CS4023 **Week11 Tutorial**

The following questions are taken from the exercises at the end of Chapter 8 of SGG (ed. 8)

1. Consider a logical address space of 64 pages with 1.024 words per page, mapped onto a physical memory of 32 frames.

   (a) How many bits are required in the logical address?

   (b) How many bits are required in the physical address?

   (**Q8.4**)

   Answer Each page/frame holds 1K; we will need 10 bits to uniquely address each of those 1024 addresses. Physical memory has 32 frames and we need 2^5 bits to address each frames, requiring in total 5+10=15 bits. A logical address space of 64 pages requires 6 bits to address each page uniquely, requiring 16bits in total.

2. Consider a paging system where the page table is stored in memory.

   (a) If a memory reference takes 200 nanoseconds, how long does a paged memory reference take?

   (b) If we add TLBs, and 75% of all page-table references are found in the TLBs, what is the effective memory reference time? (Assume that finding the page-table entry in the TLBs takes zero time, if the entry is there.)

   (**Q8.20**)

   Answer in the first case two memory accesses are required to resolve the address, requiring 400ns; in the second case the effective access time is $0.75 \times 200 + 0.25 \times 400 = 250$ nanoseconds.

3. Most systems allow a program to allocate more memory to its address space during execution. Allocation of data in the heap segments of programs is an example of such allocated memory – for example, `malloc()` in `C/C++`. What is required to support dynamic memory allocation in the following schemes?

   (a) Contiguous memory allocation

   (b) Pure segmentation

   (c) Pure paging

   (**Q8.12**)

Answer

(a) contiguous-memory allocation: might require relocation of the entire program since there is not enough space for the program to grow its allocated memory space

(b) pure segmentation: might also require relocation of the segment that needs to be extended since there is not enough space for the segment to grow its allocated memory space

(c) pure paging: incremental allocation of new pages is possible in this scheme without requiring relocation of the programs address space.

4. What is the purpose of paging the page tables? (**Q8.24**)

   Answer In certain situations the page tables could become large enough that by paging the page tables, one could simplify the memory allocation problem (by ensuring that everything is allocated as fixed-size pages as opposed to variable-sized chunks) and also enable the swapping of portions of page table that are not currently used.

5. Compare paging with segmentation with respect to the amount of memory required by the address translation structures in order to convert virtual addresses to physical addresses. (**Q8.15**)

   Answer Paging requires more memory overhead to maintain the translation structures. Segmentation requires just two registers per segment: one to maintain the base of the segment and the other to maintain the extent of the segment. Paging on the other hand requires one entry per page, and this entry provides the physical address in which the page is located.

6. Consider the following process for generating binary executables. A compiler is used to generate the object code for individual modules (files of source code), and a linkage editor is used to combine multiple object modules into a single program binary. How does the linkage editor change the binding of instructions and data to memory addresses? What information needs to be passed from the compiler to the linkage editor to facilitate the memory-binding tasks of the linkage editor? (**Q8.10**)

   Answer The linkage editor has to replace unresolved symbolic addresses with the actual addresses associated with the variables in the final program binary. In order to perform this, the modules should keep track of instructions that refer to unresolved symbols. During linking, each module is assigned a sequence of addresses in the overall program binary and when this has been performed, unresolved references to symbols exported by this binary could be patched in other modules since every other module would contain the list of instructions that need to be patched.