Report :

I declared a large size global array (int arrGlobal[10000]) and I printed all page table entries with page directory entry number ,page table entry number along with virtual address , physical address using pgtPrint() syscall.Then I found out there are total 12 pages that are allocated. When I declare a large size local array (int arrLocal[10000]) , then the number of valid entries or number of pages allocated decreases (only 2 pages are allocated) ,if we try to initialise or write /access all numbers ,then pages allocated might increase. The reason is when we declared a global array , kernel allocated memory for all integers (10000) but in the case of local array it is different . The local array should be stored in stack , stack has limited size .

When I repeatedly execute the user program , I analysed the following points:

1) The Number of entries remain same.

2) Virtual address remains same.

3) Physical address might change , I observed that physical address assigned to process is changing , when I executed the program repeatedly.

I printed all pages tables with valid entry , by using `myproc()->pgdir` . This gives pointer to the page directory to the current process. As there are total **NPDENTRIES** (1024) (Number of page directory entries) . I iterated through all entries by checking whether it's valid entry and accessible by user. For every page directory it maps to specific page table (It has physical address to access and I converted it to virtual address to acces it) and it has **NPTENTRIES** (1024) (Number of page table entries) . Then I printed all page table entries which are valid and accessible by user.

First I will explain how I handled page fault by modifying trap function in trap.c file . I modified it to check if the trap corresponds to page fault by using trap number stored in trap frame of a process.
If it is equal to **T_PGFLT** (14) . Then I check whether the faulting address is valid address (present in range of virtual memory) using `myproc()->sz` , if faulting address is greater than process size it is invalid address ,then I kill the process and print the message as it was there before modifying. We get faulting address using rcr2( ) function. If it is vaild address , I try to assign a page using kalloc( ) and I map it to page directory using mappages( ) function . To use mappages function ,we need to declare it in trap.c file.

Initially I am not allocating all memory ,which is usually done by xv6 . It allocated all memory using allocuvm function . I allocated only memory for file size rather than memory size by modifying exec.c file.

Still due to this assignment ,I learned lot more things about virtual address , physical address and about allocating page . I learned to use lot more functions and I learned how lot of functions in xv6 like mappages( ), rcr2( ), allocuvm, kalloc( ) ,   memset( ) , PGROUNDUP , PGROUNDDOWN and lot more. I learned about elf format .