Administrivia

• Labs start this week

• Lab homepage
Agenda

• Computer architecture and OS
  – CPU, memory, disk
  – Architecture trends and their impact to OS
  – Architectural support for OS
Recap

• Batch
  – One at a time. CPU is idle while waiting for I/O
  – Low throughput

• Multiprogramming
  – If a job needs to way for I/O, switch to a new job
  – Maximize throughput

• Timesharing
  – Switch to next job after some time
  – Minimize response time
Recap

• Essential components of a computer
  – CPU
  – Memory
  – I/O

• Memory hierarchy

• Caching
Recap: Memory Hierarchy

Fast, Expensive

Slow, Inexpensive
Recap: Caching

- A very important principle applied in all layers of hardware, OS, and software
  - Put frequently accessed data in a small amount of faster memory
  - Fast, most of the time (hit)
  - Copy from slower memory to the cache (miss)
  - Low cost, good performance
Architectural Support for OS

- Interrupts and exceptions
- Protected modes (kernel/user modes)
- Memory protection and virtual memory
- Synchronization instructions
Interrupt

• What is an interrupt?
  – A signal to the processor telling “do something now!”

• Hardware interrupts
  – Devices (timer, disk, keyboard, ...) to CPU

• Software interrupts (exceptions)
  – Divide by zero, special instructions (e.g., int 0x80)
Interrupt Handling

- save CPU states (registers)
- execute the associated interrupt service routine (ISR)
- restore the CPU states
- return to the interrupted program
Timesharing

• Multiple tasks share the CPU at the same time
  – But there is only one CPU (assume single-core)
  – Want to schedule different task at a regular interval of 10 ms, for example.

• Timer and OS scheduler tick
  – The OS programs a timer to generate an interrupt at every 10 ms.
Dual (User/Kernel) Mode

• Some operations must be restricted to the OS
  – accessing registers in the disk controller
  – updating memory management unit states
  – ...

• User/Kernel mode
  – Hardware support to distinguish app/kernel
  – Privileged instructions are only for kernel mode
  – Applications can enter into kernel mode only via pre-defined system calls
User/Kernel Mode Transition

- System calls
  - Programs ask OS services (privileged) via system calls
  - Software interrupt. “int <num>” in Intel x86
Memory Protection

• How to protect memory among apps/kernel?
  – Applications shouldn’t be allowed to access kernel’s memory
  – An app shouldn’t be able to access another app’s memory
Virtual Memory

• How to overcome memory space limitation?
  – Multiple apps must share limited memory space
  – But they want to use memory as if each has dedicated and big memory space
  – E.g.,) 1GB physical memory and 10 programs, each of which wants to have a linear 4GB address space
Virtual Memory

![Diagram showing virtual memory and physical memory, with processes A, B, and C each having different sections for stack, heap, data, and text. The MMU (memory management unit) is connected to the processes and the physical memory.]
MMU

- Hardware unit that translates *virtual address* to *physical address*
  - Defines the boundaries of kernel/apps
  - Enable efficient use of physical memory
Synchronization

• Synchronization problem with threads

Deposit(account, amount) {
    account->balance += amount;
}

Thread 1: Deposit(acc, 10)

LOAD R1, account->balance
ADD R1, amount
STORE R1, account->balance

Thread 2: Deposit(acc, 10)

LOAD R1, account->balance
ADD R1, amount
STORE R1, account->balance
Synchronization Instructions

• Hardware support for synchronization
  – TestAndSet, CompareAndSwap instructions
  – Atomic load and store
  – Used to implement lock primitives
  – New TSX instruction → hardware transaction

• Another methods to implement locks in single-core systems
  – Disabling interrupts
Summary

• OS needs to understand architecture
  – Hardware (CPU, memory, disk) trends and their implications in OS designs

• Architecture needs to support OS
  – Interrupts and timer
  – User/kernel mode and privileged instructions
  – MMU
  – Synchronization instructions
## OS Abstractions

<table>
<thead>
<tr>
<th>Reality</th>
<th>Abstraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>A single computer</td>
<td>Multiple computers</td>
</tr>
<tr>
<td>Limited RAM capacity</td>
<td>Infinite capacity</td>
</tr>
<tr>
<td>Mechanical disk</td>
<td>File system</td>
</tr>
<tr>
<td>Insecure and unreliable</td>
<td>Reliable and secure</td>
</tr>
<tr>
<td>networks</td>
<td></td>
</tr>
</tbody>
</table>
Acknowledgements

• Some slides are from
  – Authors of the textbook: Abraham Silberschatz, Greg Gagne, and Peter Baer Galvin
  – Dr. Prasad Kulkarni