## 7.1.2 Logical addresses Vs Physical addresses :

**The logical address :** It is the address generated by the CPU or user or programs. Examples of logical addresses are given below.

- Offset address generated by the CPU.
- Re-locatable addresses generated by compiler or assembler.
- Addresses generated by the linker.
- Address generated by the loader.

This logical address is also called as a virtual address. Now it is the duty of memory management module to converts virtual addresses into actual address that means physical address.

The physical address : This address is generated by memory management unit with the help of special hardware of modern processors or some programs. This address is generated by addition of logical address and some base address stored in base register as explained in figure 7.2.

- Base register is used to store starting address.
- Offset address is added to base address then physical address is created.

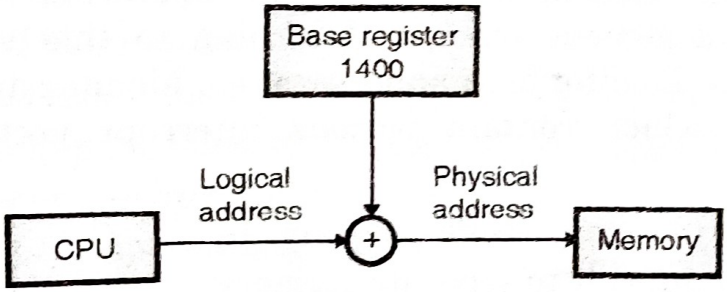



Fig. 7.2

## 7.1.3 Dynamic Linking Vs Static Linking :

Every program requires some library functions for execution, it is combined in the object code by loader or linker or linkage editor and is called as a linking. When it is linked at the time of compilation then it is called as static linking or early linking or compile time linking. When it is linked at the time of execution then it is called as a dynamic linking or late linking or run time linking. It is duty of memory management to provide this facility.

### 7.1.4 Overlays :

This is very good facility. This is also called as a dynamic loader. It uses overlay structure scheme. In order for the overlay structure to work it is necessary for the module loader to load the various procedures as they are needed. We will not go into their specific details, but there are many binders capable of processing and allocating an overlay structure.

The portion of loader that actually interprets the "calls" and loads the necessary procedure is called the overlay supervisor or simply the flipper. This overall scheme is called dynamic loading or load on call (LOCAL).

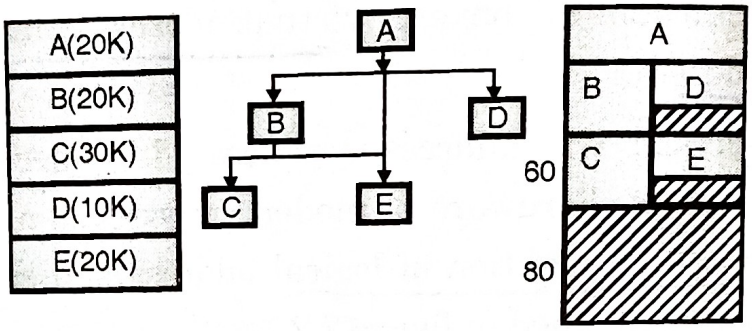



**Fig. 7.3 : Dynamic loading**

### 7.1.5 Resident Monitor :

We can say that resident monitor is first operating system which uses simplest memory management scheme. According to this scheme memory is divided in to two parts. Monitor area and user area. Monitor area is used to store program of monitor which contain various interrupt vectors and resource controlled functions.

User area is used to store user's program. In early days it was very difficult to separate memory into two partitions. So they used hardware lock in between monitor area and user area. So if user's program was trying to access monitor area it was not allowed to do so by the monitor. This hardware lock is called as a protection hardware.

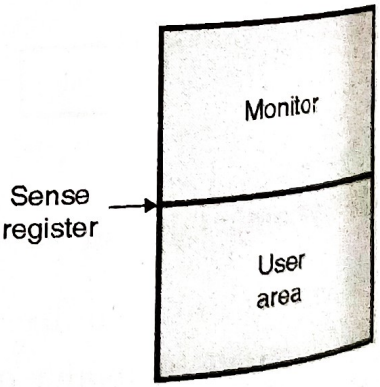


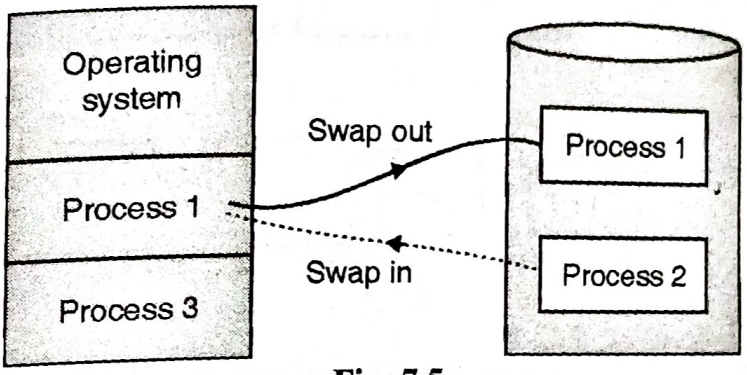

**Fig. 7.4**

### 7.1.6 Swapping :

All modern memory management module uses swapping techniques. This is divided in to two part Swap in and Swap out. In this scheme when CPU executes the process, then some part of the process is loaded into the main memory and remaining part is stored in to secondary memory.

During execution if CPU needs remaining part of the process from secondary memory then memory management loads it from secondary memory to main memory. This process is called as a swap in process.

Sometimes there is no main memory available so memory management is required to free some used memory. Hence it stores process of that main memory to secondary memory this process is called as a swap out process. This is explained in the following Fig.7.5




B.J.S. College Library

**Fig. 7.5**

Sometimes memory management can skip this swap out process. One copy of each process is already available in secondary memory if there is no change in copy of main memory then it not required to store that process in to the secondary memory. We cannot skip swap in process.

- **Swap time :** Time required for swapping process is called as swap time. If swap time increases then throughput of the operating system decreases. Generally swapping is done at the time of context switching. So if swap time increases context switching time is also increases. So for increasing performance of the operating system, it needs to use overlapped swapping.
- **Overlapped swapping :** In this scheme, when operating system performs the swap out process, then simultaneously executes some other programs. After swap out immediately it perform swap in process as well as it also executes users program, like I/O operation.

### 7.1.7 Memory Management Schemes :

There are different memory management scheme they are mainly divided into two categories Contiguous Memory Management Scheme and Non Contiguous Memory Management Scheme. They are as follows:

## 7.8 Demand Paging :

Demand paging is very similar to paging so it supports non contiguous memory allocation. Here memory is divided into equal sizes of the pieces called as a frames, Job is also divided into same sizes like as a frame called as a pages. When each job is stored into memory then it is identified as a process CPU scheduler creates PCB for each process. Similarly paged memory management creates separate page table for each process and store it into its PCB.

This algorithm stores only some pages of the job into main memory. And it keeps remaining pages in the secondary memory. Actually it kept all pages of the job in to main memory. When during the execution of such process if it requires any page which is not available in the main memory, then it is called as page faults.

It handles page faults by using following steps.

1. It checks whether there is any free frame or not? If yes then go to step 2 else step 3.
2. If there is no free frame then select victim page with the help of page removal algorithm. And make that frame free using swap out process. Means it store the content of that page from main memory to back, into secondary memory.
3. Free frame is used to load, that means swap in, required page from secondary memory to main memory.

There is separate process which can perform swap in swap out process. That process is called as swapper. This swap can be reduced using dirty bit concept

**Dirty bit concept :** It is given below.

- A bit is associated with each frame called as a dirty bit.
- When frame is loaded with any page of secondary memory then memory management monitors if page is modified during execution or not?
- If that page is modified then only it set dirty bit to 1, otherwise not.

This dirty bit is checked by the swapper process that is as follows;

- It checks if dirty bit is one or not? If it is one then it performs swap out process because copy of secondary memory is different than page in main memory,
- If dirty bit is not set then it will not perform swap out process because there are same copy of page is presents in main memory as well as in secondary memory.

**Copy on write process is explained in below :**

- When any process creates child process using fork() system called duplicate copy of page is allocated for parent as well as child.
- Copy on write permits share pages by parent as well as child process.
- There pages are called as a copy on write pages.
- Once any page is marked as a copy on write page then operating system give another copy of that page to childe process.
- Now child and parent not mix their data into each other

**Page table :**

It is used to store two types of information that is page number and frame number where that page is stored into the main memory as well as it also stores information of availability of that page.

**Page Table**

| Page number | Frame number | Status bit |
|---|---|---|
| | | |

This demand paged memory management maintain frame table which is used to store information of each and every frame like frame number, page number, process ID, status of frame and other necessary information. It is very essential for page and frame management.

**Frame Table**

| Frame number | Page number | Process ID | Status | Other information |
|---|---|---|---|---|
| | | | | |

All modern processors also support paging because the address generated by the CPU is divided into two parts that are as follows;

1. Page number 2. Page offset

This address is called as a logical address. Now paged memory management converts this logical address into physical address with the help of page table of that process. Here Physical address is divided into two parts that are as follows;

1. Frame number
2. Frame offset

Which is already explained in Fig. 7.9.

Following steps are used to calculate physical address from logical address.

1. Find out page number and offset address from the logical address

2. Find out frame number with the help of page table of current process and page number generated in logical address.
3. Find out base address of the frame=frame number * frame size
4. Check offset address is greater than frame length then generate trap.
5. Calculate physical address = base address of the frame + offset address the logical address.

**Data structure :**