Solution notes

Comparative Architectures 2005 – Paper 8 Question 1 (IAP)

All modern processors provide support for memory access protection and translation. Describe the Translation Lookaside Buffer (TLB) architecure of a modern microprocessor, and hence what information a typical TLB entry would contain. [6 marks]

- virtual addresses must be converted into physical addresses and access checks performed. The TLB is effectively a cache of V to P translations that is consulted whenever a load, store or instruction fetch is performed. If a cache miss occurs, some mechanism must be used to query the OS virtual memory data structures to determine whether the virtual address is valid in the current execution context, and what the translated physical address should be. If a valid translation exists it will be loaded into the TLB (evicting some other entry) and the instruction replayed. If no valid translation exists an exception will be generated.

- TLB entries will be indexed by: virtual frame number (the offset within the page is ignored), and possible an address space identifier.

- TLB entries will contain e.g.: a bit to indicate whether the entry is valid, translated physical frame number, bits to indicate whether the page is read only or read-write, and whether it is user or supervisor-only accessible.

- Separate instruction and data TLBs are used (for the same reasons as there are separate L1 caches). The I&D paths are naturally separate and the overlap between entries would probably be relatively small.

- Some implementations have a L2 joint TLB (shared between I&D), which can have more entries, but is typically slower to access.

Some architectures are described as having software-managed TLBs, whereas others have entries loaded entirely by hardware. Describe and contrast the two approaches. [6 marks]

- Software managed TLB: generate exception on a TLB miss, leave it to the OS to consult its virtual memory structures then use a special instruction to load the TLB withe the new translation (or generate exception). The hardware may provide assistance to nominate a suitable entry to evict (e.g. LRU)

- Hardware managed TLB: upon a TLB miss, the hardware consults some architecture-defined data structure to determine whether a valid translation exists. Typically, the data structure is a multi-level pagetable, where the msb's of the virtual address are used to index a top-level table (contained entirely within one page) that contains an entry that points to the next table that is used to resolve the next msb's. Assuming a valid translation exists, at the leaf level there will be the physical frame number for the page along with the access and protection information.

- Software managed TLBs provide more flexibility in the data structure used to hold the VM information, but handling TLB misses is typically much slower. Hardware page table

walking uses extra hardware, but can be very fast, particularly if intermediate levels of the lookup are cached.

Why might a TLB that supports 'superpages' (entries that cover multiple pages) benefit applications that use lots of memory? What steps must the operating system take to enable superpages to be used? [4 marks]

- Applications that access lots of memory in a 'random' fashion can cause the TLB to thrash (have insufficient entries to map all of the active pages), leading to a dramatic loss of performance.

- If aligned regions of virtual address space map to consecutive physical pages, it may be possible to use superpages, hence reducing the number of active TLB entries that are required.

- The OS must be employing a memory allocator that tries to avoid fragmentation of physical memory, hence enabling applications to be allocated contiguous ranges of physical address space.

Even hardware-filled TLBs usually rely on software to determine when entries should be invalidated. How might you design a hardware soloution to automatically keep the TLB coherent with OS pagetables? What extra complications would Symetric Multiprocessor Systems (SMP) pose? [4 marks]

- When filling a TLB entry, remember the physcical address the entry was read from. Snoop all writes to see if the address ever gets written, in which case invalidate the TLB entry. It my be easier to monitor the cache line or even the page the entry came from; false positives will not effect corrctness.

- SMP poses problems because another CPU may write to the location the TLB entry was loaded from. Cache coherency logic would have to be extended to ensure that CPUs that had the line in the TLB were notified if another CPU ever tried to get Exclusive access to the line.