☐ 1. Last time
☐ 2. x86-64: addresses
☐ 3. x86-64: page table structures
☐ 4. TLBs   ONE HANDOUT

---

1. Last time

VA: | VPN | offset |

page table

→ PPN
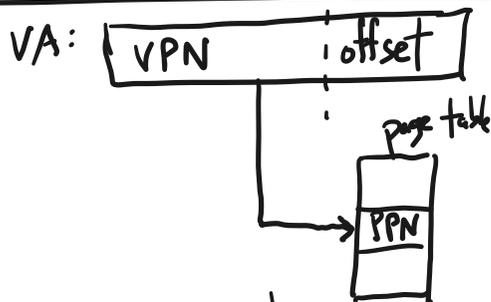
- purpose of VM (virtual mem)

- central mechanism: page table

- idealized page table: VPN is an index into a giant table, PPN is the contents of the table at the index

- thus, pg table implements a map from VPN → PPN

- in reality, it's a map from VPN → PPN ∪ {ϕ}, because a VPN might not have a valid mapping in the table.

- NOTE: VPN + PPN do not necessarily have the same # of bits.

· Because the table would be gigantic, it's not materialized as a linear table. Instead, the architecture specifies multilevel page tables

2. x86-64: addresses

VAs:

64 bits = 8 bytes
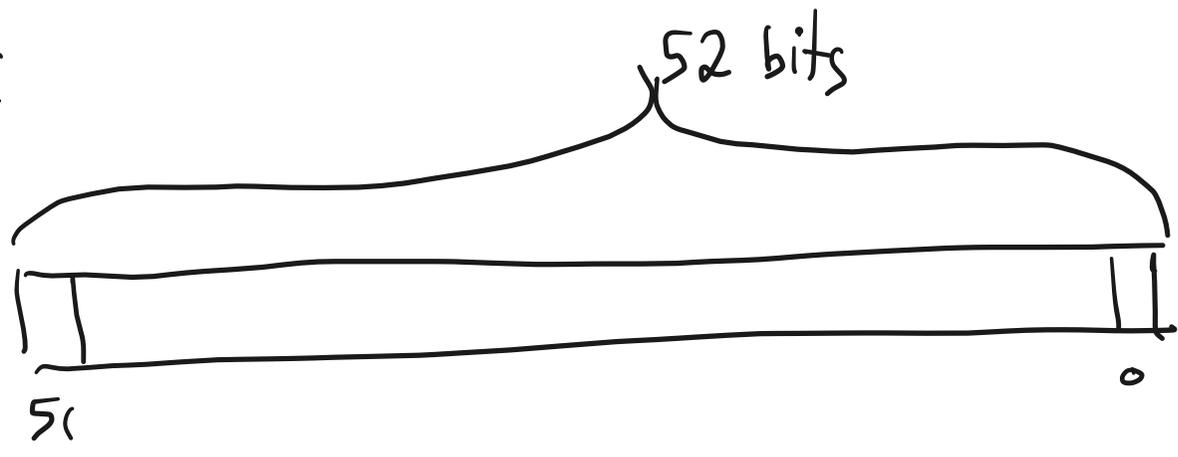
36 bits for VPN

bit 63   48 47   12 11   0
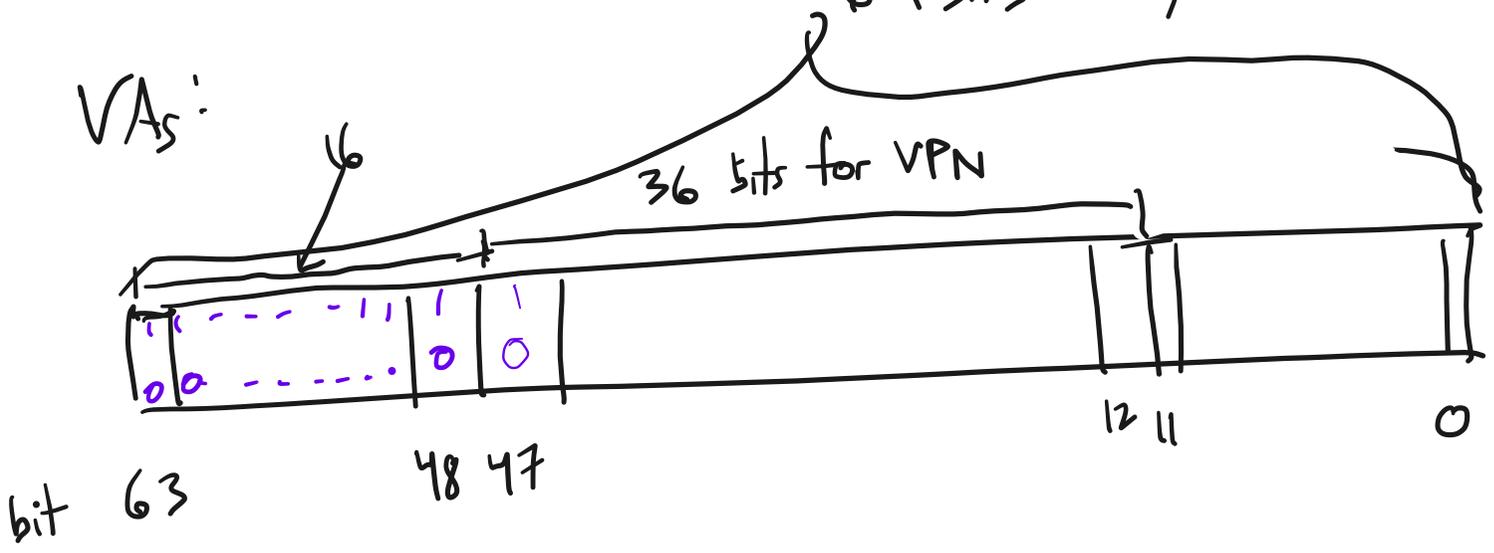
$$[-2^{47}, 2^{47}-1]$$

Address space has 48 usable bits, 2^{48} possible addresses (each addresses a byte).
Thus, 256 TB.

57 bits : 128 PB

PAs:

52 bits

5(

Physical memory can be addressed up to
52 bits.
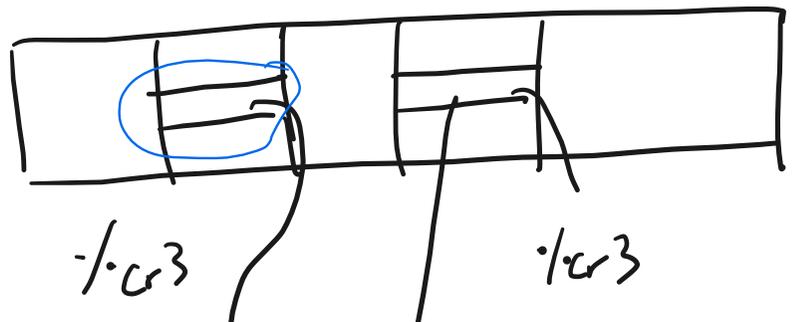
How much physical memory can thus be supported?

4 PB

Mapping: going from 48-bit number (VA) to 52-bit number (PA) at the granularity of ranges of 2^{12}.

So it's really a mapping from 36-bit numbers to 40-bit numbers.

---

3. Page table structures

[see handout]

proc table

-/·cr3   ·/·cr3
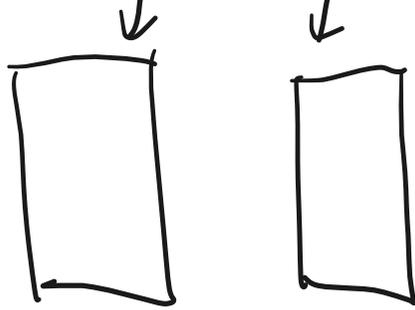
Exercise: OS wants to map

↳ VA 0x0202000 to
↳ PA 0x3000

VPN: 0x0202 →
PPN 0x3

and make it accessible to user-level but read-only.

What do pg structures look like?

0x202:

L1 page table

18
{0......0} 0 0 0 0 0 0 0 | 0 0 1 0 | 0 0 0 0 | 0 0 1 0

36 bits in all

36 = 48 - 12

-/.cr3

[@A]

511

S0   @A

L2

u/s R/w P

511

L3

U/s R/w P

511

2

0

L4

U/s R/w P

511

2   0x3   101

1

0

40 bits
PPN

U/s R/w P

0 ----- 011   //// | | 0 | |
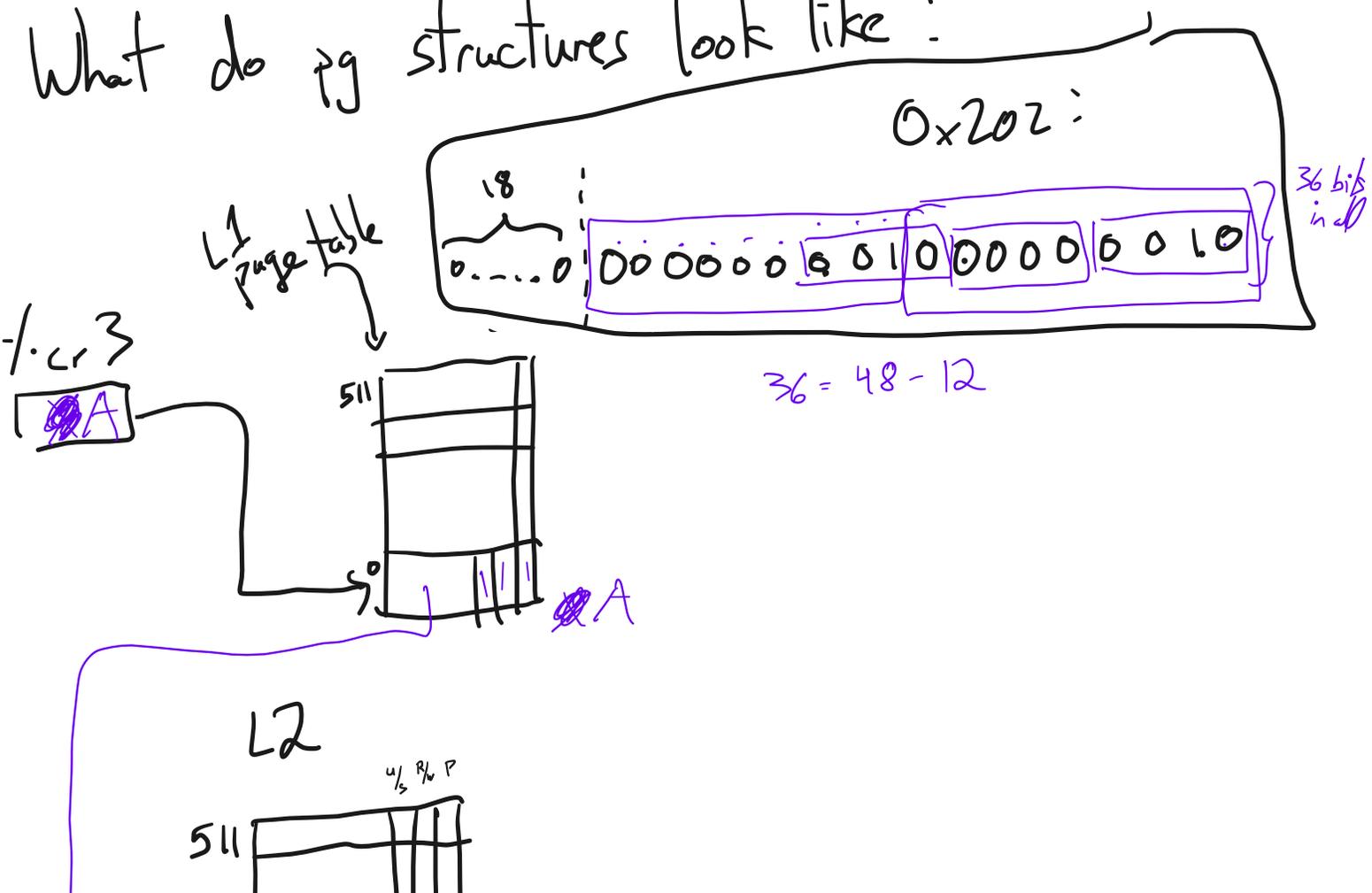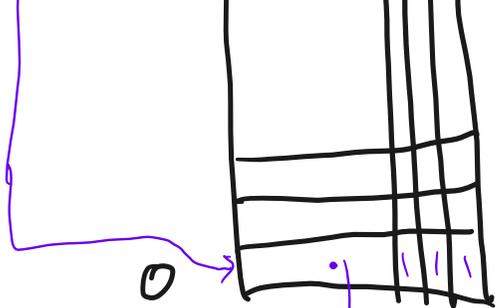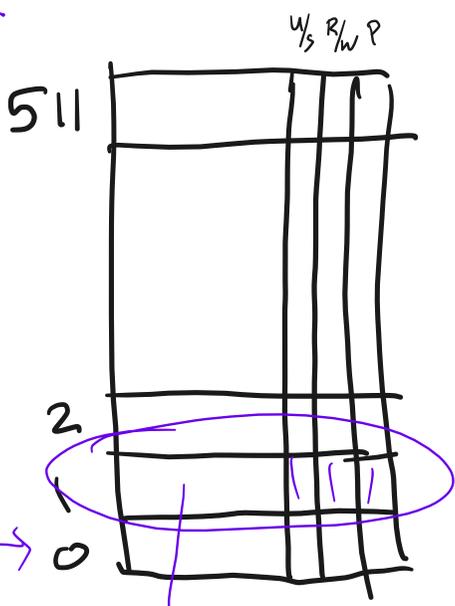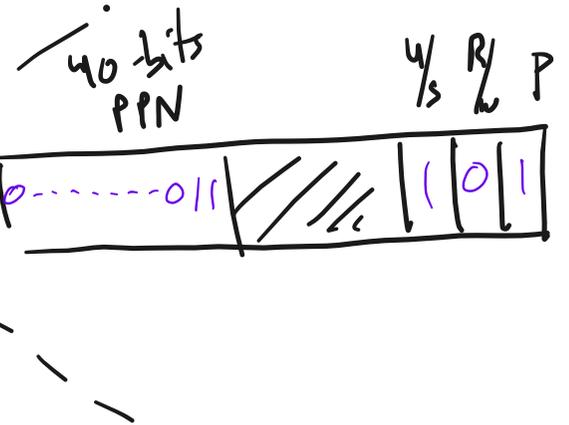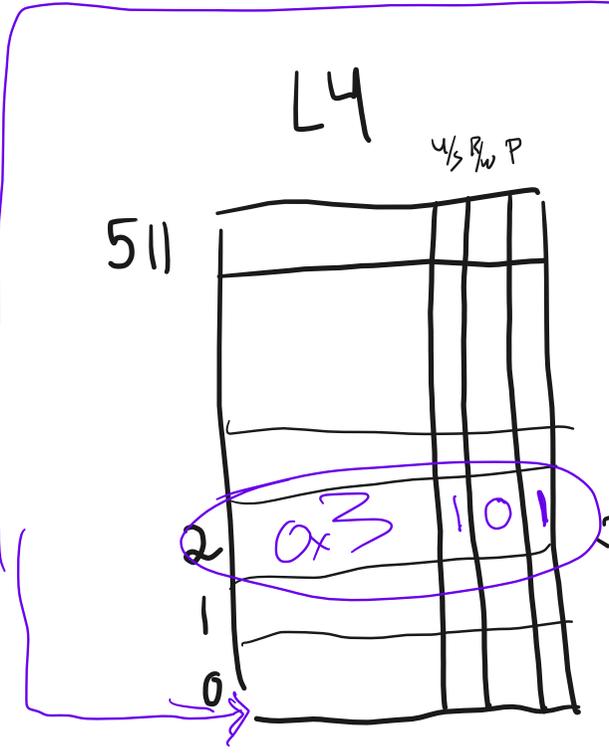
4. TLB

$$\langle VPN, (PPN, perms) \rangle$$

$\langle 0x202, (0x3, \; 0...101) \rangle$

H/w managed: x86, ARM

S/w managed: MIPS

TLB miss $\xrightarrow{?}$ pg fault

pg fault $\xrightarrow{?}$ TLB miss

Core i7 Page Table Translation

Virtual address

9	9	9	9	12
VPN 1	VPN 2	VPN 3	VPN 4	VPO

L1 PT
Page global directory

L2 PT
Page upper directory

L3 PT
Page middle directory

L4 PT
Page table

CR3
Physical address of L1 PT

40

L1 PTE

40

L2 PTE

40

L3 PTE

40

L4 PTE

512 GB region per entry

1 GB region per entry

2 MB region per entry

4 KB region per entry

Physical address of page

Offset into physical and virtual page

/12

40

Physical address

40	12
PPN	PPO

Review of Symbols

- **Basic Parameters**
  - **N = 2^n** : Number of addresses in virtual address space
  - **M = 2^m** : Number of addresses in physical address space
  - **P = 2^p** : Page size (bytes)
- **Components of the virtual address (VA)**
  - **TLBI**: TLB index
  - **TLBT**: TLB tag
  - **VPO**: Virtual page offset
  - **VPN**: Virtual page number
- **Components of the physical address (PA)**
  - **PPO**: Physical page offset (same as VPO)
  - **PPN:** Physical page number
  - **CO**: Byte offset within cache line
  - **CI:** Cache index
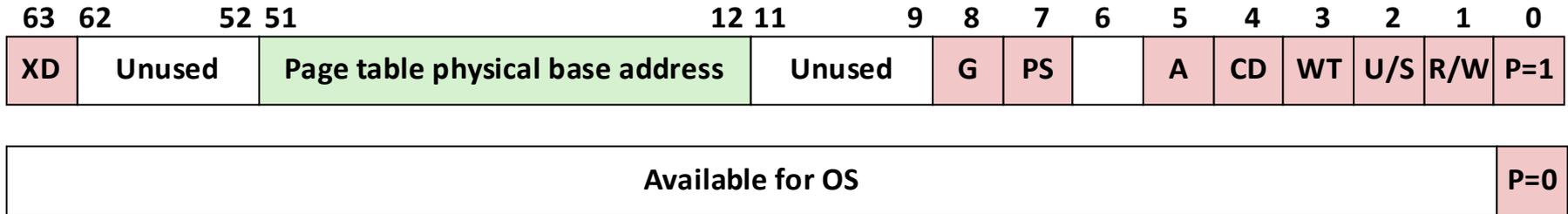 - **CT**: Cache tag

Core i7 Level 1-3 Page Table Entries

63	62 52	51 12	11 9	8	7	6	5	4	3	2	1	0
XD	Unused	Page table physical base address	Unused	G	PS		A	CD	WT	U/S	R/W	P=1

Available for OS												P=0

Each entry references a 4K child page table. Significant fields:

P: Child page table present in physical memory (1) or not (0).

R/W: Read-only or read-write access access permission for all reachable pages.

U/S: user or supervisor (kernel) mode access permission for all reachable pages.

WT: Write-through or write-back cache policy for the child page table.

A: Reference bit (set by MMU on reads and writes, cleared by software).

PS: Page size: if bit set, we have 2 MB or 1 GB pages (bit can be set in Level 2 and 3 PTEs only).

Page table physical base address: 40 most significant bits of physical page table address (forces page tables to be 4KB aligned)

XD: Disable or enable instruction fetches from all pages reachable from this PTE.

Core i7 Level 4 Page Table Entries

63	62 52	51 12	11 9	8	7	6	5	4	3	2	1	0
XD	Unused	Page physical base address	Unused	G		D	A	CD	WT	U/S	R/W	P=1

												P=0
Available for OS (for example, if page location on disk)												P=0

Each entry references a 4K child page. Significant fields:

P: Virtual page is present in memory (1) or not (0)

R/W: Read-only or read-write access permission for this page

U/S: User or supervisor mode access

**WT:** Write-through or write-back cache policy for this page

A: Reference bit (set by MMU on reads and writes, cleared by software)

D: Dirty bit (set by MMU on writes, cleared by software)
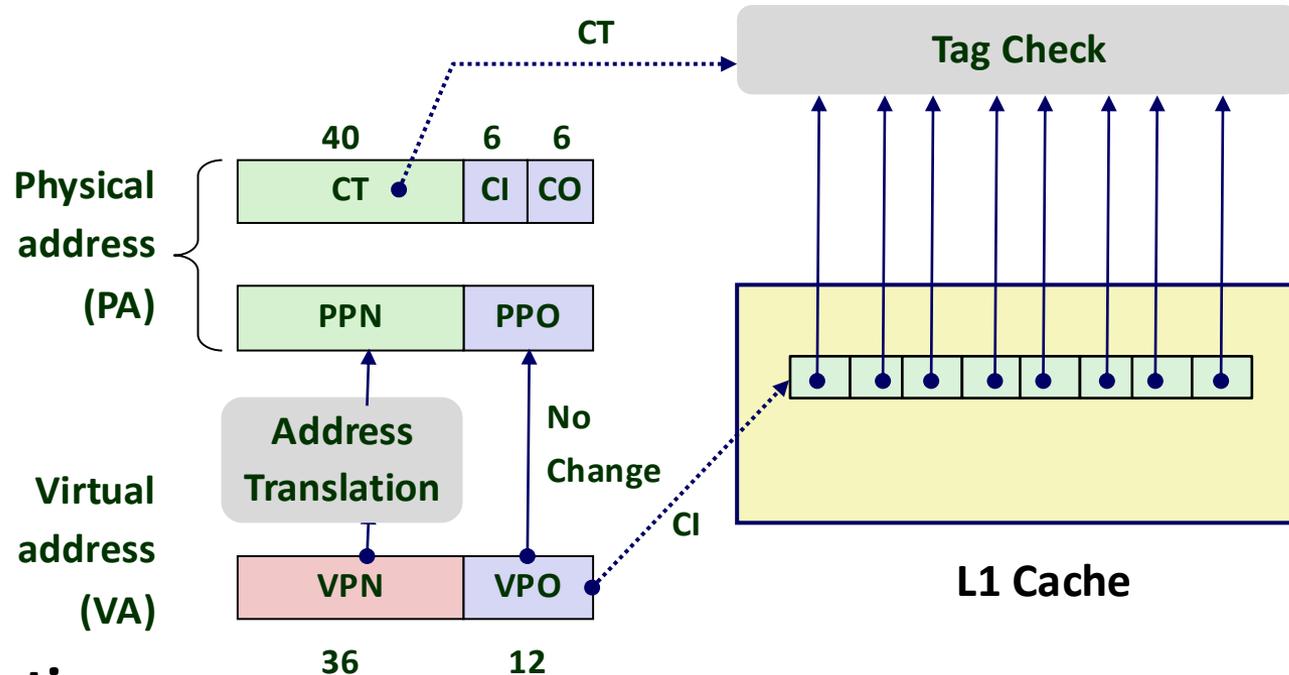
Page physical base address: 40 most significant bits of physical page address (forces pages to be 4KB aligned)

XD: Disable or enable instruction fetches from this page.

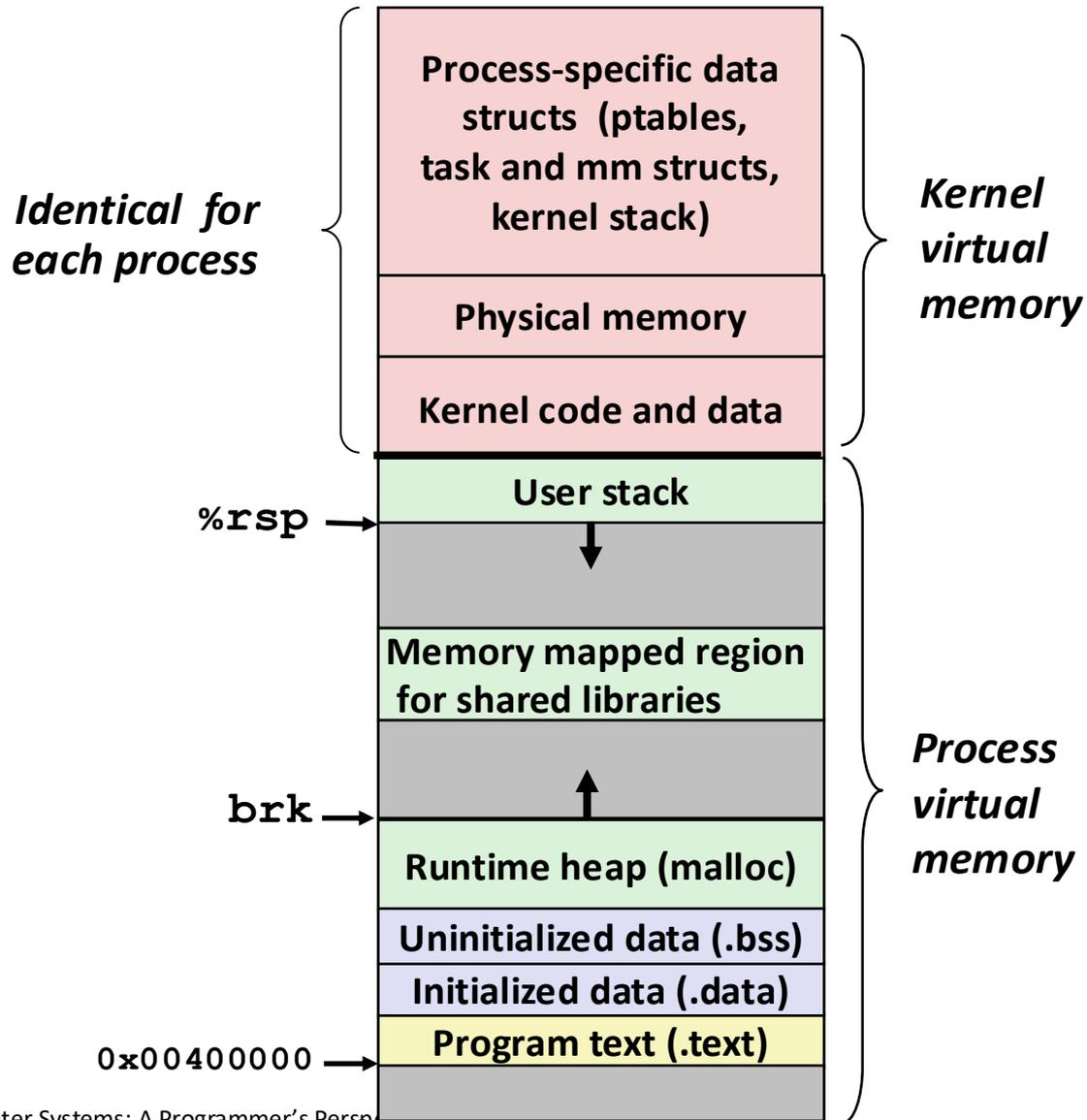# End-to-end Core i7 Address Translation
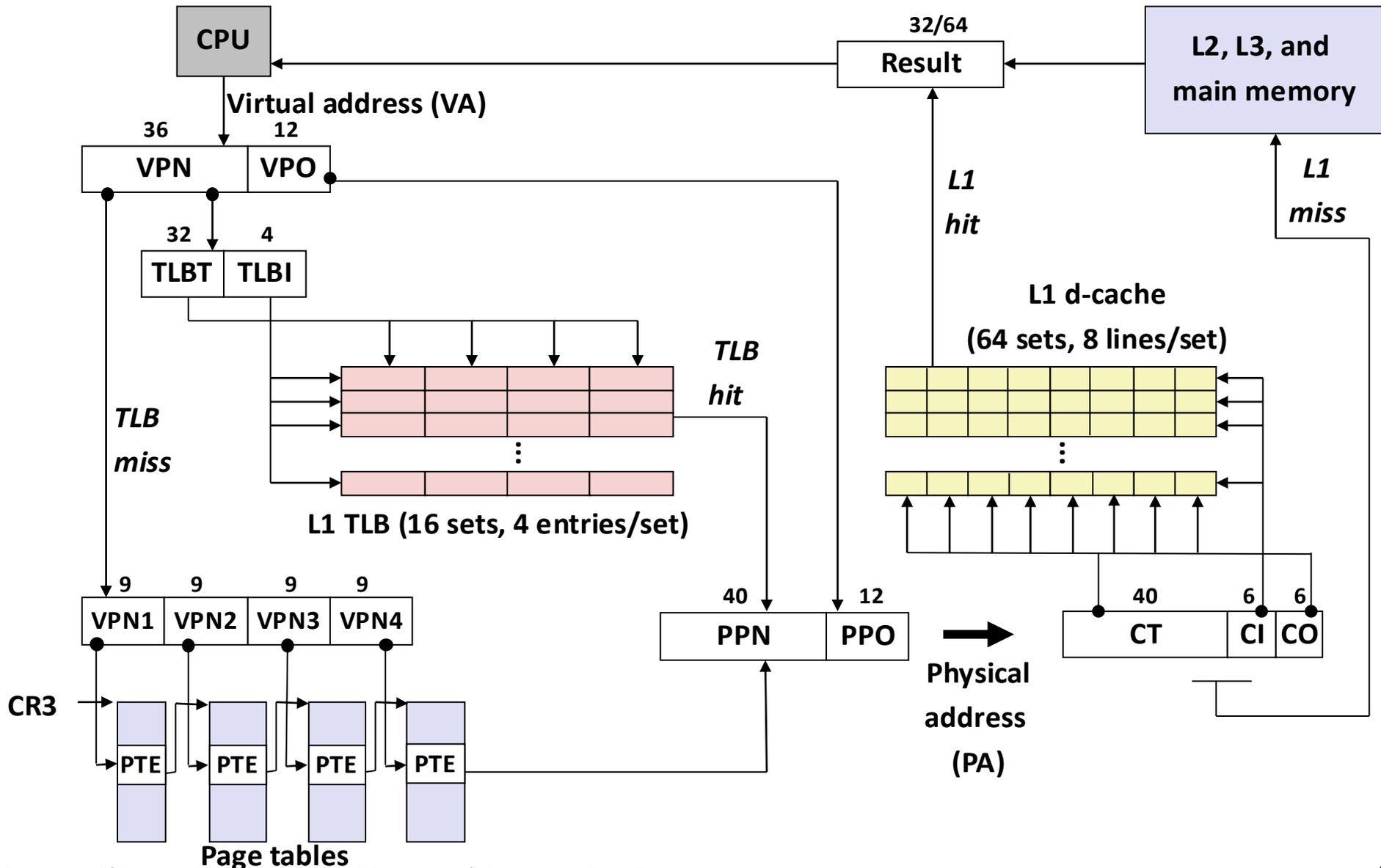
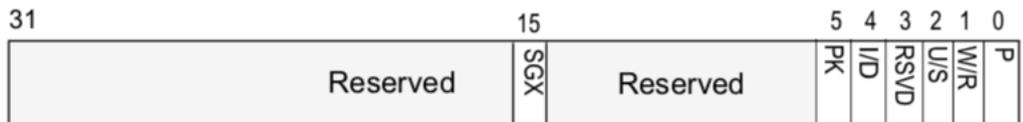# Cute Trick for Speeding Up L1 Access

Observation

- Bits that determine CI identical in virtual and physical address
- Can index into cache while address translation taking place
- Cache carefully sized to make this possible: 64 sets, 64-byte cache blocks
- Means 6 bits for cache index, 6 for *cache* offset
- That's 12 bits; matches *VPO, PPO* → One reason pages are 2^{12} bits = 4 KB

Virtual Address Space of a Linux Process

Identical for each process

Kernel virtual memory

Process-specific data structs (ptables, task and mm structs, kernel stack)

Physical memory

Kernel code and data

User stack

`%rsp` →

Memory mapped region for shared libraries

Process virtual memory

`brk` →

Runtime heap (malloc)

Uninitialized data (.bss)

Initialized data (.data)

`0x00400000` →

Program text (.text)

0

31			15		5	4	3	2	1	0
	Reserved	SGX	Reserved		PK	I/D	RSVD	U/S	W/R	P

P   0 The fault was caused by a non-present page.
        1 The fault was caused by a page-level protection violation.

W/R   0 The access causing the fault was a read.
        1 The access causing the fault was a write.

U/S   0 A supervisor-mode access caused the fault.
        1 A user-mode access caused the fault.

RSVD   0 The fault was not caused by reserved bit violation.
        1 The fault was caused by a reserved bit set to 1 in some
           paging-structure entry.

I/D   0 The fault was not caused by an instruction fetch.
        1 The fault was caused by an instruction fetch.

PK   0 The fault was not caused by protection keys.
        1 There was a protection-key violation.

SGX   0 The fault is not related to SGX.
        1 The fault resulted from violation of SGX-specific access-control
           requirements.

Figure 4-12. Page-Fault Error Code