Zack Chambers

CST 221

January 20, 2019

John Zupan

Memory Management Unit

Virtual memory is a storage allocation scheme where a running program is larger than the physical amount of memory on board. Part of the CPU is called upon; the Memory Management Unit or MMU, and by placing them on the storage disk and using main memory as a kind of cache for the most heavily executed parts. It requires remapping memory addresses on the fly to convert the address the program generated to the physical address in RAM where the word is located (Tanenbaum & Bos., 2015).

Example:

void access (uint x, bool kernel, bool write)

{

if (!(x & PG\_P)

=> page fault -- page not present

if (!(x & PG\_U) && user)

=> page fault -- not access for user

if (write && !(x & PG\_W))

if (user)

=> page fault -- not writable

else if (!(x & PG\_U))

=> page fault -- not writable

else if (%CR0 & CR0\_WP)

=> page fault -- not writable

}

maps va => pa

Uint translate (uint va, bool kernel, bool write)

{

uint pde;

pde = read\_mem (%CR3 + 4\*(va >> 22));

access (pde, kernel, read);

pte = read\_mem ( (pde & ~0xfff) + 4\*((va >> 12) & 0x3ff));

access (pte, kernel, read);

return (pte & ~0xfff) + (va & 0xfff);

}

Page Fault Handling

When a process references a page that it is not currently mapped in, it results in a page fault. During a page fault, the page fault handler locates the machine holding the page and sends a message asking it to un-map the page and resend the page. When the page arrives, it is mapped in and the faulting instruction is restarted (Tanenbaum & Bos., pg. 163, 2015).

**Pages Sent**

**NO**

**NO**

**NO**

**YES**

**YES**

**YES**

**Does this page belong?**

**Was the exception in user mode?**

**Does it match the memory access?**

**Kill the process**

**Illegal access sends**

**SIG SENDV**

**Legal access allocates new page frame**

Separation of Policy and Mechanism

The kernel internally has many ways of scheduling: first-come-first-served (FIFO), shortest job first, shortest remaining time next, round robin scheduling, priority scheduling, multiple ques, shortest process next, guaranteed scheduling, lottery scheduling, and fair-share scheduling. None of these have any inputs from user processes about scheduling decisions. To work around this issue, a solution was made to separate the scheduling mechanism from the scheduling policy. In other words, the scheduling algorithm is parameterized in some way, but the parameters can be filled in by user processes (Tanenbaum & Bos., 2015).

Reference

Tanenbaum, A. S., & Bos, H. (2015). Modern operating systems. Boston, MA: Prentice Hall

(protected-mode) address translation on the x86. (n.d.).

GitHub: <https://github.com/zchambers3/CST-221/tree/master/MemoryManagement>