Anuska Pant
CSCE 611 - OPERATING SYSTEMS

Machine Problem 3: Page Table Management

Introduction

The objective of this machine problem was to get started with the demand paging of virtual memory on an x86 architecture.

Page Management in Our Kernel

The memory layout in our kernel looks as follows:

- The total amount of memory in the machine is 32MB.
- The memory layout is such that the first 4MB are reserved for the kernel (code and kernel data) and are shared by all processes.
- Memory within the first 4MB will be direct-mapped to physical memory.
- Page Size is 4KB.

Approach

The x86 architecture consists of a two level page table, where the logical address is divided into 3 parts as follows:

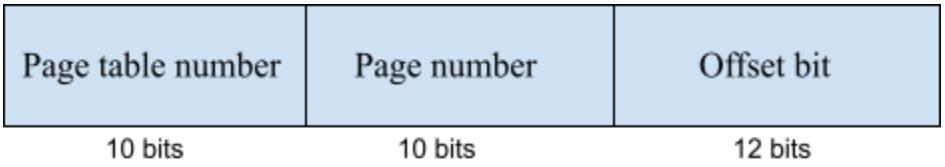

Fig: Logical Address of x86 architecture

The CR3 register is the page table base register that contains the base address of the page directory and the 10 bit page table number is used to access the page table in the page directory. Likewise, the 10 bit page number is used to access the frame number in the page table pointed by the page directory. The frame number and the offset bit gives the address in the physical memory.

The first step would be to implement direct mapping of the first 4MB of the physical memory. Thus we fill the first entry in the page directory to a page table that maps to the frames in the kernel pool and the first entry in the page directory has it's attribute set to supervisor level, read/write and present(011 in binary), while the rest of the entries are initialized to invalid, i.e the present bit is set to 0.

We then load the page table by writing it to the CR3 register, it now contains the base address of the page directory. After that paging is enabled by setting the value in CR0 register to 1.

Next problem is to handle the page fault. We have access to the faulty address in the register CR2. We read from the register and extract the page table number and page number using bit manipulation. While handling a page fault two cases might arise.

1. The entry in the page directory is invalid.
2. The entry in the page table is invalid.

When the entry in the page directory is invalid we handle it by getting a frame and allocating it to a page table that maps to the page directory at index obtained from the 10 bit page table number. Likewise, when the entry in the page table is invalid we get a frame and allocate it to the page table entry and store it in the index obtained from the 10 bit page number.

The output with the test passed is attached below.