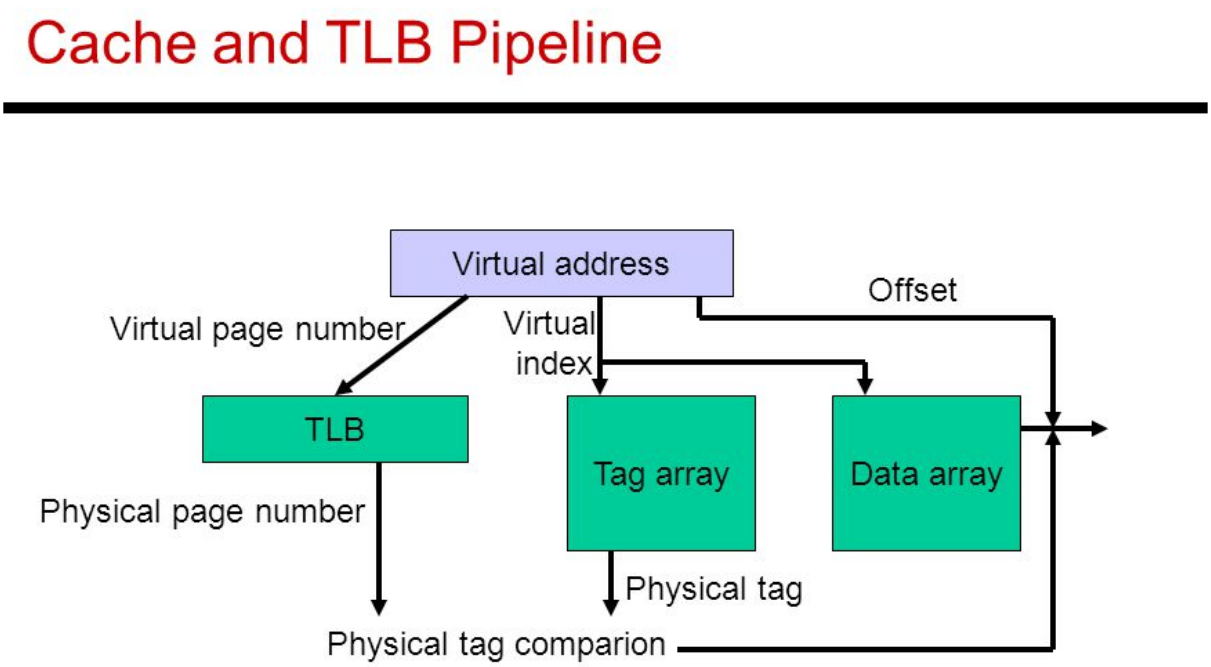
KEPHA MOSE

CS 6210-001

Homework

1. Consider a processor that supports virtual memory. It has a virtually indexed physically tagged cache, TLB, and page table in memory. Explain what happens in such a processor from the time the CPU generates a virtual address to the point where the referenced memory contents are available to the processor.




Once the CPU generates a virtual address, we break it down into the cache offset, index and tag, but we don't use the tag. We do use the index bits from the virtual address to find the set that we want in the cache. We read the valid bits and the tags here. But meanwhile, we use the page number from the virtual address to access the TLB and get the frame number. So now that we got the physical address, we perform the tag check using the physical address not the virtual one. This is why this is called a virtual index. The index bits come from the virtual address. The tag bits come from the physical address, cache. And from here, it proceeds normally, we know whether we have a hit and so on.

2. Distinguish between segmentation and paging.**Segmentation**Segmentation is a memory-management scheme where a logical address space is a collection of segments. Segmentation permits the physical address space of a process to be noncontiguous. Each segment has a name and a length. The address specify both the segment and the offset within the segment. The logical address consists of a segment number and an offset.**Paging**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 their

source (a file system or the backing store). The backing store is divided into fixed-sized blocks that are the same size as the memory frames or clusters of multiple frames. However, paging avoids external fragmentation and the need for compaction, whereas segmentation does not. A logical address is divided into two part: a page number and a page offset.

3. Explain all the actions from the time a process incurs a page fault to the time it resumes execution. Assume that this is the only runnable process in the entire system.
    1. We check an internal table (usually kept with the process control block) for this process to determine whether the reference was a valid or an invalid memory access.
    2. If the reference was invalid, we terminate the process. If it was valid but we have not yet brought in that page, we now page it in.
    3. We find a free frame (by taking one from the free-frame list, for example).
    4. We schedule a disk operation to read the desired page into the newly allocated frame.
    5. When the disk read is complete, we modify the internal table kept with the process and the page table to indicate that the page is now in memory.
    6. We restart the instruction that was interrupted by the trap. The process can now access the page as though it had always been in memory.
4. Explain the following terms: working set of a process, thrashing, paging daemon, swapper, loader, and linker.**Working set**The set of pages that a process is currently using.**Thrashing**A program causing page faults every few instructions is said to be thrashing. A process is thrashing if it is spending more time paging than executing.**Paging daemon**Is a background process that periodically to inspects the state of memory and If too few page frames are free, it begins selecting pages to evict using some page replacement algorithm. If these pages have been modified since being loaded, they are written to disk.**Swapper**Is a process used by early UNIX systems to move entire processes between memory and disk whenever not all active processes could fit in the physical memory.**Loader**A loader is a part of the operating system responsible combining a number of program segments that have been independently compiled into an executable program. It does this by binding the relocatable addresses to absolute addresses. **Linker**A linker or link editor is a computer program that takes one or more object files generated by a compiler and combines them into a single executable file, library file, or another object file.
5. Explain page coloring and how it may be used in memory management by an operating system. Page coloring is a performance optimization designed to ensure that accesses to contiguous pages in virtual memory make the best use of the processor cache. Physical memory pages are "colored" so that pages with different "colors" have different positions in CPU cache memory. When allocating sequential pages in virtual memory for processes, the kernel collects pages with different "colors" and maps them to the virtual memory. In this way, sequential pages in virtual memory do not contend for the same cache line.
6. Explain clearly the costs associated with a process context switch. The cost of context switch may come from several aspects. The processor registers need to be saved and restored, the OS kernel code (scheduler) must execute, the TLB entries need to be reloaded, and processor pipeline must be flushed. These costs are directly associated with almost every context switch in a multitasking system. We call them direct costs. In addition, context switch leads to cache sharing between multiple processes, which may result in performance degradation. This cost varies for different workloads with different memory access behaviors and for different architectures. We call it cache interference cost or indirect cost of context switch.
7. Explain the functionality of the different layers found in the network protocol stack of an operating system such as Linux.
    1. **Layer 1: Physical layer.** The physical layer is responsible for handling both the mechanical and the electrical details of the physical transmission of a bit stream. At the physical layer, the communicating systems must agree on the electrical representation of a binary 0 and 1, so that when data are sent as a stream of electrical signals, the receiver is able to interpret the data properly as binary data. This layer is implemented in the hardware of the networking

device. It is responsible for delivering bits.
2. **Layer 2: Data-link layer.** The data-link layer is responsible for handling frames, or fixed-length parts of packets, including any error detection and recovery that occurs in the physical layer. It sends frames between physical addresses.
3. **Layer 3: Network layer.** The network layer is responsible for breaking messages into packets, providing connections between logical addresses, and routing packets in the communication network, including handling the addresses of outgoing packets, decoding the addresses of incoming packets, and maintaining routing information for proper response to changing load levels. Routers work at this layer.
4. **Layer 4: Transport layer.** The transport layer is responsible for transfer of messages between nodes, including partitioning messages into packets, maintaining packet order, and controlling flow to avoid congestion.
5. **Layer 5: Session layer.** The session layer is responsible for implementing sessions, or process-to-process communication protocols.
6. **Layer 6: Presentation layer.** The presentation layer is responsible for resolving the differences in formats among the various sites in the network, including character conversions and half duplex–full duplex modes (character echoing).
7. **Layer 7: Application layer.** The application layer is responsible for interacting directly with users. This layer deals with file transfer, remote-login protocols, and electronic mail, as well as with schemas for distributed databases.