C502 – Operating Systems Tutorial *

Memory Management – Solutions

Note: The solution notes below only briefly list (some of) the key points that should be included in an answer. They are by no means complete. In an exam, you are expected to spell out the solution more fully and include a detailed explanation of your reasoning.

1. Describe the difference between swapping and paging in the context of virtual memory management.

   Answer: *Swapping means moving entire address spaces between the disk and the memory. Paging means moving individual pages, so that part of an address space may be on disk while the other part is in main memory.*

2. What is an associative memory? How does it work and how is it implemented?

   Answer: *An associative memory is one that is accessed by content rather than by address. Often an entry in an associative memory will have two parts, a tag and a value. To access the associative memory, one supplies a tag (Page number from virtual address). If that tag matches the tag of an entry in the associative memory, that entry is returned giving the frame number. In a hardware implementation (e.g., a TLB), the tags of all entries are checked in parallel.*

3. 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 MB. The page table for the running process includes the following entries:

Page	Frame
0	4
1	8
2	16
3	17
4	9

   The page size is 1024 bytes and the maximum physical memory size of the machine is 2 MB.

   (a) How many bits are required for each page table entry?
   (b) What is the maximum number of entries in a page table?
   (c) How many bits are there in a virtual address?
   (d) To which physical address will the virtual address 1524 translate to?
   (e) Which virtual address will translate to physical address 10020?

   Answer:

   (a) There can be at most $2^{21}/2^{10}$ frames in main memory, so a page entry will require 11 bits for a frame number, plus protection and reference bits, etc.
   (b) A page table may have upto $2^{24}/2^{10} = 2^{14}$ entries.
   (c) A virtual address has 24 bits.

*with thanks to Morris Sloman

(d) *Virtual address 1524 is on page 1 which lies in frame 8 with an offset of 1524 modulo 1024, or 500. So, 1524 maps to physical address 8 * 1024 + 500 = 8192 + 500 = 8692.*

(e) *Physical address 10020 is in frame 9, with an offset of 804. Virtual page 4 maps to frame 9, so virtual address 4 * 1024 + 804 = 4900, which will map to 10020.*

4. Calculate the access times for a four-level paging system assuming a TLB hit ratio of 80% and 98%. Assume that time for a memory access is 100 ns and for TLB access 20 ns.

   How does this compare to a single-level paging system?

   Answer:

   If the hit ration is 80%, the time for translation is:

   *Effective access time = 0.8 * 120 + 0.2 * 520 = 200 ns, thus 100% slower than unpaged memory access (compared to 40% in the case of single-level paging)*

   If the hit ratio is 98%, the time for translation is:

   *Effective access time = 0.98 * 120 + 0.02 * 520 = 128 ns, thus 28% slower than unpaged memory access (compared to 22% in the case of single-level paging).*

5. Compare small and large page sizes for a paged virtual memory system. Consider fragmentation, data structure requirements, page tables, page transfer time, TLB space etc. (Exam question in 2015-16).

Answer:

Small Pages	**Large Pages**
Less internal fragmentation. For n segments and page size p, average fragmentation = $\frac{n*p}{2}$.	*More internal fragmentation.*
Small data structures would typically fit in 4KB pages.	*Large page size e.g 32 KB is inefficient for small data structures of < 4KB.*
Many pages and large page table required e.g. 1Mb program requires page table of 2K 512 byte pages. Space and time overhead e.g. for loading page table on process switching.	*Smaller, easier to manage page tables with less overhead for process switching.*
Transfer time = seek + rotational delay e.g. 10ms for 512 byte but 12 ms for 8KB.	*More efficient to transfer fewer larger pages as compared to more smaller pages.*
Small pages require more TLB entries e.g. working set of 64KB and 4KB pages requires 16 TLB entries.	*Fewer TLB entries needed for working set for large pages. More efficient to flush when switching processes.*