OS/161 – Two-Level Page Table

Files changed/added:

/kern/arch/mips/conf/conf.arch

Added new vm option: arch/mips/mips/vm.c

/kern/arch/mips/include/vm.h

Defines vm system variables.

The struct pt_node describes the mapping of a virtual page to a physical page, and the struct page_table is used for the top level of the page table, containing nodes.

The macro PADDR_TO_KVADDR(paddr) converts a physical address into a kernel virtual address. KVADDR_TO_PADDR(paddr) was created to do the opposite.

/kern/arch/mips/syscall.c

Exit syscall implemented.

/kern/arch/mips/mips/vm.c

The custom virtual memory system using the node structure defined in vm.h to create an inverted page table. Each entry in the page table keeps track of the size of the contiguous allocation starting at the corresponding page. The address “lo” is used to indicate the lowest address of the managed memory.

Vm_bootstrap allocates the physical memory of the system using a for loop, and allocates pages for the inverted page table. Variable “bootstrapped” is set to 1 after it runs.

Alloc_ppages allocates physical pages, alloc_kpages allocates physical pages if bootstrapped (use ram_stealmem if early in boot).

Free_ppages frees the entire region of memory allocated to the physical page, free_kpages simply uses free_ppages with the address ran through the KVADDR_TO_PADDR macro to convert it from a kernel virtual address to a physical address.

Pt_node_create creates a page table node with no children.

Pt_create creates a page table, allocating memory for the top level of the page table only.

Pt_node_destroy frees all the memory a node points to and is destroyed.

Pt_destroy destroys a page table and frees all the nodes associated with it.

Walk_pt translates virtual kernel addresses to physical addresses, going through each page table layer until it finds the physical address referenced by the fault. If the address doesn't exist, it's allocated.

Vm_fault handles access to invalid virtual memory addresses. If it's not in virtual memory yet, run walk_pt. If an illegal access, return EFAULT.

Flush_tlb write completely invalid TLB entries to the TLB, flushing it. It's used for every context switch.

/kern/conf/conf.kern

Exit syscall added to configuration: userprog/exit.c

Link page table draft: vm/addrspace.c

/kern/conf/proj

Main kernel configuration for the project based on assignment 0 config (ASST0). Original dumbvm commented out.

/kern/include/addrspace.h

New #include for vm.h file, defines various variables for use in page tables, and functions for addrspace.c.

/kern/include/vm.h

Defines flush_tlb, page_table struct, and pt_destroy functions for vm.c.

/kern/userprog/exit.c

The exit system call. Prints exit code, exits thread, and returns 0.

/kern/vm/addrspace.c

As_create: Create a new address space. May return NULL on out-of-memory error.

As_copy: Create an exact copy of a preexisting address space

As_destroy: Destroy the pointer to the address space and free it from memory.

As_activate: Make a specified address space the one currently seen by the processor. Can input NULL to specify no particular address space.

As_define_region: Align a region in the address space to a page and set npages of either the first or second level depending on if it equals 0.

As_check_access: Check legality of a virtual memory access for an address space.

As_prepare_load: Called when preparing an address space for an executable.

As_complete_load: Called when loading an executable is complete.

/testbin/vm_t1/

Added makefile, depend.mk, and vm_t1.c to this new directory. Simply returns 0.

/testbin/vm_t2/

Added makefile, depend.mk, and vm_t2.c to this new directory. The test program creates a large array that's twice the size of physical memory, and iterates through it. Each time, it sets the index (i) of the array to 1, and increments i to 10 times the page size (set to 4096). If nothing fails during the process, 0 is returned.

/testbin/Makefile

Makefile updated to include vm_t1 and vm_t2.

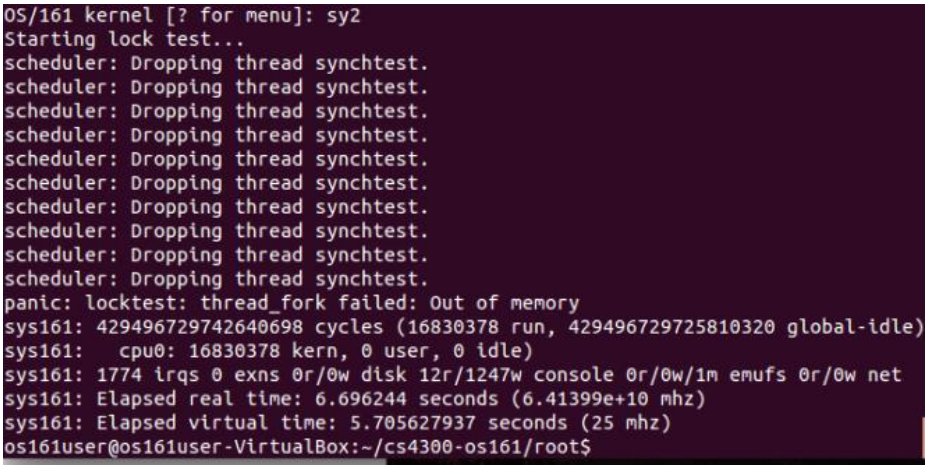
/kern/main/main.c

Boot message modified: "Peter and Ryan's system version..."

Report:

OS/161 is an educational operating system which runs within the terminal and is missing some features. When compiling and running the kernel without any changes, the system is prone to panicking as programs run and quit. Even if the processes properly deallocate the memory they used, it's still marked as allocated by the operating system. This is a result of the default memory manager that it comes with. In it, memory is never freed, as the function handling the freeing of virtual pages is unimplemented. Memory is also always allocated in contiguous chunks, meaning internal and external fragmentation occur.

The image below demonstrates this shortcoming. The test program sy2 tests the lock functionality of the operating system. Doing so allocates memory to the process. If the test program is run once, it works as expected. However, after it runs two more times, the OS runs out of memory. This result can be seen in the image below.



To solve these problems, we opted to implement a more sophisticated two-level page table system. With paging, the physical and virtual memory are split into units of fixed sizes, eliminating external fragmentation. The second level of paging also helps to alleviate the page table from becoming too large. The primary page table links to a secondary page table, which links to a virtual address space leading to physical memory.

Once everything was finalized, running the same test program multiple times no longer resulted in a crash. Other test programs also apply, since they all use memory and attempt to free it after the process finishes.