

COS214 Tutorial 4

Roberto Togneri, 2000

1. Explain how the following algorithms work in allocating memory:
 - (a) First fit.
 - (b) Best fit.
 - (c) Worst fit.

Given free memory blocks of 100K, 500K, 200K, 300K and 600K (in this order on a linked list), how would each of the above algorithms place requests for 212K, 417K, 112K and 426K (requested in that order)? Which algorithm makes best use of memory?

2.
 - (a) What is the difference between internal and external fragmentation? In these days of large RAM memories is this any longer important?
 - (b) What is the difference between physical and virtual addresses?
 - (c) Is it reasonable to have the following situations:
 - (i) physical address space \gg virtual address space
 - (ii) virtual address space \gg physical address space
 - (iii) virtual address space \approx physical address space
 - (d) In the lecture the MMU is shown as translating every virtual address to a physical address. Explain what is required to make this an efficient process given that even a simple instruction will need at least one translation and some may require several.
3.
 - (a) With the ever increasing number of bits per memory chip (e.g. 1M, 4M, 16M, and 64M), will there still be the need to consider virtual memory techniques?
 - (b) The Intel 8086 processor did not 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.
4.
 - (a) A machine has a CPU with a 32 bit address space and uses 8K pages. 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 100nsec. If each process runs for 100msec (including time to load the page table), what fraction of the CPU time is devoted to loading the page tables?
 - (b) A computer with a 32 bit address space 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 virtual address space?

5. Draw up a flow-chart to cover all the situations that arise in the operation of virtual memory paging with a TLB. Ignore the detail of running another waiting process whilst waiting for a missing page to be read in from the swap/paging disk area.

6.
 - (a) 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 500 nsec. To reduce this overhead, the computer has an associative memory, which holds 32 (virtual page, physical page frame) pairs, and can do a look up in 100 nsec. What hit rate is needed to reduce the mean overhead to 200 nsec?
 - (b) A group of operating system designers for the Frugal Computer Company are thinking about ways of reducing the amount of backing store needed in their new OS. The head guru has just suggested not bothering to save the program text in the swap area at all, but just page it in directly from the binary file whenever it is needed. Are there any problems with this approach?
 - (c) Consider a paging system with the page table stored in memory. If a memory reference takes 200 nanoseconds, how long does a paged memory reference take? If we add associative registers, and 75 percent of all page-table references are found in the associative registers, 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.)

7.
 - (a) For a 2-level page table system with pure paging detail the exact software and hardware implementation requirements.
 - (b) A 32-bit OS uses a 2-level page table scheme with 4K pages. The page table structures are stored in main memory (this is the hint!). Explain why $p = 10$ (10-bits to index top-level page table) and $q = 10$ (10-bits to index the 2nd level page table) is the only solution of assigning p and q (such that $p + q = 20$) to minimise internal fragmentation?
 - (c) The OS in (b) gets a VA = 00D4D578H. Detail which page table structures and entries are accessed (in the event of a TLB miss) and how the PA is formed.