**MINI** **PROJECT** (COM-312)

Presented By Submitted To

Amjid Khan (2021A1R106) Arjumand Ayub (2021A1R111)

Ashu Pal (2021A1R116) Jatin Koul (2021A1R120)

Miss. Pragti Jamwal

**DEPARTMENT** **OF** **COMPUTER** **SCIENCE** **AND** **ENGINEERING** **MIET(Autonomous),** **JAMMU**

**Develop a code to convert virtual address to** **physical address**

**TABLE** **OF** **CONTENT**

v PROJECT TITTLE

v TABLE OF CONTENT v INTRODUCTON

v LOGICAL ADDRESS v PHYSICAL ADDRESS

v MEMORY MANAGEMENT UNIT (MMU)

v WHY WE NEED TO CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS v MAPPING VIRTUAL ADDRESS TO PHYSICAL ADDRESS

v CODE TO CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS v OUTPUTS

**INTRODUCTION**

**Memory** **is** **one** **of** **the** **most** **important** **host** **resources.** **For** **workloads** **to** **access** **global** **system** **memory,** **we** **need** **to** **make** **sure** **virtual** **memory** **addresses** **are** **mapped** **to** **the** **physical** **addresses.** **There** **are** **several** **components** **working** **together** **to** **perform** **these** **translations** **as** **efficient** **as** **possible.** **In** **this** **miniproject** **we** **develop** **a** **code** **to** **convert** **a** **virtual** **memory** **address** **to** **physical** **address.**

**What** **is** **logical** **address** **?**

v**Logical** **Address** is generated by CPU while a program is running. The logical address is virtual address as it does not exist physically, therefore, it is also known as Virtual Address. This address is used as a reference to access the physical memory location by CPU. The term Logical Address Space is used for the set of all logical addresses generated by a program’s perspective.

vThe hardware device called Memory-management Unit is used for mapping logical address to its corresponding physical address.

**PHYSICAL** **ADDRESS**

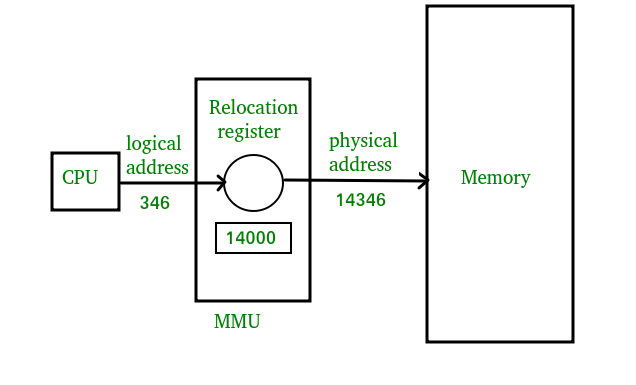
v **Physical** **Address** identifies a physical location of required data in a memory. The user never directly deals with the physical address but can access by its corresponding logical address. The user program generates the logical address and thinks that the program is running in this logical address but the program needs physical memory for its execution, therefore, the logical address must be mapped to the physical address by MMU before they are used. The term Physical Address Space is used for all physical addresses corresponding to the logical addresses in a Logical address space.

**mmU(memory** **management** **Unit)** **:**

vThe run time mapping between Virtual address and Physical Address is done by a hardware device known as MMU.

vIn memory management, the Operating System will handle the processes and move the processes between disk and memory for execution . It keeps track of available and used memory.

***Dynamic*** ***relocation*** ***using*** ***a*** ***relocation*** ***register.***

**1.CPU** **will** **generate** **logical** **address** **for** **eg:** **346**

**2.MMU** **will** **generate** **a** **relocation** **register** **(base** **register)** **for** **eg:** **14000**

**3.In** **memory,** **the** **physical** **address** **is** **located** **eg:** **(346+14000=** **14346)**

MMU Scheme:

CPU-----------MMU-----------Memory

**WHY** **WE** **NEED** **TO** **CONVERT** **VIRTUAL** **ADDRESS** **TO** **PHYSICAL** **ADDRESS?**

**To** **store** **the** **data** **and** **to** **manage** **the** **processes,** **we** **need** **a** **large-sized** **memory** **and,** **at** **the** **same** **time,** **we** **need** **to** **access** **the** **data** **as** **fast** **as** **possible.** **But** **if** **we** **increase** **the** **size** **of** **memory,** **the** **access** **time** **will** **also** **increase** **and,** **as** **we** **know,** **the** **CPU** **always** **generates** **addresses** **for** **secondary** **memory,** **i.e.** **logical** **addresses.** **But** **we** **want** **to** **access** **the** **main** **memory,** **so** **we** **need** **Address** **translation** **of** **logical** **address** **into** **physical** **address.**

**Mapping** **Physical**

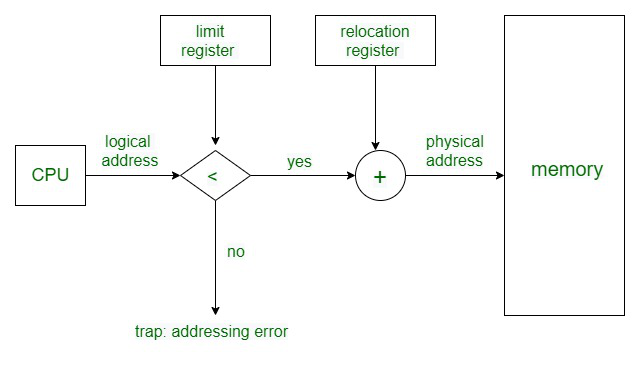
**Virtual** **Addresses** **to** **Addresses** **:**

In Contiguous memory allocation mapping from virtual addresses to physical addresses is not a difficult task, because if we take a process from secondary memory and copy it to the main memory, the addresses

will be stored in a contiguous manner, so if we know the base address of the process, we can find out the next addresses.

The Memory Management Unit is a combination of 2 registers – 1. Base Register (Relocation Register)

2. Limit Register.

**Mapping** **Physical**

**Virtual** **Addresses** **to** **Addresses** **:**

**Code** **to** **convert** **virtual** **address** **to** **physical** **address**

#include <stdio.h> #include <stdlib.h> #include <unistd.h> #include <string.h> #include <sys/mman.h> #include <sys/types.h> #include <sys/stat.h> #include <fcntl.h>

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

// Get the virtual address from the user printf("Enter a virtual address: ");

char input[128]; fgets(input, 128, stdin);

// Parse the virtual address from the input string

**CONTINUE…**

void \*virtual\_address = (void \*)strtoul(input, NULL, 16);

// Open the /proc/self/pagemap file

int pagemap\_fd = open("/proc/self/pagemap", O\_RDONLY); if (pagemap\_fd < 0) {

perror("Error opening /proc/self/pagemap"); return 1;

}

// Seek to the correct entry in the pagemap file

off\_t pagemap\_offset = (unsigned long)virtual\_address / getpagesize() \* 8;

if (lseek(pagemap\_fd, pagemap\_offset, SEEK\_SET) != pagemap\_offset)

**CONTINUE…** perror("Error seeking in /proc/self/pagemap");

return 1; }

// Read the entry from the pagemap file unsigned long pagemap\_entry;

if (read(pagemap\_fd, &pagemap\_entry, 8) != 8) { perror("Error reading from /proc/self/pagemap"); return 1;

}

// Extract the physical page number from the pagemap entry

unsigned long physical\_page\_number = pagemap\_entry & ((1ull << 55) - 1);

**CONTINUE…**

// Compute the physical address by adding the offset within the page

void \*physical\_address = (void \*)(physical\_page\_number \* getpagesize() +

(unsigned long)virtual\_address % getpagesize());

printf("Virtual address: %p\n", virtual\_address); printf("Physical address: %p\n", physical\_address);

return 0;

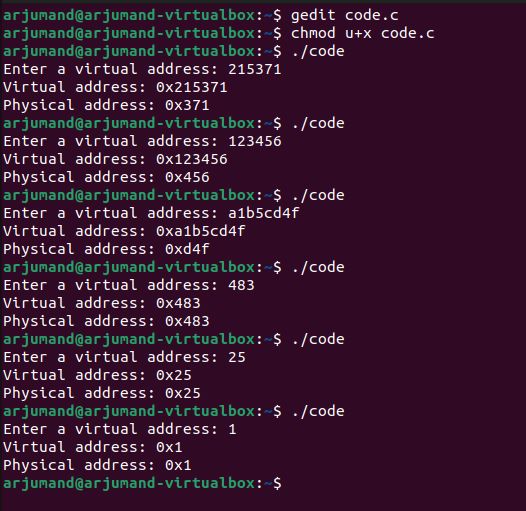
}

/\*This program prompts the user to enter a virtual address and then uses the /proc/self/pagemap file to look up the corresponding physical address. The /proc/self/pagemap file is a special file that is provided by the Linux kernel and contains the entries in the current process's page table.

Keep in mind that this is just one way to convert a virtual address to a physical address, and the exact method for doing so will depend on the specific operating system and hardware being used.

\*/

OUTPUTS:



***THANK*** ***YOU***