Virtual Memory - Sean Corzo 8/14/2023

In a system that uses paging for memory management (such as many modern operating systems), the goal is to provide the illusion of a larger amount of physical memory (main memory) than is physically available by using virtual memory.

Here's how it works:

Virtual Addresses: Processes operate using virtual addresses, which are addresses within their own virtual address space. This address space is divided into fixed-size blocks called pages. Each process interacts with memory using these virtual addresses, and these addresses are used in the code and data instructions of the program.

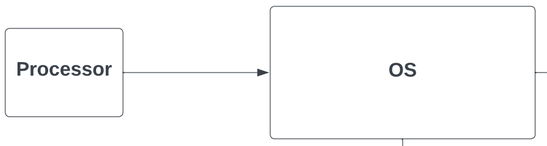
Virtual Page Numbers: When a process references a memory location using a virtual address, the virtual address is divided into two parts: the virtual page number and the page offset. The virtual page number identifies which page within the process's virtual address space is being accessed.

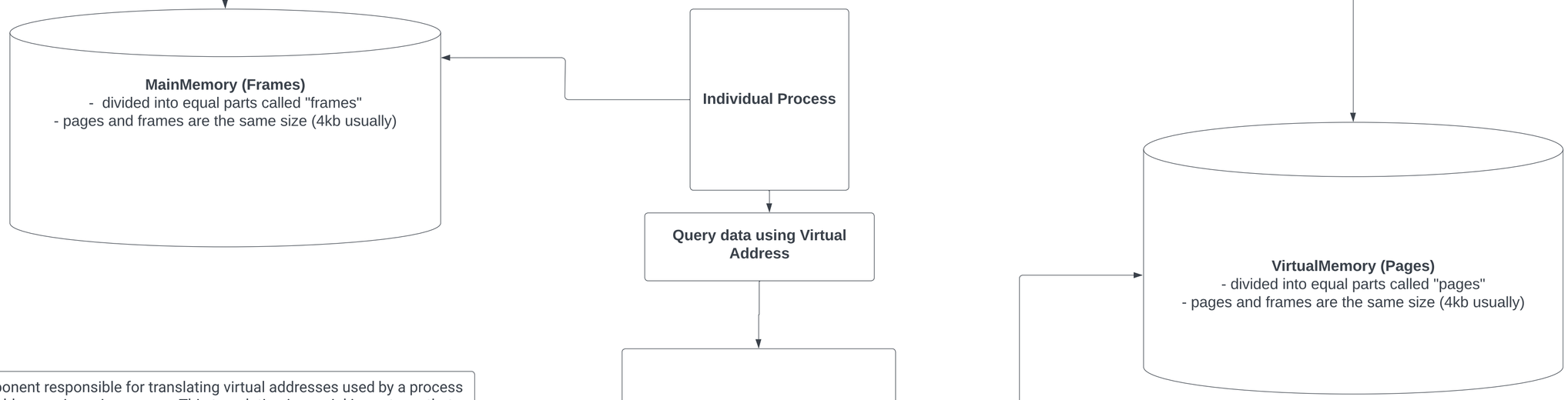
Paging Tables: The operating system maintains paging tables that map virtual page numbers to corresponding physical frame numbers in main memory. When a process uses a virtual address, the memory management unit (MMU) of the system uses the paging tables to translate the virtual page number to the corresponding physical frame number.

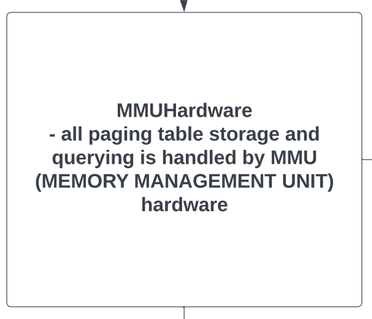
Physical Addresses: Once the MMU translates the virtual page number to a physical frame number using the paging tables, the process's virtual page offset is combined with the physical frame number to generate the actual physical memory address where the data is stored in main memory.

Page Fault:

- processor notifies OS that frame address is missing for a requested page
- handled by OS
- OS retrieves page from secondary storage (virtual memory)
- OS brings missing page into a free FRAME in main memory
- page table gets updated with new mapping
- initial CPU instruction is rerun since the missing item is now in main memory







The MMU is a hardware component responsible for translating virtual addresses used by a process into corresponding physical addresses in main memory. This translation is crucial in systems that use virtual memory and paging.

Here's the general sequence of events when a process accesses data and how the MMU is involved:

Process Accesses Data: When a process running on the CPU accesses data in memory, it uses a virtual memory address. This virtual address corresponds to a location in the process's virtual address space.

Virtual Address Translation: The process's virtual address is divided into a virtual page number and a page offset. The virtual page number indicates which page in the process's address space the data is located in.

MMU Translation: The MMU takes the virtual page number and uses it to look up the corresponding physical frame number in the paging tables. This translation ensures that the virtual memory address used by the process is mapped to a location in the physical memory (main memory).

Physical Address Generation: Once the MMU has translated the virtual page number to the physical frame number, it combines this physical frame number with the page offset from the original virtual address. This combination generates the actual physical memory address where the data is stored in main memory.

Data Access: With the physical memory address generated, the CPU can access the data directly from the main memory location corresponding to the physical address. The data can then be used by the process as needed.

PagingTable

Each process has its own separate paging table

Stored in main memory

Contains mapping of page index to frame address in main memory

PAGE FAULT: Paging table frame value may be empty in which case the contents must be retrieved from virtual memory - this is called a PAGE FAULT