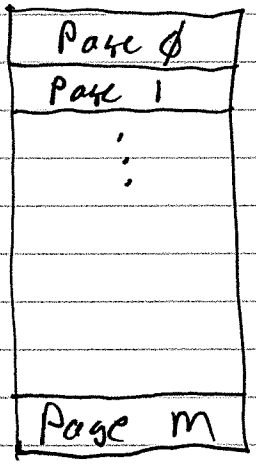
CS 411 Paging

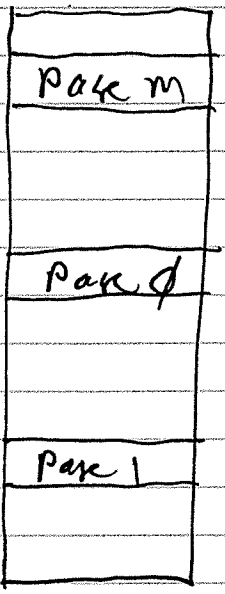
Idea: Break a process' virtual address space into fixed-size blocks (pages):

Process virtual address space



These pages are placed into page frames in physical memory:

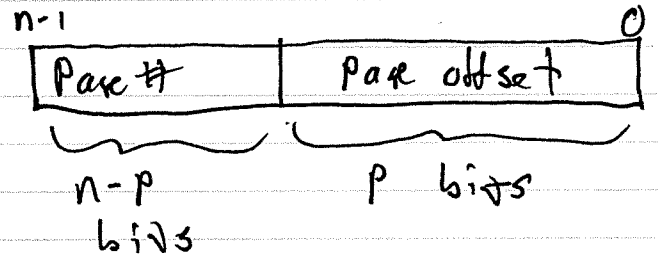
Physical memory



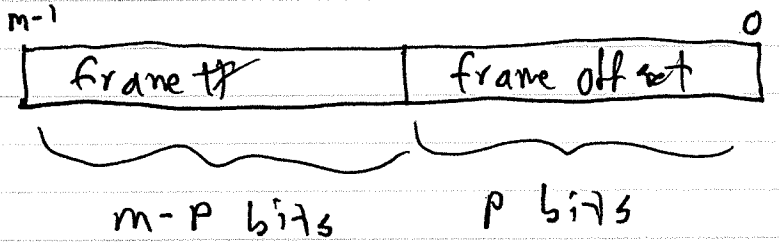
Pages and page frames are the same size, a power of 2, to simplify address translation

If page size is 2^p then page offset is p bits

Virtual address:

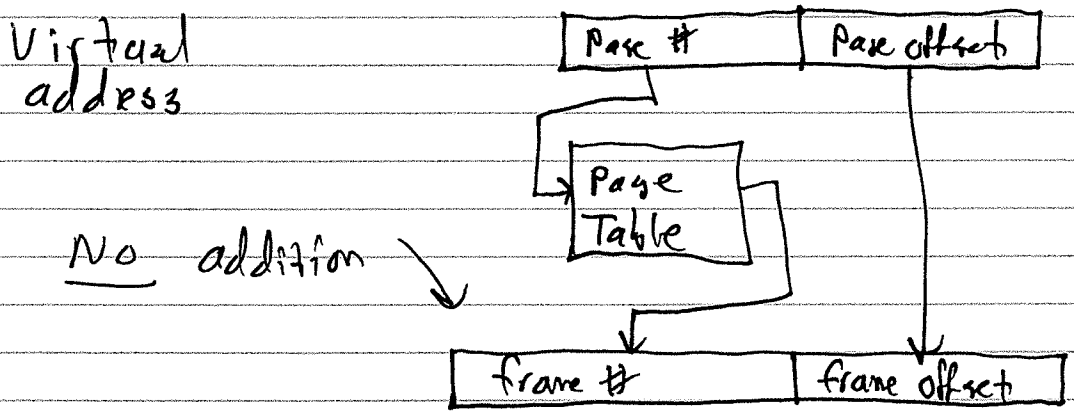


Physical address:



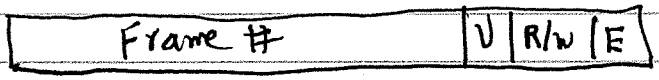
For now, we assume $n \leq m$

Address translation:



The page table is an "array" of page table entries, indexed by page #

Page Table Entry:



The page table is stored in memory, in Kernel space

Each process has its own page table

Problems with Paging

(1) It (they) increases memory access time

(2) Page tables consume Memory

Assume a 32-bit virtual address with a page size of 4KB

$4K = 2^{12}$

The page # field is $32 - 12 = 20$ bits in size. Therfore, the page table has 2^{20} entries. Assuming each entry is 4 bytes, the page table is $2^{22} = 4MB$ in size

Phoenix has 800 processes running The total size of all page tables is 3.2 GB, 4% of memory

(3) Internal Fragmentation

Questions

(1) For a system with a page size of 1KB, Virtual address space of 4MB, and a physical memory of 512MB
what is the layout of a virtual address and a physical address?
How large is the page table?
Assume 3 control bits per page table entry

(2) Problem 2 in the text with
-u 50