CISC 3595 Operating Systems
Introduction

Tuesday / Friday 11:00-12:45am
Outline

• What is operating system?
• Necessary background: computer hardware organization
  ▸ Basic Elements
  ▸ Processor: Instruction Execution
  ▸ Memory Hierarchy
    ▸ Cache Memory
  ▸ I/O Communication Techniques
Four Components of a Computer System

People, machines, other computers

CPU, memory, I/O devices
What is an Operating System?

• A program that acts as an intermediary between a user of a computer and the computer hardware

• Operating system goals:
  – Execute user programs and make solving user problems easier
  – Provide abstractions to application programs: make computer system convenient to use
  – Provide orderly, controlled, efficient allocation of resources
History of Operating Systems

- First generation (1945–55): vacuum tubes, bugs, no operating systems, programming in machine code or plugboard
- Second generation (1955–65): transistors, batch systems, programming in assembly, FORTRAN
- Third generation (1965–1980): ICs, multiprogramming OS (to increase utilization of CPU), timesharing
- Fourth generation (1980–present) personal computers, (CP/M, DOS, Windows), GUI
- The fifth generation (1990–present) mobile computers
Glen Beck and Betty Snyder program ENIAC. (U.S. Army photo)

ENIAC (Electronic Numerical Integrator And Computer) was the first electronic general-purpose computer. It was Turing-complete, digital, and can be reprogrammed to solve "a large class of numerical problems"
IBM/360 (2 million, 5 billion to built (budget 2 million)),
jewel of the office, Batch system
1970s, Unix System, PDF-7 (mini-computer)
DOS for Apple II: Contract
Current Generation OS

- Mobile devices (Smartphone, wearable computers…)
  - Limited power
  - Different I/O: touch screen, cellular communication, sensors, …

- Android, iOS, Windows
  - Android's kernel is based on one of the Linux kernel's long-term support (LTS) branches

- iOS (based upon Mac OS)
  - iPhone, iPod, iPad, …

- …
Outline

- What is operating system?
- Necessary background: computer hardware organization
  - Basic Elements
  - Processor: Instruction Execution
  - Memory Hierarchy
    - Cache Memory
  - I/O Communication Techniques
Computer System Organization

- One or more CPUs, device controllers connected through common bus providing access to shared memory
Basic Hardware Elements

- Processor or Central Processing Unit (CPU)
  - Controls operation, performs data processing

- Memory (main/primary memory)
  - Volatile, i.e., data is typically lost when power is removed
  - Used to store data and instructions

- I/O Modules: disk controller, USB controller, ...
  - Moves data between computer and external device such as storage (e.g. hard drive), communication equipment, terminals

- System Bus
  - Wires or backplane connecting CPUs, I/O modules and main memory
Processor (CPU)

- CPU: the physical heart of entire computer system
  - Execute instructions
- Instruction set: the set of machine instructions that a processor can execute, main categories:
  - Processor-memory: move data between memory and processor
  - processor-I/O: move data between peripheral device and CPU
  - Data processing: arithmetic or logic operation on data
  - Control: alter execution sequence (jump, if and loop structure)
- Some instructions are privilege instructions (can only be executed by os kernel).
Stored Program Computer

- A program: a sequence of instructions stored in memory (main memory or disk)
  - when running a program, it is loaded (from disk) into main memory
- Basic Instruction Execution cycle:

![Diagram of Basic Instruction Execution Cycle]

Stored program computer (due to John von Neumann (1903-1957))

**Figure 1.2 Basic Instruction Cycle**
Main memory: a large array of words or bytes. Each word has its own address.

CPU registers: in-CPU memory, faster and smaller than main memory

Figure 1.1 Computer Components: Top-Level View

<table>
<thead>
<tr>
<th>PC</th>
<th>IR</th>
<th>MAR</th>
<th>MBR</th>
<th>I/O AR</th>
<th>I/O BR</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC = Program counter</td>
<td>IR = Instruction register</td>
<td>MAR = Memory address register</td>
<td>MBR = Memory buffer register</td>
<td>I/O AR = Input/output address register</td>
<td>I/O BR = Input/output buffer register</td>
</tr>
</tbody>
</table>
CPU Registers: data and address registers

- **Data/Address registers**: store data or memory address
  - Store frequently used variable in register:
    ```c
    for (register int i = 0; i < bufsize; i++) * p++ = assigned_val;
    ```
  - Usually compiler decides what variables to store in register
CPU: Control and Status Registers

- **Control CPU operation, and stores CPU status**
  - Program counter (PC): holds address of the instruction to be fetched next
  - Instruction register (IR): store the instruction to be executed
  - Stack pointer: store address of the top of the current stack
  - Program status word (PSW): contain bits set by processor hardware as a result of operations
    - Condition code bits (set by comparison instructions)
    - CPU priority
    - Mode (user, or kernel mode): OS kernel runs in kernel supervisor mode, other programs run in user mode
    - Other …
Machine code and Assembly Language

- CPU only understands binary $\Rightarrow$ machine instructions are **coded in binary strings**
  - e.g., 10110000 01100001 (Hexadecimal: B0 61)
- **Assembly language**: mnemonic language (helping to remember)
  - MOV AL, #61h
  - Move the value 61h (or 97 decimal; the h-suffix means hexadecimal; the pound sign means move the immediate value, not location) into the processor register named "AL".
Considering a Hypothetical Machine

(a) Instruction format

(b) Integer format

Program counter (PC) = Address of instruction
Instruction register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory

(d) Partial list of opcodes

Figure 1.3 Characteristics of a Hypothetical Machine
Figure 1.4 Example of Program Execution
(contents of memory and registers in hexadecimal)
Most systems use a single general-purpose processor

Multiprocessors systems growing in use and importance
- parallel systems, tightly-coupled systems

Advantages include:
1. Increased throughput
2. Economy of scale
3. Increased reliability – graceful degradation or fault tolerance
Outline

• What is operating system?
• Necessary background: computer hardware organization
  ▶ Basic Elements
  ▶ Processor: Instruction Execution
  ▶ Memory Hierarchy
    ▶ Cache Memory
  ▶ I/O Communication Techniques
Memory Hierarchy

- CPU register
- CPU L1 cache
- Main memory, RAM – volatile storage media that CPU can access directly
  - Random access: take same amount of time to access any part of the memory
- Hard disk: Large nonvolatile storage capacity
  - disk controller: determines logical interaction between the device and the computer
- Magnetic tape:
  - sequential access: recall that you have to rewind VCR?
Performance measure: maximum transfer rate; spindle Rotation Speed: decides the reading/writing rate; seek time: avg time to locate the data
Why so many different memory/storage

- Major constraints in memory
  - Amount, Speed, Expense

- Generally:
  - Faster access time, greater cost per bit

---

ns: nanosecond (10⁻⁹ seconds), 1 billionth second
Memory Hierarchy

- Goal: to obtain good trade-off between cost and performance
- Going down hierarchy
  - Decreasing cost per bit
  - Increasing capacity
  - Increasing access time
  - Decreasing frequency of access
Memory Hierarchy and Caching

- **Caching**: information *in use* is copied from slower and larger storage to faster and smaller storage (cache) temporarily
  - Faster storage (cache) checked first to determine if information is there
    - If it is, information used directly from the cache (fast)
    - If not, data copied to cache and used there
- **Exploit temporal locality and spatial locality**
  - Data which is required *soon* is often *close in memory* to the current data
- **Caching is performed at many levels in a computer**
  - Hardware: L1 cache
  - Operating system: disk cache in memory
  - Software/application level caching
Primary Cache (L1 cache)

- Processor must access memory at least once per instruction cycle
  - Processor speed faster than memory access speed

- Solution: L1 Cache, storage on CPU for temporary storage of instructions

  - Cache contains copy of a portion of main memory
  - CPU first checks cache
    - If not found, block of memory read into cache
  - Because of locality of reference, likely future memory references are in that block
Outline

• What is operating system?
• Necessary background: computer hardware organization
  ▸ Basic Elements
  ▸ Processor: Instruction Execution
  ▸ Memory Hierarchy
    ▸ Cache Memory
  ▸ I/O Communication Techniques
Input/Output **Device Controller**

- **Device Controller:**
  - A chip or sets of chip that control I/O device to perform I/O
  - Special purpose processor: execute concurrently with CPU
    - Basic functionalities: moving data between device and controller’s local buffer
  - Has local buffer (registers): CPU communicates with device controller through these, e.g., command, status, input/output data
- **Device driver:** software that talks to device controller
What happens when you read from disk file?

Roughly speaking:

1. Your program (C++ library) make a system call to OS to **read from a file**
2. OS calls device driver for the disk controller
3. As the read call is blocking call, your program is put on hold;
4. Disk driver issues command to disk controller’s register: read section 11,206 from disk 2
5. Disk controller: which cylinder? Move disk arm, Wait until proper sector rotates under head, and then read disk content to its buffer
6. When data is ready (stored in disk controller’s buffer), they are transferred from disk controller’s register to main memory
7. Your program is ready to run again…
I/O operation modes

- Coordination between CPU and Device Controller
  - **Programmed I/O**: I will wait here until you are done with this, “busy waiting”
    - Problem: most I/O devices are slower than CPU => inefficient for CPU to wait for I/O to complete
  - **Interrupt-driven I/O**: let me know as soon as you are done...
  - **Direct memory access (DMA)**: Work on the reports and when you are done, put them in my mail box, and let me know...

- Needs a mechanism to change the sequential instruction execution of CPU
Interrupts

- A mechanism to interrupt normal execution sequence of CPU

**Figure 1.6 Transfer of Control via Interrupts**

**Interrupt Handler:** Predefined routines to be called when a certain interrupt occurs.
Simple Interrupt Processing

Figure 1.10  Simple Interrupt Processing
Different Types of Interrupts

- **Program generated**: arithmetic overflow, division by zero, illegal machine instruction, reference outside of user’s allowed memory space, also called *exception, trap or software interrupt*
- **Timer**: generated by a timer with CPU
- **I/O**: generated by I/O controller to signal completion of operation or error condition
  - Delivered to CPU via bus, together with device number
- **Hardware failure**: triggered by power failure, memory parity error…

An operating system is **interrupt driven**.
Interrupt-driven I/O

- When CPU encounters an I/O related instruction
  - it executes that instruction by issuing a command to appropriate I/O module, and ask I/O module to give an interrupt when done
  - CPU continues to do its work without waiting for I/O completion (maybe running other programs if it’s a blocking call, such as cin, scanf…)
- After I/O device controller finish I/O operation, it generate an interrupt to inform CPU
- CPU, in interrupt handler, read data from device controller or write next block of data to device controller, and wake up processes waiting for the interrupt...
Direct Memory Access (DMA) I/O

- DMA: allow device to access memory for reading and/or writing independently of CPU.
  - Used in high-speed I/O devices: disk controllers, graphics cards, network cards and sound cards

- DMA module (DMA device controller)
  - Transfers blocks of data from buffer storage directly to main memory
  - Only one interrupt is generated per block

- CPU only involved at beginning and ending transfer.
  - Less CPU intervention => much more efficient in terms of processing times
Outline

• What is operating system?
• Necessary background: computer hardware organization
  ▶ Basic Elements
  ▶ Processor: Instruction Execution
  ▶ Memory Hierarchy
    ▶ Cache Memory
  ▶ I/O Communication Techniques
• Operating System services
Operating System

A program that acts as an intermediary between users/programs and computer hardware

- Execute user programs and make solving user problems easier
- Provide orderly, controlled, efficient allocation of resources
- Provide abstractions to application programs: make computer system convenient to use
  - System calls
System calls

- Abstraction/Interface that OS provides to user programs
  - For process management
  - For file management
  - Directory and file system management …
Live Demo on storm

- Keep in mind: what services are we using?
- System information
- Shell: a command line interpreter vs graphical desktop
- Commands related to file systems
  - Mount, ls,
- User/Group management
  - Security and protection
- Multi-programming, time-sharing aspects
  - Command: ps, top, kill,
- Interprocess communication
- Networking:
  - netstat, ifconfig, …
Multiprogramming

- Multiple programs are kept in memory, i.e., process (a program in execution)
  - One process is selected and run via CPU scheduling
  - When it has to wait (e.g., for I/O), OS switches to another process
    - After saving status of pending process (PC, stack, …)
  - After an interrupt handler completes, control may not return to the program that was executing at the time of the interrupt

- Timesharing: allocate CPU to different processes in small quota, and round-robin manner
All above commands are not part of kernel!

- You can write your own shell, ps, ... using system calls

- Demo of strace command
- One assignment: write your own shell
Summary

• What is operating system?
• Necessary background: computer hardware organization
  ▸ Basic Elements
  ▸ Processor: Instruction Execution
  ▸ Memory Hierarchy
    ▹ Cache Memory
  ▸ I/O Communication Techniques
  ▸ Operating System services
  ▸ Next class:
    ▹ OS abstraction: process, file, and related system calls