Overview of Architecture

- Basic Computer Hardware
- How CPU handles Input Output?
- How Interrupts are supported?

The ability of OS are dictated in part by the architecture of the machine.

- Architectural support can greatly simplify, or greatly complicate the OS

OS and Hardware

<table>
<thead>
<tr>
<th>OS Service</th>
<th>Hardware Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O</td>
<td>Interrupts &amp; DMA</td>
</tr>
<tr>
<td>Protection</td>
<td>Protected Instruction</td>
</tr>
<tr>
<td></td>
<td>Kernel and User modes</td>
</tr>
<tr>
<td></td>
<td>Interrupt &amp; Trap Vectors</td>
</tr>
<tr>
<td></td>
<td>Base &amp; Limit Registers</td>
</tr>
<tr>
<td>Scheduling</td>
<td>Timers</td>
</tr>
<tr>
<td>Synchronization</td>
<td>Atomic Instructions</td>
</tr>
</tbody>
</table>
The Computer System

The computer system

System Calls
The computer system

System Calls

Users and User Programs

Operating System

Hardware

CPU
RAM
I/O Devices

BUS

Hardware revision

Architecture.
Booting the system.
Instruction execution.
I/O and interrupts.
Memory
Disks
Hardware Protection Mechanisms
Hardware Concepts.

Most computers are based on the Von Neumann architecture.

Four main elements of hardware
- Central Processor Unit (CPU)
- Main Memory
- I/O Modules or Devices (printers, disk drives, keyboard)
- System Interconnections (the bus)
CPU Registers.

**Program Counter, PC**
Address of NEXT instruction.

**Instruction Register, IR**
Contains the current instruction

**Stack Pointer, SP.**
Points to the top of the process stack.

**Processor Status Word**
Contains information about the state of the CPU, Interrupts, condition statements.

**Others.**
Memory management etc.

**User Registers.**
Used for various programming purposes.

---

Instruction Execution.

- CPU, memory and device controllers connects via common bus
- When computer starts, it loads OS
- OS runs users programs as requested
- Constantly checks for Interrupts:
  - From hardware
  - From software

Start

 Fetch Instruction.

 Handle Interrupts.

 Execute Instruction.

Interrupts Enabled

Interrupts Disabled.
**Booting a System.**

H/W looks for a boot device. (such as c:\ drive)

Boot block contains a loader program. (generally in a preset location in c:\drive)

Loader program installs OS.

OS performs various initialisation procedures

Some form of process commences (such as init)

---

**Interrupts.**

An Interrupt is a mechanism that allows the normal processing of the processor to be interrupted.

Interrupts are used to increase efficiency.

Especially between components that operate at different speeds.
Classes of Interrupts.

<table>
<thead>
<tr>
<th>Interrupt Class</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>program</td>
<td>generated by a s/w condition (e.g. error, system call)</td>
</tr>
<tr>
<td>timer</td>
<td>generated by system clock</td>
</tr>
<tr>
<td>I/O</td>
<td>generated by I/O controller</td>
</tr>
<tr>
<td>hardware failure</td>
<td>power failure, memory parity</td>
</tr>
</tbody>
</table>

Handling Interrupts.

OS must provide some form of Interrupt handler.

Each hardware platform carries out some particular procedure for an Interrupt.

Eventually the h/w hands over to the s/w.
Simple Interrupt Handling.

- Interrupt occurs.
- CPU finishes current instruction.
- CPU pushes PSW, PC onto stack.
- CPU loads new PC value. (May be a single value all the time, might vary)
- OS saves remainder of process information.
- Interrupt is processed.
- Hardware
- Software
- OS restores process state information.
- Pops old PSW, PC values off the stack.

Interrupt Handling

- The Operating System preserves the state of the CPU by storing registers and program counter
- Determine which type of interrupt has occurred:
  - polling
  - vectored interrupt
- Jumps to appropriate pre-stored code segment to determine what action should be taken to handle each type of interrupt.
Multiple Interrupts.

What happens when an interrupt interrupts an interrupt?

Disable Interrupts
- new interrupts remain pending
- some interrupts are more important than others

Interrupt Priorities.

Performing I/O

- I/O is a major part of a computer's life. Managing I/O efficiently is a major OS responsibility.

- Each I/O device is controlled by one device controller: Device controller has its own processor, and executes *asynchronously* with CPU.

- Three major mechanisms to perform I/O:
  - program driven I/O
  - interrupt driven I/O
  - direct memory access (DMA)
Example.
Programmed Input/Output (I/O)

When I/O must happen
- CPU tells I/O device controller what to do
- I/O device controller does it
- CPU must check to see if I/O device has finished
- If it has the CPU can go on

Problems caused when I/O device is slow.

Interrupt Driven I/O.

When I/O is to be done
- CPU tells I/O device controller what to do (data+operation)
- I/O device does I/O
- When each I/O is complete device controller interrupts the CPU
  » (CPU maintains status table and a link list of jobs per I/O device)
- CPU can perform data transfer (if necessary)

CPU is still responsible for transferring data from memory to the I/O device.
Direct Memory Access.

When I/O is to be done

- CPU tells I/O device controller what to do and where to put/get the information
- CPU continues as normal
- I/O device with the help of DMA unit puts/gets information straight into memory
- DMA interrupts CPU at the end of entire data transfer

Overview: Interrupt Handling and I/O

- [Text content not legible]
- [Text content not legible]
- [Text content not legible]
- [Text content not legible]
- [Text content not legible]