• Last class:
  – Course administration
  – OS definition, some history

• Today:
  – Background on Computer Architecture
Canonical System Hardware

- **CPU**: Processor to perform computations
- **Memory**: Programs and data
- **I/O Devices**: Disk, monitor, printer, …
- **System Bus**: Communication channel between the above
von Neuman Architecture (1946)

Instructions and data are stored in the same memory for which there is a single link (the von Neumann bottleneck) to the CPU which decodes and executes instructions.

The CPU can have multiple functional units.

The memory access can be enhanced by use of caches made from faster memory to allow greater bandwidth and lower latency.

J. Presper Eckert Jr. and John Mauchly were the first to develop the von Neuman architecture. John von Neumann wrote "First Draft of a Report to the EDVAC" describing the ideas of a stored memory computer. The complicated story is described in the wonder history of computers "Engines of the Mind" by Joel Shurkin.
CPU

• CPU
  – Semiconductor device, digital logic (combinational and sequential)
  – Can be viewed as a combination of many circuits

• Clock
  – Synchronizes constituent circuits

• Registers
  – CPU’s scratchpads; very fast; loads/stores
  – Most CPUs designed so that a register can store a memory address
    • n-bit architecture

• Cache
  – Fast memory close to CPU
  – Faster than main memory, more expensive
  – Not seen by the OS
CPU Instruction Execution

- Arithmetic Logic Unit (ALU)
- Program counter
  - Instruction address
- Instruction from the control unit (F)
- CPU data registers
  - Input A and B and Output R
Memory/RAM

- Semiconductor device
  - DIMMs mounted on PCBs
  - Random access: RAM
  - DRAM: Volatile, need to refresh
    - Capacitors lose contents within few tens of msecs

- CPU accesses RAM to fill registers

- OS sees and manages memory
  - Programs/data need to be brought to RAM

- Memory controller: Chip that implements the logic for
  - Reading/Writing to RAM (Mux/Demux)
  - Refreshing DRAM contents
Memory Access

• Instructions
  – Program counter is used to fetch into control unit
  – Fetched into instruction register

• Data
  – Load/store instructions
  – Move data between memory locations
I/O Devices

• Large variety, varying speeds
  – Disk, tape, monitor, mouse, keyboard, NIC
  – Serial vs parallel

• Each has a controller
  – Hides low-level details from OS
  – Manages data flow between device and CPU/memory
Hard Disk

- Secondary storage
- Mechanically operated
  - Sequential access
- Cheap => Abundant
- Very slow
  - Orders of magnitude
Interconnects

- A bus is an interconnect for flow of data and information
  - Wires, protocol
  - Data arbitration
- System Bus
- PCI Bus
  - Connects CPU-memory subsystem to
    - Fast devices
    - Expansion bus that connects slow devices
- SCSI, IDE, USB, …
  - Will return to these later
ATX & Mini ATX

The ATX design is one of the most commonly used motherboard form factors among Pentium and Pentium II processor-based PCs. The ATX offers one major innovation—a parallel, serial, and PS/2 mouse and keyboard ports that are soldered directly to the motherboard. This feature increases reliability and serviceability over the AT and baby AT designs' system of cable connections running from ports located to the computer's case. Another design innovation is the ATX's power supply use of a single 20-pin connector instead of the two six-pin connectors in older boards. This prevents users from plugging the two cables into the wrong holes and damaging the motherboard's circuitry. An ATX motherboard is typically 12 inches wide and 9.6 inches deep. ATX boards follow the same basic layout, but are sized at about 11.2 inches by 8.2 inches.
Architectural Support
Expected by Modern OSes
Services & Hardware Support

- **Protection**: Kernel/User mode, Protected Instructions, Base & Limit Registers
- **Scheduling**: Timer
- **System Calls**: Trap Instructions
- **Efficient I/O**: Interrupts, Memory-mapping
- **Synchronization**: Atomic Instructions
- **Virtual Memory**: Translation Lookaside Buffer (TLB)
Kernel/User Mode

• A modern CPU has at least two modes
  – Indicated by status bit in protected CPU register
  – OS runs in privileged mode
    • Also called kernel or supervisor mode
  – Applications run in normal mode
  – Pentium processor has 4 modes

• Events that need the OS to run switch the processor to priv. mode
  – E.g., division by zero

• OS can switch the processor to user mode

• OS definition: Software than runs in priv. mode
Protected Instructions

- Privileged instructions & registers:
  - Direct access to I/O
  - Modify page table pointers, TLB
  - Enable & disable interrupts
  - Halt the machine, etc.
Base and Limit Registers

- Hardware support to protect memory regions
  - Loaded by OS before starting program
- CPU checks each reference
  - Instruction & data addresses
- Ensures reference in range
Interrupts

• Polling = “are we there yet?” “no!” (repeat…)
  – Inefficient use of resources
  – Annoys the CPU
• Interrupt = silence, then: “we’re there”
  – I/O device has own processor
  – When finished, device sends interrupt on bus
  – CPU “handles” interrupt
Interrupts

- Asynchronous *signal* indicating need for attention
  - Replaces *polling* for events
- Represent
  - Normal events to be noticed and acted upon
    - Device notification
    - Software system call
  - Abnormal conditions to be corrected
  - Abnormal conditions that cannot be corrected
Hardware Interrupts

• Signal from a device
  – Implemented by a controller (e.g., memory)

• Examples
  – Timer
  – Keyboard, mouse
  – End of DMA transfer

• Response to processor request
• Unsolicited response
Timer

• OS needs timers for
  – Time of day
  – CPU scheduling
• Interrupt vector for timer

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x2ff00000</td>
<td>Keyboard</td>
</tr>
<tr>
<td>0xfc000b0</td>
<td>Mouse</td>
</tr>
<tr>
<td>0x2df00000</td>
<td>Timer</td>
</tr>
<tr>
<td>0x2ff6810</td>
<td>Disk 1</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Software Interrupts

- Software interrupts (Traps)
  - Special interrupt instructions
    - `int 0x80` -- System call
  - Exceptions
    - Some can be fixed (e.g., page fault)
    - Some cannot (e.g., divide by zero)
- All invoke OS, just like a hardware interrupt
Interrupt Handling

• Each interrupts has a corresponding
  – Interrupt Handler

• When an interrupt request (IRQ) is received
  – If interrupt mask allows interrupt
  – Save state of current processing
    • At time of interrupt something else may be running
    • State: Registers (stack ptr), program counter, etc.
  – Execute handler
  – Return to current processing
Interrupt Handling

Interrupt

System service call

Hardware exceptions → Exception dispatcher → Exception handlers
Software exceptions

(Exception frame)

Virtual address exceptions

Trap handlers

Interrupt service routines

Virtual memory manager's pager
Multiple Interrupts

- Clock interrupt
- Disk interrupt
- Makes a system call
- Executing in user mode

<table>
<thead>
<tr>
<th>Kernel context Layer 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute clock</td>
</tr>
<tr>
<td>Save Register context of disk</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Kernel Context layer 2.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute Disk handler</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Save register context of sys call</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Kernel Context layer 1.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute system call</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Save user level Registers</th>
</tr>
</thead>
</table>
Device Access

- **Port I/O**
  - Uses special I/O instructions
  - Port number, device address
    - Separate from process address space

- **Memory-mapped I/O**
  - Uses memory instructions (load/store)
    - To access memory-mapped device registers
  - Does not require special instructions
    - But consumes some memory for I/O
Device Access

Disadvantage:
A portion of the memory space is used for I/O devices.

Advantage:
IORC and IOWC not required.
Any data transfer instruction.

Disadvantage:
Hardware using M/I/O and WR needed to develop signals IORC and IOWC.
Requires IN, OUT, INS and OUTS.
Direct Memory Access

• Direct access to I/O controller *through memory*

• Reserve area of memory for communication with device (“DMA”)
  – Video RAM:
    • CPU writes frame buffer
    • Video card displays it

• Fast and convenient
Synchronization

• How can OS synchronize concurrent processes?
  – E.g., multiple threads, processes & interrupts, DMA

• CPU must provide mechanism for *atomicity*
  – Series of instructions that execute as one or not at all
Synchronization: How-To

• One approach:
  – Disable interrupts
  – Perform action
  – Enable interrupts

• Advantages:
  – Requires no hardware support
  – Conceptually simple

• Disadvantages:
  – Could cause starvation
Synchronization: How-To, II

• Modern approach: \textit{atomic instructions}
  
  – Small set of instructions that cannot be interrupted
  
  – Examples:
    
    • Test-and-set (“TST”)
      if word contains given value, set to new value
    
    • Compare-and-swap (“CAS”)
      if word equals value, swap old value with new
    
    • Intel: LOCK prefix (XCHG, ADD, DEC, etc.)

• Used to implement \textit{locks}
Process Address Space

• All locations addressable by the process
  – Virtual address space
• Can restrict use of addresses (RW)
• Restrictions enforced by OS
Virtual Memory

- Provide the illusion of infinite memory
- OS loads *pages* from disk as needed
  - Page: Fixed sized block of data
- Many benefits
  - Allows the execution of programs that may not fit entirely in memory (think MS Office)
- OS needs to maintain mapping between physical and virtual memory
  - Page tables stored in memory
Translation Lookaside Buffer (TLB)

- Initial virtual memory systems used to do translation in software
  - Meaning the OS did it
  - An additional memory access for each memory access!
    - S.l.o.w.!!!
- Modern CPUs contain hardware to do this: the TLB
  - Fast cache
  - Modern workloads are TLB-miss dominated
  - Good things often come in small sizes
    - We have see other instances of this
Translation Lookaside Buffer (TLB)
Modern architectures provide lots of features to help the OS do its job:

- Protection mechanisms (modes)
- Interrupts
- Device I/O
- Synchronization
- Virtual Memory (TLB)

Otherwise impossible or impractically slow in software.

Which of these are essential? Which are useful but not essential?
• Next time: Operating system structures