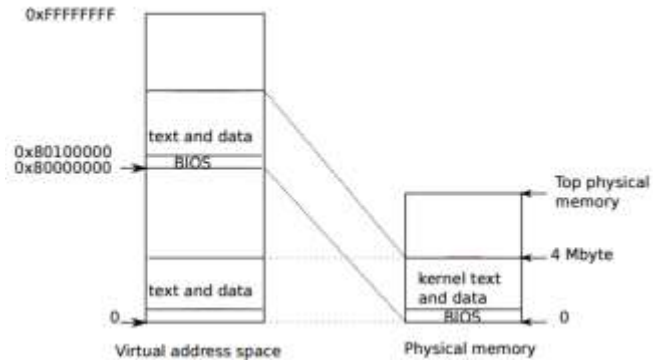
Yilin Yang

Lab 6

Ch. 1 & 2 Report

It is the operating system's responsibility to ensure that all processes make progress to completion, even if there are not enough hardware processors to support them. It must also ensure that the failure of a sole process does not impede others without strictly isolating the process as inter-process communication is still necessary. The xv6 operating system implements this through the use of page tables, assigning each process an address space or pseudo-private memory space. The page table maps virtual addresses to physical ones on chip. Every process possesses its own page table, which holds the kernel's instructions, data, and program memory.



When first powering on, the xv6 operating system loads the kernel from the disk into memory at a physical address of 0x100000. Addresses from 0xa0000 to 0x100000 are occupied by I/O devices. A page table is established that maps virtual addresses to physical ones. As a result of this, kernel instructions and data are limited to 4 MB. With a page table created, the first process, *userinit*, begins allocating processes to the page table using the process *allocproc*. *Allocproc* is called for every process while *userinit* is only called for the first one. *Allocproc* scans the page table for vacant slots marked as UNUSED and flags them as EMBRYO to mark them as used, assigning the process a PID and a kernel stack. In the event this assignment fails, the state is reverted from EMBRYO back to UNUSED. Otherwise, *userinit* will call on initcode.s to finalize the build procedure and set the process state to RUNNABLE. After *userinit* is first called, *scheduler* is then started to search for processes that are declared RUNNABLE. Initially, there is only *initproc* which satisfies this condition. *Initproc* alerts the kernel to begin using the page table. *Scheduler* runs the process, setting the process state to RUNNING and calls swtch to switch to the target process' kernel thread.

Each entry in a page table contains a 20-bit physical page number. Page tables map virtual addresses to page table entries using 20 bits, replacing an address' top 20 bits with the page table number from the page table entry. The 12 low bits from the virtual address are copied into physical address, becoming a unit of 4096 (2^{12}) bytes or a page. Each page table entry also contains flags identifying the state of the entry, such as PTE_P for present, PTE_W for writable, and so on.