Assignment Operating system

Name:- Ashwin Kumar
Roll - CE21BTECH11008

Task-1: Printing the page table entries

Modify the xv6 kernel to implement a new system call named pgtPrint() which will print the page table entries for the current process. Since the total number of page table entries can be very large, the system call should print the entries only if it is valid and the page is allowed access in user mode.

Solution:-

Steps needed in making pgtPrint() system call.

As it is not safe to direclty call system call thus for that I have made user program with the same name pgtPrint.c that calls a system call pgtPrint.

Changes to add system call pgtPrint()

1.syscall.c file

added :-[SYS_pgtPrint] sys_pgtPrint,

extern int sys_pgtPrint(void);

2.syscall.h (adding system call number)

added:- #define SYS_pgtPrint 22

3.sysproc.c

added:-

```
int sys_pgtPrint()
{
  // accessign the pointer to the page directory
 // pte_t *pte = myproc()->pagetable;
  pagetable_t pagetable = myproc()->pagetable;

  // counter for the total number of valid entry
  int counter = 0;

  //storeage for virtual address
  uint64 va=0;

  // for each entry in the page directory
  for(int i = 0 ; i < 512 ; i++)
  {
    pte_t pte= pagetable[i];

    if(pte & PTE_V)
    {

      uint64 pa1 = PTE2PA(pte);

      // for table in directory
      for(int j = 0 ; j < 512 ; j++)
      {

        pte_t pte2 = ((pagetable_t)pa1)[j];

        if (pte2 & PTE_V)
        {

          uint64 pa2 = PTE2PA(pte2);

        for(int k=0;k<512;k++)

        {

           pte_t pte3 = ((pagetable_t)pa2)[k];

           //page tabel entry will the address

        if(pte3 & PTE_V)
        {

          uint64 pa3 = PTE2PA(pte3);
              printf("PTE No::%d, Virtual page address-: %p , Physical page address-: %p\n" , counter , va , pa3);
         counter++;

        }

         //increasing the page size
```

As xv6 in risc v uses three level paging thus three for loop has been used and each for loop is going inside the page table to access

```
        //increasing the page size
        va+= PGSIZE;  //page size is definde in riscv.h

       }
      }
      va= va + 512 * PGSIZE;
    }
  }
  va=va+512*512*PGSIZE; //as we have crossed 512 * 512 entry thus increasing the virtual addrss
  }
  return 0;
}
```

other entry.

explaination of the above code :- first making pagatable pointer pointing to the output get by myproc()->pagetable as return address. Then initializing the counter to keep track of number of entry.

Outer for loop to access first entry in each page table and we know there will be 512 entry. And this is running three times in the for loop .To get page address from the page table entry using PTE2PA i.e. is defined in types.h. Another for loop for accesing end level entry. And checking entry is non zero , valid using “PTE_V” and “PTE_U” user bit as accesible by user, and at last printing those entry.

Also I have written the commend at the place of code to explain the things.

4.usys.S

added:- SYSCALL(pgtPrint)

5.user.h (map the system call to the array of system call defined)

added:- int pgtPrint(void)

6. In makefile UPROGS

added:- _pgtPrint\

at last add the pgtPrint.c in user part of os

ithout any array

```
$ pgtPrint
page table 0x0000000087f43000
..0: pte 0x0000000021fcfc01 pa 0x0000000087f3f000
.. ..0: pte 0x0000000021fcf801 pa 0x0000000087f3e000
.. .. ..0: pte 0x0000000021fd001b pa 0x0000000087f40000
.. .. ..2: pte 0x0000000021fcf407 pa 0x0000000087f3d000
.. .. ..3: pte 0x0000000021fcf017 pa 0x0000000087f3c000
..255: pte 0x0000000021fd0801 pa 0x0000000087f42000
.. ..511: pte 0x0000000021fd0401 pa 0x0000000087f41000
.. .. ..510: pte 0x0000000021fd8007 pa 0x0000000087f60000
.. .. ..511: pte 0x0000000020001c0b pa 0x0000000080007000
PTE No::0, Virtual page address-: 0x0000000000000000 , Physical page address-: 0x0000000087f40000
PTE No::1, Virtual page address-: 0x0000000000002000 , Physical page address-: 0x0000000087f3d000
PTE No::2, Virtual page address-: 0x0000000000003000 , Physical page address-: 0x0000000087f3c000
PTE No::3, Virtual page address-: 0x00000040401fe000 , Physical page address-: 0x0000000087f60000
PTE No::4, Virtual page address-: 0x00000040401ff000 , Physical page address-: 0x0000000080007000
$
```

Local arry

Defined a local array of size 10000

No changes has been seen after declaring.As address space of the system call in has no chance of distruption with local array definition.

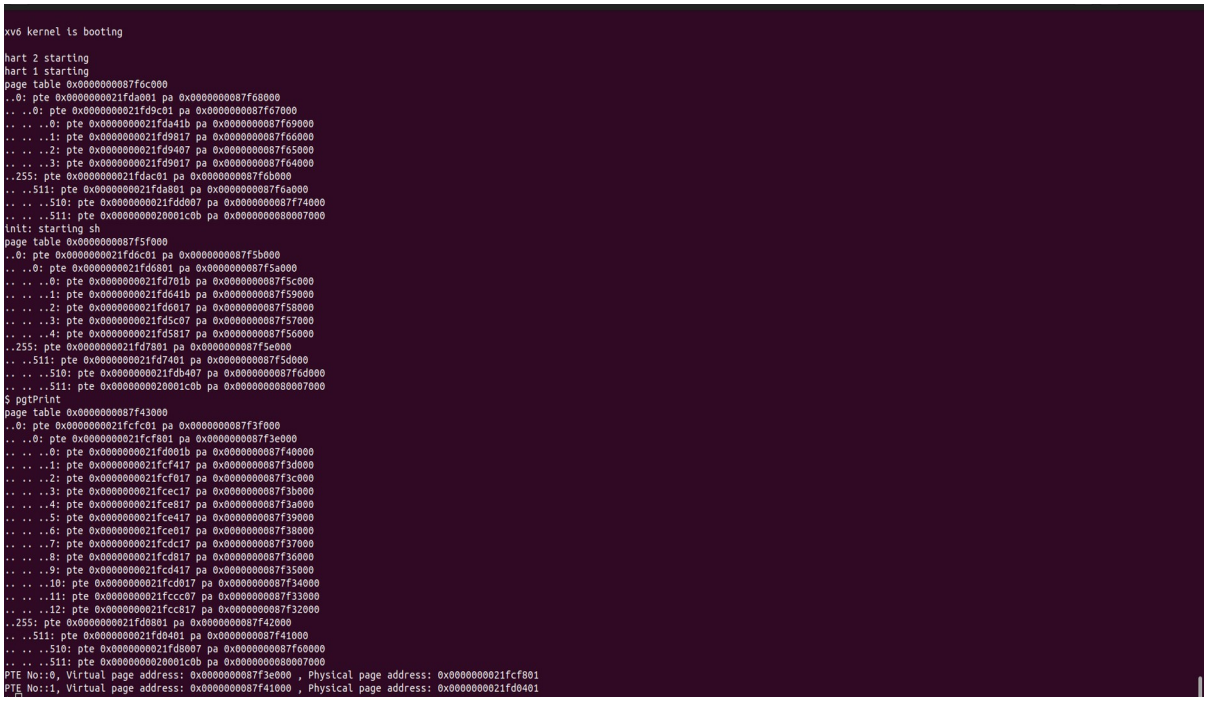
```
$ pgtPrint
page table 0x0000000087f43000
..0: pte 0x0000000021fcfc01 pa 0x0000000087f3f000
.. ..0: pte 0x0000000021fcf801 pa 0x0000000087f3e000
.. .. ..0: pte 0x0000000021fd001b pa 0x0000000087f40000
.. .. ..2: pte 0x0000000021fcf407 pa 0x0000000087f3d000
.. .. ..3: pte 0x0000000021fcf017 pa 0x0000000087f3c000
..255: pte 0x0000000021fd0801 pa 0x0000000087f42000
.. ..511: pte 0x0000000021fd0401 pa 0x0000000087f41000
.. .. ..510: pte 0x0000000021fd8007 pa 0x0000000087f60000
.. .. ..511: pte 0x0000000020001c0b pa 0x0000000080007000
This is for the local declaration
PTE No::0, Virtual page address-: 0x0000000000000000 , Physical page address-: 0x0000000087f40000
PTE No::1, Virtual page address-: 0x0000000000002000 , Physical page address-: 0x0000000087f3d000
PTE No::2, Virtual page address-: 0x0000000000003000 , Physical page address-: 0x0000000087f3c000
PTE No::3, Virtual page address-: 0x000000040401fe000 , Physical page address-: 0x0000000087f60000
PTE No::4, Virtual page address-: 0x000000040401ff000 , Physical page address-: 0x0000000080007000
$
```

Global array

defined a global size array arrGlobal[10000]

After making the global array the size of the page table entry has increased as seen.

Initial in local array page table first entry contain only4 entry 0,1,2,3 but after declareation of global array this entry has been changed to 14 – 0,1,2,3,4 ...,13.



As global resides in data segment i.e. outside the stack while local array resides in the stack.This may be the reason to make more entry in global.

Repeating the experiment

Repeating the experiment leading two two different address as can be seen.It is obvious that running the code again leads to different system call than prevoius thuse different address has been alloccated for this table.

Number of pages are in same untill and unless changing in this experiment.

```
$ pgtPrint
page table 0x0000000087f43000
..0: pte 0x0000000021fcfc01 pa 0x0000000087f3f000
.. ..0: pte 0x0000000021fcf801 pa 0x0000000087f3e000
.. .. ..0: pte 0x0000000021fd001b pa 0x0000000087f40000
.. .. ..2: pte 0x0000000021fcf407 pa 0x0000000087f3d000
.. .. ..3: pte 0x0000000021fcf017 pa 0x0000000087f3c000
..255: pte 0x0000000021fd0801 pa 0x0000000087f42000
.. ..511: pte 0x0000000021fd0401 pa 0x0000000087f41000
.. .. ..510: pte 0x0000000021fd8007 pa 0x0000000087f60000
.. .. ..511: pte 0x0000000020001c0b pa 0x0000000080007000
PTE No::0, Virtual page address-: 0x0000000000000000 , Physical page address-: 0x0000000087f40000
PTE No::1, Virtual page address-: 0x0000000000002000 , Physical page address-: 0x0000000087f3d000
PTE No::2, Virtual page address-: 0x0000000000003000 , Physical page address-: 0x0000000087f3c000
PTE No::3, Virtual page address-: 0x00000040401fe000 , Physical page address-: 0x0000000087f60000
PTE No::4, Virtual page address-: 0x00000040401ff000 , Physical page address-: 0x0000000080007000
$ pgtPrint
page table 0x0000000087f52000
..0: pte 0x0000000021fdbc01 pa 0x0000000087f6f000
.. ..0: pte 0x0000000021fd8c01 pa 0x0000000087f63000
.. .. ..0: pte 0x0000000021fd541b pa 0x0000000087f55000
.. .. ..2: pte 0x0000000021fd8807 pa 0x0000000087f62000
.. .. ..3: pte 0x0000000021fdb817 pa 0x0000000087f6e000
..255: pte 0x0000000021fd4c01 pa 0x0000000087f53000
.. ..511: pte 0x0000000021fd5001 pa 0x0000000087f54000
.. .. ..510: pte 0x0000000021fd0c07 pa 0x0000000087f43000
.. .. ..511: pte 0x0000000020001c0b pa 0x0000000080007000
PTE No::0, Virtual page address-: 0x0000000000000000 , Physical page address-: 0x0000000087f55000
PTE No::1, Virtual page address-: 0x0000000000002000 , Physical page address-: 0x0000000087f62000
PTE No::2, Virtual page address-: 0x0000000000003000 , Physical page address-: 0x0000000087f6e000
PTE No::3, Virtual page address-: 0x00000040401fe000 , Physical page address-: 0x0000000087f43000
PTE No::4, Virtual page address-: 0x00000040401ff000 , Physical page address-: 0x0000000080007000
$ pgtPrint
page table 0x0000000087f70000
..0: pte 0x0000000021fd1001 pa 0x0000000087f44000
.. ..0: pte 0x0000000021fd8001 pa 0x0000000087f60000
.. .. ..0: pte 0x0000000021fdcc1b pa 0x0000000087f73000
.. .. ..2: pte 0x0000000021fd0007 pa 0x0000000087f40000
.. .. ..3: pte 0x0000000021fcf817 pa 0x0000000087f3e000
..255: pte 0x0000000021fdc401 pa 0x0000000087f71000
.. ..511: pte 0x0000000021fdc801 pa 0x0000000087f72000
.. .. ..510: pte 0x0000000021fd4807 pa 0x0000000087f52000
.. .. ..511: pte 0x0000000020001c0b pa 0x0000000080007000
PTE No::0, Virtual page address-: 0x0000000000000000 , Physical page address-: 0x0000000087f73000
PTE No::1, Virtual page address-: 0x0000000000002000 , Physical page address-: 0x0000000087f40000
PTE No::2, Virtual page address-: 0x0000000000003000 , Physical page address-: 0x0000000087f3e000
PTE No::3, Virtual page address-: 0x00000040401fe000 , Physical page address-: 0x0000000087f52000
PTE No::4, Virtual page address-: 0x00000040401ff000 , Physical page address-: 0x0000000080007000
$
```

Task-2: Implement demand paging

We discussed demand paging in our lectures where pages are not allocated on process creation but based on demand. The base implementation of xv6 does not implement demand paging and our task in this assignment is to implement demand paging. We would implement a simpler version of demand paging where the read-only code associated with the process is mapped during the process creation, but the memory required for heap and globals is not assigned pages during process creation but allocated on demand. Also, our demand paging would be simpler to assume that sufficient memory is available and we do not need to replace/evict any page during the demand paging process.

Sol:-
Demand paging is not present in vanila xv6 but can be implented making some changes, basicly idea is tell the proces that memory has been allocated for all the variable asked but actually no storage should be allocated untill and unless it is being used or when it is demanded it will go raise page fault error and then that block will be bring to the memory.

Changes:-

1.user.h
added: //here adding demand page function signature (header)
int demandPage(void);

2.vm.c

```
70
71 //making file mapping for static use
72
73 int mappings(pte_t* pagetable, uint64 va, uint64 size, uint64 pa, int perm)
74 {
75   return mappages(pagetable , va , size , pa , perm);
76 }
77
```

3.trap.c

In user trap function checking -
To check if the trap corresponds to page fault and if yes, then implement a handler to implement demand paging. The handler is checking for the faulting address and if it is a valid address in the virtual memory range of the process. If yes, assign a page and map it to the pagetable. If it is not a valid page, then generate errors as was happening earlier.

r_scause return reason of trap if occured – it return 13 or 15 when the page fault occured

```
//changes start
    else if(r_scause() == 13 || r_scause() == 15) {
  uint64 va = r_stval();

  printf("Page Fault occured: doing demand paging for address 0x%x\n" , va);
  // Rounding  the fault virtual address to a page boundary
  va = PGROUNDDOWN(va);

  char *mem = kalloc();
  memset(mem, 0, PGSIZE);
  if(mappages(p->pagetable, va, PGSIZE, (uint64)mem, PTE_W | PTE_R | PTE_U) != 0){
    kfree(mem);
  }
} else {
  printf("usertrap(): unexpected scause for %p pid=%d\n", r_scause(), p->pid);
  printf("        sepc=%p stval=%p\n", r_sepc(), r_stval());
  p->killed = 1;
}
```

4.exec.c

in function exec making changes.

The below code is th ensure that memory is allocatd for the read only and rest dynamica memoey is based on the demand. Here checking ELF information filesz and memsz.Size of the memsz must be at least equal to filesz. Uvmalloc does this work.

```
  uint64 sz1;
  //change start

    //added this at place of above
    if((sz1 = uvmalloc(pagetable, sz, ph.vaddr + ph.filesz, flags2perm(ph.flags))) == 0)
    goto bad;

  sz = sz1;
      // so we now add the remaining space of dynamic data
  sz += ph.memsz - ph.filesz;

  if(ph.vaddr % PGSIZE != 0)
    goto bad;

    //change end
```

2nd question

N=3000

```
$ demandPage
page table 0x0000000087f43000
..0: pte 0x0000000021fcfc01 pa 0x0000000087f3f000
.. ..0: pte 0x0000000021fcf801 pa 0x0000000087f3e000
.. .. ..0: pte 0x0000000021fd001b pa 0x0000000087f40000
.. .. ..4: pte 0x0000000021fcf407 pa 0x0000000087f3d000
.. .. ..5: pte 0x0000000021fcf017 pa 0x0000000087f3c000
..255: pte 0x0000000021fd0801 pa 0x0000000087f42000
.. ..511: pte 0x0000000021fd0401 pa 0x0000000087f41000
.. .. ..510: pte 0x0000000021fd8007 pa 0x0000000087f60000
.. .. ..511: pte 0x0000000020001c0b pa 0x0000000080007000
Address from user space is : 1010
Page Fault occured: doing demand paging for address 0x1010
PTE No::0, Virtual page address-: 0x0000000000000000 , Physical page address-: 0x0000000087f40000
PTE No::1, Virtual page address-: 0x0000000000001000 , Physical page address-: 0x0000000087f44000
PTE No::2, Virtual page address-: 0x0000000000004000 , Physical page address-: 0x0000000087f3d000
PTE No::3, Virtual page address-: 0x0000000000005000 , Physical page address-: 0x0000000087f3c000
PTE No::4, Virtual page address-: 0x00000040401fe000 , Physical page address-: 0x0000000087f60000
PTE No::5, Virtual page address-: 0x00000040401ff000 , Physical page address-: 0x0000000080007000
Page Fault occured: doing demand paging for address 0x2000
PTE No::0, Virtual page address-: 0x0000000000000000 , Physical page address-: 0x0000000087f40000
PTE No::1, Virtual page address-: 0x0000000000001000 , Physical page address-: 0x0000000087f44000
PTE No::2, Virtual page address-: 0x0000000000002000 , Physical page address-: 0x0000000087f73000
PTE No::3, Virtual page address-: 0x0000000000004000 , Physical page address-: 0x0000000087f3d000
PTE No::4, Virtual page address-: 0x0000000000005000 , Physical page address-: 0x0000000087f3c000
PTE No::5, Virtual page address-: 0x00000040401fe000 , Physical page address-: 0x0000000087f60000
PTE No::6, Virtual page address-: 0x00000040401ff000 , Physical page address-: 0x0000000080007000
Page Fault occured: doing demand paging for address 0x3000
Printing final page table:
PTE No::0, Virtual page address-: 0x0000000000000000 , Physical page address-: 0x0000000087f40000
PTE No::1, Virtual page address-: 0x0000000000001000 , Physical page address-: 0x0000000087f44000
PTE No::2, Virtual page address-: 0x0000000000002000 , Physical page address-: 0x0000000087f73000
PTE No::3, Virtual page address-: 0x0000000000003000 , Physical page address-: 0x0000000087f72000
PTE No::4, Virtual page address-: 0x0000000000004000 , Physical page address-: 0x0000000087f3d000
PTE No::5, Virtual page address-: 0x0000000000005000 , Physical page address-: 0x0000000087f3c000
PTE No::6, Virtual page address-: 0x00000040401fe000 , Physical page address-: 0x0000000087f60000
PTE No::7, Virtual page address-: 0x00000040401ff000 , Physical page address-: 0x0000000080007000
final value in globArray is 1
$
```

N=5000

```
.. .. ..510: pte 0x0000000021fd8007 pa 0x0000000087f60000
.. .. ..511: pte 0x0000000020001c0b pa 0x0000000080007000
Address from user space is : 1010
Page Fault occured: doing demand paging for address 0x1010
PTE No::0, Virtual page address-: 0x0000000000000000 , Physical page address-: 0x0000000087f40000
PTE No::1, Virtual page address-: 0x0000000000001000 , Physical page address-: 0x0000000087f44000
PTE No::2, Virtual page address-: 0x0000000000006000 , Physical page address-: 0x0000000087f3d000
PTE No::3, Virtual page address-: 0x0000000000007000 , Physical page address-: 0x0000000087f3c000
PTE No::4, Virtual page address-: 0x00000040401fe000 , Physical page address-: 0x0000000087f60000
PTE No::5, Virtual page address-: 0x00000040401ff000 , Physical page address-: 0x0000000080007000
Page Fault occured: doing demand paging for address 0x2000
PTE No::0, Virtual page address-: 0x0000000000000000 , Physical page address-: 0x0000000087f40000
PTE No::1, Virtual page address-: 0x0000000000001000 , Physical page address-: 0x0000000087f44000
PTE No::2, Virtual page address-: 0x0000000000002000 , Physical page address-: 0x0000000087f73000
PTE No::3, Virtual page address-: 0x0000000000006000 , Physical page address-: 0x0000000087f3d000
PTE No::4, Virtual page address-: 0x0000000000007000 , Physical page address-: 0x0000000087f3c000
PTE No::5, Virtual page address-: 0x00000040401fe000 , Physical page address-: 0x0000000087f60000
PTE No::6, Virtual page address-: 0x00000040401ff000 , Physical page address-: 0x0000000080007000
Page Fault occured: doing demand paging for address 0x3000
PTE No::0, Virtual page address-: 0x0000000000000000 , Physical page address-: 0x0000000087f40000
PTE No::1, Virtual page address-: 0x0000000000001000 , Physical page address-: 0x0000000087f44000
PTE No::2, Virtual page address-: 0x0000000000002000 , Physical page address-: 0x0000000087f73000
PTE No::3, Virtual page address-: 0x0000000000003000 , Physical page address-: 0x0000000087f72000
PTE No::4, Virtual page address-: 0x0000000000006000 , Physical page address-: 0x0000000087f3d000
PTE No::5, Virtual page address-: 0x0000000000007000 , Physical page address-: 0x0000000087f3c000
PTE No::6, Virtual page address-: 0x00000040401fe000 , Physical page address-: 0x0000000087f60000
PTE No::7, Virtual page address-: 0x00000040401ff000 , Physical page address-: 0x0000000080007000
Page Fault occured: doing demand paging for address 0x4000
PTE No::0, Virtual page address-: 0x0000000000000000 , Physical page address-: 0x0000000087f40000
PTE No::1, Virtual page address-: 0x0000000000001000 , Physical page address-: 0x0000000087f44000
PTE No::2, Virtual page address-: 0x0000000000002000 , Physical page address-: 0x0000000087f73000
PTE No::3, Virtual page address-: 0x0000000000003000 , Physical page address-: 0x0000000087f72000
PTE No::4, Virtual page address-: 0x0000000000004000 , Physical page address-: 0x0000000087f71000
PTE No::5, Virtual page address-: 0x0000000000006000 , Physical page address-: 0x0000000087f3d000
PTE No::6, Virtual page address-: 0x0000000000007000 , Physical page address-: 0x0000000087f3c000
PTE No::7, Virtual page address-: 0x00000040401fe000 , Physical page address-: 0x0000000087f60000
PTE No::8, Virtual page address-: 0x00000040401ff000 , Physical page address-: 0x0000000080007000
Page Fault occured: doing demand paging for address 0x5000
Printing final page table:
PTE No::0, Virtual page address-: 0x0000000000000000 , Physical page address-: 0x0000000087f40000
PTE No::1, Virtual page address-: 0x0000000000001000 , Physical page address-: 0x0000000087f44000
PTE No::2, Virtual page address-: 0x0000000000002000 , Physical page address-: 0x0000000087f73000
PTE No::3, Virtual page address-: 0x0000000000003000 , Physical page address-: 0x0000000087f72000
PTE No::4, Virtual page address-: 0x0000000000004000 , Physical page address-: 0x0000000087f71000
PTE No::5, Virtual page address-: 0x0000000000005000 , Physical page address-: 0x0000000087f70000
PTE No::6, Virtual page address-: 0x0000000000006000 , Physical page address-: 0x0000000087f3d000
PTE No::7, Virtual page address-: 0x0000000000007000 , Physical page address-: 0x0000000087f3c000
PTE No::8, Virtual page address-: 0x00000040401fe000 , Physical page address-: 0x0000000087f60000
PTE No::9, Virtual page address-: 0x00000040401ff000 , Physical page address-: 0x0000000080007000
final value in globArray is 1
$
```

N=10000

```
$ demandPage
page table 0x0000000087f43000
..0: pte 0x0000000021fcfc01 pa 0x0000000087f3f000
.. ..0: pte 0x0000000021fcf801 pa 0x0000000087f3e000
.. .. ..0: pte 0x0000000021fd001b pa 0x0000000087f40000
.. .. ..11: pte 0x0000000021fcf407 pa 0x0000000087f3d000
.. .. ..12: pte 0x0000000021fcf017 pa 0x0000000087f3c000
..255: pte 0x0000000021fd0801 pa 0x0000000087f42000
.. ..511: pte 0x0000000021fd0401 pa 0x0000000087f41000
.. .. ..510: pte 0x0000000021fd8007 pa 0x0000000087f60000
.. .. ..511: pte 0x0000000020001c0b pa 0x0000000080007000
Address from user space is : 1010
Page Fault occured: doing demand paging for address 0x1010
PTE No::0, Virtual page address-: 0x0000000000000000 , Physical page address-: 0x0000000087f40000
PTE No::1, Virtual page address-: 0x0000000000001000 , Physical page address-: 0x0000000087f44000
PTE No::2, Virtual page address-: 0x000000000000b000 , Physical page address-: 0x0000000087f3d000
PTE No::3, Virtual page address-: 0x000000000000c000 , Physical page address-: 0x0000000087f3c000
PTE No::4, Virtual page address-: 0x00000040401fe000 , Physical page address-: 0x0000000087f60000
PTE No::5, Virtual page address-: 0x00000040401ff000 , Physical page address-: 0x0000000080007000
Page Fault occured: doing demand paging for address 0x2000
PTE No::0, Virtual page address-: 0x0000000000000000 , Physical page address-: 0x0000000087f40000
PTE No::1, Virtual page address-: 0x0000000000001000 , Physical page address-: 0x0000000087f44000
PTE No::2, Virtual page address-: 0x0000000000002000 , Physical page address-: 0x0000000087f73000
PTE No::3, Virtual page address-: 0x000000000000b000 , Physical page address-: 0x0000000087f3d000
PTE No::4, Virtual page address-: 0x000000000000c000 , Physical page address-: 0x0000000087f3c000
PTE No::5, Virtual page address-: 0x00000040401fe000 , Physical page address-: 0x0000000087f60000
PTE No::6, Virtual page address-: 0x00000040401ff000 , Physical page address-: 0x0000000080007000
Page Fault occured: doing demand paging for address 0x3000
PTE No::0, Virtual page address-: 0x0000000000000000 , Physical page address-: 0x0000000087f40000
PTE No::1, Virtual page address-: 0x0000000000001000 , Physical page address-: 0x0000000087f44000
PTE No::2, Virtual page address-: 0x0000000000002000 , Physical page address-: 0x0000000087f73000
PTE No::3, Virtual page address-: 0x0000000000003000 , Physical page address-: 0x0000000087f72000
PTE No::4, Virtual page address-: 0x000000000000b000 , Physical page address-: 0x0000000087f3d000
PTE No::5, Virtual page address-: 0x000000000000c000 , Physical page address-: 0x0000000087f3c000
PTE No::6, Virtual page address-: 0x00000040401fe000 , Physical page address-: 0x0000000087f60000
PTE No::7, Virtual page address-: 0x00000040401ff000 , Physical page address-: 0x0000000080007000
Page Fault occured: doing demand paging for address 0x4000
PTE No::0, Virtual page address-: 0x0000000000000000 , Physical page address-: 0x0000000087f40000
PTE No::1, Virtual page address-: 0x0000000000001000 , Physical page address-: 0x0000000087f44000
PTE No::2, Virtual page address-: 0x0000000000002000 , Physical page address-: 0x0000000087f73000
PTE No::3, Virtual page address-: 0x0000000000003000 , Physical page address-: 0x0000000087f72000
PTE No::4, Virtual page address-: 0x0000000000004000 , Physical page address-: 0x0000000087f71000
PTE No::5, Virtual page address-: 0x000000000000b000 , Physical page address-: 0x0000000087f3d000
PTE No::6, Virtual page address-: 0x000000000000c000 , Physical page address-: 0x0000000087f3c000
PTE No::7, Virtual page address-: 0x00000040401fe000 , Physical page address-: 0x0000000087f60000
PTE No::8, Virtual page address-: 0x00000040401ff000 , Physical page address-: 0x0000000080007000
Page Fault occured: doing demand paging for address 0x5000
PTE No::0, Virtual page address-: 0x0000000000000000 , Physical page address-: 0x0000000087f40000
PTE No::1, Virtual page address-: 0x0000000000001000 , Physical page address-: 0x0000000087f44000
PTE No::2, Virtual page address-: 0x0000000000002000 , Physical page address-: 0x0000000087f73000
PTE No::3, Virtual page address-: 0x0000000000003000 , Physical page address-: 0x0000000087f72000
PTE No::4, Virtual page address-: 0x0000000000004000 , Physical page address-: 0x0000000087f71000
```

```
PTE No::8, Virtual page address-: 0x000000000000b000 , Physical page address-: 0x0000000087f3d000
PTE No::9, Virtual page address-: 0x000000000000c000 , Physical page address-: 0x0000000087f3c000
PTE No::10, Virtual page address-: 0x00000040401fe000 , Physical page address-: 0x0000000087f60000
PTE No::11, Virtual page address-: 0x00000040401ff000 , Physical page address-: 0x0000000080007000
Page Fault occured: doing demand paging for address 0x8000
PTE No::0, Virtual page address-: 0x0000000000000000 , Physical page address-: 0x0000000087f40000
PTE No::1, Virtual page address-: 0x0000000000001000 , Physical page address-: 0x0000000087f44000
PTE No::2, Virtual page address-: 0x0000000000002000 , Physical page address-: 0x0000000087f73000
PTE No::3, Virtual page address-: 0x0000000000003000 , Physical page address-: 0x0000000087f72000
PTE No::4, Virtual page address-: 0x0000000000004000 , Physical page address-: 0x0000000087f71000
PTE No::5, Virtual page address-: 0x0000000000005000 , Physical page address-: 0x0000000087f70000
PTE No::6, Virtual page address-: 0x0000000000006000 , Physical page address-: 0x0000000087f61000
PTE No::7, Virtual page address-: 0x0000000000007000 , Physical page address-: 0x0000000087f45000
PTE No::8, Virtual page address-: 0x0000000000008000 , Physical page address-: 0x0000000087f46000
PTE No::9, Virtual page address-: 0x000000000000b000 , Physical page address-: 0x0000000087f3d000
PTE No::10, Virtual page address-: 0x000000000000c000 , Physical page address-: 0x0000000087f3c000
PTE No::11, Virtual page address-: 0x00000040401fe000 , Physical page address-: 0x0000000087f60000
PTE No::12, Virtual page address-: 0x00000040401ff000 , Physical page address-: 0x0000000080007000
Page Fault occured: doing demand paging for address 0x9000
PTE No::0, Virtual page address-: 0x0000000000000000 , Physical page address-: 0x0000000087f40000
PTE No::1, Virtual page address-: 0x0000000000001000 , Physical page address-: 0x0000000087f44000
PTE No::2, Virtual page address-: 0x0000000000002000 , Physical page address-: 0x0000000087f73000
PTE No::3, Virtual page address-: 0x0000000000003000 , Physical page address-: 0x0000000087f72000
PTE No::4, Virtual page address-: 0x0000000000004000 , Physical page address-: 0x0000000087f71000
PTE No::5, Virtual page address-: 0x0000000000005000 , Physical page address-: 0x0000000087f70000
PTE No::6, Virtual page address-: 0x0000000000006000 , Physical page address-: 0x0000000087f61000
PTE No::7, Virtual page address-: 0x0000000000007000 , Physical page address-: 0x0000000087f45000
PTE No::8, Virtual page address-: 0x0000000000008000 , Physical page address-: 0x0000000087f46000
PTE No::9, Virtual page address-: 0x0000000000009000 , Physical page address-: 0x0000000087f47000
PTE No::10, Virtual page address-: 0x000000000000b000 , Physical page address-: 0x0000000087f3d000
PTE No::11, Virtual page address-: 0x000000000000c000 , Physical page address-: 0x0000000087f3c000
PTE No::12, Virtual page address-: 0x00000040401fe000 , Physical page address-: 0x0000000087f60000
PTE No::13, Virtual page address-: 0x00000040401ff000 , Physical page address-: 0x0000000080007000
Page Fault occured: doing demand paging for address 0xa000
Printing final page table:
PTE No::0, Virtual page address-: 0x0000000000000000 , Physical page address-: 0x0000000087f40000
PTE No::1, Virtual page address-: 0x0000000000001000 , Physical page address-: 0x0000000087f44000
PTE No::2, Virtual page address-: 0x0000000000002000 , Physical page address-: 0x0000000087f73000
PTE No::3, Virtual page address-: 0x0000000000003000 , Physical page address-: 0x0000000087f72000
PTE No::4, Virtual page address-: 0x0000000000004000 , Physical page address-: 0x0000000087f71000
PTE No::5, Virtual page address-: 0x0000000000005000 , Physical page address-: 0x0000000087f70000
PTE No::6, Virtual page address-: 0x0000000000006000 , Physical page address-: 0x0000000087f61000
PTE No::7, Virtual page address-: 0x0000000000007000 , Physical page address-: 0x0000000087f45000
PTE No::8, Virtual page address-: 0x0000000000008000 , Physical page address-: 0x0000000087f46000
PTE No::9, Virtual page address-: 0x0000000000009000 , Physical page address-: 0x0000000087f47000
PTE No::10, Virtual page address-: 0x000000000000a000 , Physical page address-: 0x0000000087f48000
PTE No::11, Virtual page address-: 0x000000000000b000 , Physical page address-: 0x0000000087f3d000
PTE No::12, Virtual page address-: 0x000000000000c000 , Physical page address-: 0x0000000087f3c000
PTE No::13, Virtual page address-: 0x00000040401fe000 , Physical page address-: 0x0000000087f60000
PTE No::14, Virtual page address-: 0x00000040401ff000 , Physical page address-: 0x0000000080007000
final value in globArray is 1
$
```

Q3.Task-3: Implement logic to detect which pages have been accessed and/or dirty

1.syscall.c file

added :-[SYS_pgaccess] sys_pgaccess,

extern int sys_pgaccess(void);

2.syscall.h (adding system call number)

added:- #define SYS_pgaccess 23

3.sysproc.c

added:-

//adding pgaccess function to sysproc.c

it is taking three argument

```
int
sys_pgaccess(void)
{
  uint64 startaddr;
  int total_page;
  uint64 useraddr;

  //taking the argumner from the pgaccess.c function
  argaddr(0, &startaddr);
  argint(1, &total_page);
  argaddr(2, &useraddr);

  uint64 bitmask = 0;

  //finding the compliment  using the negation sign
  uint64 compliment = ~PTE_A;

  struct proc *p = myproc();
  for (int i = 0; i < total_page; ++i) {
    pte_t *pte = walk(p->pagetable, startaddr+i*PGSIZE, 0);
    if (*pte & PTE_A & PTE_D) {
      bitmask |= (1 << i);
      *pte &= compliment;
    }
  }
  copyout(p->pagetable, useraddr, (char *)&bitmask, sizeof(bitmask));
  printf("coming out\n");
  return 0;
}
```

4.usys.S

added:- SYSCALL(pgaccess)

5.user.h (map the system call to the array of system call defined)

added:- int pgaceess(void)

6. In makefile UPROGS

added:- _pgaccess\

at last add the pgaccess.c in user part of os

IN riscv.h added defined the dirty bit

```
#define PGROUNDUP(sz)  (((sz)+PGSIZE-1) & ~(PGSIZE-1))
#define PGROUNDDOWN(a) (((a)) & ~(PGSIZE-1))

#define PTE_V (1L << 0) // valid
#define PTE_R (1L << 1)
#define PTE_W (1L << 2)
#define PTE_X (1L << 3)
#define PTE_U (1L << 4) // user can access
#define PTE_A (1L << 6) // show accesed or not
#define PTE_D (1L << 7) // show dirty bit or not

//address in the page table
#define PTE_ADDR(pte) ((pte) & ~0x3FF)
```

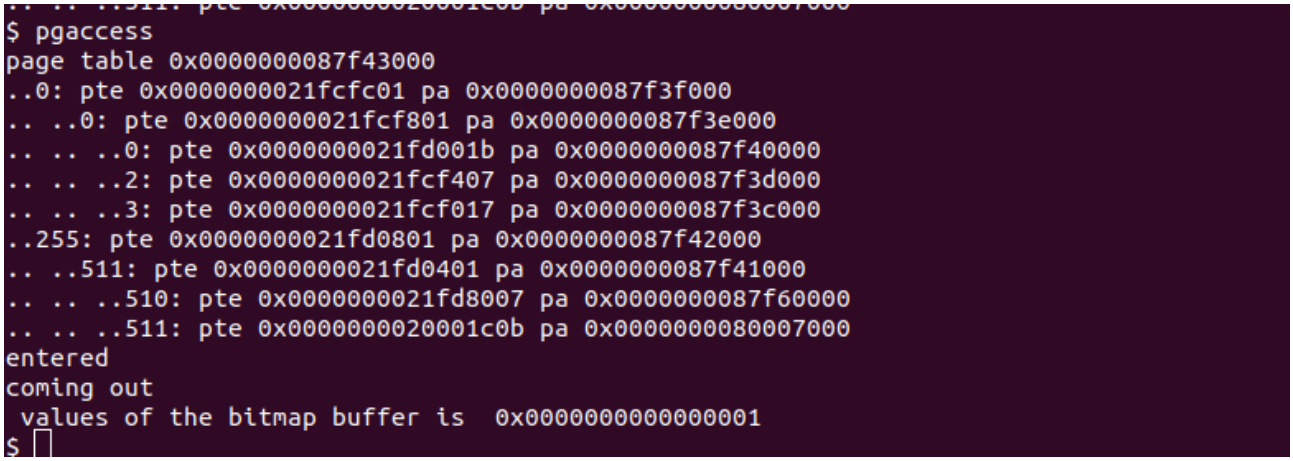
simply I am taking three argumner first is start virtual address , 2nd is page upto i am goint to check while third the addres of the buufer address.
For each pte tabel entry using walk function to check the valid entry and also checking it was dirty or not and using bitmask to store the store check access bit.
This is the pgaccess function in the user part of the xv6

Open pgaccess.c ~/xv6-riscv1/user Save

```
#include "kernel/types.h"
#include <kernel/stat.h>
#include <user/user.h>

int main(int argc, char*argv[]){

    uint64 start_va = 6;
    uint64 num_pages =30;
    uint64 buffer ;

    printf("entered\n");
    pgaccess(&start_va,num_pages,&buffer);

    printf(" values are %p", buffer);
    exit(0);
}
```

start_va = is the start of virtual addree and 30 is the numper of pages i am goint to chek c
then calling pgacces function passing address of three argumnet – and buffer address to store the returned bitmask.

This is the output – showing from 6 -30 one page has been accessd – can be seen in the address.



Your understanding of how system call works and what you learnt from this assignment.

Understanding:-

--system call is used to invoke the process that runs in the kernal mode. The working of system call i have explained like we are first calling user process and that user process is calling to the system call. For the system call some necessary modification are done that are explained along with like changing in the some file and making unique entry in some file of the kernel.

I have learned a lot from this assignment. First is how to make system call 2nd is how we can acesss the actual address from the virtual address. Then after implemented and saw how page table pointer and page table entry work.
Next I have learn how the modification in the kernel code work when applying demand paging in which is not present in vaniall xv6 risc v os. I have also get used to some assemebely code when i was adding the function. Like it uses some fixed set of address where the function write like a0 and thus at that location our cpu reads and take the decision
And last I have implemented checking of access and dirty bit during allocation of page and how to use bitmask to store and closely saw the implementation of this.

Reference:-

I have taken some refernce from various githubs that gave me interior knowledge to do this assignment and then I have implemented by own.

Its stucking when i am making xv6.tar.gz file. So i am submittin full code for xv6-riscv