PAGING

Paging:

Most virtual system uses a techniques called paging that permits the physical address space of a process to be non-contiguous. These program-generated addresses are called **virtual addresses** and form the **virtual address space.**

On computers without virtual memory, the virtual address is put directly onto the memory bus and causes the physical memory word with the same address to be read or written. When virtual memory is used, the virtual addresses do not go directly to the memory bus. Instead, they go to an MMU (Memory Management Unit) that maps the virtual addresses onto the physical memory addresses as illustrated in Fig

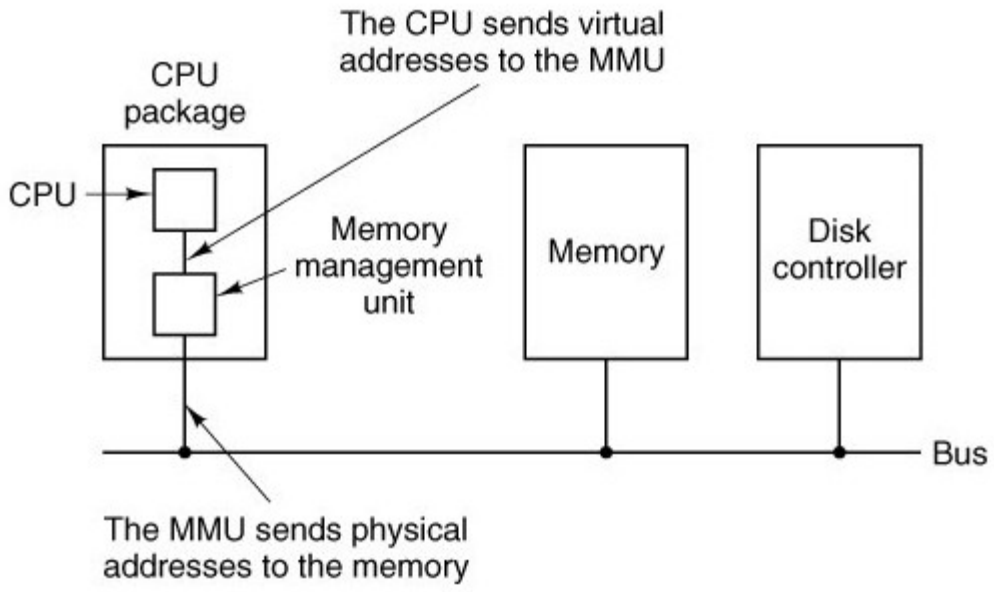

Fig:The position and function of the MMU. Here the MMU is shown as being a part of the CPU chip because it commonly is nowadays

The basic method for implementing paging involves breaking physical memory into fixed size block called **frames** and breaking logical memory into blocks of the same size called **pages.** size is power of 2, between 512 bytes and 8,192 bytes

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-size block that are of the same size as memory frames.

Backing store(2) is typically part of a hard disk that is used by a paging or swapping system to store information not currently in main memory. Backing store is slower and cheaper than main memory.

A very simple example of how this mapping works is shown in Fig. below. In this example, we have a computer that can generate 16-bit addresses, from 0 up to 64K. These are the virtual addresses. This computer, however, has only 32 KB of physical memory, so although 64-KB programs can be written, they cannot be loaded into memory in their entirety and run.

With 64 KB of virtual address space and 32 KB of physical memory, we get 16 virtual pages and 8 page frames. Transfers between RAM and disk are always in units of a page.
When the program tries to access address 0, for example, using the instruction

MOV REG,0

virtual address 0 is sent to the MMU. The MMU sees that this virtual address falls in page 0 (0 to 4095), which according to its mapping is page frame 2 (8192 to 12287). It thus transforms the address

to 8192 and outputs address 8192 onto the bus. The memory knows nothing at all about the MMU and just sees a request for reading or writing address 8192, which it honors. Thus, the MMU has effectively mapped all virtual addresses between 0 and 4095 onto physical addresses 8192 to 12287.

Similarly, an instruction **MOV REG,8192** is effectively transformed into **MOV REG,24576.** because virtual address 8192 is in virtual page 2 and this page is mapped onto physical page frame 6 (physical addresses 24576 to 28671). As a third example, virtual address 20500 is 20 bytes from the start of virtual page 5 (virtual addresses 20480 to 24575) and maps onto physical address 12288 + 20 = 12308.

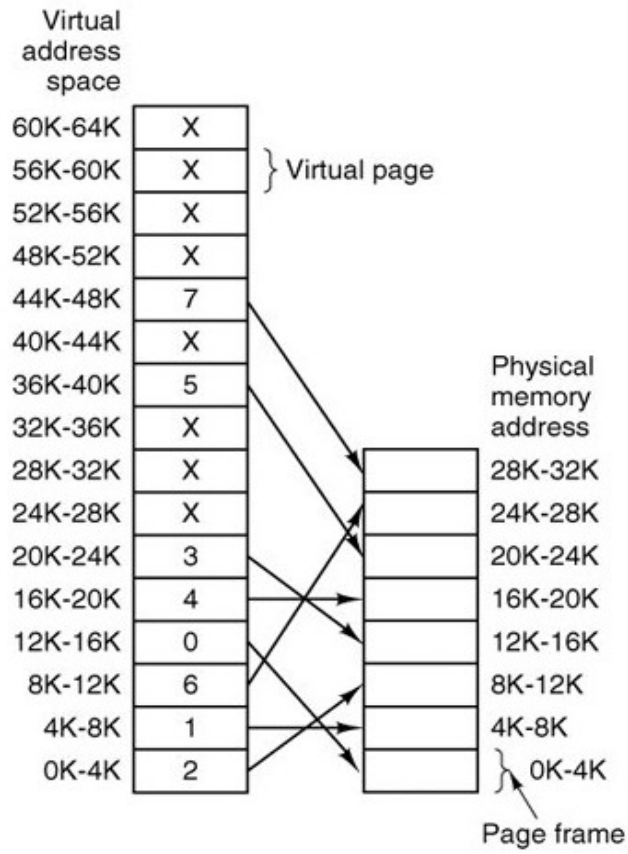

Fig:The relation between virtual addresses and physical memory addresses is given by the page table.