4/24/24 Paging

-> divide virtual & physical memory into fixed size pages

-> page level translation 52 bits 12 bits page # 10995et virtual addr V 1265b frame # 0480t physical addr VAS Z offect -> -> offset 250 -> -> the same page frame A 4 VAS 1 translate physical memory

stores translations for every page (hus page table walk) Page Table : -> Kernel sets up the page table, translation performed by hu Indexed by page # 5 frame # & page table lives in memory Physical for every vithal memory access, ne now have to make 2 physical Physical Address Memory access 4 Frame Offset Memory Frame 0 Frame 1 page table array Page Table Address Frame Access Virtual Address Page # Offset page table base register -Physical padde of the -Address % cr3 in x80 Offset Tik's vspace install water to this register. Frame M

Translation bookaside Buffer (TLB) -> caches the result of memory translation

& also caches the permission of the page mapping (may be out of sym from the Physical Memory Virtual Address Page# Offset Translation Lookaside Buffer (TLB) hit Virtual Page Page Frame Access Physical Address Matching Entry Frame Offset Miss Page Table Lookup

The Cost of single array page table

-> per process data structure -> 252 ontries take up a lot of space

How to reduce the page table size?

-> Larger page size = 2 MB & 16B page size (large 1 Super pages)

less pages, feurer entries page # 0H Smaller page taldes 252 => 2" entries

but ... Ibye internal fragmentation! Wasted [618

Maybe the problem is too newy page tables? -> Inverted page table : trades frame mapping instead (global) indexed by frame -> page the, pid -> How to look up given a virtual address ? -> search through each entry of the array worth we find a matching page # w/pict. works but very slow " -> Use a hash function to place pages > look up = hash (page #, pid) % frames => frame # inverted poge table what about hash collisions ? shared nemony ? Lost of unmapped page?

Page => frame mapping still better for look up!

Multilevel Page Tables : use indirection to only allocate entries for pages in use

