Demonstrating VM and Paging concepts using the simulator

Memory management essentially is about three main concepts: placement (where in memory data and code are placed), replacement (which part(s) of a full memory to swap out or to overwrite) and address translation (from logical to physical addresses). The simulator can demonstrate these concepts.

You can view the main physical memory (i.e. RAM) by clicking on the **VIEW MEMORY...** button under the **Views** tab. This should display the main memory window. At the bottom of the window there are two frames titled **Manual** and **Memory Management**. The **Manual** section allows you to manually allocate and de-allocate memory. In the **Memory Management** section placement policies can be selected. By using the **ADD** and **REMOVE** buttons and the placement policies you can demonstrate how these policies work. You need to experiment to get to know how these work. First use the **ADD** button to allocate memory frames of various sizes then use the **REMOVE** button to create 'holes' of desired sizes.

In the **Memory Management** section you can see three check boxes. If none is checked, then this simulates a system that has no VM implementation. In this case a process requiring memory size greater than the available free memory size will not be created and run. If you check the **VM Required** check box and do the same then you'll see that the process will be created but its memory will be swapped out. When you run in Round Robin fashion then you'll notice that the memory will be swapped in and out as the processes change state from running to ready state. This then demonstrates the importance of VM. It allows running many processes requiring memory larger than the physical memory available.

The **Paging Enabled** check box simulates paged memory management. If this is unchecked, then the process will need to have all its memory to occupy contiguous memory space; if no such space then the OS will swap out some other process's memory to make room even though there may be enough total free space available in the fragmented memory. So this essentially simulates segmentation without paging. With paging enabled, the memory pages of the process do not need to occupy contiguous memory space; pages of the memory can be anywhere there is free space. This then demonstrates the advantages of paging.

Demonstrating VM and Paging concepts using the simulator

The **Demand Paging** check box is used to simulate 'working set' concept and works in conjunction with paging. If you check this box, then not all pages of a process will be required to be in the memory at the same time. The simulator assumes that a minimum of one page will suffice in this case. Of course, this is not very realistic as some process' working sets can be larger than one page. However, what is demonstrated here is just the concept. The details can be the subject of a class discussion.

To demonstrate VM and paging it is best to use a program. You create this using the compiler. Below is a simple source you can use for this:

```
program MemTest
  for n = 1 to 40
     i = 1
  next
  read(k)
end
```

This program simply loops 40 times doing a silly assignment but this is not important. It then waits for an input. The input is there to force the program to be suspended in the waiting queue as we do not want to terminate once it completes the loop.

When you create a process using the **CREATE NEW PROCESS** button in the OS window you can also specify the number of data pages a process should be allocated to. By using different sizes, you can clearly see the VM activity as the process pages are swapped in and out (visibility of this is aided by assigning different colours for process pages in the main memory view). You can also view a process's data pages by using one of the **SHOW MEMORY...** buttons. In the process memory view, you can use the **SHOW PAGE TABLE...** button to see the page table information. Here you can see the count of page faults against each page. So, you can run several instances of the above code with different memory sizes and when all instances are in waiting state (i.e. waiting on the read statement) then you get the opportunity to inspect the page fault counts before the processes die and their memories disappear. You can then use the total page faults as a measure of the frequency of page swaps.

Demonstrating VM and Paging concepts using the simulator

This should give some measure of the merits of different VM strategies. However, in order to fine tune this and demonstrate it to the students in a clear way requires some experimentation.