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

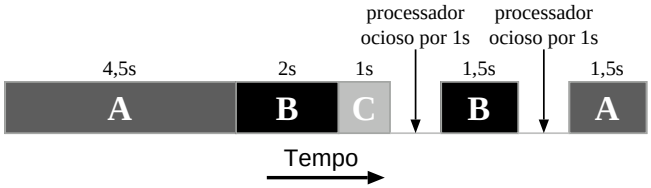
Quarto Período
Gabarito da AD1 - Primeiro Semestre de 2016

**Atenção**: 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 três programas, A, B e C, tenham executado em um sistema operacional de acordo com a figura a seguir, e que esse sistema use a multiprogramação somente para evitar ociosidade do processador durante as operações de E/S. Sabendo que A e B fazem somente uma operação de E/S cada, e que C não faz operações de E/S, responda, justificando a sua resposta:




(a) (1,0) Qual será a ociosidade do processador se C agora executar antes de A e de B?

**Resp.:** Se C executar antes de A e de B, então a execução de C não poderá ser usada para evitar parte da ociosidade, de 1s, ocorrida após B ter iniciado a sua operação de E/S. Logo, o tempo de ociosidade agora será de 2s (o tempo de execução da operação de E/S de B, pois B volta a executar antes de A voltar, cuja operação de E/S ainda não terminou) mais a ociosidade original de 1s após B terminar a sua execução (e antes de A poder executar novamente devido a sua operação de E/S ter terminado), ou seja, a ociosidade será agora de 3s.

(b) (0,5) Qual será a ociosidade do processador se a multiprogramação não for mais usada?

**Resp.:** Se a multiprogramação não for mais usada, então o tempo de ociosidade do processador será a soma dos tempos das operações de E/S de A e de B, pois C não faz operações de E/S. Como o tempo da operação de E/S de A ou de B é igual ao tempo entre o final da sua primeira execução e o início da sua última execução, já que somente uma operação de E/S é feita por A e por B, então os tempos das operações de E/S de A e B são de, respectivamente, 6,5s e 2s. Logo, o tempo total de ociosidade será agora de 8,5s.

2. (1,5) Qual é o número de instruções requeridas por cada chamada ao sistema operacional (incluindo a instrução TRAP e todas as outras instruções necessárias à troca de contexto) se o computador no qual o sistema executa está limitado a fazer no máximo 4 000 chamadas ao sistema por segundo, e se o processador pode executar 3 500 000 instruções por segundo, sendo que 4/5 da capacidade do processador são

usados exclusivamente para executar códigos do usuário? Justifique a sua resposta.

**Resp.:** Se denotarmos por $x$ o número de instruções requeridas por cada chamada, então o processador vai executar no máximo $4\,000x$ instruções por segundo ao executar no máximo 4 000 chamadas ao sistema por segundo. Agora, como 4/5 da capacidade do processador estão disponíveis para executar códigos do usuário, então 1/5 da capacidade está disponível para executar as chamadas ao sistema, ou seja, o processador pode executar $3\,500\,000/5 = 700\,000$ instruções por segundo para tratar as chamadas. Logo, temos que $4\,000x = 700\,000$ e, portanto, $x = 175$, significando que cada chamada usa 175 instruções do processador.

3. (2,0) Suponha que o sistema operacional esteja executando diretamente sobre o hardware de um computador cujas operações de E/S demorem $t$ ms. Suponha ainda que um processo tenha executado por 6s e que, durante a sua execução, tenha feito 5 000 operações de E/S. Se o sistema operacional agora executar sobre uma máquina virtual que reduza a velocidade do processador em 35% e a velocidade das operações de E/S em 50%, qual será o valor de $t$ se o processo executar na máquina virtual por 12s segundos e fizer agora 4 000 operações de E/S? Justifique a sua resposta.

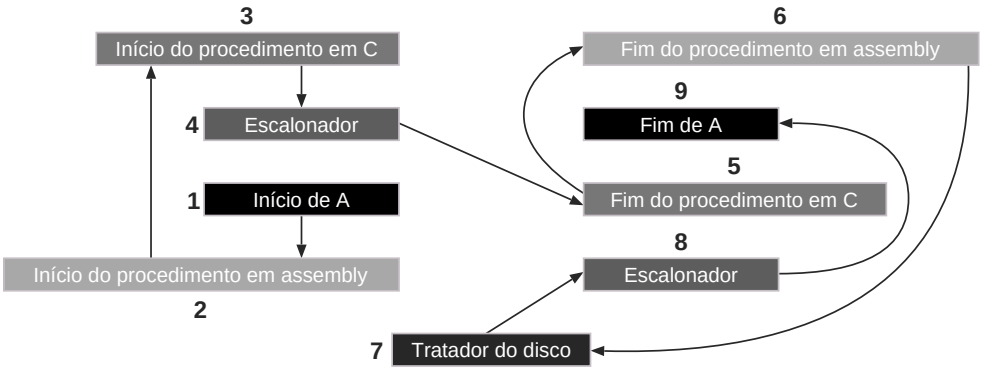
**Resp.: A resposta a seguir está dada para o caso em que a velocidade do processdor é reduzida em 75%, já que não é possível resolver para uma redução de 35%**. Como o tempo total de execução é de 6s ou 6 000 ms, e como o processo faz 5 000 operações de E/S, então $5\,000t$ ms do tempo de execução de 6000 ms desse processo são gastos com operações de E/S, quando ele executa no sistema operacional sobre o hardware do computador. Logo, o processo executa no processador do hardware por $6\,000 - 5\,000t$ ms. Note que a velocidade do processador ser reduzida em 75% significa que a velocidade do processador virtual é 25% da velocidade do processador real, o que por sua vez significa que, durante os $6\,000 - 5\,000t$ ms, somente 25% das instruções são executadas. Com isso, quando o processo executa no sis-

tema operacional sobre a máquina virtual, o tempo de execução dele no processador virtual é de $\frac{6\,000-5\,000t}{0,25}$ ms. Agora, como o processo executou 4 000 operações de E/S na máquina virtual, e como o novo tempo de cada operação de E/S é de $\frac{t}{0,5} = 2t$ ms (porque, similarmente à redução do tempo do processador, a redução da velocidade de cada operação de E/S em 50% significa que no mesmo tempo podemos, na máquina virtual, executar somente 50% das operações de E/S originais), então $4000 \times 2t = 8000t$ ms dos 12s ou 12 000 ms do tempo de execução do processo na máquina virtual foram gastos com E/S. Logo, o tempo de execução do processo no processador virtual é de $12\,000 - 8000t$ ms e, com isso, podemos concluir que $\frac{6\,000-5\,000t}{0,25} = 12\,000 - 8000t$, ou seja, que o tempo da operação de E/S $t$ no hardware real é de 1 ms.

4. (1,5) Suponha que tenha ocorrido uma interrupção do disco enquanto um processo A estava em execução. Mostre quais setas precisam ser adicionadas entre as ações dadas na figura a seguir, de tal modo que a figura resultante dê a ordem correta em que essas ações foram feitas quando a interrupção foi tratada pelo sistema operacional.


**Resp.:** Como vimos na transparência 21 da aula 4, o processo A precisa estar executando quando a interrupção do disco ocorre, porque os procedimentos em assembly e em C, assim como o escalonador são executados exatamente para suspender A quando a interrupção ocorre para, depois disso, executar o tratador do disco. Além disso, o escalonador deve ser chamado depois do tratador do disco, para poder reiniciar A depois de a interrupção ser tratada. O diagrama correto é

o dado a seguir:




5. (2,0) Suponha que uma pilha tenha inicialmente $e$ elementos, e que ela possa armazenar até $n$ elementos. Suponha ainda que a pilha ofereça três operações para os processos que desejem acessá-la: **Remover**, para remover o elemento do topo da pilha; **Inserir**, para colocar um novo elemento no topo da pilha; e **Buscar**, para verificar se um elemento está na pilha. Como três semáforos, um binário e dois de contagem, podem ser usados para garantir que essas operações não levem a condições de corrida? Justifique a sua resposta.

**Resp.:** A seguir mostramos como um semáforo binário e dois de contagem podem ser usados para implementar as funções. O semáforo binário, chamado *acesso*, é usado para garantir o acesso exclusivo à pilha. O primeiro semáforo de contagem, chamado *vazia*, conta o número de entradas não usadas na pilha, e é usado para bloquear o processo que executar a operação **Inserir** quando a pilha estiver cheia. Finalmente, o segundo semáforo de contagem, chamado *cheia*, conta o número de entradas usadas na pilha, e é usado para bloquear o processo que executar a operação **Remover** quando a pilha estiver vazia. Como inicialmente a pilha tem $e$ elementos e não está sendo usada, então os semáforos *vazia*, *cheia* e *acesso* são inicializados, respectivamente, com $n-e$, $e$ e 1. A seguir mostramos os códigos para as operações **Remover**, **Inserir** e **Buscar**, sendo que a função *removetopo*() remove e

retorna o elemento no topo da pilha, a função $inseretopo(a)$ insere o elemento $a$ no topo da pilha, e a função $buscaelemento(a)$ procura o elemento $a$ na pilha, retornando $true$ se ele for achado e $false$ em caso contrário:

```
elemento Remover()
{
  // Função que implementa a operação Remover
  // Garante que existe pelo menos um elemento na pilha.
  P(cheia);
  // Garante o acesso exclusivo à pilha.
  P(acesso);
  // Remove o elemento do topo da pilha e o coloca em a.
  a = removetopo();
  // Libera o acesso exclusivo à pilha.
  V(acesso);
  // Usa a operação V sobre vazia para registrar que um
  // elemento foi removido da pilha.
  V(vazia);
  // Termina a função retornando a.
  return(a);
}

void Inserir(elemento a)
{
  // Função que implementa a operação Inserir
  // Garante que existe pelo menos um espaço vazio na pilha.
  P(vazia);
  // Garante o acesso exclusivo à pilha.
  P(acesso);
  // Insere a no topo da pilha.
  inseretopo(a);
  // Libera o acesso exclusivo à pilha.
  V(acesso);
  // Usa a operação V sobre cheia para registrar que um
  // elemento foi inserido na pilha.
  V(cheia);
}
```

```
bool Buscar(elemento a)
{
  // Função que implementa a operação Buscar
  // Garante o acesso exclusivo à pilha.
  P(acesso);
  // Procura a na pilha.
  result = buscaelemento(a);
  // Libera o acesso exclusivo à pilha.
  V(acesso);
  return(result);
}
```

6. (1,5) Suponha que um sistema operacional use o algoritmo de escalonamento por *round robin*, sendo que cada quantum equivale a quatro unidades de tempo. Suponha ainda que os processos tenham sido escalonados no processador na ordem ABCABCABCABABAAAA, 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 reduzida de 2 unidades a cada cinco unidades de tempo, não é menor do que a de um outro processo, e que as prioridades iniciais dos processos A, B e C são, respectivamente, de 28, 14 e 20? Justifique a sua resposta.

   **Resp.:** Devido a um empate nas prioridades, temos duas respostas, mostradas nas tabelas a seguir. Nessas tabelas mostramos, na primeira linha, o tempo antes de o processo da coluna ter sido executado. Na segunda linha mostramos, da esquerda para a direita, a ordem de execução dos processos no processador. Finalmente, na última linha mostramos, para cada coluna, a prioridade do processo antes de ele executar no processador no tempo dado na mesma coluna. Pelas duas tabelas, vemos que nova sequência de escalonamento é AAAAACCAACBABBB ou AAAAACCAACABBBB.

| 0 | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 47 | 52 | 53 | 58 | 63 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | A | A | A | A | C | C | A | A | C | B | A | B | B | B |
| 28 | 26 | 24 | 22 | 20 | 20 | 18 | 18 | 16 | 16 | 14 | 14 | 12 | 10 | 8 |

| 0 | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 47 | 48 | 53 | 58 | 63 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | A | A | A | A | C | C | A | A | C | A | B | B | B | B |
| 28 | 26 | 24 | 22 | 20 | 20 | 18 | 18 | 16 | 16 | 14 | 14 | 12 | 10 | 8 |