Curso de Tecnologia em Sistemas de Computação
Disciplina de Sistemas Operacionais
Professores: Valmir C. Barbosa e Felipe M. G. França
Assistente: Alexandre H. L. Porto

Quarto Período
AD1 - Segundo Semestre de 2006

Nome -
Assinatura -

---

1. O que é **multiprogramação**?

   Resp.: É um conceito que permite o compartilhamento do processador entre os vários processos, sendo que cada processo executará no processador por um dado intervalo de tempo, com o objetivo de maximizar o tempo de uso do processador. Outro processo poderá executar no processador se o processo atualmente em execução precisar executar uma operação de E/S, como a leitura de um dado do disco.

2. Por que a tabela de processos é necessária em um sistema de tempo compartilhado? Ela também é necessária em sistemas de computadores pessoais em que só um processo existe e toma conta da máquina inteira até que se encerre?

   Resp.: -Porque em um sistema de tempo compatilhado cada usuário pode executar um ou mais processos. Como a interação com os usuários

deve ser aceitável, nenhum processo poderá executar em um processador até o seu término. Deve-se então alternar a sua execução com a de outros processos, o que exigirá um espaço especial, uma entrada na tabela de processos, para garantir a correta execução deste processo. -A tabela de processos não é necessária em um sistema que possui um único processo em execução porque este processo nunca será suspenso.

3. As versões iniciais do processador Pentium não podiam suportar um monitor de máquina virtual. Qual característica essencial é necessária para permitir que uma máquina seja "virtualizável"?

   Resp.: Para que um monitor de máquina virtual possa criar máquinas virtuais que são cópias exatas do hardware, este deve ser capaz de interceptar todas as instruções privilegiadas do processador, que somente podem ser executadas no modo supervisor (como as que acessam os dispositivos físicos do hardware). Logo, o processador deve ser capaz de gerar uma interrupção, que será tratada pelo monitor de máquina virtual, sempre que uma destas instruções for executada no modo usuário (os primeiros processadores Pentium ignoravam a execução destas instruções).

4. Suponha que você vá projetar uma arquitetura avançada de computador que faça a comutação de processos em hardware, em vez de ter interrupções. De que informações a CPU precisaria? Descreva como a comutação de processos por hardware poderia funcionar.

   Resp.: Como o processador deve somente executar a comutação dos processos, este precisará ter um registrador com um ponteiro para a entrada, na tabela de processos, do processo que está atualmente em execução. Ao comutar um processo, quando, por exemplo, um dispositivo terminar de executar uma operação de E/S, o processador, usando este registrador, copiará o estado corrente da máquina (o valor atual dos registradores) para a entrada da tabela de processos do processo que acabou de ser suspenso. Depois copiará, para este registrador, o ponteiro para a entrada, na tabela de processos, do processo de serviço que trata daquele dispositivo (armazenado na entrada do vetor de interrupção deste dispositivo). Mais tarde, após o processo de serviço terminar a sua execução, o processo que foi suspenso poderá execu-

tar novamente, e o processador poderá restaurar, usando a entrada da tabela para este processo, o estado em que a máquina estava no momento da sua suspensão, o que permitirá reiniciar corretamente a sua execução.

5. Considere um computador que não tem uma instrução TEST AND SET LOCK mas tem uma instrução para comutar o conteúdo de um registrador e de uma palavra da memória em uma única ação indivisível. É possível utilizar isso para escrever uma rotina **ENTER_REGION** como a reproduzida a seguir (vista na Aula 5)?

```
ENTER_REGION:
    TSL REGISTER, LOCK
    CMP REGISTER, #0
    JNE ENTER_REGION
    RET
```

Resp.: Sim, pois podemos usar a variável LOCK como um sinalizador, com dois possíveis valores: 0, indicando que o processo pode executar a sua seção crítica, e 1, indicando que o processo não pode executar a sua seção crítica. Com isso, bastará que o registrador, no caso o REGISTER, seja igual a 1 antes de executarmos a instrução, que poderia se chamar, por exemplo, SWAP, como no código dado a seguir. Assim como antes, o processo, ao sair da seção crítica, deverá executar o procedimento **LEAVE_REGION**, que colocará um valor 0 no sinalizador, para permitir que outros processos possam executar a sua seção crítica.

```
ENTER_REGION:
    MOVE REGISTER, #1
    SWAP REGISTER, LOCK
    CMP REGISTER, #0
    JNE ENTER_REGION
    RET
```

6. Cinco programas estão esperando para serem executados. Seus tempos esperados de execução são 9, 6, 3, 5 e X. Em que ordem eles devem ser executados para minimizar o tempo médio de resposta? (Sua resposta

dependerá de X).

Resp.: Como vimos na Aula 6 sobre escalonadores, o algoritmo de escalonamento que garante que o tempo médio de resposta será o mínimo, quando conhecemos os tempos esperados de execução dos programas, é o escalonamento do trabalho mais curto primeiro. Como X é o único tempo que pode variar, poderemos ter as seguintes ordens de execução dos programas:

- Se $0 < X < 3$, a ordem será X, 3, 5, 6 e 9.
- Se $X = 3$, a ordem poderá ser X, 3, 5, 6 e 9, ou 3, X, 5, 6 e 9.
- Se $3 < X < 5$, a ordem será 3, X, 5, 6 e 9.
- Se $X = 5$, a ordem poderá ser 3, X, 5, 6 e 9, ou 3, 5, X, 6 e 9.
- Se $5 < X < 6$, a ordem será 3, 5, X, 6 e 9.
- Se $X = 6$, a ordem poderá ser 3, 5, X, 6 e 9, ou 3, 5, 6, X e 9.
- Se $6 < X < 9$, a ordem será 3, 5, 6, X e 9.
- Se $X = 9$, a ordem poderá ser 3, 5, 6, X e 9, ou 3, 5, 6, 9 e X.
- Se $X > 9$, a ordem será 3, 5, 6, 9 e X.