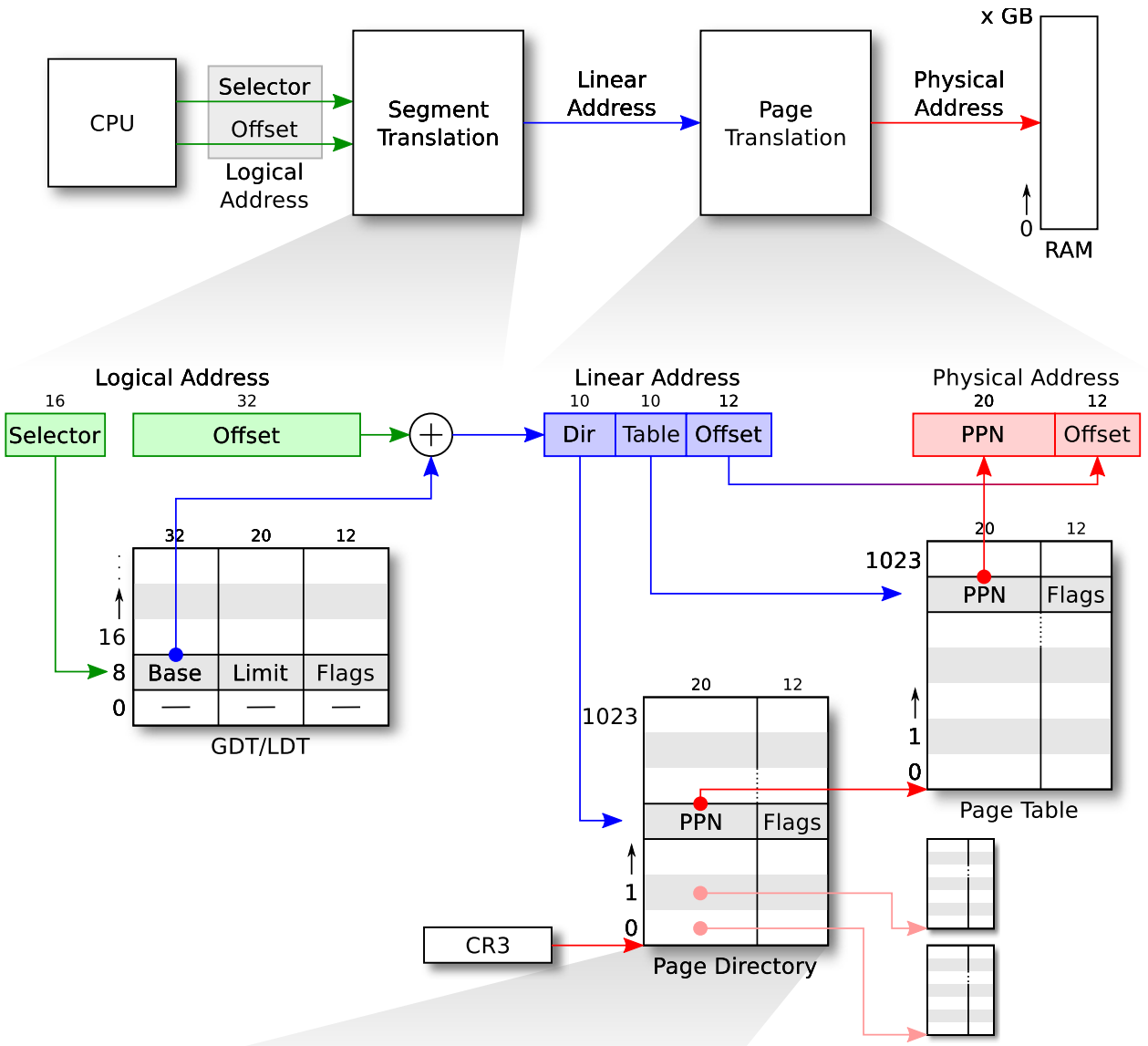
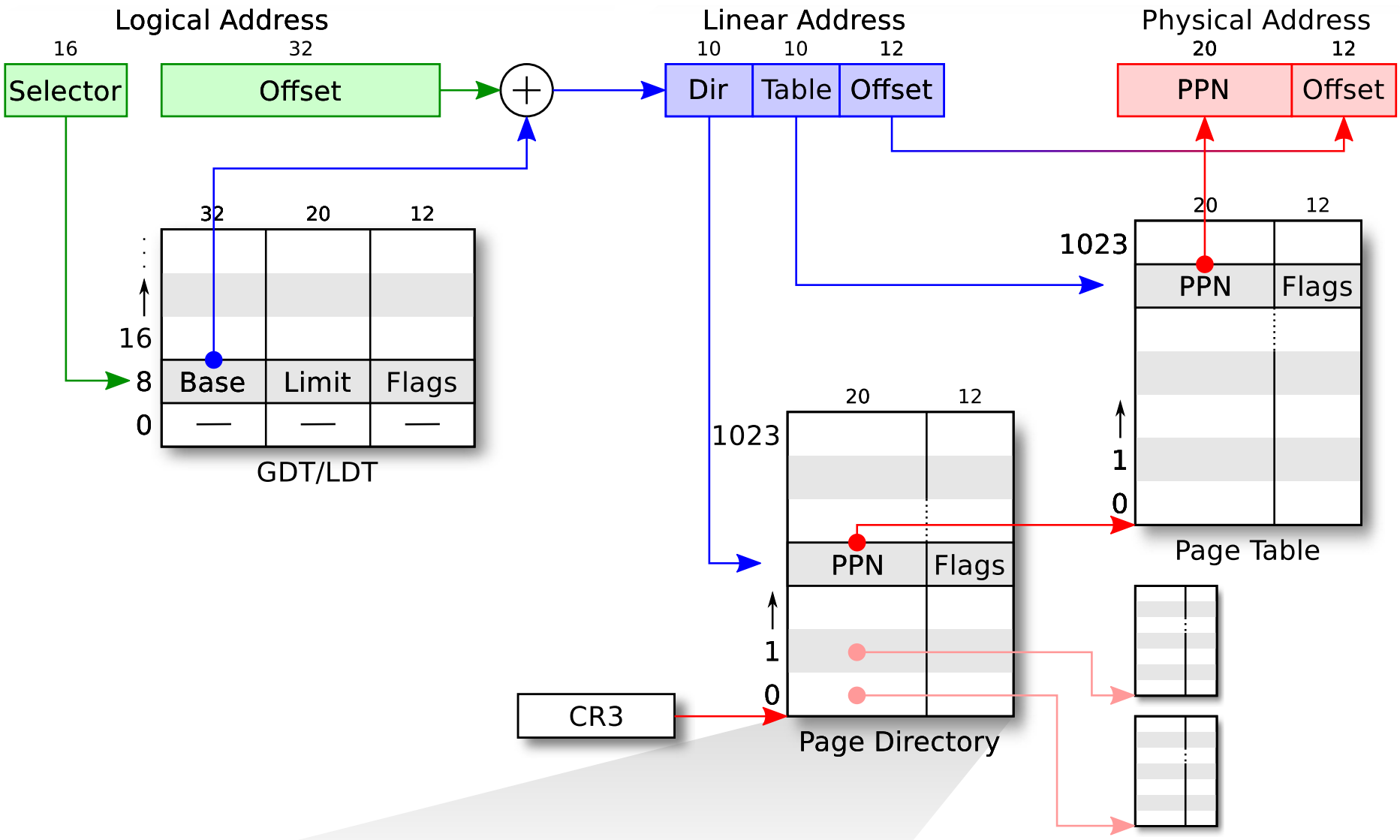
236366 Operating Systems Engineering
Recitation #3 (part 1):
Address Translation and Paging

Presented by:
Ilia Kravets
<ilia@cs>

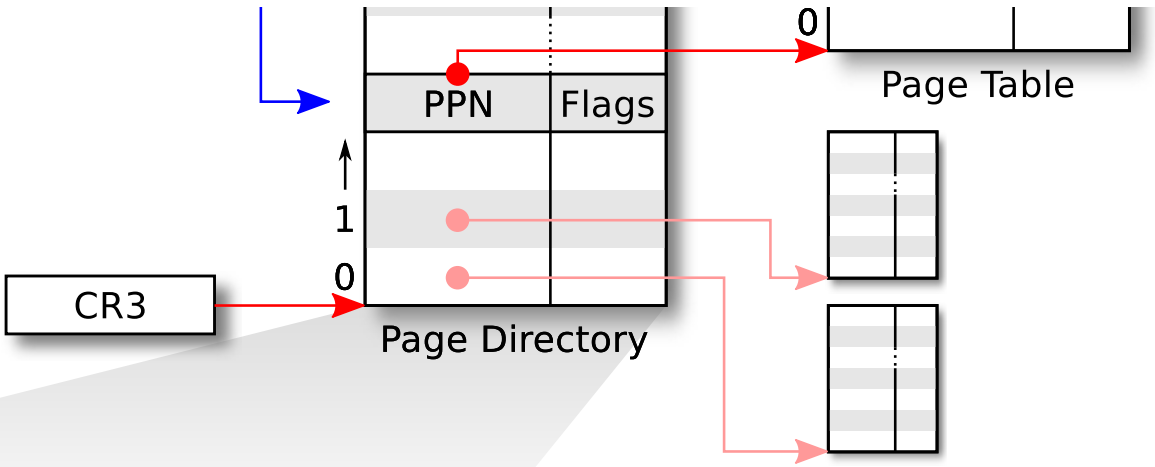
x86 address translation

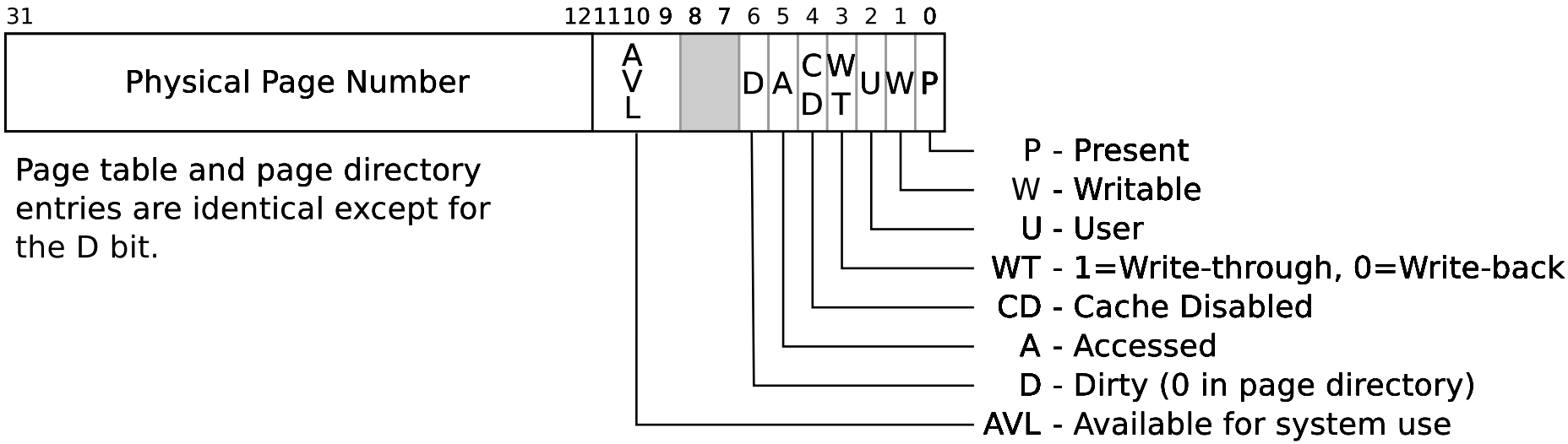


x86 address translation



x86 address translation





Page table and page directory entries are identical except for the D bit.

Using x86 paging

- start using paging
  - setup PD, PT's
  - point to PD: `%cr3 = PHYSADDR(PD)`
  - enable paging: `%cr0 = %cr0 | CR0_PG`
  - may "disable" segmentation
- flush TLB when updating PD/PT
  - full flush: reload `%cr0`
  - specific entry only: `invlpg VA`
- access control
  - S/U bit
  - R/W bit (& `CR0_WP`)
  - `%cr3` can be modified by kernel only
    - what about PD/PT?

Using x86 paging

- start using paging
  - setup PD, PT's
  - point to PD: `%cr3 = PHYSADDR(PD)`
  - enable paging: `%cr0 = %cr0 | CR0_PG`
  - may "disable" segmentation
- flush TLB when updating PD/PT
  - full flush: reload `%cr0`
  - specific entry only: `invlpg VA`
- access control
  - S/U bit
  - R/W bit (& `CR0_WP`)
  - `%cr3` can be modified by kernel only
    - what about PD/PT?

Using x86 paging

- start using paging
  - setup PD, PT's
  - point to PD: `%cr3 = PHYSADDR(PD)`
  - enable paging: `%cr0 = %cr0 | CR0_PG`
  - may "disable" segmentation
- flush TLB when updating PD/PT
  - full flush: reload `%cr0`
  - specific entry only: `invlpg VA`
- access control
  - S/U bit
  - R/W bit (& `CR0_WP`)
  - `%cr3` can be modified by kernel only
    - what about PD/PT?

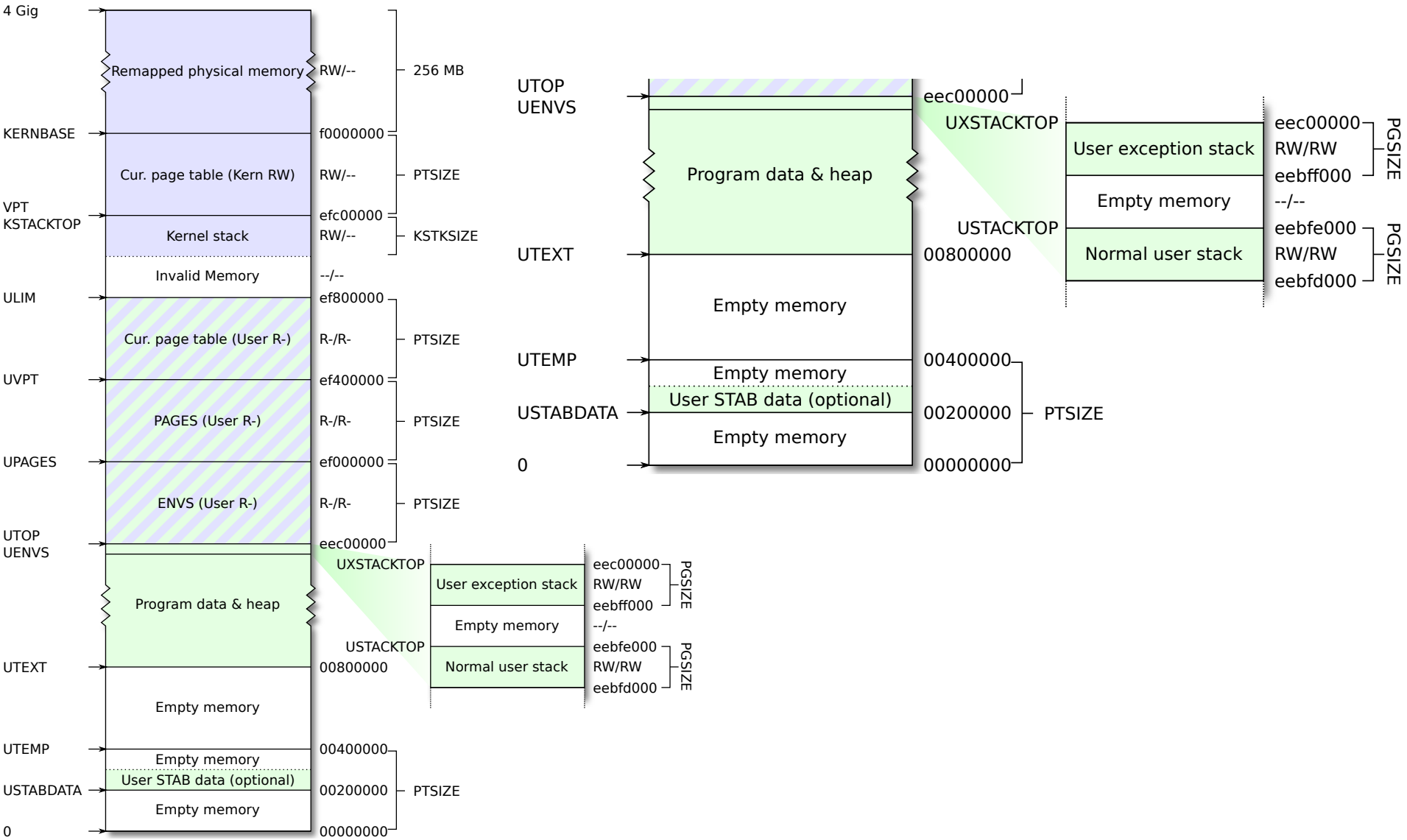
MMU: linear to physical address translation

```
uint translate (uint la, bool user, bool write)
{
  uint pde;
  pde = read_mem (%CR3 + 4*(la >> 22));
  access (pde, user, write);
  pte = read_mem ( (pde & 0xfffff000) + 4*((la >> 12) & 0x3ff));
  access (pte, user, write);
  return (pte & 0xfffff000) + (la & 0xfff);
}

// check protection. pxe is a pte or pde.
// user is true if CPL==3
void access (uint pxe, bool user, bool write)
{
  if (!(pxe & PG_P)
     => page fault -- page not present
  if (!(pxe & PG_U) && user)
     => page fault -- not access for user

  if (write && !(pxe & PG_W)) {
    if (user)
       => page fault -- not writable
    if (%CR0 & CR0_WP)
       => page fault -- not writable
  }
}
```

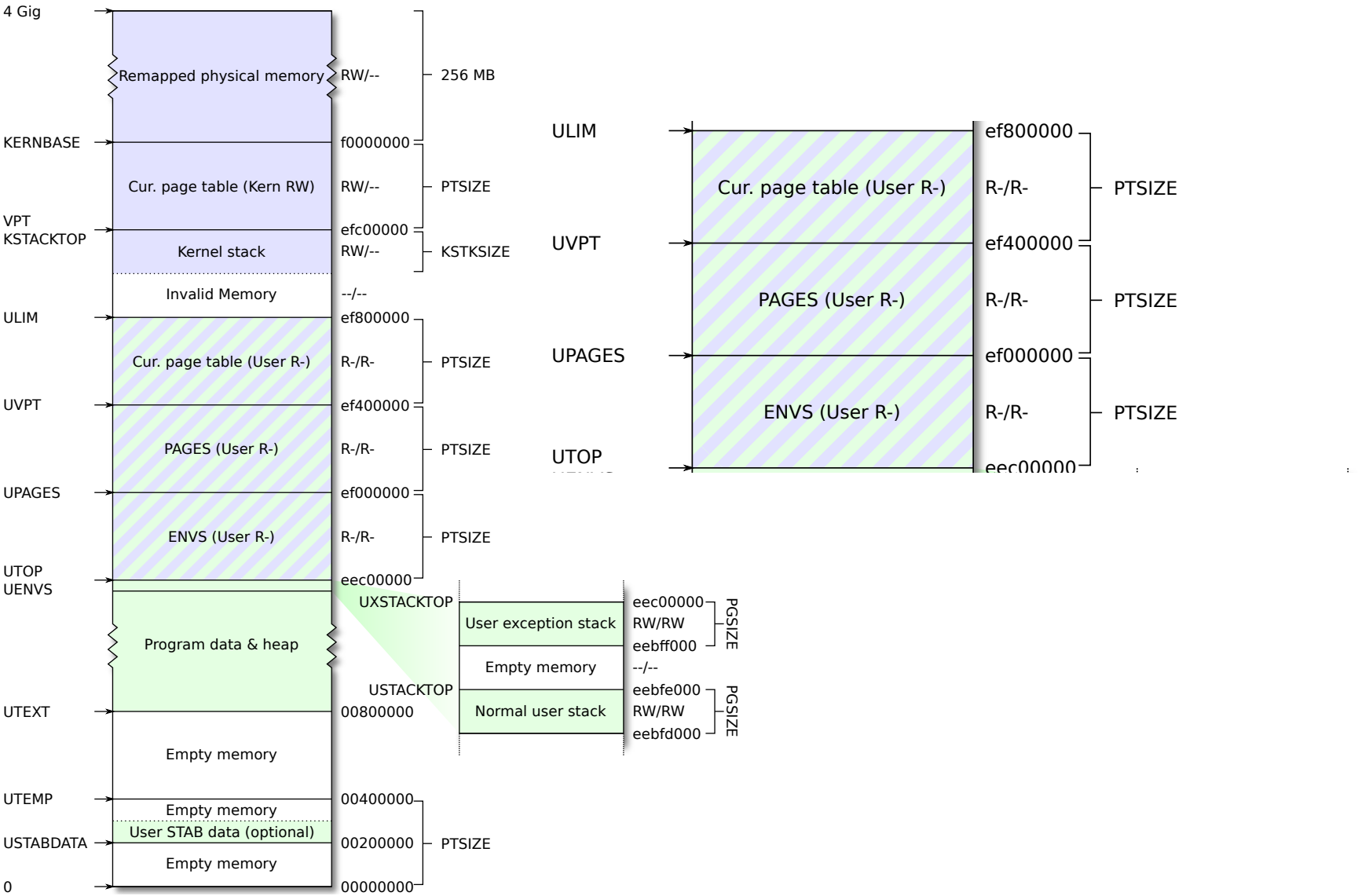
JOS virtual address space

- segmentation is “disabled”
  - identity $VA \rightarrow LA$ mapping (base=0, limit=4G)
  - user/kernel differs in DPL only (about it later. . .)
- per process page table
- same address space for both user and kernel
  - kernel is at the top
  - kernel part is the same for all processes
- limit user access (non-user addresses are usually inaccessible)
- JOS tricks
  - some kernel memory is remapped read only for user
  - physical memory is remapped continuously
  - virtual page table (VPT)

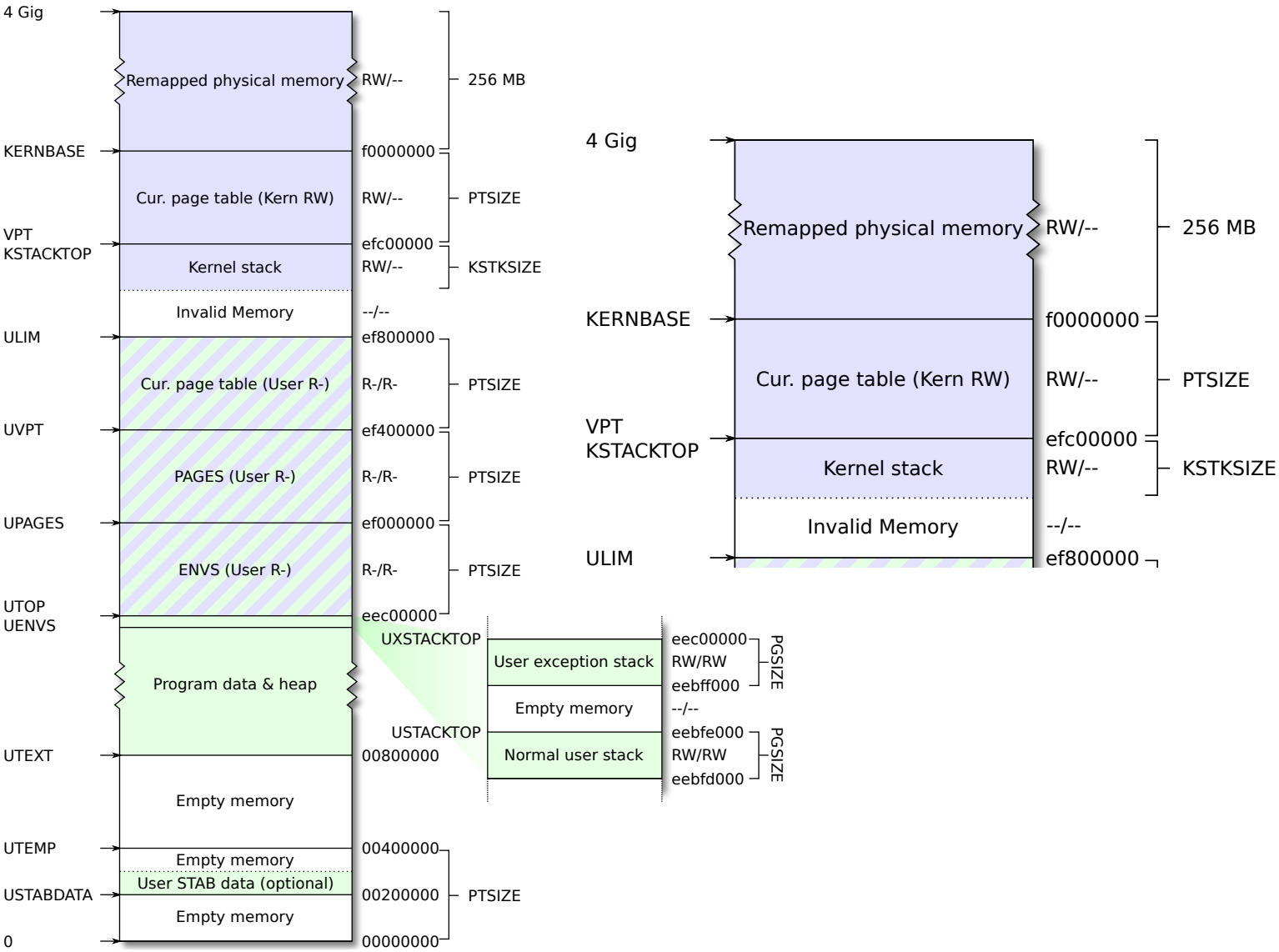
JOS virtual address space map



JOS virtual address space map



JOS virtual address space map



JOS virtual page table

- PT is just a mapping function $f(linear) \rightarrow physical$
- implemented in x86 as 2-level page walk
- OS needs to re-implement $f()$ in software
- wouldn't it be nice to just use `pagetable[linear]`?
- so 2nd level PTs should form a continuous 4MB array
  - continuous in *virtual* memory
  - MMU translates all memory accesses using PT, remember?
  - just need to setup one 2nd level PT which will map *all* 2nd level PTs
    - and maintain it whenever PD is updated
  - but PD does exactly this!
  - let's just use PD as 2nd level PT
    - bonus: "self maintained" mapping
  - put physical address of PD into PD's entry #*x*
  - we got a 4MB array starting at VA $x << 22$
    - what's the value of *x* in JOS?
  - UVPT – read-only user's view of VPT
    - how to setup?

JOS virtual page table

- PT is just a mapping function $f(linear) \rightarrow physical$
- implemented in x86 as 2-level page walk
- OS needs to re-implement $f()$ in software
- wouldn't it be nice to just use `pagetable[linear]`?
- so 2nd level PTs should form a continuous 4MB array
  - continuous in *virtual* memory
  - MMU translates all memory accesses using PT, remember?
  - just need to setup one 2nd level PT which will map *all* 2nd level PTs
    - and maintain it whenever PD is updated
  - but PD does exactly this!
  - let's just use PD as 2nd level PT
    - bonus: "self maintained" mapping
  - put physical address of PD into PD's entry #*x*
  - we got a 4MB array starting at VA $x << 22$
    - what's the value of *x* in JOS?
  - UVPT – read-only user's view of VPT
    - how to setup?

JOS virtual page table

- PT is just a mapping function $f(linear) \rightarrow physical$
- implemented in x86 as 2-level page walk
- OS needs to re-implement $f()$ in software
- wouldn't it be nice to just use `pagetable[linear]`?
- so 2nd level PTs should form a continuous 4MB array
  - continuous in *virtual* memory
  - MMU translates all memory accesses using PT, remember?
  - just need to setup one 2nd level PT which will map *all* 2nd level PTs
    - and maintain it whenever PD is updated
  - but PD does exactly this!
  - let's just use PD as 2nd level PT
    - bonus: "self-maintained" mapping
  - put physical address of PD into PD's entry #*x*
  - we got a 4MB array starting at VA $x << 22$
    - what's the value of *x* in JOS?
  - UVPT – read-only user's view of VPT
    - how to setup?

JOS virtual page table

- PT is just a mapping function $f(linear) \rightarrow physical$
- implemented in x86 as 2-level page walk
- OS needs to re-implement $f()$ in software
- wouldn't it be nice to just use `pagetable[linear]`?
- so 2nd level PTs should form a continuous 4MB array
  - continuous in *virtual* memory
  - MMU translates all memory accesses using PT, remember?
  - just need to setup one 2nd level PT which will map *all* 2nd level PTs
    - and maintain it whenever PD is updated
  - but PD does exactly this!
  - let's just use PD as 2nd level PT
    - bonus: "self-maintained" mapping
  - put physical address of PD into PD's entry #*x*
  - we got a 4MB array starting at VA $x << 22$
    - what's the value of *x* in JOS?
  - UVPT – read-only user's view of VPT
    - how to setup?

JOS virtual page table

- PT is just a mapping function $f(linear) \rightarrow physical$
- implemented in x86 as 2-level page walk
- OS needs to re-implement $f()$ in software
- wouldn't it be nice to just use `pagetable[linear]`?
- so 2nd level PTs should form a continuous 4MB array
  - continuous in *virtual* memory
  - MMU translates all memory accesses using PT, remember?
  - just need to setup one 2nd level PT which will map *all* 2nd level PTs
    - and maintain it whenever PD is updated
  - but PD does exactly this!
  - let's just use PD as 2nd level PT
    - bonus: "self-maintained" mapping
  - put physical address of PD into PD's entry #x
  - we got a 4MB array starting at VA $x << 22$
    - what's the value of x in JOS?
  - UVPT – read-only user's view of VPT
    - how to setup?

JOS virtual page table

- PT is just a mapping function $f(linear) \rightarrow physical$
- implemented in x86 as 2-level page walk
- OS needs to re-implement $f()$ in software
- wouldn't it be nice to just use `pagetable[linear]`?
- so 2nd level PTs should form a continuous 4MB array
  - continuous in *virtual* memory
  - MMU translates all memory accesses using PT, remember?
  - just need to setup one 2nd level PT which will map *all* 2nd level PTs
    - and maintain it whenever PD is updated
  - but PD does exactly this!
  - let's just use PD as 2nd level PT
    - bonus: "self-maintained" mapping
  - put physical address of PD into PD's entry #x
  - we got a 4MB array starting at VA $x << 22$
    - what's the value of x in JOS?
  - UVPT – read-only user's view of VPT
    - how to setup?

JOS virtual page table

- PT is just a mapping function $f(linear) \rightarrow physical$
- implemented in x86 as 2-level page walk
- OS needs to re-implement $f()$ in software
- wouldn't it be nice to just use `pagetable[linear]`?
- so 2nd level PTs should form a continuous 4MB array
  - continuous in *virtual* memory
  - MMU translates all memory accesses using PT, remember?
  - just need to setup one 2nd level PT which will map *all* 2nd level PTs
    - and maintain it whenever PD is updated
  - but PD does exactly this!
  - let's just use PD as 2nd level PT
    - bonus: "self-maintained" mapping
  - put physical address of PD into PD's entry #x
  - we got a 4MB array starting at VA $x << 22$
    - what's the value of x in JOS?
  - UVPT – read-only user's view of VPT
    - how to setup?

JOS virtual page table

- PT is just a mapping function $f(linear) \rightarrow physical$
- implemented in x86 as 2-level page walk
- OS needs to re-implement $f()$ in software
- wouldn't it be nice to just use `pagetable[linear]`?
- so 2nd level PTs should form a continuous 4MB array
  - continuous in *virtual* memory
  - MMU translates all memory accesses using PT, remember?
  - just need to setup one 2nd level PT which will map *all* 2nd level PTs
    - and maintain it whenever PD is updated
  - but PD does exactly this!
  - let's just use PD as 2nd level PT
    - bonus: "self-maintained" mapping
  - put physical address of PD into PD's entry #x
  - we got a 4MB array starting at VA $x << 22$
    - what's the value of x in JOS?
  - UVPT – read-only user's view of VPT
    - how to setup?

JOS virtual page table

- PT is just a mapping function $f(linear) \rightarrow physical$
- implemented in x86 as 2-level page walk
- OS needs to re-implement $f()$ in software
- wouldn't it be nice to just use `pagetable[linear]`?
- so 2nd level PTs should form a continuous 4MB array
  - continuous in *virtual* memory
  - MMU translates all memory accesses using PT, remember?
  - just need to setup one 2nd level PT which will map *all* 2nd level PTs
    - and maintain it whenever PD is updated
  - but PD does exactly this!
  - let's just use PD as 2nd level PT
    - bonus: "self-maintained" mapping
  - put physical address of PD into PD's entry #x
  - we got a 4MB array starting at VA $x << 22$
    - what's the value of x in JOS?
  - UVPT – read-only user's view of VPT
    - how to setup?

JOS virtual page table

- PT is just a mapping function $f(linear) \rightarrow physical$
- implemented in x86 as 2-level page walk
- OS needs to re-implement $f()$ in software
- wouldn't it be nice to just use `pagetable[linear]`?
- so 2nd level PTs should form a continuous 4MB array
  - continuous in *virtual* memory
  - MMU translates all memory accesses using PT, remember?
  - just need to setup one 2nd level PT which will map *all* 2nd level PTs
    - and maintain it whenever PD is updated
  - but PD does exactly this!
  - let's just use PD as 2nd level PT
    - bonus: "self-maintained" mapping
  - put physical address of PD into PD's entry #x
  - we got a 4MB array starting at VA $x << 22$
    - what's the value of x in JOS?
  - UVPT – read-only user's view of VPT
    - how to setup?

JOS virtual page table

- PT is just a mapping function $f(linear) \rightarrow physical$
- implemented in x86 as 2-level page walk
- OS needs to re-implement $f()$ in software
- wouldn't it be nice to just use `pagetable[linear]`?
- so 2nd level PTs should form a continuous 4MB array
  - continuous in *virtual* memory
  - MMU translates all memory accesses using PT, remember?
  - just need to setup one 2nd level PT which will map *all* 2nd level PTs
    - and maintain it whenever PD is updated
  - but PD does exactly this!
  - let's just use PD as 2nd level PT
    - bonus: "self-maintained" mapping
  - put physical address of PD into PD's entry #x
  - we got a 4MB array starting at VA $x << 22$
    - what's the value of x in JOS?
  - UVPT – read-only user's view of VPT
    - how to setup?

JOS virtual page table

- PT is just a mapping function $f(linear) \rightarrow physical$
- implemented in x86 as 2-level page walk
- OS needs to re-implement $f()$ in software
- wouldn't it be nice to just use `pagetable[linear]`?
- so 2nd level PTs should form a continuous 4MB array
  - continuous in *virtual* memory
  - MMU translates all memory accesses using PT, remember?
  - just need to setup one 2nd level PT which will map *all* 2nd level PTs
    - and maintain it whenever PD is updated
  - but PD does exactly this!
  - let's just use PD as 2nd level PT
    - bonus: "self-maintained" mapping
  - put physical address of PD into PD's entry #x
  - we got a 4MB array starting at VA $x << 22$
    - what's the value of x in JOS?
  - UVPT – read-only user's view of VPT
    - how to setup?

JOS virtual page table

- PT is just a mapping function $f(linear) \rightarrow physical$
- implemented in x86 as 2-level page walk
- OS needs to re-implement $f()$ in software
- wouldn't it be nice to just use `pagetable[linear]`?
- so 2nd level PTs should form a continuous 4MB array
  - continuous in *virtual* memory
  - MMU translates all memory accesses using PT, remember?
  - just need to setup one 2nd level PT which will map *all* 2nd level PTs
    - and maintain it whenever PD is updated
  - but PD does exactly this!
  - let's just use PD as 2nd level PT
    - bonus: "self-maintained" mapping
  - put physical address of PD into PD's entry #*x*
  - we got a 4MB array starting at VA $x << 22$
    - what's the value of *x* in JOS?
  - UVPT – read-only user's view of VPT
    - how to setup?

JOS virtual page table

- PT is just a mapping function $f(linear) \rightarrow physical$
- implemented in x86 as 2-level page walk
- OS needs to re-implement $f()$ in software
- wouldn't it be nice to just use `pagetable[linear]`?
- so 2nd level PTs should form a continuous 4MB array
  - continuous in *virtual* memory
  - MMU translates all memory accesses using PT, remember?
  - just need to setup one 2nd level PT which will map *all* 2nd level PTs
    - and maintain it whenever PD is updated
  - but PD does exactly this!
  - let's just use PD as 2nd level PT
    - bonus: "self-maintained" mapping
  - put physical address of PD into PD's entry #x
  - we got a 4MB array starting at VA $x << 22$
    - what's the value of x in JOS?
  - UVPT – read-only user's view of VPT
    - how to setup?

JOS virtual page table

- PT is just a mapping function $f(linear) \rightarrow physical$
- implemented in x86 as 2-level page walk
- OS needs to re-implement $f()$ in software
- wouldn't it be nice to just use `pagetable[linear]`?
- so 2nd level PTs should form a continuous 4MB array
  - continuous in *virtual* memory
  - MMU translates all memory accesses using PT, remember?
  - just need to setup one 2nd level PT which will map *all* 2nd level PTs
    - and maintain it whenever PD is updated
  - but PD does exactly this!
  - let's just use PD as 2nd level PT
    - bonus: "self-maintained" mapping
  - put physical address of PD into PD's entry #x
  - we got a 4MB array starting at VA $x << 22$
    - what's the value of x in JOS?
  - UVPT – read-only user's view of VPT
    - how to setup?

JOS virtual page table

- PT is just a mapping function $f(linear) \rightarrow physical$
- implemented in x86 as 2-level page walk
- OS needs to re-implement $f()$ in software
- wouldn't it be nice to just use `pagetable[linear]`?
- so 2nd level PTs should form a continuous 4MB array
  - continuous in *virtual* memory
  - MMU translates all memory accesses using PT, remember?
  - just need to setup one 2nd level PT which will map *all* 2nd level PTs
    - and maintain it whenever PD is updated
  - but PD does exactly this!
  - let's just use PD as 2nd level PT
    - bonus: "self-maintained" mapping
  - put physical address of PD into PD's entry #x
  - we got a 4MB array starting at VA $x << 22$
    - what's the value of x in JOS?
  - UVPT – read-only user's view of VPT
    - how to setup?

JOS virtual page table

- PT is just a mapping function $f(linear) \rightarrow physical$
- implemented in x86 as 2-level page walk
- OS needs to re-implement $f()$ in software
- wouldn't it be nice to just use `pagetable[linear]`?
- so 2nd level PTs should form a continuous 4MB array
  - continuous in *virtual* memory
  - MMU translates all memory accesses using PT, remember?
  - just need to setup one 2nd level PT which will map *all* 2nd level PTs
    - and maintain it whenever PD is updated
  - but PD does exactly this!
  - let's just use PD as 2nd level PT
    - bonus: "self-maintained" mapping
  - put physical address of PD into PD's entry #x
  - we got a 4MB array starting at VA $x << 22$
    - what's the value of x in JOS?
  - UVPT – read-only user's view of VPT
    - how to setup?