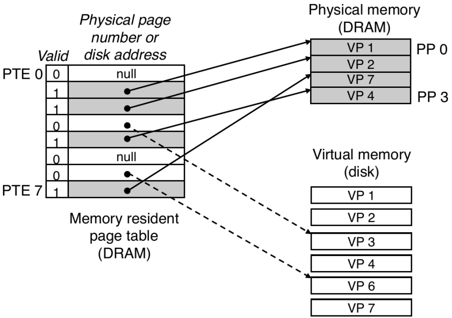
Virtual Memory Systems

CPSC 235 - Computer Organization

References

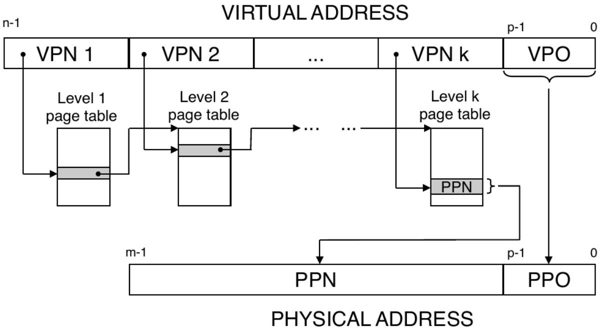
- Slides adapted from CMU

Review: Virtual Memory and Physical Memory

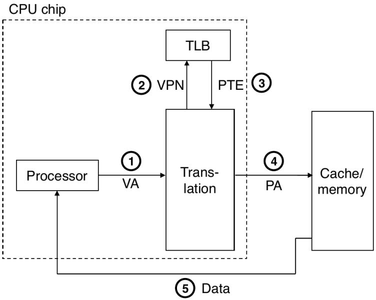


- A *page table* contains page table entries (PTEs) that map virtual pages to physical pages

Translating with a k-level Page Table

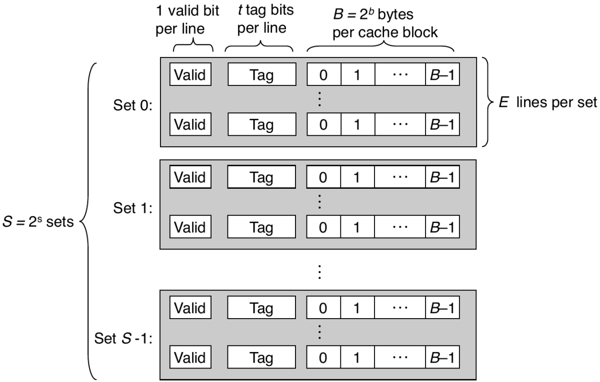


Translation Lookaside Buffer (TLB)



- A TLB hit eliminates the *k* memory accesses required to do a page table lookup

Recall: Set Associative Cache



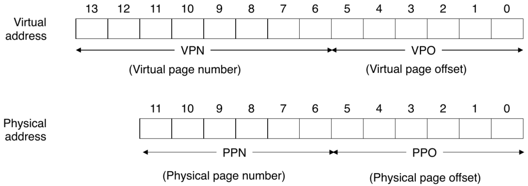
Cache size: $C = B \times E \times S$ data bytes

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: translation lookaside buffer index
  - TLBT: translation lookaside buffer 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

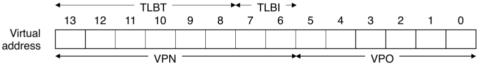
Simple Memory System Example

- Addressing
  - 14-bit virtual addresses
  - 12-bit physical addresses
  - Page size = 64 bytes



Simple Memory System TLB

- 16 entries
- 4-way associative



Set	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	03	–	0	09	0D	1	00	–	0	07	02	1
1	03	2D	1	02	–	0	04	–	0	0A	–	0
2	02	–	0	08	–	0	06	–	0	03	–	0
3	07	–	0	03	0D	1	0A	34	1	02	–	0

Simple Memory System Page Table

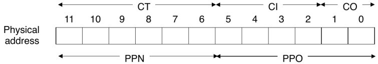
- Only showing the first 16 entries (out of 256)

VPN	PPN	Valid
00	28	1
01	–	0
02	33	1
03	02	1
04	–	0
05	16	1
06	–	0
07	–	0

VPN	PPN	Valid
08	13	1
09	17	1
0A	09	1
0B	–	0
0C	–	0
0D	2D	1
0E	11	1
0F	0D	1

Simple Memory System Cache

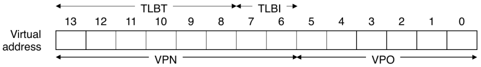
- 16 lines, 4-byte cache line size
- Physically addressed
- Direct mapped



Idx	Tag	Valid	Blk 0	Blk 1	Blk 2	Blk 3
0	19	1	99	11	23	11
1	15	0	–	–	–	–
2	1B	1	00	02	04	08
3	36	0	–	–	–	–
4	32	1	43	6D	8F	09
5	0D	1	36	72	F0	1D
6	31	0	–	–	–	–
7	16	1	11	C2	DF	03
8	24	1	3A	00	51	89
9	2D	0	–	–	–	–
A	2D	1	93	15	DA	3B
B	0B	0	–	–	–	–
C	12	0	–	–	–	–
D	16	1	04	96	34	15
E	13	1	83	77	1B	D3
F	14	0	–	–	–	–

Address Translation Example

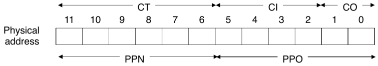
- Virtual Address: `0x3d4` = `00001111 010100`
  - VPN: `0x0F`, TLBI: `0x03`, TLBT: `0x03`, PPN: `0x0D`
  - Hit, no fault



Set	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	03	–	0	09	0D	1	00	–	0	07	02	1
1	03	2D	1	02	–	0	04	–	0	0A	–	0
2	02	–	0	08	–	0	06	–	0	03	–	0
3	07	–	0	03	0D	1	0A	34	1	02	–	0

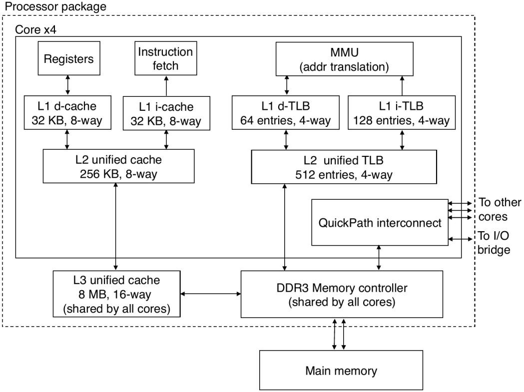
Address Translation Example

- Physical Address:
  - PPN: `001101`, PPO: `010100`
  - CO: `0`, CI: `0x5`, CT: `0x0D`, Hit: yes, Byte: `0x36`

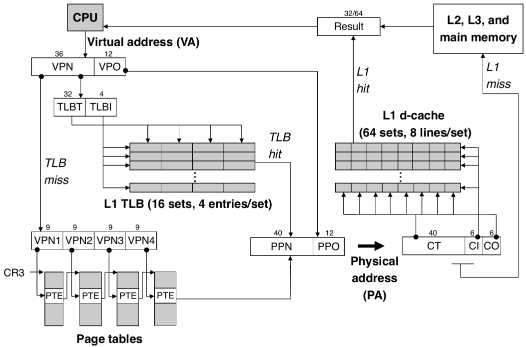


Idx	Tag	Valid	Blk 0	Blk 1	Blk 2	Blk 3
0	19	1	99	11	23	11
1	15	0	–	–	–	–
2	1B	1	00	02	04	08
3	36	0	–	–	–	–
4	32	1	43	6D	8F	09
5	0D	1	36	72	F0	1D
6	31	0	–	–	–	–
7	16	1	11	C2	DF	03
8	24	1	3A	00	51	89
9	2D	0	–	–	–	–
A	2D	1	93	15	DA	3B
B	0B	0	–	–	–	–
C	12	0	–	–	–	–
D	16	1	04	96	34	15
E	13	1	83	77	1B	D3
F	14	0	–	–	–	–

Intel Core i7 Memory System



End-to-end Core i7 Address Translation



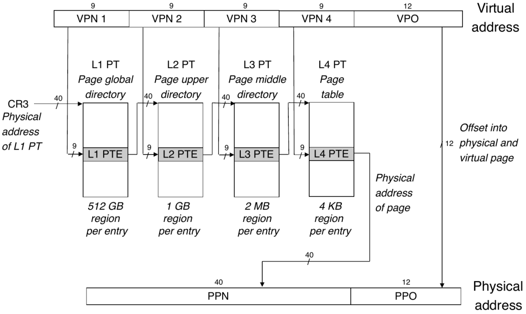
Core i7 Level 1-3 Page Table Entries

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

Available for OS (page table location on disk)	**P=0**

- Each entry references a 4K child page table:
    - P: child page table present in physical memory
    - R/W: read-only or read-write 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 child page table
    - A: reference bit (set by MMU on reads and writes, cleared by software)
    - PS: Page size either 4KB or 4MB (defined for level 1 PTEs only)
    - Page table physical base address: 40 most significant bits or physical page table address (forces page tables to be 4KB aligned)
    - XD: disable or enable instruction fetches from all pages reachable from this PTE

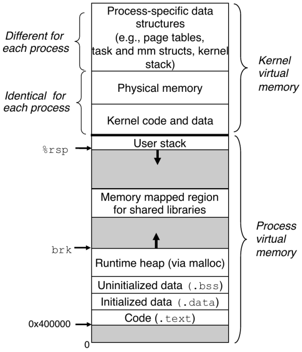
Core i7 Page Table Translation



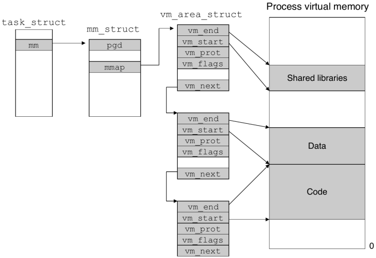
Trick for Speeding Up L1 Access

- Observation
  - Bits that determine the CI are identical in virtual and physical address
  - Can index into cache while address translation is taking place
  - Generally there is a hit in the TLB, so PPN bits (CT bits) are available quickly
  - "Virtually indexed, physically tagged"
  - Cache carefully sized to make this possible

Virtual Address Space of a Linux Process



Linux Organizes VM as Collection of "Areas"



- pgd: page global directory address; points to L1 page table
- vm_prot: read/write permissions for this area
- vm_flags: pages shared with other processes or private to this process

Linux Page Fault Handling

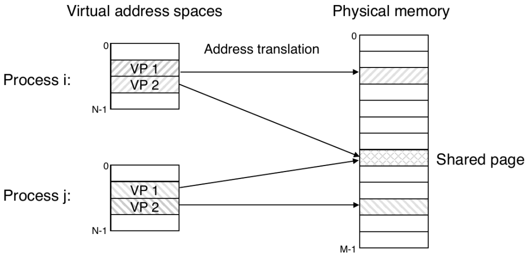
- Read from a non-existing page: segmentation fault
- Read from data area: normal page fault
- Write to text area: violating permission by writing to a read-only page; Linux reports a segmentation fault

Memory Mapping

- VM areas initialized by associating them with disk objects
  - Called memory mapping
- Area can be backed by (that is, get its initial values from):
  - Regular file on disk (for example, an executable object file)
    - Initial page bytes come from a section of a file
  - Anonymous file (that is, nothing)
    - First fault will allocate a physical page full of zeros
    - Once the page is written to (dirtied), it is like any other page
- Dirty pages are copied back and forth between memory and a special swap file

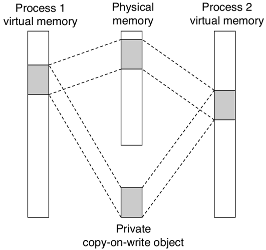
Review: Memory Management and Protection

- Code and data can be isolated or shared among processes



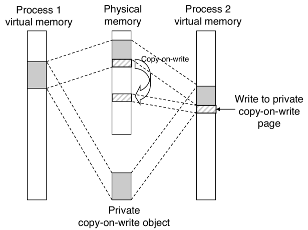
Sharing Revisited: Shared Objects

- Process 1 maps the shared object (on disk)
- Process 2 maps the same shared object
- Note that the virtual addresses can be different, but the difference must be a multiple of the page size
- Two processes mapping a private copy-on-write (COW) object
- Area flagged as private copy-on-write
- PTEs in private areas are flagged as read-only



Sharing Revisited: Private Copy-on-Write (COW) Objects

- Instruction writing to private page triggers protection fault
- Handler creates new R/W page
- Instruction restarts upon handler return
- Copying deferred as long as possible



Finding Shareable Pages

- Kernel Same-Page Merging
  - OS scans through all of physical memory looking for duplicate pages
  - When found, merge into a single copy marked as copy-on-write
  - Implemented in Linux kernel in 2009
  - Limited to pages marked as likely candidates
  - Especially useful when processor running many virtual machines

Summary

- VM requires hardware support
  - Exception handling mechanism
  - TLB
  - Various control registers
- VM requires OS support
  - Managing page tables
  - Implementing page replacement policies
  - Managing file system
- VM enables many capabilities
  - Loading programs from memory
  - Providing memory protection