# Smaller Tables Summary - SistOp

Lautaro Bachmann

# Contents

# Paging: Smaller Tables

## Simple Solution: Bigger Pages

### Effect

If we use bigger pages, the page table size reduction exactly mirrors the increase in page size.

### The Problem

is that big pages lead to internal fragmentation

## Hybrid Approach: Paging and Segments

### our hybrid approach:

instead of having a single page table for the entire address space of the process, why not have one per logical segment? we might thus have three page tables, one for the code, heap, and stack parts of the address space.

### with segmentation,

we had a **base** register that told us where each segment lived in physical memory, and a **bound** register that told us the size of said segment.

### In our hybrid,

we still have those structures in the MMU; we use the base not to point to the segment itself but rather to hold the **physical address of the page table** of that segment. **The bounds register** is used to indicate the end of the page table

### On a TLB miss

the hardware uses the segment bits (SN) to determine which base and bounds pair to use. The hardware then takes the physical address therein and combines it with the VPN to form the address of the page table entry (PTE):

### The critical difference in our hybrid

is the presence of a bounds register per segment; each bounds register holds the value of the maximum valid page in the segment. unallocated pages between the stack and the heap no longer take up space in a page table

### Problems

**Use of segmentation,**
it still requires us to use segmentation;

**External fragmentation,**
this hybrid causes external fragmentation to arise again.

## Multi-level Page Tables

### The basic idea

First, chop up the page table into page-sized units; then, if an entire page of page-table entries (PTEs) is invalid, don't allocate that page of the page table at all. To track whether a page of the page table is valid use a new structure, called the page directory.

### page directory.

The page directory thus either can be used to tell you where a page of the page table is, or that the entire page of the page table contains no valid pages.

### Explained Behaviour

it just makes parts of the linear page table disappear and tracks which pages of the page table are allocated with the page directory.

### page directory entries

The page directory, in a simple two-level table, contains one entry per page of the page table. It consists of a number of page directory entries (PDE).

A PDE (minimally) has a **valid bit** and a **page frame number** (PFN), similar to a PTE.

**the meaning of this valid bit is slightly different:**
if the PDE is valid, it means that at least one of the pages of the page table that the entry points to (via the PFN) is valid, If the PDE is not valid the rest of the PDE is not defined.

### advantages

the multi-level table only allocates page-table space in proportion to the amount of address space you are using;

each portion of the page table fits neatly within a page, making it easier to manage memory; the OS can simply grab the next free page when it needs to allocate or grow a page table.

### Disadvantages

on a TLB miss, two loads from memory will be required to get the right translation information from the page table

Another obvious negative is increased complexity.

## More Than Two Levels

to make each piece of the page table fit within a single page if the page directory gets too big we build a further level of the tree, by splitting the page directory itself into multiple pages, and then adding another page directory on top of that, to point to the pages of the page directory.

## The Translation Process: Remember the TLB

#### process of address translation

before any of the complicated multilevel page table access occurs, the hardware first checks the TLB;

#### upon a hit,

the physical address is formed directly without accessing the page table at all,

#### upon a TLB miss

the hardware need to perform the full multi-level lookup.

The cost of our traditional two-level page table: two additional memory accesses to look up a valid translation.

## Inverted Page Tables

#### Description

instead of having many page tables we keep a single page table that has an entry for each **physical page** of the system.

#### The entry

tells us which process is using this page, and which virtual page of that process maps to this physical page.

#### Finding the correct entry

is now a matter of searching through this data structure. a hash table is often built over the base structure to speed up lookups.

## Swapping the Page Tables to Disk

the size of page tables, may be too big to fit into memory all at once. Thus, some systems place such page tables in **kernel virtual memory,** thereby allowing the system to **swap** some of these page tables to disk when memory pressure gets a little tight.