HW 3

1. (10 points) Select whether each of the following statements is true or false.

1) true
2)true
3)true
4)false
5)false

2. (16 points) In a system using segmentation as the address translation approach, the virtual address space is split into 2 segments. The top bit of the virtual address (VA) determines which segments the VA is in:

1) segment 0 : base = 250(Decimal), bound = 5 (Decimal). I looked at the valid translations and since we were given a physical address we could count down until 0 which got us a physical address of 250 and since the virtual address has 16 bits (0-15) divided by two it gets us bits 0-7 for seg0, and 8-15 seg1. Another approach is just simply using the formula PA = base + va – vaseg which will get us 250 base.

2) segment 1: Base = 881, limit =2. Same approach as part 1. Pa = base + va – va_seg-→ 880 = base + 14 -5 ….base = 881.

3)same approach as previous answers…. Base = 100, limit = 3

4) same approach as previous answers….Base = 502, limit = 3

3. (34 points) Let us assume the following in a system which uses paging for address translation.

1) since virtual address space is 16k--> 2^14 , thus 14 bits are in the VA

2) address space / page size . 2^14/2^4 = 2^10…..10 bits in VPN.

3) since it's a 14 bit address and we have 10 bits for vpn …. That leaves us with 4 bits for offset.

4) 2^10 pages since vpn is 10 bits so we do 2^10 to get total pages

5) 8 bits for top level. We have 1024 entries and each one 4 bytes so 1024 x 4 = 2^12 (table size). Then we device 2^12 by the page size and we get 256 pte of 16 byte pages. Since our top level has these 256 pte we get 8 bits for top level

6) since our vpn was 10 bits all that is left is 2 bits for PTI

7) 256 pages as explained in question 5.

8) 2^10 vs 258. 2 level paging has the 256 entries from first level and since first and last pages are mapped, we add 2 to get 258 vs the 1024 of linear paging. 766 pages saved

9) 6 bits since we have 64 entries. I had to cut the original entries by 4

10) 2 bits since since pti is 2

11) since we need 6 bits for gpdi we get 2^6 = 64

12)64+2 +2 = 68 for three level paging and 1024 for linear paging. Since only first and last are mapped we get 2 from second level and 2 from third level, thus getting 68 in total. 956 pages saved.

4. (20 points) A system uses linear paging for virtualizing memory. In this system, address space size is 128 bytes, page size is 4 bytes, addressable physical memory is 128 KB (yes, this system has larger physical memory than virtual address space). A page table entry (PTE) consists of a valid bit followed by physical frame number (PFN) - in other words, the most significant bit of a PTE is the valid bit, and the rest of the PTE is the PFN.

1) since pte is represented as a hex in the given inputs, we can easily count that the size of pte is 2 bytes

2) since we have 32 entries, we divide by 2 gets us 16 pages.

3) occupies 8 pages. Since we have 16 entries, we divide by 2 which gets us 8 pages.

4) its 10 pages in this 2-level paging example and 16 in the linear approach. I checked the entries and only two of them had a valid bit of 1 meaning 8 pages (from directory) + 2 from second level. Thus saving 6 pages.

5. (20 points) Let us assume we have a system using paging for address translation. A virtual address in this system is 32-bit long, VPN and page offset are both 16-bit long. The following is the content of the TLB in this system. Each TLB entry contains a 16-bit VPN (virtual page number), 12-bit PFN (physical frame number), and a 4 bit ASID. The ASID of a TLBs entry equals to the PID of the entrys corresponding process.

1)HIT, PA: A000000
2)HIT, PA: F000000
3)MISS
4)MISS
5)MISS
6)HIT, PA: FF00350
7)HIT, PA: AFF4321
8)MISS