Gabarito da prova 3

1 a) a[i] = a[i] + a[i-1];

1 b) a i 4 * [] a i 4 * [] a i 1 – 4 * [] + :=

1 c) Propriedade algébrica: (i-1)*4 equivale a i*4 – 4; eliminação de sub-expressões comuns: i*4.

1 d) Se arranjo tem base 1, cada acesso a um de seus elementos é precedido por uma subtração (i-1)

2 a) 12 KiBytes (8 KiB para os quadros, 4 KiB para a tabela de páginas)

2 b) Página 9; endereço físico 0x129E

2 c) Página 2

2 d) Página 1

3 a) P_A às 19h03m30s, P_B às 19h06m00s e P_C às 19h04m30s

3 b) P_A às 19h09m, P_B às 19h12m e P_C às 19h08m

3 c) P_A às 19h06m, P_B às 19h05m e P_C às 19h03m

3 d) Processo de menor prioridade entra na região crítica e perde a CPU (pelo escalonador, para entrada de processo de maior prioridade) antes de concluí-la. Processo de maior prioridade tenta entrar na região crítica e entra em espera ocupada. Não perde a CPU e nenhum dos dois processos pode avançar. Exemplos: P_A na região crítica, P_B ou P_C em espera ocupada; ou P_B na região crítica, P_C em espera ocupada.

4 a) Bloqueio mútuo (*deadlock*). Exemplo: Processo 0 executa entra(0), faz flag[0]=true e perde CPU (escalonador). Processo 1 executa entra(1), faz flag[1]=true e fica em espera ocupada (flag[0] é true). Quando Processo 0 volta a executar, também entra em espera ocupada (flag[1] é true). Nenhum dos dois processos consegue entrar na região crítica.

4 b) Quando o sistema operacional não suportar semáforos e a linguagem de programação não suportar monitores.

4 c) Usuário não tem como garantir atomicidade na execução das operações *down* e *up*.

4 d) Não se as threads estiverem num mesmo processo – o processo como um todo será bloqueado e a outra thread não terá como executar.