Greg Matthews
February 22nd 2018

CS 543
Homework #3 – Questions

1.) Who owns the pages in physical memory (process, user, or system)? How would you denote ownership?
Answer:

The owner of each page frame in physical memory is the process. This depends on whether it is a free page in memory or a page that is currently allocated by another process. If the page frame is free, then any process can take ownership of it so long as the operating system allows the process to map to that particular address in memory. However, if another process attempts to read/write from a page frame that was created by a different process, it would depend on if the memory management unit (MMU) has set any reading/writing privileges for other processes who want access to that page frame. If a process was to read from a page frame that does not have read access, that process would simply receive a page fault.

Ownership of a frame can be denoted by adding more bits (descriptor and flags) to a page table entry (PTE) that give more information on the page's privilege level and read/write permissions when being handled by the MMU. For example, a typical 32bit page table entry has 20bits associated with addressing a 4KB address in physical memory, and the last 12 bits of the page directory are used for specifying information about that page. Of those 12 bits, one of the bits is a User/Supervisor bit, which denotes whether the page has user access (0), or only the supervisor can access (1), as well as having other bits denoting whether other processes can read or write to it.

2.) When you replace an entry in the TLB do you have to write it back to physical memory? Why or why not?
Answer:

Whenever an entry in the TLB is replaced, you would have to write it back to physical memory depending whether read & write capabilities are applicable. Using a dirty bit to keep track of page modifications, if a page is to be replaced in the TLB that has a modify bit of 1, meaning it has been modified, then we must write the page to disk because the page value stored on disk has an older value and must be updated with whatever value was stored in the TLB to ensure next time it is accessed it has the most up to date value.

3.) Does the size of the TLB make a difference in your statistics? Experiment by simulation.
Answer:

Increasing the TLB size makes a big difference towards the TLB hit-rate. The higher number of TLB entries that are available, the higher the TLB hit rate will be because there are much less page replacements happening, which allows for more pages and frames to be stored. For example, when simulating the virtual memory manager in C, using a TLB entry size of 16, the TLB hit rate was calculated at 0.0570. Increasing the TLB entry size to 32 resulted in a TLB hit rate of 0.1160 and increasing the TLB entry size to 64 resulted in a TLB hit rate of 0.2210, which shows a linear increase in TLB hit rate with increase in TLB entry size, up to a maximum of 1.0.

(Extra Credit) Can this virtual memory manager have more levels of memory (for example, cache) – If so, how? If not, why not?

Answer:

I believe the virtual memory manager is capable of having more levels of memory, because with the help of page tables allowing virtual memory to seem infinitely big, this means it has more than enough addressable pages to be able to address into account multiple levels of physical memory frames. Implementing a page table hierarchy may be necessary to allow for addressing of different levels of memory that correspond to different levels of the page table. Using hierarchical page tables allows for even more addressing capabilities.