**images­Universidade Federal Rural do Semi-Árido - UFERSA**

**Centro de Ciências Exatas e Naturais - CCEN**

**Departamento de Computação - DC**

**Curso de Ciência da Computação**

**Disciplina: Sistemas Operacionais**

**Prof. Leiva Casemiro Oliveira**

# 2.2-Tarefa de Sistemas Operacionais

**INSTRUÇÕES:** Essa atividade pode ser feita em grupo e os códigos (projeto Eclipse) e respostas correspondentes as questões devem ser enviados em um único arquivo (.zip) até a data estabelecida no SIGAA pelo líder do grupo como parte da nota da Unidade em avaliação. A avaliação também conta com a apresentação dos alunos em sala de aula.

**Orientações gerais**

* A implementação deve ser feita no MARS versão 4.5, disponível em <http://courses.missouristate.edu/kenvollmar/mars/>
* A implementação corresponde de parte da infraestrutura de sistema operacional para realizar gerenciamento de processos.
* Para se ter acesso ao código do MARS, após baixar o arquivo .jar utilize uma ferramenta de descompactação (ex: Winrar) e extraia seu conteúdo. Em seguida crie um projeto em uma IDE para Java (ex: Eclipse) com os arquivos extraídos.
  + O “Apêndice A” do livro “PATTERSON, D. A.; HENNESSY, J.L. Organização e projeto de computadores – a interface hardware software. 3. ed. Editora Campus, 2005” apresenta detalhes sobre o MIPS e chamadas de sistema.

**Descrição da Implementação:** Gerenciador de memória virtual

* O Gerenciador de Memória Virtual (GMV) será responsável por criar e manter um espaço de endereçamento virtual para cada processo como ilustrado na figura a seguir
* Assim, cada processo terá uma tabela que mapeia seus endereços virtuais para endereços de memória física (até o limite daquele processo)
* As páginas de endereços virtuais não mapeados estão no disco
* Assim o GMV deve:
  + Criar tabelas de mapeamento de endereços virtuais para os processos
  + Utilizar blocos de alocação (páginas) para troca de informações entre disco e memória física
  + Contabilizar os endereços físicos disponíveis para o processo
  + Mapear endereço virtual (páginas) em endereço físico disponível (molduras)
  + Desmapear um endereço físico para mapear outro que vem do disco quando o limite dos endereços dos processos for atingido
  + Desmapear todos os endereços quando o processo terminar

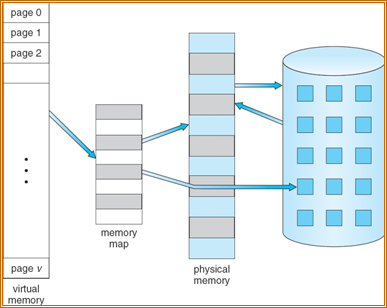
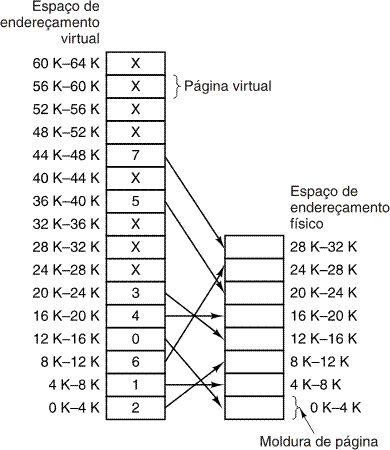
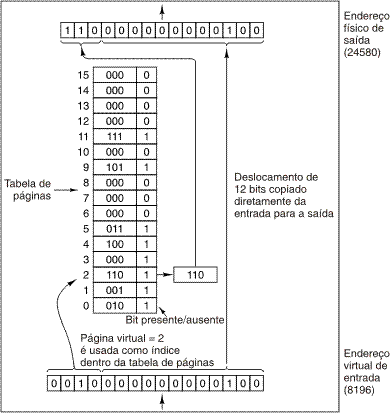


Figure 1 - Espaço de endereçamento virtual - Fonte: http://slideplayer.com.br/slide/2264627/

* Essa atividade consiste no desenvolvimento de uma ferramenta para simulação, logo não há um disco propriamente dito. Assim, serão feitas as seguintes considerações:
  + O “disco” na nossa simulação é a memória do MARS, onde sempre poderão ser obtidos os dados/instruções requisitadas (mas nos limitaremos apenas a endereços de instruções). Assim, neste texto, a memória do simulador será chamada de “disco” e a memória proposta simplesmente de memória.
  + Na simulação não será necessário fazer transferência propriamente dita entre o disco (memória do simulador) e a memória (proposta).
  + Sua ferramenta apenas avaliará os endereços (de instruções, indicadas pelo PC) e atualizará a tabela de memória virtual, levando em consideração quais páginas estão na memória e quais estão no disco, apenas para que saibamos quando um *trap* deveria acontecer e quando o algoritmo de substituição de páginas é invocado.
* É necessário implementar uma MMU (*Memory Manegment Unit*) que mantém a tabela de memória virtual para traduzir endereços virtuais em físicos.
* Criar uma classe “entrada da tabela virtual” com, no mínimo, os atributos:
  + Página Referenciada
  + Página Modificada
  + Proteção (bits R, W e X)
  + Bit Presente/ausente
  + Número da moldura mapeada
* Criar uma classe “tabela virtual” com uma coleção de “entradas da tabela virtual”
  + Cada processo terá sua própria tabela virtual, então pode-se optar por implementar um objeto “tabela virtual” para cada processo ou 1 único objeto que reserve uma quantidade fixa de “entradas da tabela” para cada processo
* O gerenciador de memória deve implementar memória virtual (considerando apenas dos endereços do segmento de código) simulada por meio de uma ferramenta que deve ser conectada ao MIPS no momento da execução do código *assembly*
  + Para implementar a ferramenta siga o tutorial disponível em: <http://courses.missouristate.edu/KenVollmar/mars/tutorial.htm>
  + Use como exemplo a ferramenta já existente “MemoryReferenceVisualization”
  + Sua ferramenta deve aproveitar o trabalho anterior e permitir realizar as seguintes configurações via interface gráfica:
    - Tamanho de bloco de alocação de memória (tamanho da página virtual) em quantidade de instruções do MIPS
    - Quantidade máxima de blocos de alocação por processo
    - Configuração do tipo de algoritmo de substituição de páginas da memória virtual (NRU, FIFO, Segunda chance, LRU)
    - As configurações não devem ser modificadas após iniciada a simulação (sugere-se haver um botão de confirmação das configurações para que fiquem desabilitadas logo após esse botão ser clicado e assim permanecer durante toda simulação)
  + Sua ferramenta também deve mostrar o estado atual da memória virtual de cada processo e as configurações escolhidas
    - O estado atual pode ser apenas do processo em execução ou de toda a memória (todos os processos)
    - O estado atual consiste em exibir informações sobre quais páginas virtuais estão mapeadas e em que molduras estão mapeadas, a quantidade de páginas virtuais disponíveis e se está acessando um bloco da memória física ou do disco. Inspire-se na figura a seguir



* + - A ferramenta deve mostrar as configurações atuais
    - E ainda deve exibir a seguintes informações (totais e por processo)
      * Quantidade de hits na memória
      * Quantidade de miss na memória (acessos ao disco)
      * Quantidade de substituições de blocos
  + Não há necessidade de modificações na memória do simulador
* Quando um processo é criado (*fork*), deve ser criada uma tabela de memória virtual para esse processo mas nenhuma moldura deve ser mapeada (ainda não acontece nenhuma transferidos do disco para memória) e os registradores de limite são setados
* A cada acesso a memória, a MMU verifica se a página está alocada na tabela (*hit*) e se o endereço é válido
  + Se estiver traduz para o endereço físico, verifica se o endereço está nos limites do espaço de endereçamento
    - Se não estiver imprime uma mensagem de erro no MARS e a simulação deve parar
  + Se estiver dentro dos limites e se tal página está presente na memória e se é válida então permite o acesso.
  + Se o endereço não estiver mapeado para memória (*miss*)
    - Verifica se a quantidade máxima de páginas alocadas foi atingida
      * Se sim, faz a substituição (usando o algoritmo de substituição de página configura) pelo novo bloco
      * Se não, aloca o novo bloco em um espaço vazio
  + Vale salientar que uma página virtual deve conter mais de 1 endereço físico (de acordo com a configuração da ferramenta antes da simulação), logo, endereços físicos simultâneos podem estar dentro de uma mesma página
    - Assim, a MMU deve verificar os campos “índice” e “deslocamento” do endereço virtual para mapeá-lo para endereço físico, assim como na figura a seguir
    - Para isso é necessário usado o endereço armazenado em PC e dividi-lo nos 2 campos mencionados



* + - Como as páginas mapeiam n instruções e como cada instrução possui 4 bytes, uma sugestão para calcular em qual página um endereço virtual pode está mapeado, é dividir tal endereço pela quantidade de bytes dentro da página.
      * Ex: Suponha que a tabela de um processo tenha apenas 2 páginas virtuais (página 0 e página 1) e cada página armazena 2 instruções. Suponha também que as instruções desse processo estejam nos seguintes endereços: 0x00400000, 0x00400004, 0x00400008 e 0x0040000c. Para calcular a página para onde o endereço é mapeado basta dividir o endereço por 8 (quantidade de bytes de 2 instruções). Assim:
        + Quando o 1º endereço for solicitado, a MMU o mapearia para a página zero. O 2º endereço também seria mapeado para a página zero.
        + O 3º e 4º endereços serão mapeados para a página 1 e assim por diante.
      * Uma sugestão para o deslocamento dentro da página é dividir o endereço pelo tamanho (4 bytes) e o resultado módulo quantidade de endereços dentro da página.
        + Ex: Considerando o mesmo exemplo, para o 1º endereço temos: (0x00400000 / 4) mod 2 = 0; para o 2º endereço: (0x00400004 / 4) mod 2 = 1; para o 3º endereço: (0x00400008 / 4) mod 2 = 0; e para o 4º endereço: (0x0040000c / 4) mod 2 = 1;
      * Se preferir o deslocamento em bytes, não realiza a divisão por 4, apenas o módulo por 16 (quantidade de bytes nas 2 entradas da tabela de página virtuais). E o resultado seria: para o 1º endereço: 0; para o 2º endereço: 4; para o 3º endereço: 8; para o 4º endereço: 12;
* Implementar os algoritmos de substituição de página: NRU, FIFO, Segunda chance, LRU, os quais serão invocado de acordo com a configuração inicial no momento que for necessário.
* Para validar seu gerenciador de memória
  + Utilize o código de teste dos trabalhos anteriores modificando os parâmetros da ferramenta e compare os resultados
  + Crie mais processos sem laços infinitos
  + Simule situações de erros que podem ser detectados pela ferramenta
* Utilize uma Tool semelhante ao do Gerente de Memória do MOSS (ver imagem abaixo).

