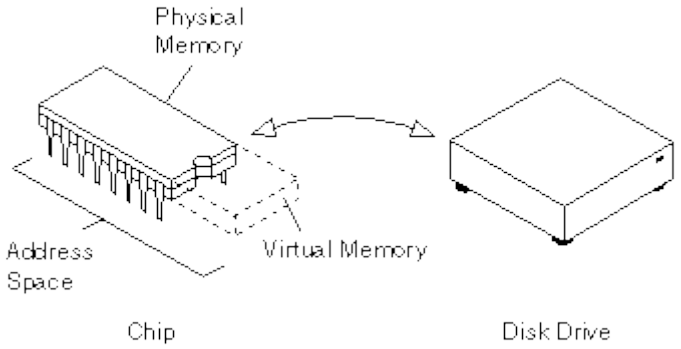
Escape from Reality



You have probably never heard of this type of RAM while taking a stroll through the aisle of your local computer store. Well quick rush out and buy as much of this new futuristic memory. WAIT WAIT!! Just Kidding. Actually ***Virtual Memory*** has been around for more than 20 years. You may have never heard of it because its something that you do not buy. It comes in two parts, a large secondary storage device, i.e ***Hard Disk Drive***, and an ***Operating System***, i.e ***Windows 95***™.

Virtual memory is a technique that allows instructions executed by a program that may not be able to fit entirely in main memory. When an instruction is needed by the program it is transferred from virtual memory to physical. The advantage virtual memory gives a programmer is that he needs to be aware of only the **LOGICAL** memory space. The **LOGICAL** memory space is a combination of physical memory and the hard disk space acting as memory.

Methods for Data Transfer

Since a program cannot actually access and manipulate data within the virtual memory, there are two methods for transferring the data from virtual memory to physical.

1. **Paging**

   In this method the data being swapped is a fixed size. Paging uses a single address space. When the program needs data that is in virtual memory, the operating system copies a certain number of pages, blocks of data, from your hard disk to main memory. When the requested data is not in main memory, the operating system goes to the hard disk and copies the correct page into main memory. Each time a page is needed that is not currently in memory, a page fault occurs. An invalid page fault occurs when the address of the page being requested is invalid. In this case, the application is usually aborted. An invalid page fault is one of the reasons computers crash.

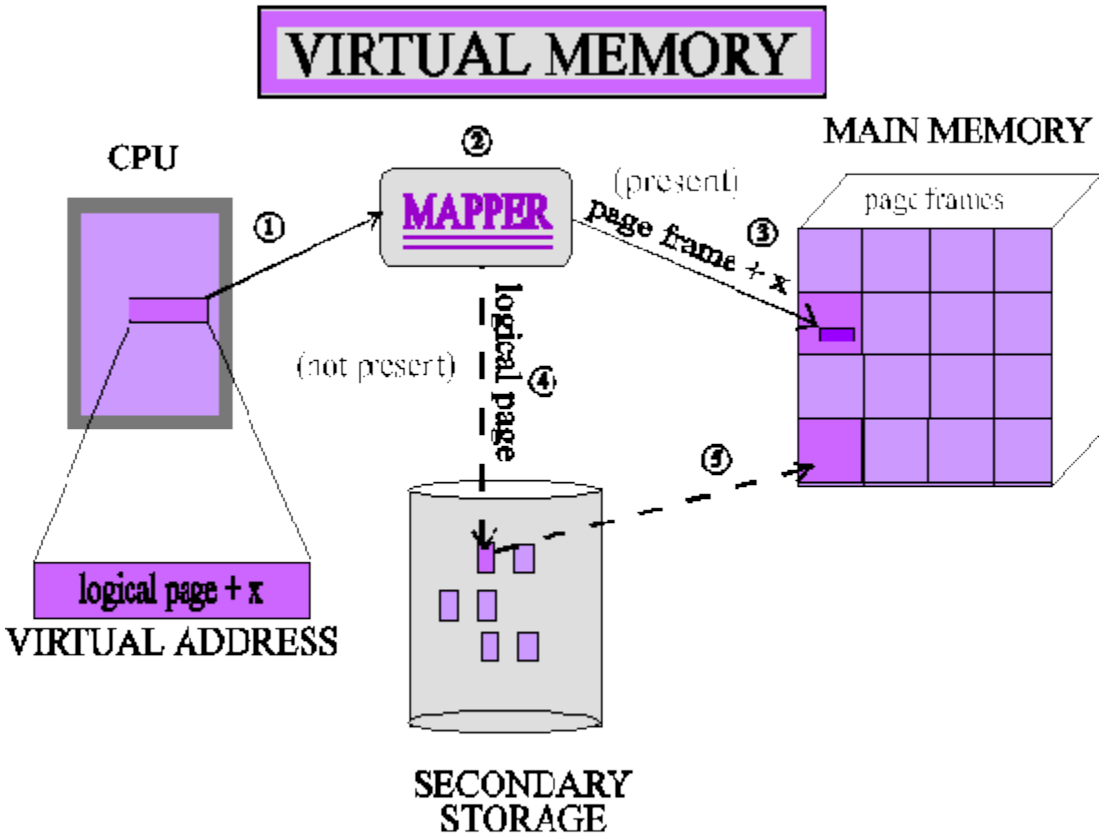
2. **Segmentation**

   This method is uses variable sized memory blocks. This method is useful because it is very common for the size of objects in a program to change dynamically. If a single address space is used, as in the paging, after the memory has been allocated it cannot change size. Resulting in wasted memory or not enough memory. To fix this problem the computer system would set up many independent address spaces. Each of these address spaces is called a segment. However, unlike paging, the programmer actively maintains the segments.

Here is a table comparing the two methods:

	Paging	**Segmentation**
Words per address	One	Two
Programmer Visible?	Invisible to application programmer	May be visible to application programmer
Replacing a block	Trivial (all blocks are the same size)	Hard (must find contiguous, variable-size, unused portion of main memory)
Memory use inefficiency	Internal fragmentation (unused portion of page)	External fragmentation (unused portion of main memory)
Efficient disk traffic	Yes (adjust page size to balance access time and transfer time)	Not always (small segments may transfer just a few bytes)

Mapping out your future...memory access



The above diagram shows one of the most important part of virtual memory implementation. The mapper is responsible for the translation of the virtual address requested from the program to a physical address that is kept in main memory. The what makes the mapper able to translate the addresses is a page table. This table identifies where all the pages of program are stored in main memory. If the mapper is unable to locate the page in main memory it creates a page fault. The page fault alerts the operating system that the data could not be found in main memory. At this point execution of the program is suspended. As the user you barely notice this stoppage, but to the computer it takes way too much time. What the computer is doing at this stage is going back to the mapper and telling it to find the desired data in secondary storage. To keep the computer running as fast as possible there has to be a way to minimize the time spent retrieving the necessary data.

Virtual Memory in the Buff

The key to the magic, its not magic its computer science, of virtual memory is a small piece of hardware called the translation look-aside buffer or TLB. The TLB is simply a cache with a special purpose; a purpose more important than the other types of cache you may have read about already (biased statement). This cache is fully associative, meaning data can be placed anywhere within the cache. Each entry in the TLB holds a pieces of information telling the

computer where to find the requested data. One piece is called the tag. This is where the virtual address is kept. The mapper sends the requested address to the TLB and the address is compared with all the tags in the TLB. If the correct one if found the second piece in the entry is activated. The second piece of information contains the physical address where the data can be found. At this point the page that is stored on the secondary storage gets swapped with the page that is in main memory. Now that the correct data has been sent to the main memory the program can now continue executing. The TLB allows the computer a very fast way of determining where the data that is not contained in memory is stored and since the address is retained in the cache its extremely easy for future requests of the same address.

Conclusion

While the need to be knowledgeable on virtual memory is not necessary to be a savvy computer memory shopper, it helps to be aware of another factor determining the overall performance of your computer. If you want your computer to be in top form, having to rely on virtual memory too often is not the way to go. On the other hand, almost all operating systems use virtual memory no matter how much main memory you have. Therefore, you would like know why there is a performance hit and what your operating system is doing under the cover. Well, just so you feel that you hadn't wasted your time and might have learned something its time slightly tax your brain.

Q & A for the Extremely Bored

1. What would be the reason a programmer would like a paged virtual memory?
2. Since TLBs are fairly expensive they only contain the most frequently accessed pages. The percentage of times a page is found in the TLB is called the Hit Ratio. The access time of the virtual memory can be calculated by multiplying the hit ratio by the access time using the TLB and adding (1 - hit ratio) times the access time using the main memory page table. Using the page table requires an extra access to main memory. The total is then compared to the time for a simple access to the main memory. If I give you an example it will be too easy but...

Example: If the access time for main memory is 120 nanoseconds and the access time for the TLB is 15 nanoseconds and the hit ratio is 85 percent, then access time = .85 x (15 + 120) + (1 - .85) x (15 + 120 + 120) = 153 nanoseconds. Since the simple access time is 120 nanoseconds, this represents a slowdown of 27 percent compared to the simple main memory access.

What would the slowdown be if the hit ratio were increased to 95 percent?

The Answers

1. A programmer would like a paged virtual memory because the system is transparent to the programmer. The programmer would have to make specific adjustments while coding for a segmented virtual memory.

2. Access time = .95 x (15 + 120) + .05 x (15 + 120 + 120) = 141 nanoseconds. This is a slowdown of only 17 percent. As you can see the higher the hit rate in the TLB the faster virtual memory becomes.