Abstract:

The September 03 class is the first class in the fall. Syllabus can be found at http://www.cs.rutgers.edu/tdnguyen/courses/cs519/fall2003/index.html and will not be covered in this node.

The remaining part of the lecture is mainly about review of concepts in Architecture that are important for Operating System people, and introduction of Operating System as its purpose and major components.

1 Review of Architecture

The computer model widely used today is so called “Von Neumann” model, where code and data are not distinguished and all stored in the main memory. The CPU has a special register PC which points to the address of the next instruction. Each instruction cycle is fetch - decode - execute.

![Von Neumann Computer Model with Memory Hierarchy](image)

Figure 1: Von Neumann Computer Model with Memory Hierarchy

1.1 Memory Hierarchy

With the speed gap between CPU and memory becomes wider, memory hierarchy becomes more important than ever, which composed of
• Registers
• Caches: L1, L2 ...
• Main Memory
• Disks

Why do we need to care about this?

• Data copied in many places may cause consistency problem in the system.
• We are in charge of the virtual memory: Main Memory (≡) Disk

1.2 IO

![Figure 2: Memory and IO](image)

How does IO fits into this model? See Figure 2. Special memory addresses are reserved for IO devices. When such an address selected, registers in the devices respond instead of the main memory.

• **Program IO** CPU take care of transferring each byte of data between memory and devices.
• **DMA** CPU tells a DMA controller go get/store such data from/to such devices. DMA controller will create an interrupt when done.

Node that when in DMA mode, memory may modified by the DMA controller while some cache may still hold old data, which is a consistency problem. Common solutions include: IO buffer in the memory is not cached, or automatic cache invalidation when writing operation appears on the IO bus with corresponding memory address.

1.3 Others

**Interruption, Trap and Exception:**
They all have a number, a handler, and when happened, the CPU enters kernel mode and begin to execute the handler whose starting address is given in the interruption table. While the differences:

• Interruption usually corresponds to some hardware events, like some device just finished last job, or the clock ticked. And the later one is important in process/thread scheduling.
• Exceptions are usually program errors like divided by zero, invalid instructions etc.. While accessing non-accessible memory address is used in implementing virtual memory.

• Trap is a way for user application ask for OS services, so that transferring to kernel mode and start executing OS code could happen at the same time.

**TLB**
This is a piece of hardware made solely for OS to implement virtual memory. Address translation happens on *every* instruction run on virtual space: each time CPU fetches an instruction, the address of that instruction needs to be translated. If it’s a load or store we have one more address to translate. We can’t afford to translate virtual address to physical one by OS.

## 2 OS Introduction

### 2.1 What is OS

**Resource Management Agent**
Operating System manages all the hardware resource, regulates the access to these resource by applications.

**A Virtual Machine**
It hides complicated and often heterogeneous hardware interface from the application and provide higher level abstraction for them.

**Mechanism/Policy**
Mechanism is the way to implement the abstraction. Policy is some rule or pattern upon which the behavior of the applications should be regulated.

### 2.2 Major Components

Usually OS is composed of File System, Memory Management and Process/Thread Management. Each of these components provide abstraction of certain aspect of the resources provided by hardware.

**File System**
It provides abstraction of persistent storage (like disks):

- **Name Space:** A tree (or DAG) of directories.
- **File:** as a continues sequence of bytes.

*Mechanism:* Tables etc. that translate directories/files onto block address on disks.
*Policies:* Security, Quota, Number of files that can be open at the same time.

**Memory Management**
In another words, Virtual Memory. It provides abstraction of memory hierarchy:

- **Size:** A much larger memory space than the hardware actually has.
- **Shape:** as a continues sequence of bytes. And use paging to enable sharing

*Mechanism:* Paging, Segementation, Address Translation, Page fault ...

**Process/Thread Management**
It provides abstraction of the CPU+memory+Sharing, gives the application a virtual private machine, and sharing at the same time if specified by the application.

*Mechanism:* Context Switch, Pipe, Message Queue, IPC ...

*Polity:* Priority of Processes, Multiplex of CPU/Memory ...