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
Gabarito da AD1 - Primeiro Semestre de 2013

**Atenção:** ADs enviadas pelo correio devem ser postadas cinco dias antes da data final de entrega estabelecida no calendário de entrega de ADs.

**Atenção**: Tem havido muita discussão sobre a importância de que cada aluno redija suas próprias respostas às questões da AD1. Os professores da disciplina, após refletirem sobre o assunto, decidiram o seguinte: Cada aluno é responsável por redigir suas próprias respostas. Provas iguais umas às outras terão suas notas diminuídas. As diminuições nas notas ocorrerão em proporção à similaridade entre as respostas. Exemplo: Três alunos que respondam identicamente a uma mesma questão terão, cada um, 1/3 dos pontos daquela questão.

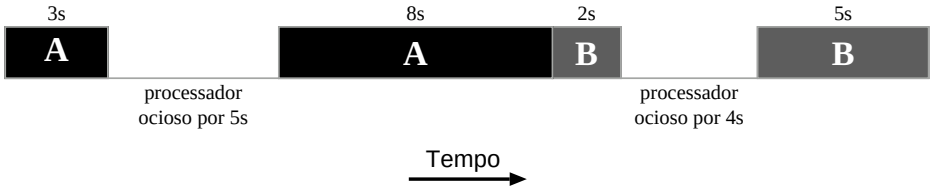
Nome -
Assinatura -

---

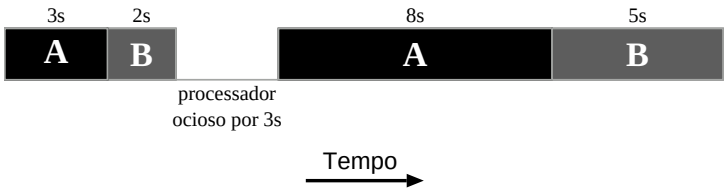
1. (1,5) Suponha que somente dois programas, A e B, estejam em execução no processador do computador. O programa A foi o primeiro a executar no processador: executou por 11s, tendo precisado fazer uma operação de E/S, com duração de 5s, após os primeiros 3s de execução. O programa B, que executou por 7s, também precisou fazer uma operação de

E/S, com duração de 4s, após os primeiros 2s de execução. Se o sistema operacional não usar a multiprogramação, qual será o tempo de ociosidade do processador? Agora, se o sistema usar a multiprogramação, o processador ficará ocioso? Justifique a sua resposta.

**Resp.:** -Se o sistema operacional não usar a multiprogramação, então não poderemos executar um outro programa quando o programa em execução fizer uma operação de E/S. Com isso, o processador ficará ocioso durante a execução dessa operação de E/S. As execuções dos programas A e B no processador serão como na figura a seguir. Como o processador ficará ocioso durante a execução de cada operação de E/S, então o tempo de ociosidade do processador será a soma dos tempos das operações de E/S executadas por A e B, isto é, $5\text{s} + 4\text{s} = 9\text{s}$.




-Agora, quando o sistema operacional usar a multiprogramação, o tempo durante o qual o processador ficar ocioso poderá ser usado para executar outros programas. Nesse caso, poderemos usar o tempo de execução da operação de E/S do programa A para executar o programa B, e o tempo da operação de E/S de B para executar A. Logo, a nova ordem de execução de A e B no processador será a dada pela figura a seguir. Isso ocorrerá porque o tempo da operação de E/S feita por A é maior em 3s do que o tempo que B executa antes de fazer a sua operação de E/S, e o tempo restante de execução da operação de E/S feita por B é suficiente para que A termine a sua execução. Pela figura, vemos que o processador ainda ficará ocioso por 3s, mas isso somente ocorrerá porque não existem outros programas para serem executados no processador.




2. (1,5) Suponha que um computador esteja limitado a executar no máximo 2 500 chamadas ao sistema operacional por segundo, e que uma chamada ao sistema requeira 1 250 instruções, incluindo a de TRAP e todas as necessárias à troca de contexto. Quantas instruções o computador poderá executar por segundo se três quartos da capacidade do processador forem usados para executar códigos do usuário? Justifique a sua resposta.

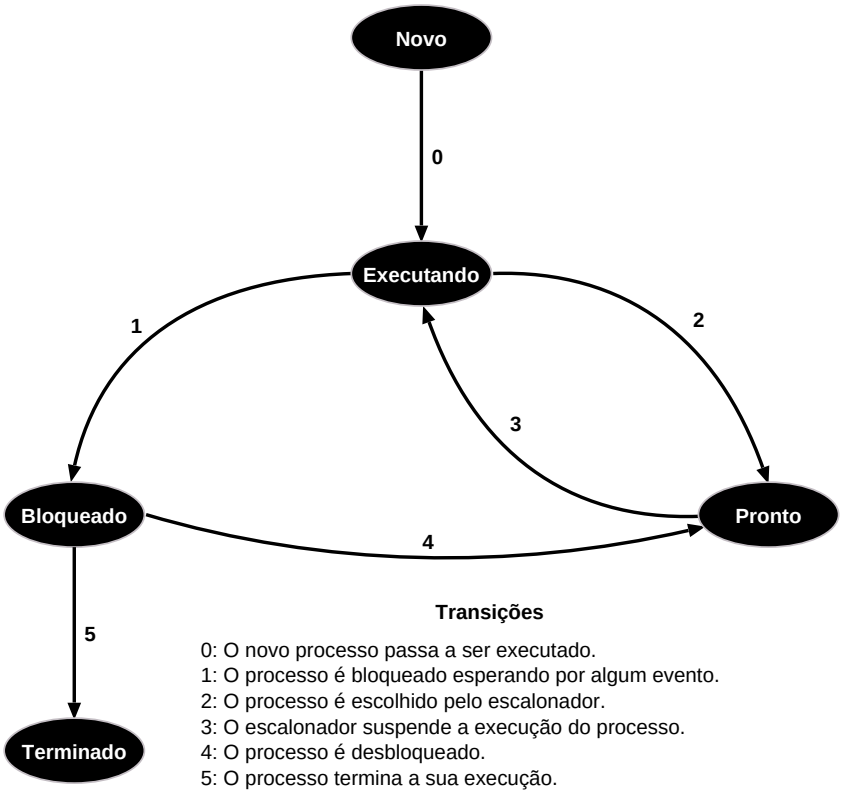
   **Resp.:** Para executar 2 500 chamadas ao sistema operacional, o computador precisa usar $2\,500 \times 1\,250 = 3\,125\,000$ instruções. Agora, como três quartos da capacidade do processador estão disponíveis para executar códigos de aplicação, então um quarto da capacidade está disponível para executar as chamadas ao sistema. Logo, se chamarmos de $x$ o número total de instruções que o computador pode executar por segundo, então temos que $\frac{1}{4}x = 3\,125\,000$, ou seja, $x$ é igual a 12 500 000.

3. (2,0) Suponha que o sistema operacional esteja executando diretamente sobre o hardware do computador, cujas operações de E/S demoram 2,5ms. Suponha ainda que um processo tenha executado por 11s e que, durante a sua execução, tenha feito 2 700 operações de E/S. Se o sistema operacional agora executar sobre uma máquina virtual, que reduz a velocidade do processador em 15% e a velocidade das operações de E/S em 80%, quantas operações de E/S o programa poderá fazer para o seu tempo de execução ainda ser de 11s? Justifique a sua resposta.

   **Resp.:** Como cada operação de E/S demora 2,5ms e como o processo fez 2 700 operações, então 6 750ms do tempo de execução de 11 000ms desse processo foram gastos em operações de E/S, quando ele executou no sistema operacional sobre o hardware do computador. Logo, o processo executou no processador do hardware por $11\,000 - 6\,750 =$

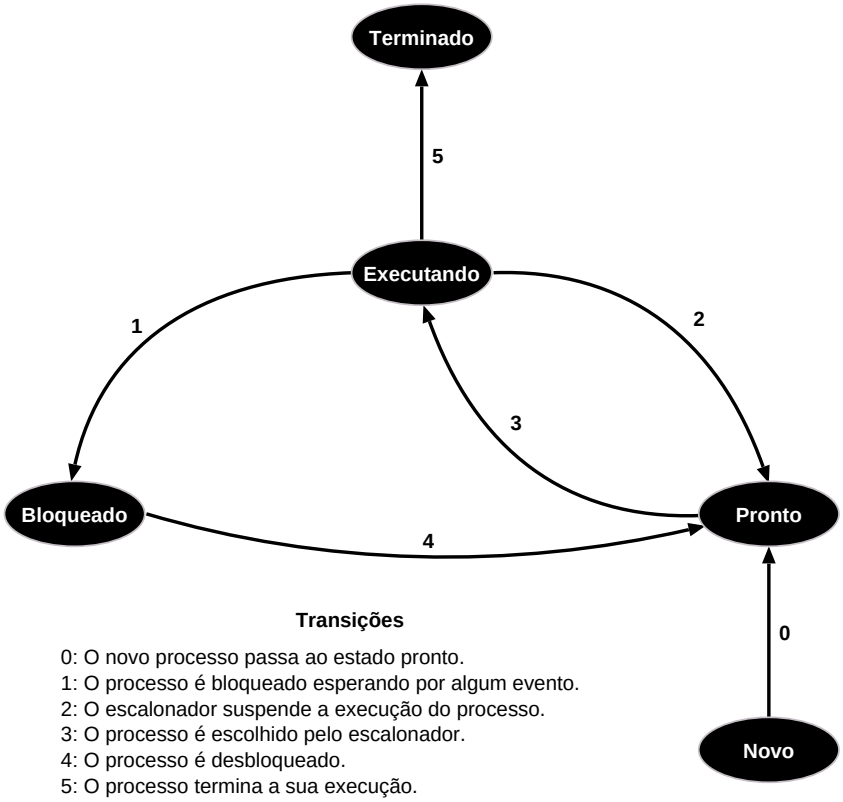
4 250ms. Note que a velocidade do processador ser reduzida em 15% significa que a velocidade do processador virtual é 85% da velocidade do processador real, o que por sua vez significa, durante os 4 250ms, que somente 85% das instruções serão executadas. Com isso, quando o processo executar no sistema operacional sobre a máquina virtual, o tempo de execução dele no processador virtual será de $4\,250/0,85 = 5\,000$ms. Agora o processo, para executar no mesmo tempo de 11 000ms, somente poderá executar operações de E/S por 6 000ms. Como o tempo gasto por cada operação de E/S na máquina virtual é de $2,5/0,2 = 12,5$ms, pois ela reduz a velocidade das operações de E/S em 80%, então o processo deverá executar $6\,000/12,5 = 480$ operações de E/S.

4. (1,5) Na figura dada a seguir mostramos uma versão estendida do diagrama de transição dos estados de um processo, com dois novos estados: o estado **Novo**, em que o processo é colocado quando é criado, e o estado **Terminado**, em que o processo é colocado quando termina a sua execução. O diagrama está correto? Justifique a sua resposta.




**Resp.:** Não, existem diversos erros no diagrama dado na figura. Quando um processo é criado, isto é, está no estado **Novo**, ele deve passar ao estado **Pronto** para esperar pela sua vez de executar no processador. Isso é necessário porque o processo em execução, ou algum dos outros processos no estado **Pronto**, pode ser mais prioritário do que o processo recém-criado. Logo, a descrição da transição 0 está incorreta, e a transição deve sair do estado **Novo** para o estado **Pronto**. A transição do estado **Bloqueado** para o **Terminado** também está incorreta, pois somente um processo em execução pode terminar a sua execução. Quando o evento que bloqueou o processo ocorrer, ele deve ser passado ao estado **Pronto**, esperando para executar novamente no processador. Com isso, a transição 5 está incorreta, e ela deve sair do estado **Executando** para o estado **Terminado**. Finalmente, com base no diagrama que vimos na aula 4, vemos que as descrições das transições 2 e 3 foram trocadas, isto é, a descrição dada na transição 2 é a da 3, e a dada na 3 é a da 2. Na figura a seguir temos o diagrama

correto, com todas as correções descritas anteriormente.




5. (2,0) Suponha que dois processos, A e B, compartilhem uma pilha, usada para armazenar números. Suponha ainda que essa pilha, inicialmente vazia, possa armazenar até $n$ números, e que exista uma variável usada para contabilizar a quantidade de números armazenados nela. O processo A continuamente insere dois números na pilha. Já o processo B continuamente remove três números da pilha, calcula o produto desses números, e depois insere o resultado do produto na pilha. Como os semáforos binários devem ser usados para garantir a correta execução dos processos A e B? Justifique a sua resposta.

**Resp.:** Precisamos de três semáforos binários, *acesso*, $bloquearA$ e $bloquearB$. O semáforo *acesso* é usado para garantir o acesso exclusivo à pilha. Já o semáforo $bloquearA$ é usado para bloquear A caso o

tamanho da pilha seja $n-1$ ou $n$, pois nesse caso A não pode colocar dois elementos na pilha. Finalmente, o semáforo $bloquearB$ é usado para bloquear B caso a pilha possua menos do que três elementos. Como a pilha inicialmente está vazia e não está sendo usada, os valores dos semáforos binários, *acesso*, $bloquearA$ e $bloquearB$ são de, respectivamente, 1, 1 e 0. A seguir mostramos dois possíveis procedimentos que podem ser executados pelos processos A e B, sendo que o tamanho atual da pilha está na variável $t$. Os procedimentos da sua resposta não precisam ser exatamente iguais aos dados a seguir, mas precisam, para estarem corretos: (i) possuírem um laço infinito como os destes procedimentos; (ii) garantirem o acesso exclusivo à pilha; e (iii) garantirem que A não coloque números em uma pilha com $t > n-2$ elementos e que B não retire números de uma pilha com $t < 3$ elementos.

```
void ProcessoA()
{
  while (1)
  {
    P(acesso);
    if (t > n - 2)
    {
      V(acesso);
      P(bloquearA);
      P(acesso);
    }
    // Código para colocar os dois números na pilha.
    t = t + 2
    if (t > 2)
      V(bloquearB);
    V(acesso);
  }
}

void ProcessoB()
{
  while (1)
  {
    P(acesso);
    if (t < 3)
```

```
    {
      V(acesso);
      P(bloquearB);
      P(acesso);
    }
    // Código para remover três números da pilha.
    t = t − 3
    // Código para multiplicar os três números.
    // Código para colocar o resultado pilha.
    t = t + 1
    if (t < n − 1)
      V(bloquearA);
    V(acesso);
  }
}
```

6. (1,5) Suponha que um sistema operacional use o algoritmo de escalonamento por *round robin*, sendo que cada quantum equivale a três unidades de tempo. Suponha ainda que os processos tenham sido escalonados no processador na ordem ABABABAAAA, e que eles usem todos os seus quanta integralmente. Qual será a nova sequência de escalonamento se o sistema operacional agora usar o algoritmo por prioridades, sendo que um processo executa enquanto a sua prioridade, que é aumentada de 1 unidade a cada unidade de tempo, não é maior do que a de um outro processo, e que as prioridades iniciais dos processos A e B são, respectivamente, de 8 e 10?

   **Resp.:** Pela ordem de execução, vemos que o processo A executou por 7 quanta e o processo B por 3 quanta. Agora, como cada quantum corresponde a três unidades de tempo, e como os processos A e B usaram todos os seus quanta, então A e B executaram por, respectivamente, 21 e 9 unidades de tempo no processador. As três tabelas dadas a seguir mostram a ordem de execução se agora usarmos o algoritmo por prioridades, sendo que cada processo executa até terminar ou até algum outro processo possuir prioridade menor, e supondo que a prioridade é aumentada de uma unidade a cada unidade de tempo. Para cada tabela, a primeira linha mostra a passagem do tempo. A segunda linha mostra os processos executados em cada unidade de tempo. Fi-

nalmente, a terceira linha mostra, para o processo de cada coluna, a prioridade desse processo antes de ele executar na unidade de tempo dessa mesma coluna. Como podemos ver pelas tabelas, a sequência de escalonamento agora será AAABBAABBAABBAABBAABAAAAAA-AAAA.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| A | A | A | B | B | A | A | B | B | A |
| 8 | 9 | 10 | 10 | 11 | 11 | 12 | 12 | 13 | 13 |

| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|---|
| A | B | B | A | A | B | B | A | A | B |
| 14 | 14 | 15 | 15 | 16 | 16 | 17 | 17 | 18 | 18 |

| 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
|---|---|---|---|---|---|---|---|---|---|
| A | A | A | A | A | A | A | A | A | A |
| 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |