Memory Management

Disclaimer: some slides are adopted from book authors’ slides with permission
Roadmap

• CPU management
  – Process, thread, synchronization, scheduling

• Memory management
  – Virtual memory

• Disk management

• Other topics
Memory Management

• Goals
  – Easy to use abstraction
    • Same virtual memory space for all processes
  – Isolation among processes
    • Don’t corrupt each other
  – Efficient use of capacity limited physical memory
    • Don’t waste memory
Concepts to Learn

• Virtual memory
• MMU and TLB
• Demand paging
Virtual Memory (VM)

- **Abstraction**
  - 4GB linear address space for each process

- **Reality**
  - 1GB of actual physical memory shared with 20 other processes

- **How?**
Virtual Memory

• Hardware support
  – MMU (memory management unit)
  – TLB (translation lookaside buffer)

• OS support
  – Manage MMU (sometimes TLB)
  – Determine address mapping

• Alternatives
  – No VM: many real-time OS (RTOS) don’t have VM
Virtual Address

- Process A
- Process B
- Process C

MMU

Physical Memory
MMU

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

- BaseAddr: base register
- Paddr = Vaddr + BaseAddr

- Advantages
  - Fast

- Disadvantages
  - No protection
  - Wasteful
A Better MMU

• Base + Limit approach
  – If Vaddr > limit, then trap to report error
  – Else Paddr = Vaddr + BaseAddr
A Better MMU

• Base + Limit approach
  – If Vaddr > limit, then trap to report error
  – Else Paddr = Vaddr + BaseAddr

• Advantages
  – Support protection
  – Support variable size partitions

• Disadvantages
  – Fragmentation
Fragmentation

- External fragmentation
  - total available memory space exists to satisfy a request, but it is not contiguous
Modern MMU

• Paging approach
  – Divide physical memory into fixed-sized blocks called frames (e.g., 4KB each)
  – Divide logical memory into blocks of the same size called pages (page size = frame size)
  – Pages are mapped onto frames via a table ➔ page table
Modern MMU

• Paging hardware
Modern MMU

• Memory view
Virtual Address Translation

Virtual address

0x12345678

Page #

Offset

Physical address

0xabcde678

frame #: 0xabcde

frame #

offset
Advantages of Paging

• No external fragmentation
  – Efficient use of memory
  – Internal fragmentation (waste within a page) still exists
Issues of Paging

• Translation speed
  – Each load/store instruction requires a translation
  – Table is stored in memory
  – Memory is slow to access
    • ~100 CPU cycles to access DRAM
Translation Lookaside Buffer (TLB)

• **Cache** frequent address translations
  – So that CPU don’t need to access the page table all the time
  – Much faster
Issues of Paging

• Page size
  – Small: minimize space waste, requires a large table
  – Big: can waste lots of space, the table size is small
  – Typical size: 4KB
  – How many pages are needed for 4GB (32bit)?
    • 4GB/4KB = 1M pages
  – What is the required page table size?
    • assume 1 page table entry (PTE) is 4bytes
    • 1M * 4bytes = 4MB
  – Btw, this is for each process. What if you have 100 processes? Or what if you have a 64bit address?
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

1st level | 2nd level | offset

Base ptr

1st level Page table

2nd level Page

Physical address

Frame # Offset
### Example

Virtual address format (24 bits)

<table>
<thead>
<tr>
<th>Vaddr</th>
<th>1st level idx</th>
<th>2nd level idx</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0703FE</td>
<td>07</td>
<td>03</td>
<td>FE</td>
</tr>
<tr>
<td>0x072370</td>
<td>__</td>
<td>__</td>
<td>__</td>
</tr>
<tr>
<td>0x082370</td>
<td>__</td>
<td>__</td>
<td>__</td>
</tr>
</tbody>
</table>
Multi-level Paging

• Can save table space
• How, why?

• Don’t need to create all mappings in the outer page table
Recap: Paging

• Advantages
  – No external fragmentation

• Two main Issues
  – Translation speed can be slow
    • TLB
  – Table size is big
    • Multi-level page table
Recap: Two Level Paging

Virtual address

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

Base ptr

1\textsuperscript{st} level Page table

2\textsuperscript{nd} level Page

Physical address

Frame # \quad \textit{Offset}
Quiz

• What is the page table size for a process that only uses 8MB memory?
  – Common: 32bit address space, 4KB page size
  – Case 1) 1-level page table
    • Assume each page table entry is 4 bytes
    • Answer: $2^{20} \times 4$ byte = 4MB
  – Case 2) two-level page table
    • Assume first 10 bits are used as the index of the first-level page table, next 10 bits are used as the index of the second-level page table. In both levels, single page table entry size is 4 bytes
    • Answer: $2^{10} \times 4 + 2 \times (2^{10} \times 4) = 4$KB + $8$KB = 12KB
Quiz

• What is the page table size for a process that only uses 16MB memory?
  – Common: 32bit address space, 4KB page size
  – Case 1) 1-level page table
    • Assume each page table entry is 4 bytes
    • Answer: \(2^{20} \times 4 \text{ byte} = 4 \text{MB}\)
  – Case 2) two-level page table
    • Assume first 10 bits are used as the index of the first-level page table, next 10 bits are used as the index of the second-level page table. In both levels, single page table entry size is 4 bytes
    • Answer: \(2^{10} \times 4 + 4 \times (2^{10} \times 4) = 4\text{KB} + 16\text{KB} = 20\text{KB}\)