Memory Management

Disclaimer: some slides are adopted from book authors’ slides with permission
Recap: Virtual Address

Process A

Process B

Process C

MMU

Physical Memory
Recap: MMU

• Hardware unit that translates *virtual address* to *physical address*
Recap: A Simple MMU

- BaseAddr: base register
- Paddr = Vaddr + BaseAddr

- Advantages
  - Fast

- Disadvantages
  - No protection
  - Wasteful
Recap: Modern MMU

- Paging hardware
Recap: Virtual Address Translation

Virtual address
0x12345678

Page #
0x12345
Offset
0x678

Physical address
0xabcde678

frame #: 0xabcde
frame #
offset
Recap: Paging

• Advantages
  – No external fragmentation

• Two main Issues
  – Translation speed can be slow
    • TLB
  – Table size is big
Multi-level Paging

• Two-level paging
Two Level Address Translation

Virtual address

1\textsuperscript{st} level \quad 2\textsuperscript{nd} level \quad \text{offset}

Base ptr

1\textsuperscript{st} level Page table

2\textsuperscript{nd} level Page

2\textsuperscript{nd} level Page table

Physical address

Frame # \quad Offset
## Example

<table>
<thead>
<tr>
<th>1st level</th>
<th>2nd level</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits</td>
<td>8 bits</td>
<td>8 bits</td>
</tr>
</tbody>
</table>

Virtual address format (24 bits)

- **Vaddr:** 0x0703FE
  - 1st level idx: **07**
  - 2nd level idx: **03**
  - Offset: **FE**

- **Vaddr:** 0x072370
  - 1st level idx: **__**
  - 2nd level idx: **__**
  - Offset: **__**

- **Vaddr:** 0x082370
  - 1st level idx: **__**
  - 2nd level idx: **__**
  - Offset: **__**
Multi-level Paging

• Can save table space
• How, why?

• Don’t need to create all mappings in the outer page table
Concepts to Learn

• Demand paging
Virtual Memory (VM)

• Abstraction
  – 4GB linear address space for each process

• Reality
  – 1GB of actual physical memory shared with 20 other processes

• Does each process use the
  (1) entire virtual memory
  (2) all the time?
Demand Paging

• Idea: instead of keeping the entire memory pages in memory all the time, keep only part of them on a on-demand basis
Page Table Entry (PTE)

- PTE format (architecture specific)
  - Valid bit (V): whether the page is in memory
  - Modify bit (M): whether the page is modified
  - Reference bit (R): whether the page is accessed
  - Protection bits (P): readable, writable, executable
Partial Memory Mapping

- Not all pages are in memory (i.e., valid=1)
Page Fault

• When a virtual address cannot be translated to a physical address, MMU generates a trap to the OS

• Page fault handling procedure
  – Step 1: allocate a free page frame
  – Step 2: bring the stored page on disk (if necessary)
  – Step 3: update the PTE (mapping and valid bit)
  – Step 4: restart the instruction
Page Fault Handling
Demand Paging
Starting Up a Process

Stack

Heap

Data

Code

Unmapped pages
Starting Up a Process

Access next instruction
Starting Up a Process

Stack

Heap

Data

Code

Page fault
Starting Up a Process

OS
1) allocates free page frame
2) loads the missed page from the disk (exec file)
3) updates the page table entry
Over time, more pages are mapped as needed