Daniel Peterson 109091561

2. In Fig. 3-3 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 example?

**The base register represents the location in memory where the program is loaded in and the limit register represents the length of the program. The fact that they are both the same is simply a coincidence because the program could have been a different length, such as 8KB, which would have made the limit register 8,192. Or the program could have been loaded in a different location in memory and the base register would have been different.**

4. Consider a swapping system in which memory consists of the following hole sizes in memory order: 10 MB, 4 MB, 20 MB, 18 MB, 7 MB, 9 MB, 12 MB, and 15 MB. Which hole is taken for successive segment requests of

(a) 12 MB

(b) 10 MB

(c) 9 MB

for first fit? Now repeat the question for best fit, worst fit, and next fit.

1. **First fit: Hole 3 (20MB)**

**Best fit: Hole 7 (12MB)**

**Worst fit: Hole 3 (20MB)**

**Next fit: Hole 3 (20MB)**

1. **First fit: Hole 1 (10MB)**

**Best fit: Hole 1 (10MB)**

**Worst fit: Hole 4(18MB)**

**Next fit: Hole 4 (18MB)**

1. **First fit: Hole 4 (18MB)**

**Best fit: Hole 6 (9MB)**

**Worst fit: Hole 8 (15MB)**

**Next fit: Hole 6(9MB)**

5. What is the difference between a physical address and a virtual address?

**Virtual addresses are viewed by processes while physical address spaces are the hardware addresses. Virtual address spaces are mapped into physical address spaces through the MMU. The virtual memory can store all of the data that processes need, but only a portion of that data is loaded into the physical memory at a given time. The size of the virtual address space and the physical address space are not equal.**

7. Using the page table of Fig. 3-9, give the physical address corresponding to each of the following virtual addresses:

(a) 20

(b) 4100

(c) 8300

**a) 8,212**

**b) 4,100**

**c) 24,684**

14. A machine has a 32-bit address space and an 8-KB page. The page table is entirely in hardware, with one 32-bit word per entry. When a process starts, the page table is copied to the hardware from memory, at one word every 100 nsec. If each process runs for 100 msec (including the time to load the page table), what fraction of the CPU time is

devoted to loading the page tables?

**2^32 / 8KB = 2 ^19 pages**

**2^19 \* 0.0001msec = 52.242 = 52msec loading**

**52/100 = 52% of CPU time**

16. You are given the following data about a virtual memory system:

(a)The TLB can hold 1024 entries and can be accessed in 1 clock cycle (1 nsec).

(b) A page table entry can be found in 100 clock cycles or 100 nsec.

(c) The average page replacement time is 6 msec.

If page references are handled by the TLB 99% of the time, and only 0.01% lead to a page fault, what is the effective address-translation time?

**1nsec \* (0.99) + 100nsec \* (0.09) + 6,000,000nsec \*(0.01) = 0.99 + 9 + 60,000 = 60,009.99 nsec**

19. A computer with a 32-bit address uses a two-level page table. Virtual addresses are split into a 9-bit top-level page table field, an 11-bit second-level page table field, and an offset. How large are the pages and how many are there in the address space?

**32 bits are available for an address. 9 used for the top level and 11 for the second level make 20. 32-20 = 12 so 12 bits are used as the page offset. 2^12 bits is 4K for a page. 2^20 bits is 1M for number of pages.**

20. A computer has 32-bit virtual addresses and 4-KB pages. The program and data together fit in the lowest page (0–4095) The stack fits in the highest page. How many entries are needed in the page table if traditional (one-level) paging is used? How many page table entries are needed for two-level paging, with 10 bits in each part?

**2^32 / 2^12 = 2^20 page entries in a one level page table. If 10 bits are used for each part of the page table, then 2^10 page entries are in one level of the page table. Then in the second page table, 2 bits are used for the remaining page entries. So one page entry from the first level is used to point to the second table, and then the two page entries in the second table are used to find the page.**

25. A computer with an 8-KB page, a 256-MB main memory, and a 64-GB virtual address space uses an inverted page table to implement its virtual memory. How big should the hash table be to ensure a mean hash chain length of less than 1? Assume that the hashtable size is a power of two.

**256MB/8KB**

**2^28/2^13 = 2^15 pages**

**If the hash table were 2^15 entries in size, then there would be a mean hash chain length of 1. If the hash table were doubled in size to 2^16 then the mean hash chain length would be cut in half to 0.5.**

28. If FIFO page replacement is used with four page frames and eight pages, how many page faults will occur with the reference string 0172327103 if the four frames are initially empty? Now repeat this problem for LRU.

**FIFO: FFFFF271F3 where F is a page fault where the first page reference is removed. There are 6 page faults**

**[][][]0 F**

**[][]01 F**

**[]017 F**

**0172 F**

**1723 F**

**1723 no F on 2**

**1723 no F on 7**

**1723 no F on 1**

**7230 F**

**7230 no F on 3**

**LRU: FFFFF271FF where F is a page fault where the least recently used page reference is removed. There are 7 page faults**

**[][][]0 F**

**[][]01 F**

**[]017 F**

**0172 F**

**1723 F**

**1723 no F on 2**

**1723 no F on 7**

**1723 no F on 1**

**1720 F and 3 is LRU so replaced**

**1730 F and 2 is LRU so replaced**

29. Consider the page sequence of Fig. 3-15(b). Suppose that the *R* bits for the pages *B* through *A* are 11011011, respectively. Which page will second chance remove?

**The second chance algorithm will remove page D because it will clear B and C because they have R bits set. Since page D does not have the R bit set it is evicted.**

32. In the WSClock algorithm of Fig. 3-20(c), the hand points to a page with *R* =0. If t =400, will this page be removed? What about if t =1000?

**If R=1 then the page was recently used. In this case, R = 0 so the algorithm must look at the time of last use for the page. If the time of last use is greater than the value for t, then the page contains a valid copy and can be replaced. If the time of last use is not greater than the value for t, then it cannot be replaced since there is not a valid copy. If Figure 3-20(c) is used, the time of last use is 1213, which is greater than both t = 400 and t = 1000 so this page contains valid copies and can be replaced.**

36. A computer has four page frames. The time of loading, time of last access, and the *R* and *M* bits for each page are as shown below (the times are in clock ticks):

**Page Loaded Last ref. R M**

0 126 280 1 0

1 230 265 0 1

2 140 270 0 0

3 110 285 1 1

(a) Which page will NRU replace?

(b) Which page will FIFO replace?

(c) Which page will LRU replace?

(d) Which page will second chance replace?

**a) NRU will replace page 2 because neither the R nor M bits are set**

**b) FIFO will replace page 3 because it was the first page loaded in at time 110**

**c) LRU will replace page 1 because it was referenced least recently at time 265**

**d) Second change will replace page 2 because it is the first loaded page without its R bit set**

46. When segmentation and paging are both being used, as in MULTICS, first the segment descriptor must be looked up, then the page descriptor. Does the TLB also work this way, with two levels of lookup?

**No, TLB does not work with two levels of lookup. The search key uses both the segment number and the virtual page number, so the exact page can be found using a single lookup.**