**Próximos Passos e Melhorias com Base no PDF T2-A:**

1. **Ajuste da Inicialização do GM em Sistema.java**:
   * A página (tamPag) não pode ser igual ao tamanho total da memória (tamMem). Precisamos definir um valor razoável para tamPag, por exemplo, 10 ou 16.
   * **Ação**: Modificar o construtor do GM em Sistema.java para gm = new GM(hw.mem, 10); ou outro valor apropriado para o tamanho da página.
2. **Implementação do Modelo de Três Estados no Scheduler e GP**:
   * O GP.PCB.State enum já existe com BLOCKED, READY, RUNNING.
   * **Ação**: Modificar a lógica do Scheduler.schedule():
     + Quando um processo é escolhido para execução, seu estado deve ser RUNNING.
     + Quando uma SYSCALL de I/O ocorre, o processo atual deve ser movido para BLOCKED e adicionado à Fila Bloqueados. Um novo processo da Fila Prontos deve ser escalonado.
     + Quando uma interrupção de I/O de um dispositivo ocorre, o processo correspondente (que está na Fila Bloqueados) deve ser movido para READY e adicionado à Fila Prontos.
     + Quando um STOP ocorre ou uma interrupção fatal (endereço inválido, instrução inválida, overflow) acontece, o processo deve ser finalizado (removido das filas, desalocado). Atualmente, ele é colocado de volta na fila de prontos, o que é incorreto.
3. **Tratamento de Contexto no Scheduler**:
   * **Ação**: Salvar os registradores da CPU (hw.cpu.reg) no PCB (chosenPCB.regs) antes de o processo ser preemptado e colocado de volta na fila (q.add(chosenPCB)).
   * **Ação**: Carregar os registradores do PCB (chosenPCB.regs) na CPU (hw.cpu.reg) quando o processo é escolhido para execução. Isso já está parcialmente feito.
4. **Implementação de Threads (CPU, Console/I/O, Shell)**:
   * **CPU Thread**: A CPU.run() deve ser chamada dentro de uma thread separada. Precisaremos de mecanismos de sincronização (semáforos ou wait/notify) para que o escalonador possa "liberar" a CPU para executar e a CPU possa "avisar" o escalonador quando uma fatia de tempo termina ou uma interrupção ocorre.
   * **Console/I/O Thread**:
     + Crie uma nova classe para o dispositivo de I/O (ex: ConsoleDevice).
     + Essa classe deve ter uma fila de pedidos de I/O (ex: BlockingQueue<IORequest>).
     + A thread do ConsoleDevice ficará em loop, consumindo pedidos dessa fila.
     + Ao finalizar um pedido, ela deve simular uma interrupção de I/O para a CPU.
   * **Shell Thread**: A classe Sistema.menu() deve ser executada em uma thread separada, permitindo que o usuário interaja enquanto o sistema está processando.
5. **Melhoria do SysCallHandling**:
   * **Ação**: Modificar SysCallHandling.handle() para:
     + Para SYSCALL de IN (leitura, reg[8] == 1), o processo atual deve ser bloqueado. O pedido de I/O (incluindo o endereço de memória reg[9]) deve ser adicionado à fila de pedidos do dispositivo de console.
     + Para SYSCALL de OUT (escrita, reg[8] == 2), o processo atual deve ser bloqueado. O pedido de I/O (incluindo o valor e o endereço) deve ser adicionado à fila de pedidos do dispositivo de console. A saída real deve ser feita pela thread do console.
     + Após adicionar o pedido à fila, a SysCallHandling deve de alguma forma avisar o escalonador para bloquear o processo e escolher outro.
6. **Novos Tipos de Interrupção**:
   * **Ação**: Adicionar um novo tipo de interrupção em Interrupts.java para "intIOCompleta" (ou similar) para sinalizar a conclusão de uma operação de I/O.
   * **Ação**: Modificar InterruptHandling.handle() para diferenciar os tipos de interrupção:
     + Para intIOCompleta, o processo bloqueado deve ser movido para a fila de prontos.
     + Para interrupções fatais (endereço inválido, instrução inválida, overflow), o processo deve ser finalizado.
   * **Interrupção do Relógio (Timer)**: Se você for implementar preempção por tempo, precisará de uma nova interrupção para o timer e uma rotina de tratamento para ela.
7. **Simulação de DMA**:
   * Quando a thread do console processar um pedido de IN, ela deve ler um valor (ex: do usuário) e escrevê-lo diretamente na memory.pos[tradutor(enderecoLogico, tabelaPaginas)].
   * Quando processar um pedido de OUT, ela deve ler o valor de memory.pos[tradutor(enderecoLogico, tabelaPaginas)] e exibi-lo.