Class Assignment - 5

Name - Rajeev Kumar
ID - 12341700

Part 1: Add a Page Fault Counter in proc and Create Syscall getpagefaults()

- **Step 1 –Add a counter field page faults to struct proc.**

  File Name: proc.h
  Code Added:

```
int page_faults ;
```

- **Step 2 – Initialize it in allocproc() (proc.c). Write the line that sets page faults counter initially to zero.**

  File Name: proc.c
  Code Added:

  Inside allocproc() function :

```
p->page_faults=0;

Int
growproc(int n)
{
  uint sz;
  struct proc *curproc = myproc();
  sz = curproc->sz;

  if(n > 0){
      sz+=n;
  } else if(n < 0){
      sz = deallocuvm(curproc->pgdir, sz, sz + n);
  }
  curproc->sz = sz;
  switchuvm(curproc);
  return 0;
}
```

- **Step 3 : Implement the syscall to fetch page faults count**

File Name: `syscall.h`
Code Added:

```
#define SYS_getpagefaults 31
```

File Name: sysproc.c
Code Added:

```
uint sys_getpagefaults(void) {
    struct proc *p=myproc();
    return p->page_faults;
}
```

File Name: syscall.c
Code Added:

```
extern int sys_getpagefaults(void);
```

Inside syscalls[ ] Table :

```
[SYS_getpagefaults] sys_getpagefaults,
```

File Name: `user.h`
Code Added:

```
int getpagefaults(void);
```

File Name: `usys.S`
Code Added:

```
SYSCALL(getpagefaults)
```

File Name: `defs.h`
Code Added:

```
int             vmfault(pde_t* pgdir,uint va, int);
```

Part 2: Lazy Page Allocation

- **Step 1 – Modify the vmfault function to allocate pages lazily. The full code is provided below; study it carefully and understand its working:**

File Name: vm.c
Code Added:

```
int vmfault(pde_t *pgdir, uint va, int write) {
    struct proc *p = myproc();
    char *mem;
    pte_t *pte;

    if (va >= p->sz)
        return -1;

    va = PGROUNDDOWN(va);


    pte=walkpgdir(pgdir, (char *)(uint)va, 0);
    if(pte && (*pte & PTE_P))
      return 0;
    //if (walkpgdir(pgdir, (void *)va, 0))
     //   return 0;
    mem = kalloc();
    if (mem == 0)
        return -1;

    memset(mem, 0, PGSIZE);

    if (mappages(pgdir, (void*)(uint)va, PGSIZE, V2P(mem),
PTE_W | PTE_U | PTE_P) < 0) {
        kfree(mem);
        return -1;
    }
    return 0;
}
```

- **Step 2 : Modify the trap handler to increase page faults count and handle faults by calling vmfault(). Replace the relevant code section with:**

  File Name: `trap.c`
  Code Added:

```
case T_PGFLT:

      {

        struct proc *p = myproc();

        p->page_faults++;

        if (vmfault(p->pgdir, rcr2(), tf->err & 2) < 0)

          p->killed = 1;

      }

      Break
```

PART 3: User Programs to Measure Page Faults

- **Step 1 : tlbrun.c**

  File Name: tlbrun.c
  Code Added:

```
#include "types.h"
#include "stat.h"
#include "user.h"

#define PAGESIZE 4096
#define MAXPAGES 1024

int main() {
    int jump = PAGESIZE / sizeof(int);

    printf(1, "PageCount\tTrials\tTicks\tPageFaults\n");

    for (int numpages = 1; numpages <= MAXPAGES;
numpages *= 2) {
        int trials = 5000000;

        int faults_before = getpagefaults();
        int start = uptime();

        int *arr = (int*) sbrk(numpages * PAGESIZE);
        if (arr == (void*) -1)
            exit();

        for (int t = 0; t < trials; t++) {
            for (int i = 0; i < (numpages/2) * jump; i += jump) {
                // Access the page → trigger faults on first use
                arr[i] = t;
            }
        }
        int end = uptime();
        int faults_after = getpagefaults();
        printf(1, "%d\t%d\t%d\t%d\n",
               numpages, trials, end - start,
               faults_after - faults_before);
    }

    exit();
}
```

File Name: `tlbtest.c`
Code Added:

```
#include "types.h"
#include "stat.h"
#include "user.h"
#define PAGESIZE 4096
int main(int argc, char *argv[]) {
    if (argc < 3) {
        printf(1, "Usage: tlbtest <numpages> <trials>\n");
        exit();
    }
    int numpages = atoi(argv[1]);
    int trials = atoi(argv[2]);
    int jump = PAGESIZE / sizeof(int);
    int faults_before = getpagefaults();
    int start = uptime();
    int *arr = (int*) sbrk(numpages * PAGESIZE);
    if (arr == (void*) -1)
        exit();
    for (int t = 0; t < trials; t++) {
        for (int i = 0; i < (numpages/2) * jump; i += jump) {
            arr[i] = t;   // Access page
        }
    }
    int end = uptime();
    int faults_after = getpagefaults();
    printf(1, "%d\t%d\t%d\t%d\n",
          numpages, trials, end - start,
                faults_after - faults_before);
            exit();
        }
```

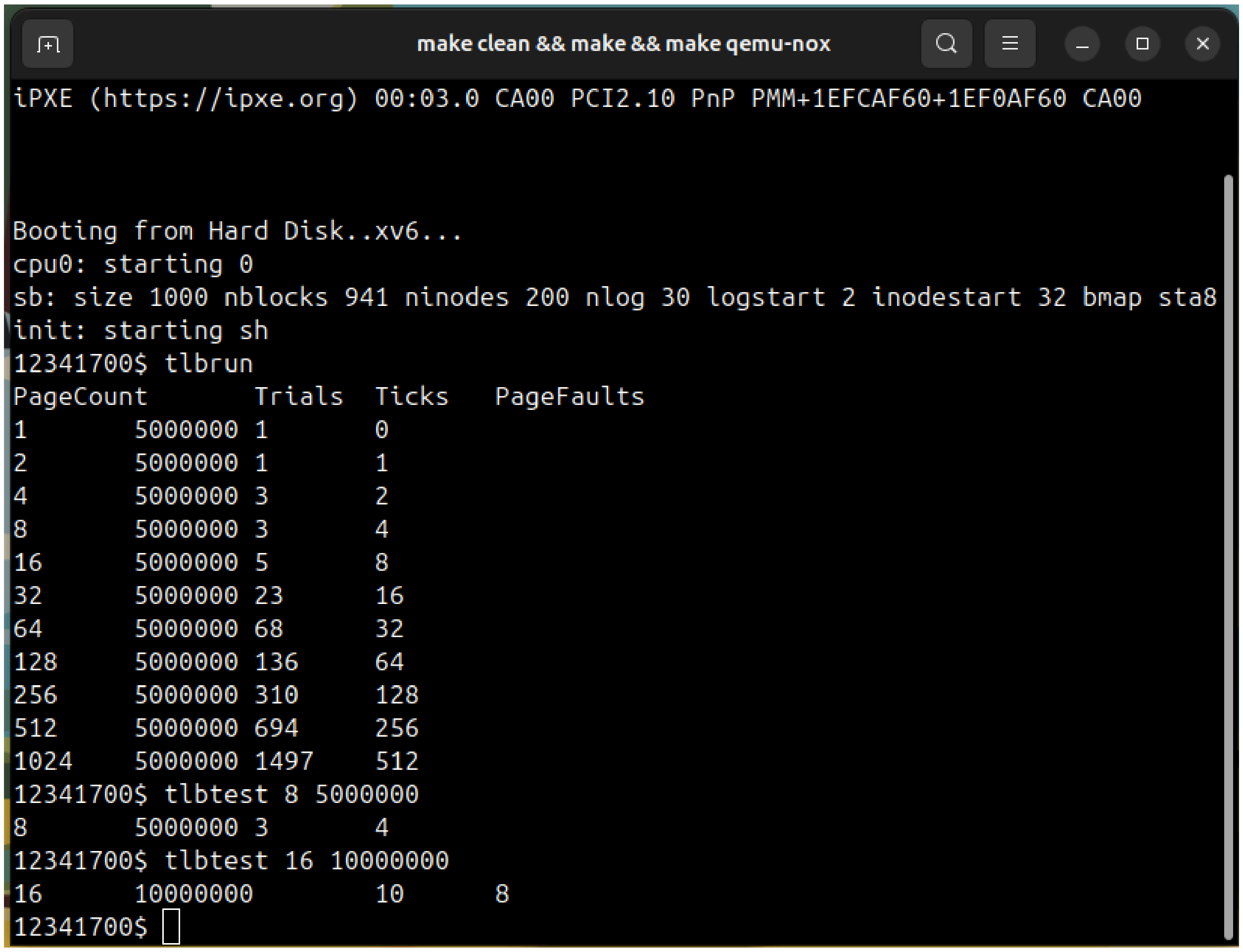
PART 4: Integration and Testing

- **Step 1 : Add programs to Makefile (UPROGS):**

  File Name: Makefile
  Code Added:

```
_tlbrun \
_tlbtest \
```

- **OUTPUT :**



```
iPXE (https://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1EFCAF60+1EF0AF60 CA00



Booting from Hard Disk..xv6...
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap sta8
init: starting sh
12341700$ tlbrun
PageCount       Trials  Ticks   PageFaults
1       5000000 1       0
2       5000000 1       1
4       5000000 3       2
8       5000000 3       4
16      5000000 5       8
32      5000000 23      16
64      5000000 68      32
128     5000000 136     64
256     5000000 310     128
512     5000000 694     256
1024    5000000 1497    512
12341700$ tlbtest 8 5000000
8       5000000 3       4
12341700$ tlbtest 16 10000000
16      10000000        10      8
12341700$ 
```