Introduction to Operating Systems (UE15CS302)

Name: Alekhya E

USN: 01FB15ECS032

Section: A

Problem Statement:

PID\_IOS\_0817\_21:

Add a new system call that, given a virtual address, returns the translated physical address (or report that the address is not valid) on xv6.

# Abstract:

This report mainly focusses on how the virtual addresses will be converted to physical addresses as a system call in xv6.

# Introduction:

The addresses generated by CPU are virtual addresses. Each of these addresses should be mapped to actual physical address and only then RAM is actually accessed.

# Steps:

1. Store the virtual addresses generated while booting of qemu in an array declared as global.

2. Generate the physical address where the kernel starts using the command : objdump –f kernel.

3. Validate the input that is the virtual address given by the user with the elements of the virtual address array generated while booting of qemu .If it is valid then finding the corresponding physical address. Otherwise send an alert to convey that the address is invalid.

4. Find the physical address by adding the given validated virtual address with the starting physical address where the kernel starts.

# 3. Files to be modified:

Modifications for the files to make it a system call

1. syscall.h:

Add #define SYS\_adress 22

2. defs.h

In proc.c add int adress(void)

3. user.h

Add int adress(void)

4. sysproc.c

Calling function adress()

int sys\_adress(void)

{

return adress();

}

5. usys.S

Add SYSCALL(adress)

6. sysall.c

Add [SYS\_adress ] sys\_adress,

7. vm.c

In loaduvm() following code has to be included:

int z=0;  
  
 char virtual[100][20];// For storing virtual addresses generated while booting .  
  
 for(z=0;z<sz; z+=PGSIZE)  
      {  
            
            
          virtual[i][z]=\*(addr);  
            
      }

https://ssl.gstatic.com/ui/v1/icons/mail/images/cleardot.gif

8. proc.c

//Adding function definition

//Declaring variables

extern char virtual[100][20];

//Externing the virtual addresses array of vm.c  
char physical[100][20];

//Array to store the physical addresses.

char va[]="80115cf4";

//User input virtual address.  
char \*base="0010000c";

//Starting physical address of the kernel  
  
int c=0;

//Function used to convert base address which is hexadecimal to decimal.

uint atoi\_hexa(char \*buf, int start){          
    uint res = 0;    // Stores the final result  
    int i = start ;  
  
    for(; buf[i] != '\0' ; i++){  
        uint temp = 0 ;  
        if(buf[i] - '0' >= 0 && buf[i] - '0' <= 9)  
            temp = buf[i] - '0' ;  
        else{  
            switch(buf[i]){  
                case 'A' :  
                case 'a' :  temp = 10 ;  
                            break ;  
                case 'B' :  
                case 'b' :  temp = 11 ;  
                            break ;  
                case 'C' :  
                case 'c' :  temp = 12 ;  
                            break ;  
                case 'D' :  
                case 'd' :  temp = 13 ;  
                            break ;  
                case 'E' :  
                case 'e' :  temp = 14 ;  
                            break ;  
                case 'F' :  
                case 'f' :  temp = 15 ;  
                            break ;  
                default :   return res ;   
            }  
        }  
        res = res\*16 + temp ;  
    }  
     
    return res ;  
}

//Convert the character array to integer for comparison

int char\_to\_int\_conversion(char \*hex\_char) {  
    int hex\_int = 0;  
    while (\*hex\_char) {  
  
        int current\_char = \*hex\_char++;   
          
    // converting hexadecimal character to the 4 bit equivalent number  
      
        if (current\_char >= '0' && current\_char <= '9')   
            current\_char = current\_char - '0';  
        else if (current\_char >= 'a' && current\_char <='f')               
            current\_char = current\_char - 'a' + 10;  
        else if (current\_char >= 'A' && current\_char <='F')           
            current\_char = current\_char - 'A' + 10;      
              
    // shift 4 to make space for new digit, and add the 4 bits of the new digit   
        hex\_int = (hex\_int << 4) | (current\_char & 0xF);  
    }  
    return hex\_int;  
}

Creating an array for physical addresses

int adress(void)  
{  
    for(int k=0;k<10;k++)  
    { for(int j=0;j<10;j++)  
        {  
            physical[k][j]=virtual[k][j]+atoi\_hexa(base,0);  
        }

}

 //To convert character array to corresponding integer.

 int va\_i= char\_to\_int\_conversion(va);

    int flag = 0;  
    cprintf("\ngiven address:%x\n",va\_i);  
    for(int i=0;i<10;i++)  
    {  
        //cprintf("\nvirtual addresses%x",virtual[i]);

//Comparing if the input given matches any of the virual addresses

//If it matches then print the corresponding physical address

        if(va\_i==virtual[i])  
        {  
            cprintf("Matching\n");  
              
            cprintf("pa for user entered va is ------%x\n",(atoi\_hexa(va,0))+(atoi\_hexa(base,0)));  
            flag=1;  
            break;  
        }  
          
    }  
    cprintf("\n");  
    if(!flag)  
        {  
            cprintf("Invalid virtual adress entered by user\n");  
        }  
      
      
        return 1;      
}

9. Create another file named address\_1.c

This file contains main() along with the function call.

Code to be included:

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

{

adress();

exit();

}

10. Makefile:

Add adress\_1\ (i.e the filename ) to UGPROGS=\

Add address\_1.c to EXTRA=\

# 3. Code

1. Declare an array which stores the virtual addresses produced during booting of xv6 in loaduvm() in vm.c file.

int z=0;  
  
char virtual[100][20];// For storing virtual addresses.  
  
 for(z=0;z<sz;z+=PGSIZE)  
  {  
            
            
          virtual[i][z]=\*(addr);  
            
   }

2. proc.c

//Externing the virtual address array which is declared in vm.c.

//Function used to convert base address which is hexadecimal to decimal

uint atoi\_hexa(char \*buf, int start){          
    uint res = 0;    // Stores the final result  
    int i = start ;  
  
    for(; buf[i] != '\0' ; i++){  
    //cprintf("hi---");  
        uint temp = 0 ;  
        if(buf[i] - '0' >= 0 && buf[i] - '0' <= 9)  
            temp = buf[i] - '0' ;  
        else{  
            switch(buf[i]){  
                case 'A' :  
                case 'a' :  temp = 10 ;  
                            break ;  
                case 'B' :  
                case 'b' :  temp = 11 ;  
                            break ;  
                case 'C' :  
                case 'c' :  temp = 12 ;  
                            break ;  
                case 'D' :  
                case 'd' :  temp = 13 ;  
                            break ;  
                case 'E' :  
                case 'e' :  temp = 14 ;  
                            break ;  
                case 'F' :  
                case 'f' :  temp = 15 ;  
                            break ;  
                default :   return res ;   
            }  
        }  
        res = res\*16 + temp ;  
    }  
     
    return res ;  
}

To convert the character array to integer for comparison

int char\_to\_int\_conversion(char \*hex\_char) {  
    int hex\_int = 0;  
    while (\*hex\_char) {  
  
        int current\_char = \*hex\_char++;   
          
    // converting hexadecimal character to the 4 bit equivalent number  
      
        if (current\_char >= '0' && current\_char <= '9')   
            current\_char = current\_char - '0';  
        else if (current\_char >= 'a' && current\_char <='f')               
            current\_char = current\_char - 'a' + 10;  
        else if (current\_char >= 'A' && current\_char <='F')           
            current\_char = current\_char - 'A' + 10;      
              
    // shift 4 to make space for new digit, and add the 4 bits of the new digit   
        hex\_int = (hex\_int << 4) | (current\_char & 0xF);  
    }  
    return hex\_int;  
}

Creating an array for physical addresses

int adress(void)  
{  
    for(int k=0;k<10;k++)  
    {      
        for(int j=0;j<10;j++)  
        {  
            physical[k][j]=virtual[k][j]+atoi\_hexa(base,0);  
        }      
              
    }

 int va\_i= char\_to\_int\_conversion(va); //Coverting the given virtual address to integer.  
    int flag = 0;  
    cprintf("\ngiven address:%x\n",va\_i);  
    for(int i=0;i<10;i++)  
    {  
        //cprintf("\nvirtual addresses%x",virtual[i]);

//Comparing if the input given matches any of the virual addresses

//If it matches then print the corresponding physical address

        if(va\_i==virtual[i])  
        {  
            cprintf("Matching\n");  
              
            cprintf("pa for user entered va is ------%x\n",(atoi\_hexa(va,0))+(atoi\_hexa(base,0)));  
            flag=1;  
            break;  
        }  
          
    }  
    cprintf("\n");  
    if(!flag)  
        {  
            cprintf("Invalid virtual adress entered by user\n");  
        }  
      
      
        return 1;      
}

1. Create file address\_1.c

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

{

adress(); //Function call for adress() defined in sysproc.

exit();

}

# References:

<http://www.cse.iitd.ernet.in/~sbansal/os/lec/l14.html>

Xv6 Russ Cox,Frans Kaashoek,Robert Morris