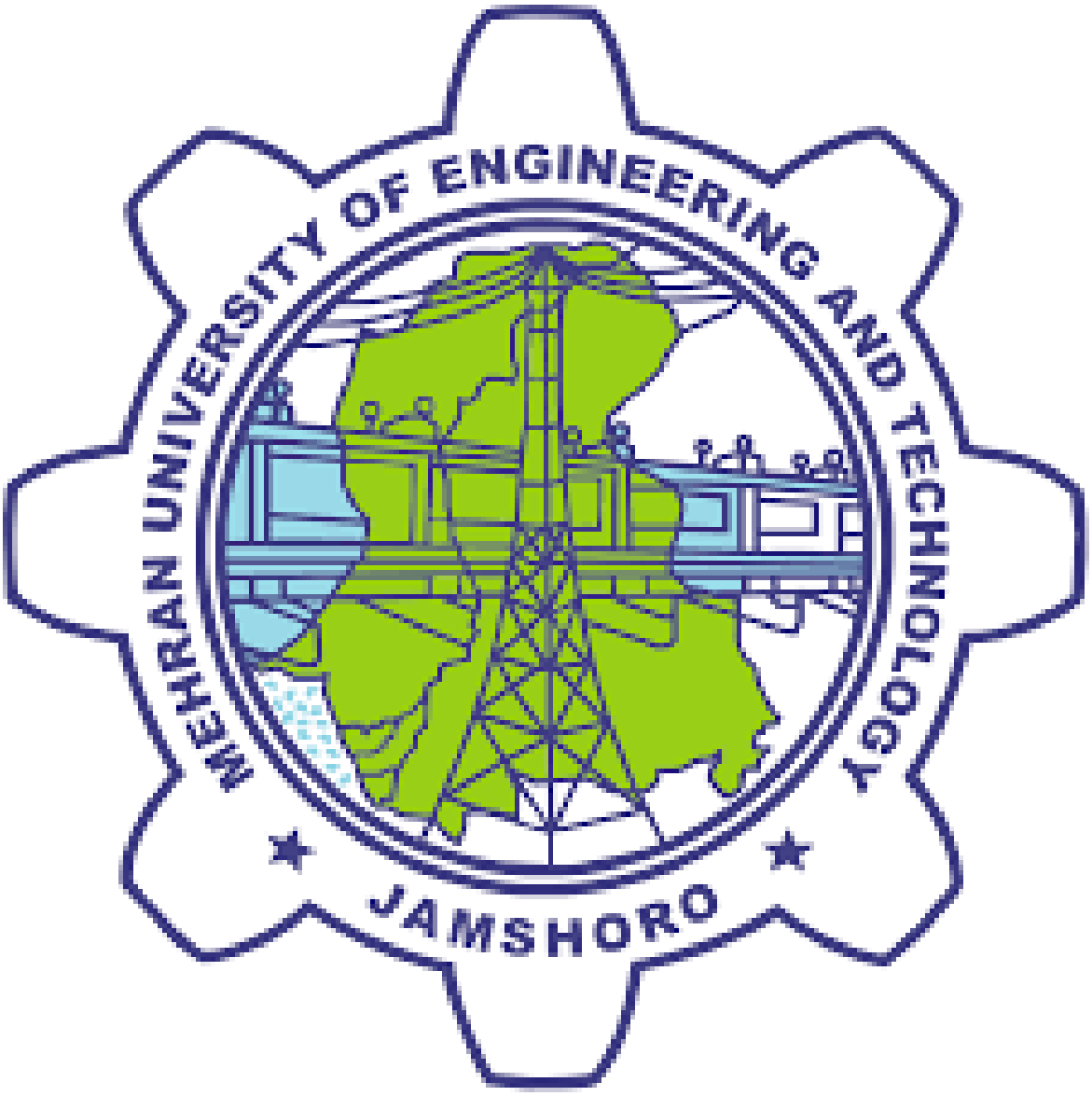
Name: ZOHAIB HASSAN SOOMRO

RollNo#: 19SW42

Subject: Operating Systems



Address-space switch

On an address-space switch, as occurs on a process switch but not on a thread switch, some TLB entries can become invalid, since the virtual-to-physical mapping is different. The simplest strategy to deal with this is to completely flush the TLB. This means that after a switch, the TLB is empty, and *any* memory reference will be a miss, so it will be some time before things are running back at full speed. Newer CPUs use more effective strategies marking which process an entry is for. This means that if a second process runs for only a short time and jumps back to a first process, the TLB may still have valid entries, saving the time to reload them

Other strategies avoid flushing the TLB on a context switch: (a) A single address space operating system uses the same virtual-to-physical mapping for all processes. (b) Some CPUs have a process ID register, and the hardware uses TLB entries only if they that match the current process ID.

For example, in the Alpha 21264, each TLB entry is tagged with an *address space number* (ASN), and only TLB entries with an ASN matching the current task are considered valid. Another example in the Intel Pentium Pro, the page global enable (PGE) flag in the register CR4 and the global (G) flag of a page-directory or page-table entry can be used to prevent frequently used pages from being automatically invalidated in the TLBs on a task switch or a load of register CR3. Since the 2010 Westmere microarchitecture Intel 64 processors also support 12-bit *process-context identifiers* (PCIDs), which allow retaining TLB entries for multiple linear-address spaces, with only those that match the current PCID being used for address translation

While selective flushing of the TLB is an option in software-managed TLBs, the only option in some hardware TLBs (for example, the TLB in the Intel 80386) is the complete flushing of the TLB on an address-space switch. Other hardware TLBs (for example, the TLB in the Intel 80486 and later x86 processors, and the TLB in ARM processors) allow the flushing of individual entries from the TLB indexed by virtual address.

Flushing of the TLB can be an important security mechanism for memory isolation between processes to ensure a process can't access data stored in memory pages of another process. Memory isolation is especially critical during switches between the privileged operating system kernel process and the user processes – as was highlighted by the Meltdown security vulnerability. Mitigation strategies such as kernel page-table isolation (KPTI) rely heavily on performance-impacting TLB flushes and benefit greatly from hardware-enabled selective TLB entry management such as PCID

Virtualization and x86 TLB

With the advent of virtualization for server consolidation, a lot of effort has gone into making the x86 architecture easier to virtualize and to ensure better performance of virtual machines on x86 hardware.

Normally, entries in the x86 TLBs are not associated with a particular address space; they implicitly refer to the current address space. Hence, every time there is a change in address space, such as a context switch, the entire TLB has to be flushed. Maintaining a tag that associates each TLB entry with an address space in software and comparing this tag during TLB lookup and TLB flush is very expensive, especially since the x86 TLB is designed to operate with very low latency and completely in hardware. In 2008, both Intel (Nehalem)and AMD (SVM) have introduced tags as part of the TLB entry and dedicated hardware that checks the tag during lookup. Even though these are not fully exploited, it is envisioned that in the future, these tags will identify the address space to which every TLB entry belongs. Thus a context switch will not result in the flushing of the TLB – but just changing the tag of the current address space to the tag of the address space of the new task.