HW - 3.1 - HW 23

Questions: Tannenbaum & Bos 3.{2,3,4,6,8,11,15,22,23,24,27,29,31,33,35,37}

**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? It is an accident. The base register is the physical address in which the beginning of a program is loaded, when a process is run, in this case 16,384. I can be a different address. The limit register is 16, 384 to hold the entire length of the program in the base register.

**3.** A swapping system eliminates holes by compaction. Assuming a random distribution

of many holes and many data segments and a time to read or write a 32-bit memory

word of 4 nsec, about how long does it take to compact 4 GB? For simplicity, assume

that word 0 is part of a hole and that the highest word in memory contains valid data.

4\* 2^30GB \*1/4bytes = 2^30

2\*4\*4^-9 \* 2^30 = 1/32678,

**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.

For first: a) 20, b) 10, c) 18, Best a) 12, b) 10, c) 9,

Worst a) 20, b) 18, c) 15, Next a) 20, b) 18, c) 9

**6.** For each of the following decimal virtual addresses, compute the virtual page number

and offset for a 4-KB page and for an 8 KB page: 20000, 32768, 60000.

* 4KB = 4096, Page number = 20000/4096 = 4, 20000-4\*4096 = 3616
* 8KB = 8192, Page number = 20000/8192 = 2, 20000-2\* 8192 = 3616
* 4KB = 4096, Page number = 32768/4096 = 8, 32768-8\*4096 = 0
* 8KB = 8192, Page number = 32768/8192 = 4, 32768-4\* 8192 = 0
* 4KB = 4096, Page number = 60000/4096 = 14, 60000-14\*4096 = 2656
* 8KB = 8192, Page number = 60000/8192 = 7, 60000-7\* 8192 = 2656
* 4KB = 4096, Page number = 63540/4096 = 15, 63540-15\*4096 = 2100
* 8KB = 8192, Page number = 63540/8192 = 7, 63540-7\* 8192 = 6196

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | * 20000 | * 32768 | * 60000 | * 63540 |
| * 4KB | * 4,3616 | * 8,0 | * 14,2656 | * 15,2100 |
| * 8KB | * 2,3616 | * 4,0 | * 7,2656 | * 7,6196 |

**8.** The Intel 8086 processor did not have an MMU or support virtual memory. Nevertheless,

some companies sold systems that contained an unmodified 8086 CPU and did

paging. Make an educated guess as to how they did it. (*Hint*: Think about the logical

location of the MMU.)

A good place to put the MMU is in the middle of the CPU and memory on a bus.

**11.** Consider the following C program:

int X[N];

int step = M; /\* M is some predefined constant \*/

for (int i = 0; i < N; i += step) X[i] = X[i] + 1;

(a) If this program is run on a machine with a 4-KB page size and 64-entry TLB, what

values of *M* and *N* will cause a TLB miss for every execution of the inner loop?

(b) Would your answer in part (a) be different if the loop were repeated many times?

Explain.

M=0, If M is at least 4096 to ensure a TLB miss every inner loop execution. For N it does not matter how many times X is accessed.

M would be the same for case b, N > 64KB and X > 256K then it will miss too.

**15.** Suppose that a machine has 48-bit virtual addresses and 32-bit physical addresses.

(a) If pages are 4 KB, how many entries are in the page table if it has only a single

level? Explain.

2^2\*2^10 = 2^12,

2^48/2^12 = 2^36 entries.

68719476736

(b) Suppose this same system has a TLB (Translation Lookaside Buffer) with 32 entries.

Furthermore, suppose that a program contains instructions that fit into one

page and it sequentially reads long integer elements from an array that spans thousands

of pages. How effective will the TLB be for this case?

Effective from 0 to 4096, then miss.

**22.** A computer whose processes have 1024 pages in their address spaces keeps its page

tables in memory. The overhead required for reading a word from the page table is 5

nsec. To reduce this overhead, the computer has a TLB, which holds 32 (virtual page,

physical page frame) pairs, and can do a lookup in 1 nsec. What hit rate is needed to

reduce the mean overhead to 2 nsec?

75% hit rate will get mean overhead to 2ns.

**23.** How can the associative memory device needed for a TLB be implemented in hardware,

and what are the implications of such a design for expandability?

By comparing a key to contents of multiple registers, we need a set of comparators that compares each bit in register to what we are looking for by the key.

**24.** A machine has 48-bit virtual addresses and 32-bit physical addresses. Pages are 8 KB.

How many entries are needed for a single-level linear page table?

2^48/2^13 = 2^35

**27.** Suppose that the virtual page reference stream contains repetitions of long sequences

of page references followed occasionally by a random page reference. For example, the

sequence: 0, 1, ... , 511, 431, 0, 1, ... , 511, 332, 0, 1, ... consists of repetitions of the

sequence 0, 1, ... , 511 followed by a random reference to pages 431 and 332.

(a) Why will the standard replacement algorithms (LRU, FIFO, clock) not be effective

in handling this workload for a page allocation that is less than the sequence

length?

If the number of page frames is not the length of the entire sequence, 512, every reference page will fault.

(b) If this program were allocated 500 page frames, describe a page replacement approach

that would perform much better than the LRU, FIFO, or clock algorithms.

Keep up to 489 then swap 501-511

**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 first page with an address bit of zero.

**31.** Give a simple example of a page reference sequence where the first page selected for

replacement will be different for the clock and LRU page replacement algorithms. Assume

that a process is allocated 3=three frames, and the reference string contains page

numbers from the set 0, 1, 2, 3.

LFU 2 will be replaced. In clock 0 will be replaced.

**33.** Suppose that the WSClock page replacement algorithm uses a of two ticks, and the

system state is the following:

**Pa ge Time stamp V R M**

0 6 1 0 1

1 9 1 1 0

2 9 1 1 1

3 7 1 0 0

4 4 0 0 0

where the three flag bits *V*, R, and *M* stand for Valid, Referenced, and Modified, respectively.

(a) If a clock interrupt occurs at tick 10, show the contents of the new table entries. Explain.

(You can omit entries that are unchanged.)

**Pa ge Time stamp V R M**

0 6 1 0 1

1 9 1 1 0

2 9 1 1 1

3 7 1 0 0

4 4 0 0 0

(b) Suppose that instead of a clock interrupt, a page fault occurs at tick 10 due to a read

request to page 4. Show the contents of the new table entries. Explain. (You can

omit entries that are unchanged.)

changes page 4, time 10, R 1

**35.** How long does it take to load a 64-KB program from a disk whose average seek time is

5 msec, whose rotation time is 5 msec, and whose tracks hold 1 MB

(a) for a 2-KB page size?(5+5+2/32\*20)\*32 = 360

(b) for a 4-KB page size?(5+5+4/32\*20)\*16=200

The pages are spread randomly around the disk and the number of cylinders is so large

that the chance of two pages being on the same cylinder is negligible.

**37.** Suppose that two processes *A* and *B* share a page that is not in memory. If process *A*

faults on the shared page, the page table entry for process *A* must be updated once the

page is read into memory.

(a) Under what conditions should the page table update for process *B* be delayed even

though the handling of process *A*’s page fault will bring the shared page into memory?

Explain. When a page is retrieved from disk or page not found would cause delay.

(b) What is the potential cost of delaying the page table update?

The cost of the running process in delay. Performance would suffer