Memory Management Unit (MMU)

Operating Systems Final Exam

Author: Gabriel Hofer

Date: May 2, 2021

TODO

1. print formatted memory map to stdin
2. write tests for code
3. make page tables for multiple processes
4. make an inverted page table

Memory Management As we saw in CoA, memory is central to the operation of a modern computer system. Memory consists of a large array of bytes, each with its own address. The CPU fetches instructions from memory according to the value of the program counter. These instructions may cause additional loading from and storing to specific memory addresses. We introduced not only the concept of paging, but also virtual memory. If you decide to this task you need to inside your dash invoke (you have the option to write this as a dash-function or a free-standing program that is forked from inside the dash) a simulation of: 1. Paging, using an array of e elements simulating the simulated physical memory divided into f frames of b bytes. Your simulation will run q processes (the user decides how many processes), needing p pages. A memory map that keeps track of the frames. Each process then accesses memory locations using: 1. Basic method, each process having their own page table (needs to be stored somewhere), requesting p pages from the physical memory. 2. Inverted page table, one page table for the system. 2. Page replacement algorithm. You decide which algorithm to use, but it needs to be tied to your paging algorithm. 1. How many pages for each process?

Terminology used in this Document **Page** **Frame** **Page number** **Offset**

Details

- we will be using a 32-bit address space (uint32_t)

Questions

How large should your simulated memory be?

How large is the virtual memory of each process?

How to simulate that a process is accessing a memory location? How often? In what order?

How to keep track of what memory locations are in use?

What about shared frames?

Are there risk for race conditions? How will you handle it?

How to display all information, what makes sense for the user to see?

Should the user be allowed to decide what information to see?

Should the user be allowed to freeze the simulation at any moment and explore the current state?

Usage

Testing