Prova de SOII 2002/01

1) O que é página vítima? Descreva um algoritmo ideal para a escolha da página vítima. Descreva um algoritmo utilizado na prática para a escolha da vítima.
Página vítima é a página física que é escolhida para ser retirada da memória em uma falta de página, quando não existe memória física disponível para uma nova página.
O algoritmo ideal é o ótimo, onde a taxa de oorrência de interrupções de páginas ausentes é baixa.
A página a ser substituiífda é a que não será usada pelo período de tempo mais longo. Só que esse algoritmo é difícil de implementar , porque requer conhecimento futuro de string de referência.
Um algoritmo utilizado na prátic é o LRU (least recently used). Substitui a página que não foi usada por mais tempo.Quando uma página precisa ser substituída ele seleciona a página que não foi usada pelo maior período de tempo. Essa estratégia é o algoritmo de sibstituição de página ótimo olhando para trás no tempo, ao invés de para frente.

2) Quais são os bits de controle existentes na tabela de páginas? Descreva o motivo de cada um destes bits estar ligado ou desligado (se existir mais de um motivo isso deve ser descrito).
Na tabela de páginas podem ser encontrados os seguintes bits de controle: bit válido-inválido, bit alterado (de modificação), bit de referência, bit de permissão de escrita, bit de permissão de execução, bit núcleo.
Bit válido-inválido- quando esse bit é definido como válido, seu valor indica que a página associada está no espaço de endereçamento lógico do processo e, portanto, é uma página legal (válida). Se o bit for definido como “inválido”, o valor indica que a página não está no espaço de endereçamento lógico do processo. Endereços ilegais são bloqueados com o uso do bit válido-inválido.
Bit alterado- é o bit ativado pelo hardware sempre que qualquer palavra ou byte na página for alterado indicando que a página foi modificada. Quando uma página ´s selecionada para substituição, examinamos seu bit de modificação. Se o bit estiver ativado, sabemos que a página foi modificada desde que foi lida do disco. Nesse caso, devemos gravar a página no disco. Se o bit não estiver ativo a página não foi modificada desde que foi carregada na memória.
Bit de referência – o bit de referência para uma página é ativado, pelo hardware, sempre que a página é referenciada (leitura ou escrita para qualquer byte na página). Eles estão associados com cada entrada na tabela de página.
Inicialmente, todos os bits são limpos (definidos como 0) pelo SO. À medida que o processo de usuário executa, o bit associado com cada página referenciada é ativado (para 1) pelo hardware. Após algum tempo, podemos determinar quais páginas foram usadas e quais não foram.
Bit de permissão de escrita de escrita é aquele que se estiver setado (1) indica que o conteúdo da página pode ser alterado, ou seja, podem ser gravados novos dados na página. Se estiver desligado o conteúdo da página não poderá ser modificado. As páginas de código tem seus bits de escrita desligados para que o usuário não possa alterar o código do processo. As páginas de dados tem seus bits de escrita ligados de forma que seus dados possam ser alterados. Há casos em que o bit de escrita está desligado na página de dados (COPY ON WRITE) que consiste no compartilhamento de uma página de dados por dois processos distintos. Este bit permanece desligado enquanto os processos efetuam leitura da página. Quando um dos processos tenta efetuar uma escrita na página, é gerada uma interrupção. Então o SO faz uma cópia da página em outro local do espaço de endereçamento físico, e esta cópia pode ser exclusiva do processo que efetuou a operação de escrita. O bit de escrita das páginas são setados e cada um agora possui sua página exclusiva.

Bit de execução indica se a página considerada pode ou não ser executada. Nas páginas de código este bit se encontra ligado e nas páginas de dados o mesmo se encontra desligado para que se tente executar uma página que não seja código.
Bit núcleo – indica que as páginas são do sistema operacional e só podem ser modificadas pelo administrador.

3 - Descreva 3 estratégias para diminuir o problema de tabelas de páginas muito grandes.
1ª estratégia: Paginação em dois níveis a tabela de páginas é dividida em dois níveis, transformando a tabela de página em uma estrutura de dados de dois níveis,
Existe uma tabela em um nível mais alto chamado de diretório de páginas e esse diretório aponta para pedaços da tabela de Páginas original. Quando nesses pedaços existirem bit de ligado a tabela de diretórios marcará a presença com 1 no campo presente; quando nesse pedaços não existirem o bit de presença a tabela de diretórios marcará 0 no campo de bit presente.
Desta forma o tamanho ocupado pela estrutura de dados é muito menor que 4MB. O que se perde com a tabela de páginas de dois níveis? Em 2 níveis quando o hardware não consegue mapear pela TLB ele vai no diretório de páginas e vê se o pedaço existe. Se existir ele vai até o diretório de páginas e marca presente no seu pedaço correspondente ao que existe a página. O que torna mais lento.
2ª estratégia: Tabela de páginas invertidas ela tem uma entrada para cada página real que passa a ser o índice da tabela. Com isso ela em o número de entradas igual ao nº de páginas reais que existem e não o nº de páginas virtuais. Com isso ela é muito menor pois o nº de entradas é igual ao nº de páginas reais. O nº de páginas reais costumam ser muito menor que o nº de páginas virtuais e é uma única tabela.
3ª estratégia: Gerar interrupção quando a conversão não pode ser feita via TLB, no caso da SUN, o sistema operacional é que faz a conversão.
O SO tem que fazer uma estrutura de dados qualquer que pode ser até uma tabela invertida para fazer a conversão de virtual para real.

4 – Quais são as áreas de memória típicas de um processo? O que é localizado em cada uma destas áreas? O que motiva o crescimento de cada uma destas áreas? Como este crescimento pode ocorrer com a segmentação? Com paginação?
Área de pilha(reservada para variáveis locais)
Área de dados (para as variáveis globais e dinâmicas)
Área de código(código do programa)
Código e dados são alocados quando o processo começa a executar. Dados locais e variáveis dinâmicas vão sendo alocados conforme o programa executa.
Na área de pilha e a área de dados sofrem modificações. Na área de pilha pode ocorrer o stack overflow, que é o estouro da pilha, ou seja, quando o topo da pilha alcança o limite da máximo da pilha.Quando isso ocorre o SO tenta aumentar o limite da pilha (UNIX), caso contrário aborta o processo. A segmentação utiliza uma tabela de segmentos que guarda o início e o tamanho do segmento, desta forma visualiza a memória unidimensional de uma forma forma bidimensional. Com isso a alocação não precisa ser contínua. A segmentação possibilita a expansão da pilha, com segmento de pilha o hardware é capaz de perceber o topo está no seu limite inferior e gera uma interrupção e o SO trata e testa o tamanho da pilha. Logo a segmentação possibilita o reconhecimento do estoura da pilha. Com a paginação passou a existir duas formas de ver a memória, o espaço de endereçamento lógico e físico. Cada processo possui seu espaço de endereçamento lógico, sendo assim, ele pensa que está sozinho na memória. Com a paginação, basta ter memória real livre em qualquer posição que será mapeada na memória lógica. O espaço de endereçamento lógico é muito grande, praticamente acaba com o problema do crescimento da área de dados e de pilha, bastando para isso coloca-la afastadas uma da outro, se uma área atingir a outra como é paginação, basta alocar outra página livre em outro lugar na memória física, já que é paginação.

A área de código pode ser modificada durante a execução do programa, havendo um aumento e diminuição (programas automodificáveis), esses programas geram códigos que são incorporados a ele mesmo, mas isso não é muito comum. Geralmente a área de código fica protegida, se o processo tentar modifica-la ele é abortado. Nos sistemas operacionais existentes hoje em dia não existe a possibilidade de alterar a área de código.