Review: Address Translation

- Stack
- Heap
- Data
- Code

Virtual Address → MMU → Physical Address

- Invalid → Exception
Review: Paging
Review: Virtual Pages

- Stack
- Heap
- Data
- Code

**Virtual Memory Address (vaddr)**

- **Page Table**
  - Frame[page#]
  - Offset
  - Frame[page#] offset

- **Page Fault**
  - NULL page or access not allowed

- **SegFault**

- **MMU**

- **Data**

<table>
<thead>
<tr>
<th>v</th>
<th>Frame</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>47</td>
<td>R,W</td>
</tr>
<tr>
<td>0</td>
<td>NULL</td>
<td>R,W</td>
</tr>
<tr>
<td>0</td>
<td>13</td>
<td>R,W</td>
</tr>
<tr>
<td>1</td>
<td>42</td>
<td>R,X</td>
</tr>
</tbody>
</table>

- paddr = Frame[page#] offset
Review: Problems with Paging

- **Memory Consumption**: page table is really big
  - Example: consider 64-bit address space, 4KB ($2^{12}$) page size, assume each page table entry is 8 bytes.
  - Larger pages increase internal fragmentation

- **Performance**: every data/instruction access requires *two* memory accesses:
  - One for the page table
  - One for the data/instruction
Traditional Paging

- page table is stored in physical memory
- implemented as array of page table entries
- Page Table Base Register (PTBR) stores physical address of beginning of page table
- Page table entries are accessed by using the page number as the index into the page table

<table>
<thead>
<tr>
<th>Page Table Entries</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 NULL R,W</td>
<td></td>
</tr>
<tr>
<td>0 NULL R,W</td>
<td></td>
</tr>
<tr>
<td>0 NULL R,W</td>
<td></td>
</tr>
<tr>
<td>1 59 R,X</td>
<td></td>
</tr>
<tr>
<td>1 47 R,W</td>
<td></td>
</tr>
<tr>
<td>0 NULL R,W</td>
<td></td>
</tr>
<tr>
<td>0 13 R,W</td>
<td></td>
</tr>
<tr>
<td>1 42 R,X</td>
<td></td>
</tr>
</tbody>
</table>

PTBR
Two-level Page Tables

- Page table is stored in virtual memory pages
- Page directory is stored in physical memory (page table for the page table)
- Implemented as array of page directory entries
- Page Table Base Register (PTBR) stores physical address of beginning of page directory
Two-level Page Tables

+ only store in-use page table entries in physical memory
+ easier to allocate page table
- more memory accesses
Exercise 1: Two-level Page Tables

- Assume you are working on an architecture with a 32-bit virtual address space in which idx1 is 4 bits, idx2 is 12 bits, and offset is 16 bits.

- How big is a page in this architecture? $2^{16}$ bytes = 64 KB

- How big is a page table entry in this architecture? 16 bytes
Exercise 2: Two-level Page Tables

Assume you are still working on that architecture.

Compute the physical address corresponding to each of the virtual address (or answer "invalid"):

- a) 0x00000000 \( \rightarrow \) 0x00470000
- b) 0x20022002 \( \rightarrow \) invalid
- c) 0x10015555 \( \rightarrow \) 0xCAFE5555

<table>
<thead>
<tr>
<th>Virtual Address</th>
<th>Frame</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>0x0047</td>
<td>R,W</td>
</tr>
<tr>
<td>0x01000000</td>
<td>0x0013</td>
<td>R,W</td>
</tr>
<tr>
<td>0x02000000</td>
<td>0x0042</td>
<td>R,X</td>
</tr>
<tr>
<td>0x03000000</td>
<td>0x004A</td>
<td>R</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Frame</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0047</td>
<td>R,W</td>
</tr>
<tr>
<td>0x0013</td>
<td>R,W</td>
</tr>
<tr>
<td>0x0042</td>
<td>R,X</td>
</tr>
<tr>
<td>0x004A</td>
<td>R</td>
</tr>
</tbody>
</table>

4 bit idx1 | 12 bit idx2 | 16 bit offset
Multi-level Page Tables

• Problem: How big does the page directory get? 128 MB
  • Assume you have a 48-bit address space
  • Assume you have 4KiB pages
  • Assume you have 8 byte page table entries/page directory entries

• Goal: Page Table Directory should fit in one frame

**Multi-level page tables:** add additional level(s) to tree
Review: Problems with Paging

- **Memory Consumption:** page table is really big
  - Example: consider 64-bit address space, 4KB ($2^{12}$) page size, assume each page table entry is 8 bytes.
  - Larger pages increase internal fragmentation

- **Performance:** every data/instruction access requires two memory accesses:
  - One for the page table
  - One for the data/instruction
Translation-Lookaside Buffer (TLB)

- General idea: if address translation is slow, cache some of the answers
- Translation-lookaside buffer is an address translation cache that is built into the MMU
Exercise 3: TLBs

• Consider the following piece of code that multiplies two matrices:

```c
int a[1024][1024], b[1024][1024], c[1024][1024]

void multiply(){
    for(int i = 0; i < 1024; i++)
        for(int j = 0; j < 1024; j++)
            for(int k = 0; k < 1024; k++)
                c[i][j] += a[i][k] * b[k][j];
}
```

• Assume that the system has 4KiB pages, the TLB stores 8 entries, and the TLB uses a LRU eviction policy. Recall that ints are 4 bytes.

• Compute the number of TLB misses (assuming the TLB is initially empty)
Example: The Linux x86 Address Space

- Use "only" 48-bit addresses (top 16 bits not used)
- 4KiB pages by default
  - supports larger "superpages"
- Four-level page table
- Physical memory stores memory pages, memory-mapped files, cached file pages
- Updates are periodically written to disk by background processes
- Page eviction algorithm uses variant of LRU called 2Q
  - approximates LRU with clock
  - maintains two lists (active/inactive)
- Stack is marked non-executable
- Virtual address of stack/heap start are randomized each time process is initialized
Example: Core i7 Address Translation

Virtual address (VA) → TLBT → TLBI → TLB

L1 TLB (16 sets, 4 entries/set) → VPN1, VPN2, VPN3, VPN4 → PPN, PPO → CR3 → Page tables

32/64 → Result → L2, L3, and main memory

L1 hit → L1 d-cache (64 sets, 8 lines/set)

L1 miss → L1 d-cache (64 sets, 8 lines/set)
Exercise 4: Feedback

1. Rate how well you think this recorded lecture worked
   1. Better than an in-person class
   2. About as well as an in-person class
   3. Less well than an in-person class, but you still learned something
   4. Total waste of time, you didn't learn anything

2. Do you have any comments or suggestions for future classes?