Nessa segunda parte do projeto da máquina virtual, começamos a implementar conceitos de particionamento em nossa memória de 1024 posições, algo que mais além veio a se tornar um gerente de memória capaz de ver quais partições estão livres para carregar um determinado programa, já calculando o offset e o endereço limite da partição, algo que será primordial quando o programa for processado.

Outro conceito fundamental para conseguirmos implementar um sistema de time-slice é o process control block (PCB), que por sua vez é um bloco de informações de controle a respeito de um processo. Cada programa terá seu PCB com seu process id, estado atual, program counter e seus próprios registradores. O gerente de processo é responsável por criar os PCBs, pois logo em seguida cria-se uma fila de execução com eles, para que cada processo utilize a CPU por uma fatia de tempo e volte para o final da fila.

Nossa maior dificuldade foi em modularizar o projeto, pois como antigamente tínhamos foco total na CPU, agora teremos que separar cada componente individualmente e fazer com que se comuniquem de forma lógica e eficiente. Outra questão eminente é que o sistema aparentava não ter estrutura suficiente para inicializar e unificar os componentes.

Pensando nisso, resolvamos criar a classe Sistema Operacional que ficará responsável por inicializar cada componente implementado, ordenando assim o gerente de memória a carregar os programas em Assembly para que as outras etapas tomem continuidade.

Quando é iniciado a execução em time-slice, que foi definido 10 instruções como fatia de tempo, a CPU recebe por parâmetro um PCB onde estará salvo todas as informações necessárias para começar a execução ou retomar de onde havia sido pausada. Para facilitar o sistema em time-slice, criamos 4 estados para um processo, sendo READY pronto para executar, RUNNING quando a CPU está computando as instruções, WAITING quando terminou a fatia de tempo e o processo volta para o final da fila e FINISHED quando o processo acaba.

Nossa máquina virtual aceita somente 4 ou 8 partições para inicializar, já que devemos usar apenas números 2^n e porque estamos sempre carregando os 4 programas.

O gerente de memória carrega todos os programas em partições aleatórias não utilizadas ainda, ou seja, sempre que o sistema operacional for inicializado ele será independente para lidar com qualquer processo em qualquer partição.

Ao final da execução de todos os processos, é printado na tela o valor final dos 8 registradores de cada PCB, além das posições da memória que foram utilizadas (programa e dados) em sua determinada partição.