Exercises on memory management

1. Assume the base and limit registers contain the same value, 16,384. Is this just an accident, or are they always the same? It is just an accident, why are they the same in this case?

   sol: It is an accident. The base register is 16,384 because the program happened tobe loaded at address 16,384. It could have been loaded anywhere. The limit register is 16,384 because the program contains 16,384 bytes. It could have been any length. That the load address happens to exactly match the program length is pure coincidence.

2. A system implements a paged virtual address space for each process using a one-level page table. The maximum size of an address space is 16 megabytes. The page table for the running process includes the following entries:

0	4
1	8
2	16
3	17
4	9

   The page size is 4KB and the maximum physical memory size of the machine is 2 megabytes.

   (a) How many bits are required for each page table entry?

   sol: There are at most $2^{21}/2^{12}$ frames in memory, so a page table entry will require 9 bits for a frame number and possibly other bits for protection.

   (b) What is the maximum number of entries in a page table?

   sol: The size of a virtual address space is 2^{24}, there the page table could have up to $2^{24}/2^{12} = 2^{12}$ entries.

   (c) How many bits are there in a virtual address?

   sol: The virtual address space has 24 bits.

   (d) To which physical address will the virtual address 1524 translate to?

   sol: Virtual address 1524 lies on page 0 which resides in frame 4 with an offset of 1524. So, 1524 maps to physical address $4 \times 4096 + 1524 = 16384 + 1524 = 17908$

   (e) Assuming that frame 0 starts at the physical address 0, that consecutive frame numbers have consecutive physical addresses (the first address of frame $i + 1$ follows immediately the last address

of frame i), then, which virtual address will translate to physical address 65536?

65536/4096 = 16. Since addresses start at number "0", the address 65536 is actually on frame 17 with offset 0. This frame has been assigned to page 3 of the logical address, with offset 0, it is the logical address 12288.

3. Consider a paging system with the page table stored in memory.

   (a) If a memory reference takes 400ns, how long does it takes to fetch data through the page table?

   sol: Need to access memory twice, one to access the page table and one for fetching the data: $2 \times 400ns = 800ns$

   (b) If we add a TLB, and 85% of all page-table references are found in the TLB (i.e. hit rate is 85%), what is the effective memory reference time? (Assume that finding a page-table entry in the associative registers takes zero time, if the entry is there.)

   sol: Effective access time = $(.85 \times 400) + (.15 \times 800) = 340 + 120 = 460ns$

4. Consider a paging system with the page table stored in memory. Assume the overhead to read an entry in the page table is 5ns (access to the page table is 5 ns). To reduce this overhead, a TLB is installed, where a look up in the TLB cost 1ns. What hit rate is needed to reduce the average overhead to access the page table to 2ns?

   sol: The effective instruction time is 1h + 5(1-h), where h is the hit rate. If we equate this formula with 2 and solve for h, we find that h must be at least 0.75.

5. A system implements a paged virtual address space using a two-level page table. The maximum size of a process address space is 2^{27} bytes (about 134 megabytes). The page table for a running process includes the following entries:

0	16
1	8
2	5
3	1
4	2

   The frame size is 2^{10} bytes (1KB) and the physical memory size is 2^{16} bytes (65 KBs).

   (a) How many bits are required for each page table entry?

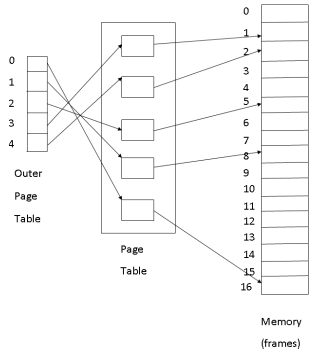

Figure 1: Two levels page table

The content of a page table entry is the address of a frame. The physical memory is 2^{16} bytes, assuming the memory is byte addressable, the size of the physical address space is 2^{16}, we need 16 bits to address a frame in the physical memory, thus 16 bits of memory (2 bytes) are required for each entry of the page table.

(b) What is the maximum number of entries in the inner page table?

The maximum size of a process is 2^{27}. Page sizes are always equal to the frame size of the computer which is 2^{10}. Thus the maximum number of pages a process can have is $2^{27}/2^{10} = 2^{17}$. The inner page table has one entry for each page, thus the maximum number of entries of the inner page table is 2^{17}.

(c) How many bits there is in virtual addresses generated by the CPU?

The maximum size of a process is 2^{27}, the virtually addressable space is 2^{27} bytes, thus addresses generated by the CPU have 27 bits.

(d) What is the maximum number of entries of the inner page table that can be stored in one frame?

We need 2 bytes to address a frame. Thus a frame can store $2^{10}/2 = 2^9 = 512$ entries of the inner page table.

(e) How many frames are needed to store a maximum size process?

The maximum number of pages a process may have is 2^{17}, thus the maximum number of entries in the inner page table is 2^{17}. One frame can store 2^9 entries of the page table, thus the number of frames needed to store the inner page table is $2^{17}/2^9 = 2^8 = 256$ frames.

(f) How many entries there will be in the outer page table?

To store the inner page table we need 2^8 frames. The outer page table must contain one entry for each frame of the inner page table. Consequently the outer page table will have 2^8 entries.

(g) To which physical address will the virtual address 9812 translate to?

A virtual address has 27 bits. 9812 in binary on 27 bits is 00000000 000001001 1001010100. The first 8 bits of the binary representation of 9812 are all zeros, they refer to entry 0 of the outer page table. Entry 0 of the outer page table has the address of the fifth frame holding the inner page table.

The next 9 bits 000001001 of the virtual address index into entry 9 of the fifth inner page table in Figure 1.

Entry 9 index into frame 16. Frame 16 starts at the address 16384 (16×1024). The last 10 bits of the virtual address 9812, $1001010100 = 596$, give us the physical address of the virtual address: $16384 + 596 = 16980$.

(h) Assuming that frame 0 starts at the physical address 0, that consecutive frame numbers have consecutive physical addresses (the first address of frame $i + 1$ follows immediately the last address of frame i), then, which virtual address will translate to physical address 33768?

$33768/1024 = 32.976 =$ frame 33. As we don't know from Figure 1 which entry of the page table index into this frame, we cannot tell the logical address of the above physical address.

6. Assume the physical memory is 64 bits addressable. Each frame is 16 KB or 2^{14} bytes. We have a process of 2^{64} bytes. The total number of pages in the process is $2^{64}/2^{14} = 2^{50}$. Rather than using a multilevel page table, we will use a hashed page table. The hashed page table has 2^{14} entries.

(a) What is the size of the hashed table?

sol: the contain of the hashed page table are pointers to link lists. Since the physical address space is 64 bits, and pointers are addressed, so there is 8 bytes per entry in the hashed table, thus $2^{14} \times 2^3 = 2^{17}$

(b) How many bits of a virtual address are used to address in the hashed page table?

sol: 50 bits

(c) Assume the bits of a virtual address that are used to index into the hashed table convert into the integer 337,656,234. What is the corresponding entry in the hash table? 14762, i.e. 337,656,234 modulo 16384