV = 20000//4KB = 4 || D = 20000 % 4KB = 3616

P8K: NPV = 20000//8KB = 2 || D = 20000 % 8KB = 3616

B)

P4K: NPV = 32768//4KB = 8 || D = 32768 % 4KB = 0

P8K: NPV = 32768//8KB = 4 || D = 32768 % 8KB = 0

C)

P4K: NPV = 60000//4KB = 14 || D = 60000 % 4KB = 2656

P8K: NPV = 60000//8KB = 7 || D = 60000 % 8KB = 2656

5) Obter endereço físico a partir dos endereços virtuais.

A)

EV = 20; NPV = 0; mapeado na moldura 2 (8K até 12K).

Portanto: EF = EM_INICIAL + D = 8K + (20 - 0) = 8212

B)

EV = 4100; NPV = 1; mapeado na moldura 1 (4K até 8K).

Portanto: EF = EM_INICIAL + D = 4K + (4100 - 4K) = 4100

C)

EV = 8300; NPV = 2; mapeado na moldura 6 (24K até 28K).

Portanto: EF = EM_INICIAL + D = 24K + (8300 - 8K) = 24684

6)

*Tamanho das Páginas: 2KB = 2^11 => 11 bits serão de deslocamento!

A)

Total de Endereços Virtuais = 2^20 = 1MB => 1024KB e 20 bits (cada página)

TPAG = T_EV/T_P = 1024KB/2KB = 512 páginas.

Dessa forma, tem-se que os **últimos** 11 bits de cada página do endereço virtual (correspondente a 20 bits) serão destinados ao deslocamento, e os outros bits restantes (neste caso, 20 - 11 = 9 bits) serão destinados para o número da página virtual.

B)

Total de Endereços Físicos = (1/4) * TOTAL_EV = (2^20)/(2^2) = 2^18 => 256KB e 18 bits (cada página)

TPAG = T_EF/T_P = 256KB/2KB = 128 páginas.

Dessa forma, tem-se que os **últimos** 11 bits de cada página do endereço físico (correspondente a 18 bits) serão destinados ao deslocamento, e os outros bits restantes (neste caso, 18 - 11 = 7 bits) serão destinados para o número da página física (ou moldura).

7)

Total de Endereços Virtuais = 2^32 => 32 bits cada página.

Foi utilizado, neste exercício, uma tabela de páginas de **dois níveis** da seguinte maneira:

Nível 1 -> 2^9 => 9 bits de seleção de páginas, o que corresponde à 512 páginas no total.
Nível 2 -> 2^11 => 11 bits de seleção de páginas, o que corresponde à 2048 páginas no total.
Deslocamento -> D = 32 - 9 - 11 = 12 bits de deslocamento para cada página.

Dessa forma, para saber o tamanho da página de fato (endereços utilizáveis para dados) basta realizar:
TAM_PG = 2^(32-9-11) = 2^12 = 4096 bytes
E, por fim, para saber quantas páginas existem no espaço de endereçamento citado (32 bits), basta realizar:
QTD_PG = (2^32)/(2^12) = 2^20 = 1048576 páginas = 512 * 2048 = (N1) * (N2)

8)
Total de Endereços Virtuais = 2^32 => 32 bits cada página
Tamanho das Páginas: 4KB = 2^12 => 12 bits serão de deslocamento!

Para saber a quantidade de entradas na tabela de páginas, utilizando a paginação tradicional de **um nível**, basta realizar o seguinte cálculo:
QTD_TP = T_EV = 2^(32 – 12) = 2^20 linhas de endereço, sendo 12 bits para deslocamento de cada página. A PV0 ficará a parte de código e dados; e a PV(2^20) ficará a pilha (duas entradas).
Agora, em caso de utilização de uma tabela de páginas com **dois níveis** de **10 bits em cada nível**, basta realizar o seguinte cálculo (sabendo previamente que existem 12 bits de deslocamento):
QTD_TP = 2^10 linhas de endereço **em cada nível**, totalizando 2^(10 + 10 + 12) = 2^32.

9)
A) A falta de página é uma indicação (interrupção) da MMU (hardware responsável por converter endereços virtuais em físicos) ao SO quando um endereço virtual é referenciado por uma instrução, mas a sua respectiva página virtual não está na memória primária (ou seja, numa “moldura”). Isso ocasiona a aplicação, pelo SO, de

um algoritmo de substituição de página e, após o processo, a instrução que necessitava do endereço virtual referenciado é continuada.

B) É dado o nome de *Thrashing* à ação frequente de falta de página por um ou mais programas em execução. Ou seja, ocorre quando o número de páginas físicas é inferior ao tamanho do chamado conjunto de trabalho (àquele que consiste de todas as páginas usadas nas “n” referências mais recentes à memória).

10) A **paginação por demanda** está relacionada ao carregamento de páginas à medida em que são usadas pelos processos. Ou seja, apenas as páginas referenciadas pelas instruções são carregadas na memória principal. Isso pode ser um problema em sistemas multiprogramados, haja visto que existe a comutação constante de processos, possibilitando a existência de *Thrashing*.

Já a **pré-paginação** consiste no carregamento das páginas mais utilizadas antes da execução novamente do processo, marcando-as na tabela de páginas. Ou seja, é nesse tipo de paginação em que o conjunto de trabalho (consiste de todas as páginas usadas nas “n” referencias mais recentes à memória) é aplicado de tal maneira que o SO busca determiná-lo para cada processo, tendo o conjunto de páginas mais utilizadas na memória antes de rodar o processo e, dessa forma, o *Trashing* é evitado.

11)

A)

NUR: remove a página de ordem mais baixa (das quatro classes) e **só leva em conta os bits R e M**. Dessa forma, a página a ser trocada é a página 2.

B)

FIFO: é mantido uma lista com páginas utilizadas, tendo a **página mais antiga como a primeira da lista (e é a que é escolhida para remoção)**. Dessa forma, a página a ser trocada é a página 3.

C)

MRU/LRU: pode ser por hardware ou software. **Existência de um contador (por página)** que incrementa automaticamente a cada referência da página na memória, e o SO vai selecionar o de **menor valor**. Dessa forma, a página a ser trocada é a página 1 (menos tempo carregada e com bit R em 0).

D)

Segunda Chance: semelhante ao FIFO, mas escolhe a **com bit de referenciada igual a 0 (coloca as páginas, conforme percorre do início da lista, de bit referenciada igual a 1 no fim da fila com esse bit valendo 0)**. Dessa forma, a página a ser trocada é a página 2.

12) Aplicando o NUR (leva em conta somente os bits R e M, tendo um *clock* para zerar o R periodicamente) tem-se:

1, 2, 4, 3 -> inicialmente na memória principal (Falta de Páginas = 4) => 1, 2, 4, 3

7 -> será colocado no lugar da página 1 (Falta de Páginas = 4 + 1 = 5) => 7, 2, 4, 3

1 -> será colocado no lugar da página 2 (Falta de Páginas = 5 + 1 = 6) => 7, 1, 4, 3

3, 4 -> já estão na memória principal (Falta de Páginas = 6) => 7, 1, 4, 3

5 -> será colocado no lugar da página 7 (Falta de Páginas = 7) => 5, 1, 4, 3

7 -> será colocado no lugar da página 1 (Falta de Páginas = 8) => 5, 7, 4, 3

2 -> será colocado no lugar da página 3 (Falta de Páginas = 9) => 5, 7, 4, 2

3 -> será colocado no lugar da página 4 (Falta de Páginas = 10) => 5, 7, 3, 2

4 -> será colocado no lugar da página 5 (Falta de Páginas = 11) => 4, 7, 3, 2

6 -> será colocado no lugar da página 7 (Falta de Páginas = 12) => 4, 6, 3, 2

Total de Falta de Páginas = 12

Páginas Virtuais Emolduradas ao Fim do Processo: 4, 6, 3, 2 (respectivamente).

13) Aplicando o FIFO (cria uma fila e remove a cabeça da fila quando necessário), tem-se:

0, 2, 1-> inicialmente na memória principal (Falta de Páginas = 3). => 0, 2, 1

3 -> será colocado no lugar da página 0 (Falta de Páginas = 4) => 3, 2, 1

4 -> será colocado no lugar da página 2 (Falta de Páginas = 5) => 3, 4, 1

3 -> já está na memória (Falta de Páginas = 5) -> 3, 4, 1

2 -> será colocado no lugar da página 1 (Falta de Páginas = 6) => 3, 4, 2

1 -> será colocado no lugar da página 3 (Falta de Páginas = 7) => 1, 4, 2

Total de Falta de Páginas = 7

Páginas Virtuais Emolduradas ao Fim do Processo: 1, 4, 2 (respectivamente).

14) Utilizando o algoritmo de envelhecimento, tem-se os seguintes contadores em cada *clock*:

Clock 1: 0, 1, 1, 1

Clock 2: 1, 1, 2, 2

Clock 3: 2, 1, 3, 2

Clock 4: 3, 2, 3, 3

Clock 5: 3, 2, 4, 3

Clock 6: 4, 2, 5, 3

Clock 7: 5, 3, 5, 3

Clock 8: 5, 3, 5, 4

Portanto, os valores finais dos contadores são, respectivamente: 5, 3, 5, 4.

15) Supondo que T = 200, a página a ser removida será a página 3 (iniciando a contagem em 0 e de baixo para cima, a primeira página de instante 2003). Isso se deve ao fato de que o tempo virtual atual é de 2204 e, tendo o conjunto de trabalho definido por **T = 200**, as **páginas "não importantes**" serão as que estarão em um **tempo virtual menor** que **TNP = 2204 - 200 = 2004 (a partir do atual)**.

Para selecionar a página correta, o **algoritmo realiza a seguinte subtração** (para bit R = 0 durante a procura da página): **IDADE = TEMPO_VIRTUAL_ATUAL - INSTANTE_DA_ULTIMA_REFERENCIA**; tendo a **"IDADE > T"** como parâmetro para a seleção da página a ser substituida (primeira ocorrência).

Desta forma, aplicando o algoritmo no exercício, obterá-se as seguintes idades (iniciando a procura de baixo para cima e lembrando que somente pode escolher os de bit R = 0): 190 (não, pois 190 <= T); 172 (não, pois 172 <= T); 184 (não, pois 184 <= T); 201 (sim, pois 201 > T). Portanto, a página 3 será selecionada para ser substituída, levando em conta o conjunto de trabalho com T = 200.

16) Tendo um sistema com processos de tamanho médio **TMP = 512KB** e tamanho de entrada da tabela de páginas sendo **TETP = 4 bytes**; **o tamanho de página ótimo (TPO)**, em bytes, será dado da seguinte maneira:

TPO = sqrt(2*TMP*TETP)

=> TPO = sqrt(2*512K*4) = sqrt(2*(2^19)*(2^2))

=> TPO = sqrt(2^22) = 2^11 bytes = 2048 bytes

Dessa forma, o tamanho de página ótimo, nesse sistema, é de 2048 bytes.