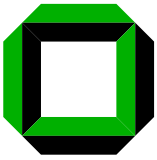

University of Karlsruhe
System Architecture Group
Gerd Liefländer

Teaching Assistant:
Philipp Kupferschmied

System Architecture 2008/09
Assignment 12

Question 12.1: Address Spaces

1. Recall the purpose of logical address spaces.
2. How can logical address spaces be mapped to physical addresses? Point out advantages of each approach.
3. Point out why it might be desirable to share parts of the logical address space with other applications. Compare shared memory with IPC.
4. What parts of address spaces can typically be shared?

Question 12.2: Comparison of Page Table Layouts

A computer with 512 MB of main memory installed has 32 bit virtual addresses and 4 kB pages. Each page table entry occupies 4 bytes. A specific program and its data together fit into the lowermost page (0–4095), whereas the stack fits into the highest page. How much memory is at least required for holding the page table under the following circumstances:

1. A one-level page table is used.
2. Two-level page tables are used.
3. An inverted page table is used.

Question 12.3: Use(less)-case Study of TLBs and Superpages

Consider an IA32-like system where the MMU supports a two-level page table with the second level containing 1024 page table entries mapping to 4 kB page frames. Each page table entry (both first and second level) is 4 bytes. The system only supports 4 kB page sizes and does neither contain a TLB nor a cache.
We want to sequentially read 16 MB consecutive bytes from virtual memory, starting at address 0 and read one word (4 bytes) at a time.

1. How many memory accesses are required to read the desired data?
2. We now introduce a 16 entry data TLB. How many memory accesses are needed to read the 16 MB of memory specified above?
3. Now consider a system supporting superpages. A superpage is 4 MB and is implemented using a first-level page table entry. The system has a 16 entry TLB. How many memory accesses are needed to read the 16 MB of memory specified above?
4. Compare the number of memory accesses needed for the systems mentioned in parts (2.) and (3.). Why does using superpages in this scenario not significantly reduce the number of memory accesses? Describe a scenario where superpages do help significantly.

Question 12.4: Pagefault Handling

When an application touches a virtual page for the very first time, it might not yet be mapped to physical memory (*demand paging*). The accesses will raise a pagefault exception, and the responsible handler is expected to map the contents of the requested page to physical memory.

1. Classify these pagefaults according to whence the data for the unmapped page has to be fetched by the pagefault handler.
2. If the application has been running for some time, modifying data along its way, there is one additional case that needs to be covered on a pagefault. Which?
3. Discuss which information is required by the pagefault handler to correctly setup (or restore) the contents of accessed pages.
4. Propose a (compact) data structure to hold the above information.
5. Can you reuse page table entries to store some of this information? Would this be a *good idea*$^{\text{TM}}$?
6. Is the required information dependent on the type of page table (one-level, multi-level, inverted, hashed inverted) used in the system?

Question 12.5: Hardware- vs. Software-Walked Page Tables

1. What's the difference between the use of hardware-walked page tables and software-walked page tables? How does this relate to TLBs?
2. What difference can you make out in the contents of software-walked vs. hardware-walked multi-level page tables?
3. Under what circumstances are TLB miss handlers or pagefault handlers invoked?
4. How can hardware-walked multi-level page tables be set up or accessed from software running on truly paged virtual memory?