

## Operating Systems Components (3)

### Lecture 4

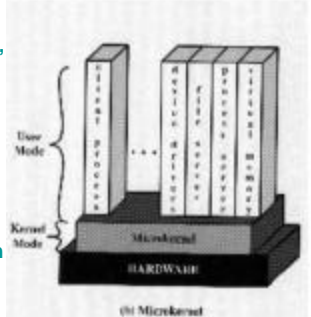
University of Cambridge  
Kings

CS153, B. D. Fleisch

1

## Microkernel OS(1)

- Client-server model, IPC between clients and servers
- Smaller, easier to tune, debug and maintain
- Microkernel provides protected communication
- Good policy/mechanism separation
  - ◆ kernel provides minimal mechanism
  - ◆ flexible expression of policy at user level



University of Cambridge  
Kings

CS153, B. D. Fleisch

2

## Microkernel OS(2)

- Many OS functions are implemented as user-level servers in a microkernel
- The OS functions can mimic the functions of UNIX or Apple OS systems or others. IBM called these *personalities*
- Efficiency is a problem, often slower response from system services in a microkernel design
- Easy to extend to distributed systems

University of Cambridge  
Kings

CS153, B. D. Fleisch

3

## Readings on IBM Microkernel

- Microkernel design isn't always successful
- See the IBM workplace OS article on my web page under link "research publications" called "The Failure of Personalities to Generalize" by Brett D. Fleisch in HOT-OS VI, May, 1997 (#9). See research publications on my web page.

University of Cambridge  
Kings

CS153, B. D. Fleisch

4

### Multithreading

- A process can be multithreaded and consist of collection of one or more **threads** that can run simultaneously
- Useful when the application consists of several tasks that 1) do not need to be serialized and 2) can be run concurrently
- Threads give the programmer greater control over the timing of application-related events
- All threads within the same process share the same data and resources and a part of the process's execution context
- It is easier to create or destroy a thread or switch among threads (of the same process) than to do the same thing with processes
- CS160 deals with concurrent programming (and multithreaded programming)

CS153, B. D. Fleisch
5

### Symmetric Multiprocessing (SMP)

- A computer with multiple processors
- Each processor can perform the same functions and share same main memory and I/O facilities (symmetric)
- The OS schedule processes/threads across all the processors (real parallelism)
- Existence of multiple processors is transparent to the user.
- Incremental growth: just add another CPU!
- Robustness: a single CPU failure does not halt the system, only the performance is reduced.
- Solaris and Windows NT are designed to handle SMP well

CS153, B. D. Fleisch
6

### Example of parallel execution on SMP

CS153, B. D. Fleisch
7

### Virtual Machine

- Real machine is emulated in a “virtual machine”
- OS functionality on top of the virtual machine
- Good protection and Isolation
- Difficult to emulate hardware

Figure 5.11 System models: (a) Normal machine, (b) Virtual machine.

CS153, B. D. Fleisch
8

### Types of Operating Systems

**FIGURE 1.3**  
A classification of advanced operating systems.

- **Distributed Operating Systems**
  - OS for autonomous computers connected via network
  - Manages hardware and software
  - Transparency
- **Characteristics**
  - ◆ No shared physical memory
  - ◆ No shared physical clock
  - ◆ Unpredictable communication delays

University of Cambridge Business School CS153, B. D. Fleisch 9

### Types of Operating Systems (continued)

- **Multiprocessor OSs**
  - ◆ Processors share common physical memory over an interconnection network
  - ◆ Tightly coupled system
  - ◆ Issues of complexity: Sharing common main memory efficiently
- **Database OSs**
  - ◆ Require support for transactions
  - ◆ Store/Retrieve large volumes of Data
  - ◆ Concurrency Control and Failure Recovery
- **Real-time OSs**
  - ◆ Deadlines
  - ◆ Scheduling of jobs to meet deadlines

University of Cambridge Business School CS153, B. D. Fleisch 10

### Computer System Overview (1)

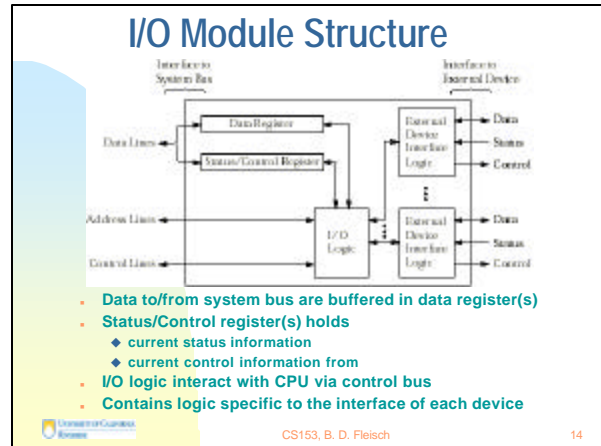
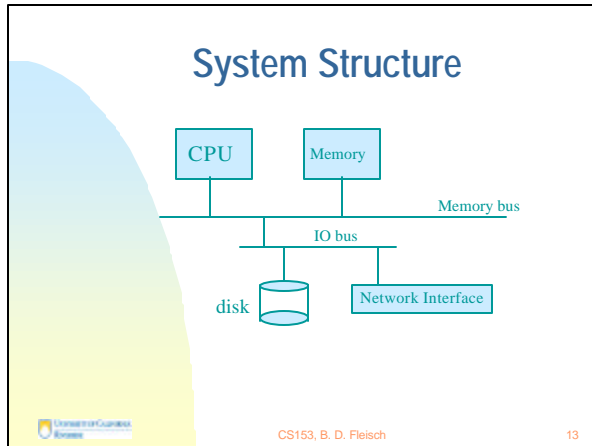
**Review of basic architecture components relevant to operating systems**

University of Cambridge Business School CS153, B. D. Fleisch 11

### Basic Components

- **Processor (CPU)**
- **Main Memory (aka real memory, aka primary memory)**
  - ◆ volatile
- **I/O modules (I/O controllers, I/O channels, I/O processors...)**
  - ◆ hardware (with registers called I/O ports) that moves data between cpu and peripherals like:
    - secondary memory devices (eg: hard disks)
    - keyboard, display...
    - communications equipment
- **System interconnection (ie: Buses)**
  - ◆ communication among processors, memory, and I/O modules

University of Cambridge Business School CS153, B. D. Fleisch 12



- ### CPU Registers (fast memory on cpu)
- Control & Status Registers**
    - Generally not available to user programs
    - some used by CPU to control its operation
    - some used by OS to control program execution
  - User-visible Registers**
    - available to system (OS) and user programs
    - holds data, addresses, and some condition codes
- University of Waterloo  
CS153, B. D. Fleisch 15

- ### Control & Status Registers (Examples)
- Program Counter (PC)**
    - Contains the address of the next instruction to be fetched
  - Instruction Register (IR)**
    - Contains the instruction most recently fetched
  - Program Status Word (PSW)**
    - A register or group of registers containing:
      - condition codes and status info bits
      - Interrupt enable/disable bit
      - Supervisor(OS)/user mode bit
- University of Waterloo  
CS153, B. D. Fleisch 16

### User-Visible Registers

- **Data Registers**
  - ◆ can be assigned by the user program to perform operations on data
- **Address Registers**
  - ◆ contain memory address of data and instructions
  - ◆ may contain a portion of an address that is used to calculate the complete address
- **Examples of Address Registers**
  - ◆ **Index/Offset**
    - involves adding an index to a base value to get an address
  - ◆ **Segment pointer**
    - when memory is divided into segments, memory is referenced by a segment and an offset
  - ◆ **Stack pointer**
    - points to top of stack

University of Waterloo  
CS153, B. D. Fleisch 17

### User-Visible Registers

- **Condition Codes or Flags**
  - ◆ Bits set by the processor hardware as a result of operations
  - ◆ Can be accessed by a program but not changed directly
  - ◆ **Examples**
    - sign flag
    - zero flag
    - overflow flag

University of Waterloo  
CS153, B. D. Fleisch 18

### The Basic Instruction Cycle

```

    graph LR
      START([START]) --> Fetch[Fetch Next Instruction]
      Fetch --> Execute[Execute Instruction]
      Execute --> HALT([HALT])
      Execute -- Basic Cycle --> Fetch
  
```

- The CPU fetches the next instruction (with operands) from memory
- Next the CPU executes the instruction
- The Program Counter (PC) holds address of the instruction to be fetched next
- PC is incremented after each fetch

University of Waterloo  
CS153, B. D. Fleisch 19

### Modes of Execution

- To provide protection for kernel data structures and entities (and other OS data) most processors support at least 2 execution modes:
  - ◆ **Privileged mode** (a.k.a. system mode, kernel mode, supervisor mode, control mode)
    - Allows manipulating control registers, primitive I/O instructions, memory management via privileged instructions
  - ◆ **User mode** – only nonprivileged instructions may be executed and a process can only access its own virtual address range
- To support modes, the CPU provides a mode bit (whether it is a bit depends on number of states) which indicates the mode the processor is executing in

University of Waterloo  
CS153, B. D. Fleisch 20

### Synchronous I/O

- WRITE is a privileged instruction that causes a system call and transfers control to the printer driver eventually (I/O program)
- I/O program prepares I/O module for printing (4)
- CPU must WAIT for I/O command to complete
- Long wait for a printer
- I/O program finishes in (5) and reports status of the operation

University of Cambridge  
CS153, B. D. Fleisch
21

### Interrupts

- Modern systems now permit I/O modules to INTERRUPT the CPU
- I/O modules just assert an interrupt request line on the control bus
- Then the CPU transfers control to an **Interrupt Handler Routine** which is normally part of the OS
- Interrupts cause:
  - 1. Processor to save the current execution context
  - 2. Set the PC to the start address of an interrupt handler
  - 3. Switch from user mode to kernel mode to execute privileged instructions in the handler

University of Cambridge  
CS153, B. D. Fleisch
22

### Instruction Cycle with Interrupts

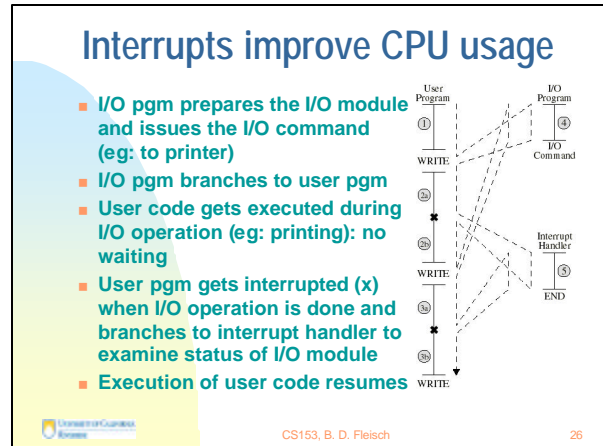
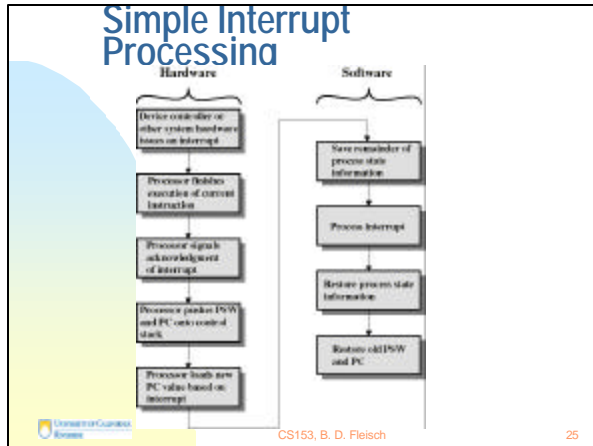
- CPU checks for interrupts after each instruction
- If no interrupts, then fetch the next instruction for the current program
- If an interrupt is pending, then suspend execution of the current program, and execute the **interrupt handler**

University of Cambridge  
CS153, B. D. Fleisch
23

### Interrupt Handler

- A program that determines nature of the interrupt and performs necessary actions
- Control is transferred to this program when the interrupt is fielded
- Control must be transferred back to the interrupted program so that it can be resumed from the point of interruption
- This point of interruption can occur anywhere in the program
- Thus: must save the state of the program (content of PC + PSW + registers + ...)

University of Cambridge  
CS153, B. D. Fleisch
24



- ### Classes of Interrupts
- I/O
    - signals normal completion of operation or error
  - Program Exception (Synchronous interrupt)
    - Division by zero
    - Overflows
    - Attempt to execute an illegal instruction e.g. a privileged instruction from User mode
    - Reference outside user's memory space
    - SVC call (invocation of a kernel call) (next viewgraph)
  - Timer
    - Preempts a program to perform another task
  - Hardware failure (eg: memory parity error)
- University of Waterloo  
CS153, B. D. Fleisch 27