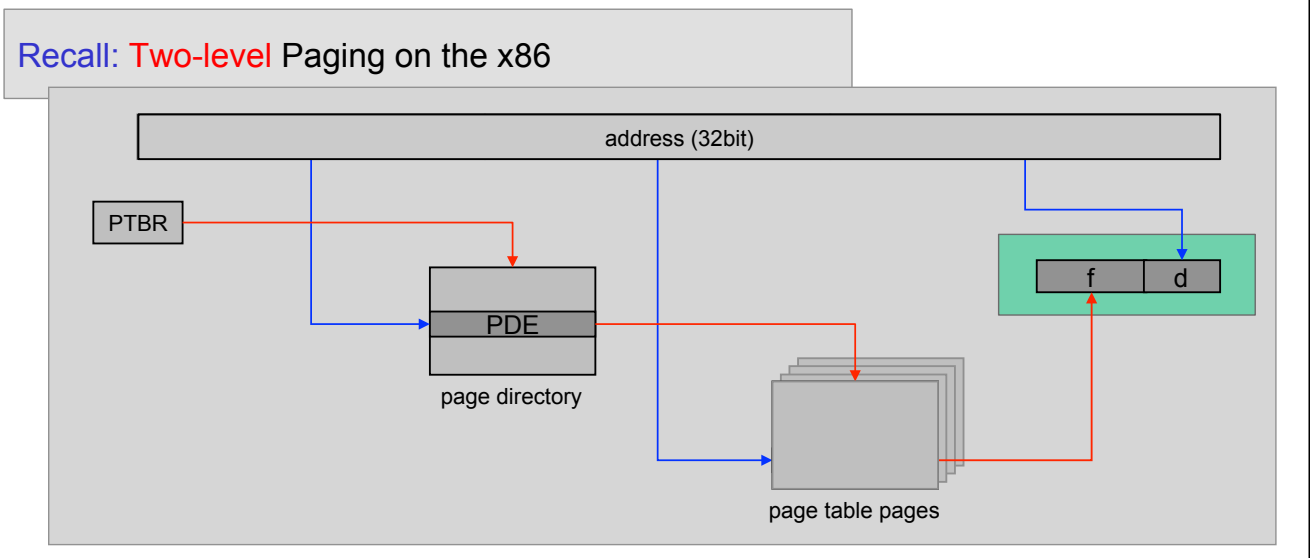
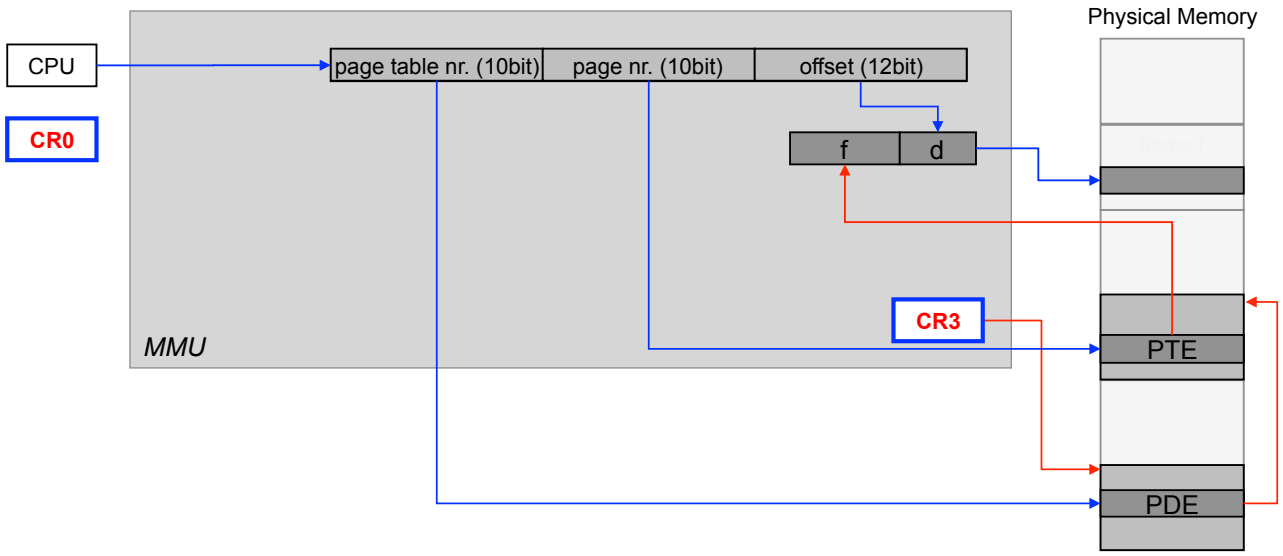
Paging on the x86

- Two-level paging in the x86
- Memory layout of page table
- Page table entries (PTEs)
- Page directory entries (PDEs)
- Mechanics of a page fault
- Managing meta data for OS
- The TLB?

The x86 has 2-Level Paging



Directories and Page Table Pages are in Memory



Directories and Page Table Pages are in Memory

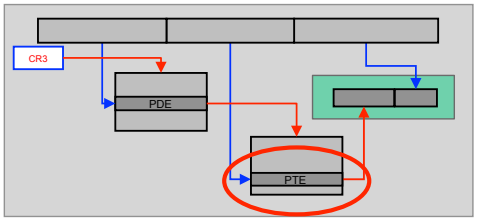
Page Table Entries

last 12 bit would always be zero!

Page Table Entry (PTE)

frame address?

32 bit



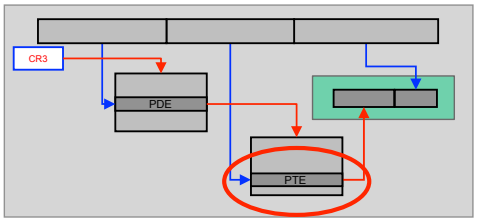
Page Table Entries

12 bit

Page Table Entry (PTE)

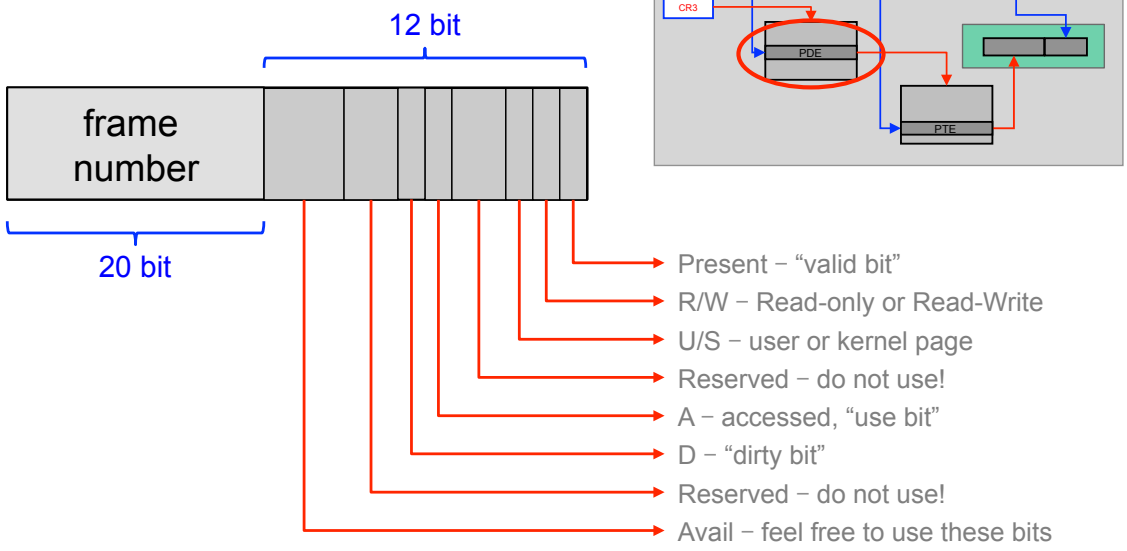
frame number

20 bit

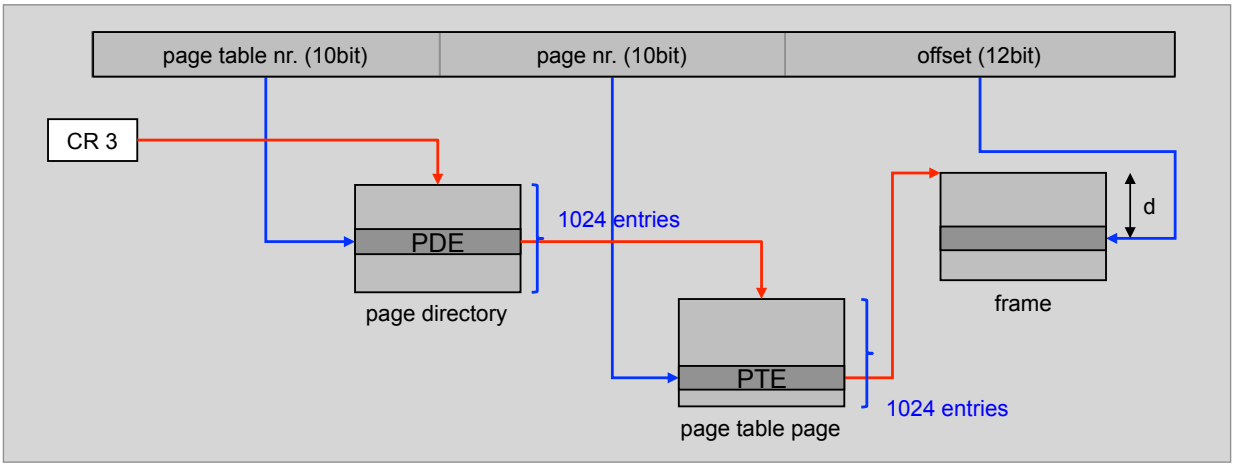


- Present – "valid bit"
- R/W – Read-only or Read-Write
- U/S – user or kernel page
- Reserved – do not use!
- A – accessed, "use bit"
- D – "dirty bit"
- Reserved – do not use!
- Avail – feel free to use these bits

What about Page Directory Entries?



Some Address Space Arithmetic



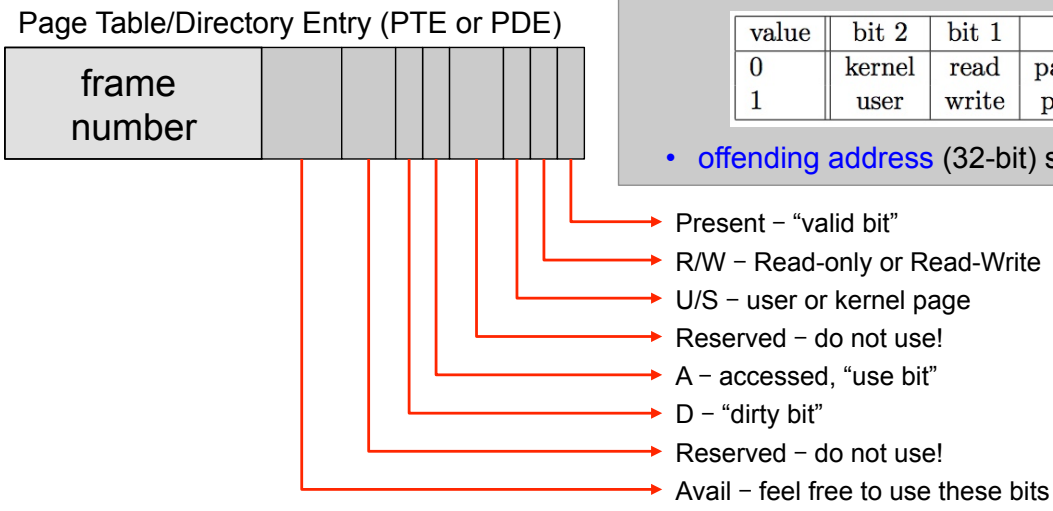
Page Fault Exceptions: Mechanics

Exception 14!

- information about fault stored in **`err_code`**

value	bit 2	bit 1	bit 0
0	kernel	read	page not present
1	user	write	protection fault

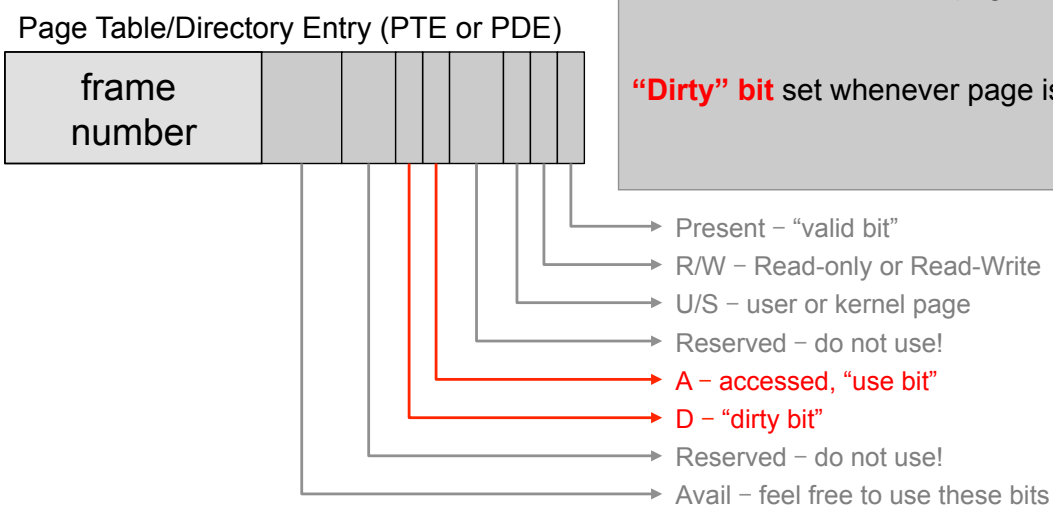
- offending address (32-bit) stored in CR2



Maintaining Meta-Data

"Use" bit set whenever page is referenced.

"Dirty" bit set whenever page is written.



Where is the TLB?!

TLB is largely invisible.

Sometimes we need to delete entries in the TLB!

- reload the CR 2 register (PTBR)

Paging on the x86

- Address translation
- Memory layout of page table
- PDEs and PTEs
- Mechanics of a page fault
- Managing meta data for OS
- TLB