Assignment 2

Memory Virtualization

In this assignment, we will learn about memory virtualization using page tables. You will be implementing single level page tables, and an API to simulate the memory. You are given 200 MB RAM, to be used by OS to create and manage user-processes. The first portion of the RAM will be used by the OS and the remaining will be assigned to processes by OS.
The OS should support creation, stopping and forking of processes (only the memory management part of it). Each process has 4 MB of virtual memory (The layout is given in Figure 1), and the page size is set to be 4KB. The OS should flexibly map the virtual pages of processes to the physical frames.

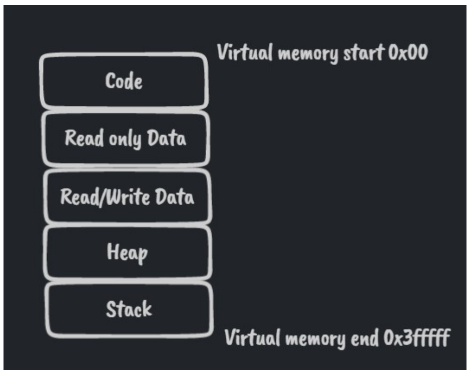

Figure 1: Process Virtual Memory Layout

You should implement the following functions: (A detailed description is given in the provided code files)

- create_ps(int code_size, int ro_data_size, int rw_data_size, int max_stack_size, unsigned char* code_and_ro_data) : This function should create a new process, allocate its virtual memory and initialize code and read only data segments.
- exit_ps(int pid) : This function should deallocate all the memory allocated to the process with the given pid. This pid can now be used for other new processes.
- fork_ps(int pid) : This function should create a new process that has identical memory with the process of given pid.
- allocate_pages(int pid, int vmem_addr, int num_pages, int flags) : This function should allocate num_pages starting at vnem_addr for a process with given pid. Set the protection bits of each page according to the flags. If any of the to be allocated pages was already allocated then this should kill the process. See comments in the code for more details.
- deallocate_pages(int pid, int vnem_addr,int num_pages) : This function should de-allocate num_pages starting at vnem_addr for a process with given pid. If any of the to be de-allocated pages was not allocated then this should kill the process. See comments in the code for more details.
- read_mem(int pid, int vmem_addr) : Read 1 byte at the given virtual address for the process with given pid, return it. In case of illegal access, kill the process.
- write_mem(int pid, int vmem_addr, unsigned char byte) :Write the byte at the given virtual address for the process with given pid. In case of illegal access, kill the process.

Other than these, there are helper functions to be implemented. See **TODOs** in code for more information

General Guidelines

- You should not use malloc to allocate new memory, you should stick to memory given in RAM.
- You can assume that the processes will not exhaust the process memory (128 MB).
- We have given you ample of memory for OS (72 MB).
- You are advised to create a free-list for managing the available physical frames

Think about which functions among the ones mentioned above, will be performed by OS software and which functions will be performed by hardware. In the assignment, we are only simulating the hardware, so everything is done by the software.
If interested, you can read about **Copy On Write** for optimizing fork. You don't need to do this for the assignment.

Submission Details

Submit a zip file named <Entry_No>_A2.zip (eg: 2019MT60648_A2.zip). Upon unzipping your submission, we should get a folder named <Entry_No>_A2 containing all your code files.