Paging: Smaller Tables
CS 537: Introduction to Operating Systems

Louis Oliphant

University of Wisconsin - Madison

Fall 2023

Administrivia

- Project 1 Graded
  - Grade in Canvas, Feedback in handin directory
  - If you have questions or concerns about the grade you received, please email **avmatthews@cs.wisc.edu** with you CSL login and a description of the problem.
- Project 3 out – Due Oct 10 @ 11:59pm
  - `SIGTTIN` and `SIGTTOU` Job Control Signals
    A process cannot read from the user's terminal while it is running as a background job. When any process in a background job tries to read from the terminal, all of the processes in the job are sent a SIGTTIN signal. The default action for this signal is to stop the process. SIGTTOU is similar to SIGTTIN, but is generated when a process in a background job attempts to write to the terminal
  - `signal()` vs. `sigaction()`

Administrivia (cont)

- Email me about great TA experiences
- Louis Oliphant Office Hours this Thursday end at 4:30pm
- **Midterm 1**: Oct 12 (Next Thursday)
    - In-class
    - McBurney CS1325, 5:45pm-7:23pm
    - Multiple-Choice / Scantron
    - **Bring #2 Pencil**
    - **Bring UW Student ID**
- **Study Group** – Discuss lecture concepts, practice problems, and prepare for quizzes.
    - Times: Tuesdays 7-8pm
    - Location: Comp Sci room B207. To find this room in the CS building: go to the Dayton Street entrance, follow signs for room 1240, take the stairs or elevator down.
    - Questions? email Andy Kuemmel, CSLC Director, kuemmel@wisc.edu

Review: TLB

- The **TLB** is a **fully associative** cache of a few (e.g. 32) PTEs to **reduce the number of memory accesses** during address translation.
- Cache's take advantage of **spatial** and **temporal** locality
- TLB misses can be handled by hardware (the MMU) or software (the OS)
- Flushing or ASID portion in TLB handles issues involving context switches
- Common replacement policies are Random and **LRU**

Quiz 7: TLBs

https://tinyurl.com/cs537-fa23-q7

Agenda

Ways of Addressing Large Page Tables:

- Larger Pages
- Hybrid Approach: Combining Segmentation and Page Tables
- Multi-level Page Tables
- Inverted Page Tables

Why Are Page Tables Large?

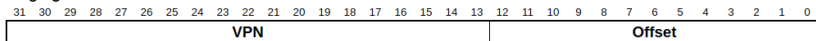| PFN	valid	prot
10	1	r-x
-	0	-
23	1	rw-
-	0	-
-	0	-
-	0	-
-	0	-

how to avoid storing these?

...many more invalid...

-	0	-
-	0	-
-	0	-
-	0	-
28	1	rw-
4	1	rw-

Larger Pages

4 bytes per entry, 4KB Pages

$2^{20} = 1$ M entries $= 4$ MB

Paging

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							VPN													Offset											

Larger Pages

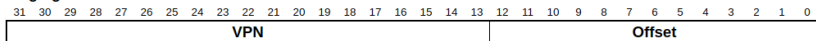| 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							VPN												Offset												
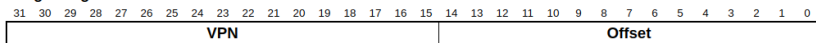
4 bytes entry, 16 KB Pages

$2^{18} = 256$ K entries $= 1$ MB

Larger Pages $=$ Waste *within* pages (internal fragmentation)

Hybrid Approach

Paging

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VPN																			Offset												

Hybrid Approach: Paging and Segments

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Seg		VPN																	Offset												
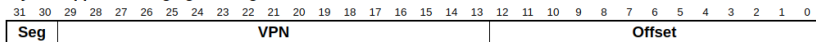
- A page table for each logical segment
- Base/Bounds for each table
  - Base – Location of table
  - Bounds – Holds maximum valid page in segment

```
SN = (VirtualAddress & SEG_MASK) >> SN_SHIFT
VPN = (VirtualAddress & VPN_MASK) >> VPN_SHIFT
if VPN > Bounds[SN]
    Raise Segmentation Violation
else
    AddressOfPTE = Base[SN] + (VPN * sizeof(PTE))
```

Hybrid Approach Drawbacks

- Large, sparsely used heap, still have a lot of page table waste

- Page tables now can be arbitrary in size (e.g. memory becomes hard to manage, external fragmentation)

Multi-Level Page Tables

- Chop up the page table into page-sized units.
- If an entire page of page-table entries are invalid, don't allocate that page of the page table at all.
- Many modern systems use this approach (e.g. x86)
- to track if a page of the page table is valid, use a **page directory**
- The page directory can be used to find a page of the page table OR indicate no valid pages

Multi-Level Page Table Example
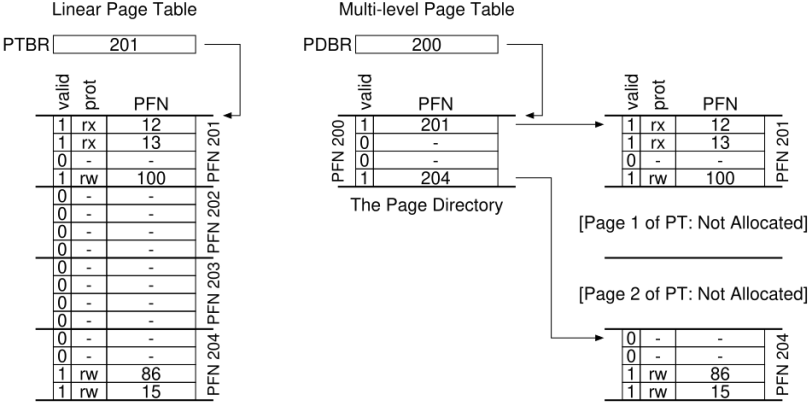
Multi-Level Page Table Advantages & Disadvantages

- Only allocates page table space in proportion to the amount of address space a process uses
- Generally compact and supports sparse address spaces
- If carefully constructed, each portion of the page table fits neatly within a page, making memory management easy
- Adds a level of indirection (**time-space tradeoff**)
    - Now a TLB miss requires 2 memory accesses to get the PTE:
        - First, lookup in the page directory where that portion of the page table is located
        - Second, lookup in that page of the page table the PTE
- Adds complexity

Detailed Multi-Level Example

- 16 KB Address Space
  (14 bits for a VA)
- 64 byte pages
  (offset is 6 bits)
- 256 # Pages
  (VPN is 8 bits)
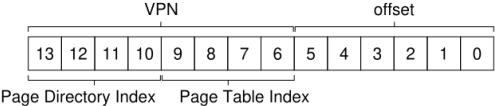
VPN	Contents
0000 0000	code
0000 0001	code
0000 0010	(free)
0000 0011	(free)
0000 0100	heap
0000 0101	heap
0000 0110	(free)
0000 0111	(free)
....	... all free ...
1111 1100	(free)
1111 1101	(free)
1111 1110	stack
1111 1111	stack

VPN								offset					
13	12	11	10	9	8	7	6	5	4	3	2	1	0

Page Directory Index   Page Table Index

A Page Directory and Pieces of Page Table

Page Directory		Page of PT (@PFN:100)			Page of PT (@PFN:101)		
PFN	valid?	PFN	valid	prot	PFN	valid	prot
100	1	10	1	r-x	—	0	—
—	0	23	1	r-x	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	80	1	rw-	—	0	—
—	0	59	1	rw-	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	55	1	rw-
101	1	—	0	—	45	1	rw-

Virtual Address: 0x3F80 (11 1111 1000 0000)
PD Index: **1111** PT Index: **1110** Offset **00 0000**
Lookup 1111 in Page Directory: Get PFN is 101
Lookup 1110 in Page 101 of PT: Get PFN is 55 (110111)
Physical Address: 55 (shifted) + offset = 00 1101 1100 0000 =
0x0DC0

More Than Two Levels

- Multi-Level page tables can be any number of levels

- With each extra level adds another memory access on a TLB
  miss

Inverted Page Tables (Another Approach)

Rather than having many page tables (one per process), have a single table with an entry for each *physical page* of the system.

To convert a virtual address scan the single page table until you find the physical page entry that holds the VPN.

Linear scan too expensive, so use a hash table.

Summary

- Multiple approaches to shrink page tables
  - Larger pages
  - Hybrid approach
  - Multi-Level page tables
  - Inverted page tables

- Select the approach based upon constraints

- Next Time: Swapping (When memory is not enough)