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 address translation
• Paging and TLB
• Page table management
• Swap
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 #**
0x12345

**Offset**
0x678

**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