

Presented by : **1. Abhay Khajuria**

# Naman Bakshi

1. **Hardeep Singh**

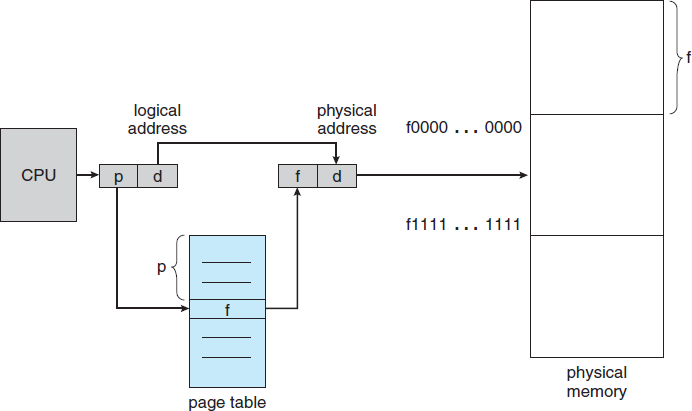
# Rishi Gupta

# Vansh Sharma

**IMPLEMENTATION OF PAGING TECHNIQUE OF MEMORY MANAGEMENT**

**INTRODUCTION TO PAGING**

* **Paging** is a memory-management scheme that permits the physical address space of a process to be noncontiguous. Paging avoids external fragmentation and the need for compaction. It also solves the considerable problem of fitting memory chunks of varying sizes onto the backing store; most memory management schemes used before the introduction of paging suffered from this problem.
* The problem arises because, when some code fragments or data residing in main memory need to be swapped out, space must be found on the backing store. The backing store has the same fragmentation problems discussed in connection with main memory, but access is much slower, so compaction is impossible. Because of its advantages over earlier methods, paging in its various forms is used in most operating systems.
* The basic method for implementing paging involves breaking physical memory into fixed-sized blocks called **frames** and breaking logical memory into blocks of the same size called **pages**.
* When a process is to be executed, its pages are loaded into any available memory frames from the backing store. The backing store is divided into fixed-sized blocks that are of the same size as the memory frames.
* The hardware support for paging is illustrated in figure below. Every address generated by the CPU is divided into two parts: a **page number (p)** and a **page offset (d)**. The page number is used as an index into a **page table**.
* The page table contains the base address of each page in physical memory. This base address is combined with the page offset to define the physical memory address that is sent to the memory unit.



## Paging hardware

* **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.   
  The hardware device called Memory-Management Unit is used for mapping logical address to its corresponding physical address.
* **Logical Address Space** is the set of all logical addresses generated by CPU for a program.

**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.

* **Physical address** mapped to corresponding logical addresses is called Physical Address Space

**EXAMPLE:**

* If Logical Address = 31 bit, then Logical Address Space = 231 words = 2 G words (1 G = 230)
* If Logical Address Space = 128 M words = 27 \* 220 words, then Logical Address = log2 227 = 27 bits
* If Physical Address = 22 bit, then Physical Address Space = 222 words = 4 M words (1 M = 220)
* If Physical Address Space = 16 M words = 24 \* 220 words, then Physical Address = log2 224 = 24 bits

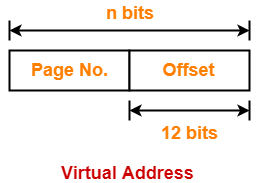
The **Mapping** from virtual to physical address is done by the memory management unit (MMU) which is a hardware device and this mapping is known as paging technique.

* The Physical Address Space is conceptually divided into a number of fixed-size blocks, called **frames**.
* The Logical address Space is also splitted into fixed-size blocks, called **pages**.
* Page Size = Frame Size

**CPU is divided into:**

**Page number(p):** Number of bits required to represent the pages in Logical Address Space or Page number.

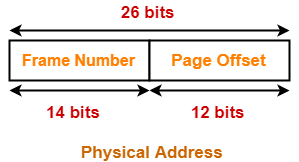
**Page offset(d):** Number of bits required to represent particular word in a page or page size of Logical Address Space or word number of a page or page offset.



**Physical Address is divided into:**

**Frame number(f):** Number of bits required to represent the frame of Physical Address Space or Frame number.

**Frame offset(d):** Number of bits required to represent particular word in a frame or frame size of Physical Address Space or word number of a frame or frame offset.



**HISTORY OF PAGING**

In the 1960s, swapping was an early virtual memory technique. An entire program or entire segment would be "swapped out" (or "rolled out") from RAM to disk or drum, and another one would be *swapped in* (or *rolled in*). A swapped-out program would be current but its execution would be suspended while its RAM was in use by another program; a program with a swapped-out segment could continue running until it needed that segment, at which point it would be suspended until the segment was swapped in.

A program might include multiple overlays that occupy the same memory at different times. Overlays are not a method of paging RAM to disk but merely of minimizing the program's RAM use. Subsequent architectures used memory segmentation, and individual program segments became the units exchanged between disk and RAM. A segment was the program's entire code segment or data segment, or sometimes other large data structures. These segments had to be contiguous when resident in RAM, requiring additional computation and movement to remedy fragmentation.

Ferranti's Atlas, and the Atlas Supervisor developed at the University of Manchester, (1962), was the first system to implement memory paging. Subsequent early machines, and their operating systems, supporting paging include the IBM M44/44X and its MOS operating system (1964), the SDS 940 and the Berkeley Timesharing System (1966), a modified IBM System/360 Model 40 and the CP-40 operating system (1967), the IBM System/360 Model 67 and operating systems such as TSS/360 and CP/CMS (1967), the RCA 70/46 and the Time Sharing Operating System (1967), the GE 645 and Multics (1969), and the PDP-10 with added BBN-designed paging hardware and the TENEX operating system (1969).

Those machines, and subsequent machines supporting memory paging, use either a set of page address registers or an in-memory page table to allow the processor to operate on arbitrary pages anywhere in RAM as a seemingly contiguous logical address space. These pages became the units exchanged between disk and RAM.

**CODE FOR PAGING**

#include<stdio.h>

int main()

{

printf("Paging technique of Memory management");

int memsize=15;

int pagesize,nofpage;

int p[100];

int frameno,offset;

int logadd,phyadd;

int i;

int choice=0;

printf("\nYour memsize is %d ",memsize);

printf("\nEnter page size:");

scanf("%d",&pagesize);

nofpage=memsize/pagesize;

for(i=0;i<nofpage;i++)

{

printf("\nEnter the frame of page%d:",i+1);

scanf("%d",&p[i]);

}

do

{

printf("\nEnter a logical address:");

scanf("%d",&logadd);

frameno=logadd/pagesize;

offset=logadd%pagesize;

phyadd=(p[frameno]\*pagesize)+offset;

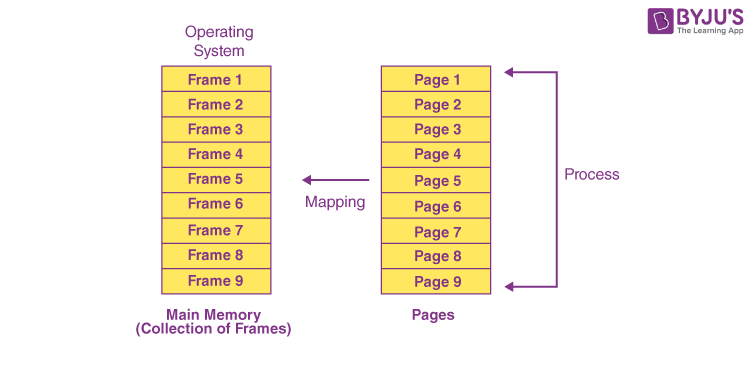
printf("\nPhysical address is:%d",phyadd);

printf("\nDo you want to continue(1/0)?:");

scanf("%d",&choice);

}while(choice==1);

}



**OUTPUT:**

Paging technique of Memory management

Your memsize is 15

Enter page size:10

Enter the frame of page1:10

Enter a logical address:552

Physical address is:2

Do you want to continue(1/0)?:0