ECE485/585 Homework No. 4 Solution

Problem No. 1

(a) Number of virtual pages = $2^{38} / 2^{12} = 2^{26}$
Thus the single level page table needs to have 2^{26} = 64M page table entries (PTEs)

(b) Amount of physical memory consumed by the page table = 64M PTEs * 4 bytes/PTE = 256MB

(c) Given a 38-bit virtual address, we have 38 – 12 = 26 bits to split between the page table directory and the page tables. Since the address bits are equally split (according to the problem statement), there will be a 13-bit index into the page directory and a 13-bit index into each page table.
Thus, the page directory will consume 2^{13} PDEs * 4 bytes/PDE = 32KB of physical memory
According to the problem statement, all the page tables are fully populated and resident in memory. Therefore, there are 2^{13} page tables taking up space in the physical memory. Each of these page tables has 2^{13} PTEs.
Thus, the physical memory consumed by all the page tables = 2^{13} page tables * 2^{13} PTEs per page table * 4 bytes/PTE = 256MB

(d)

Page Size (P)	VPN bits	VPO bits	PPN bits	PPO bits
8K	25	13	20	13
16K	24	14	19	14
32K	23	15	18	15
1M	18	20	13	20

Doing these calculations is pretty straightforward. First, figure out how many bits are needed for the offset (that is, how many bits to address each location in a page). Use the same number for the Virtual and Physical page offset. The Virtual and Physical page numbers are calculated from the total address bits minus the offset. For example, for an 8K page size we need 13 address bits. The virtual page number is then 38 – 13 = 25 address bits and the physical page number is 33 bits (8GB takes 33 bits) – 13 = 20 address bits.

Problem No. 2

(a) The "E" state enables the processor to distinguish the non-shared blocks (blocks with only one copy) in its cache from shared blocks (blocks with multiple copies) which are kept in "S" state. When a processor writes new data to a cache block in the "E" state, it does not need to send any RFO (or invalidation) messages to other processors, thereby reducing coherence traffic.

(b) The state transition diagrams for the MSI protocol are similar to the diagrams for the MESI protocol (Lecture 11, slides 21 and 22), except for the following differences, (i) The "E" state and all the transitions into and out of the "E" state are removed, (ii) The "Hit" signal is not considered during the "I" to "S" state transition.

(c) When using the MSI protocol, any processor which writes new data to a clean block needs to send an RFO (or invalidation) message to the other processors. In the absence of any actual data sharing, these messages are unnecessary and consume extra network traffic. However, these messages are unavoidable (they are a necessary evil), since there is no state in the MSI protocol to indicate blocks which are not shared by multiple processors.

Problem No. 3

(a) Segments were too large to provide sufficient granularity. Entire segment had to be in memory, which resulted in inefficient memory capacity utilization. While variable sized segments addressed this problem, they introduced complications in memory allocation.

(b) A TLB as a cache for virtual memory to physical memory mappings. In CPUs that support virtual addressing the hardware or software must "walk the page table" to find the base in physical memory for a page of instructions and/or data. This is a slow operation (particularly if it is done in software) that could cripple the throughput of the CPU. The TLB, in effect, "short circuits" the page walk by directly providing the base address of the physical page in memory.
A TLB can be used in conjunction with a cache to improve CPU performance by enabling the virtual page translation to proceed in parallel with the cache lookup.

(c) An ASID is an **A**ddress **S**pace **Id**entifier. It is a unique number often tied to a CPU and process ID. ASIDs can be made part of the tag in a virtual memory system (e.g. they become part of the tag) thus eliminating the need to flush the TLB on a context switch.

Problem No. 4

(a) Because of the closed page memory policy, each main memory access requires an ACTIVATE command followed by a READ command. Thus the memory access latency is tRCD + CL + (burst length/2) = 9 + 9 + 4 = 22 cycles. The DIMM clock on PC3-10600 is 10600/8/2 or 666 MHz or 1.5ns cycle length. So 22 x 1.5 = 33ns for a read.
AMAT = (L1 hit ratio * L1 hit time) + (L1 miss ratio * ((L2 hit ratio * L2 hit time) + (L2 miss ratio * L2 miss time))
= (.95 x 1ns) + (.05 x [.8 x (1+8) ns + .2 x (1+8+33) ns]) = 1.73ns

Note: The TLB latency has not been included in the AMAT calculations for this part because: (i) TLB hits proceed in parallel with L1 access and take less time (< 1ns) than a L1 hit, and (ii) there are no TLB misses (since the TLB hit ratio is assumed to be 100%).

(b) Each TLB hit takes 1ns. On a TLB miss, we need to "walk the page tables".
In a 2-level paging hierarchy (page table directory + page table), each page table walk requires 2 memory accesses: first access is to the page directory and second access is to the page table pointed to be the page directory.
Thus, TLB miss latency = page walk latency = 2 * 33 = 66ns
Average time to obtain the virtual to physical translation = (TLB hit ratio * TLB hit time) + (TLB miss ratio * TLB miss time)
= (0.98 * 1ns) + (0.02 * (1 + 66) ns) = 2.32ns

Note: The above calculations are done under the assumption (specified in the problem statement) that all the page tables are resident and locked in physical memory (not paged out or cache). If these assumptions were to change, the calculations would also change. For example, if the page tables were allowed to be cached, then the page table accesses which hit in the L1 or L2 cache would not require main memory accesses and would have to be accounted differently.