2023 Fall ECE 344: Operating Systems

Lecture 12 1.0.0

Page Tables

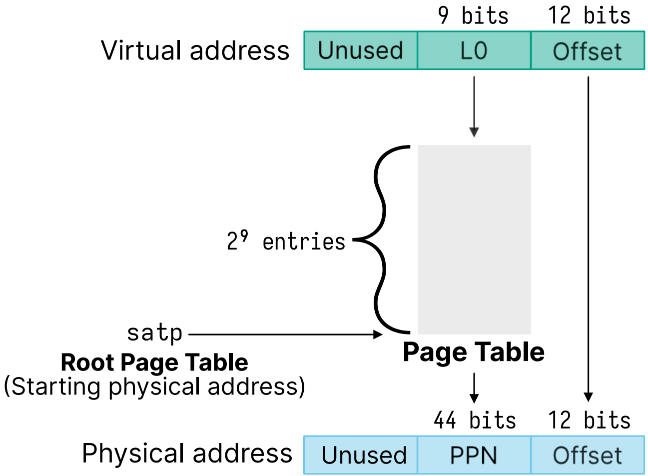
Jon Eyolfson

2023 Fall

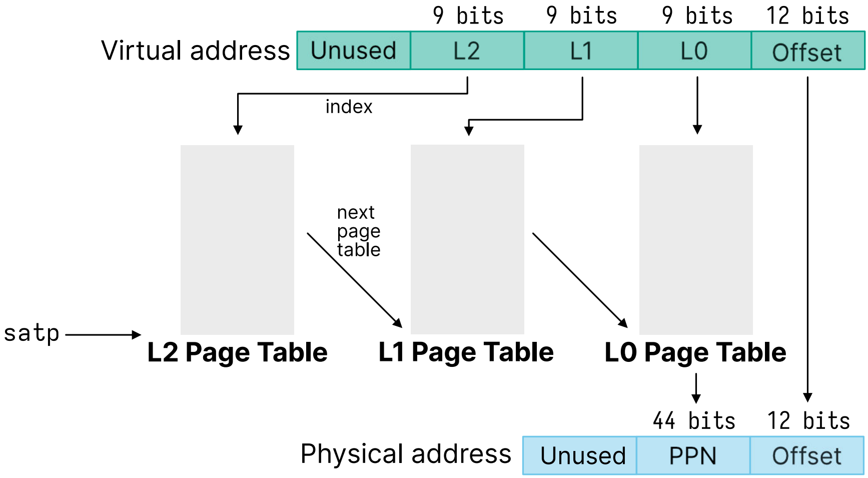
What Should We Do About the Page Table Size?

Most programs don’t use all the virtual memory space, how can we take advantage?

We Can Make Our Page Table Fit on a Page



Multi-Level Page Tables Save Space for Sparse Allocations



Page Allocation Uses A Free List

Given physical pages, the operating system maintains a free list (linked list)

The unused pages themselves contain the `next` pointer in the free list

- Physical memory gets initialized at boot

To allocate a page, you remove it from the free list

- To deallocate a page you add it back to the free list

Insight: Use a Page for Each Smaller Page Table

There are 512 (2^9) entries of 8 bytes(2^3) each, which is 4096 bytes

The PTE for L(N) points to the page table for L(N-1)

You follow these page tables until L0 and that contains the `PPN`

The Smaller Page Tables are Just Like Arrays

Instead of:

```
int page_table[512] // What's the size of this?
```

or

```
x = page_table[2]; // What's the offset of index 2?
```

You have:

```
PTE page_table[512]
```

where:

```
sizeof(page_table) == PAGE_SIZE
```

and

```
sizeof(page_table) = number of entries * sizeof(PTE)
```

Consider Just One Additional Level

Assume our process uses just one virtual address at `0x3FFFF008`
  or `0b11_1111_1111_1111_1111_0000_0000_1000`
  or `0b111111111_111111111_000000001000`

We'll just consider a 30-bit virtual address with a page size of 4096 bytes
  We would need a 2 MiB page table if we only had one ($2^{18} \times 2^{3}$)

Instead, we have a 4 KiB L1 page table ($2^{9} \times 2^{3}$) and a 4 KiB L0 page table
  Total of 8 KiB instead of 2 MiB

Note: worst case if we used all virtual addresses we would consume 2 MiB + 4 KiB

Translating `3FFFF008` with 2 Page Tables

Consider the L1 table with the entry:

Index	PPN
511	0x8

Consider the L0 table located at `0x8000` with the entry:

Index	PPN
511	0xCAFE

The final translated physical address would be: `0xCAFE008`