Slide 1

Computer System Overview

Operating Systems

2004/S2

Slide 2

Operating System

➜ Provides an abstraction layer over the concrete hardware
➜ Allocation of resources
- Processor
- Memory
- I/O devices

➜ Optimisation of resource utilisation
➜ Protection and Security

Understanding operating systems therefore requires some basic understanding of computer systems

Slide 3

Basic Elements

Simplified view:

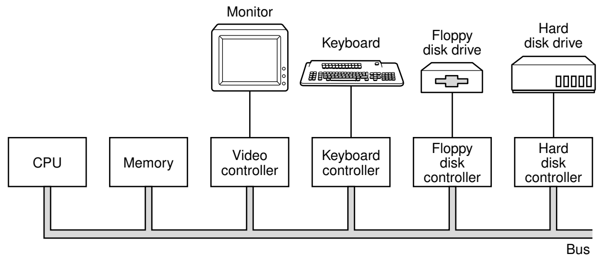
➜ Processor
➜ Main Memory
- referred to as real memory or primary memory
- volatile

➜ I/O modules
- secondary memory devices
- communications equipment
- terminals

➜ System bus
- communication among processors, memory, and I/O modules

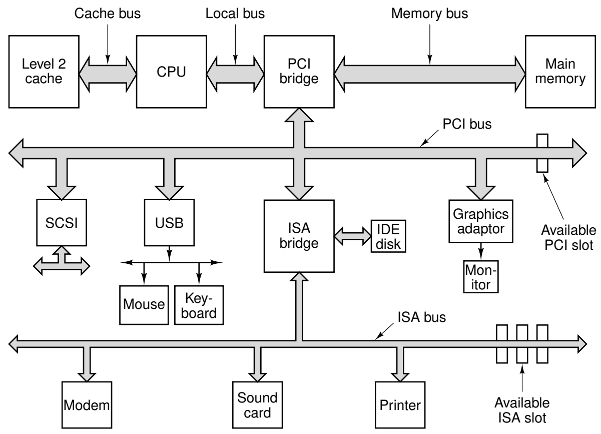
Slide 4

Top-Level Components



Slide 5

Example: Large Pentium System



Slide 6

Processor

- ➜ Fetches intructions from memory, decodes and executes them
- ➜ Set of instructions is processor specific
- ➜ Instructions include:
  - ✶ load value from memory into register
  - ✶ combine operands from registers or memory
  - ✶ branch
- ➜ All CPU's have registers to store
  - ✶ key variables and temporary results
  - ✶ information related to control program execution

Slide 7

Processor Registers

- ➜ Data and address registers
  - Hold operands of most native machine instructions
  - Enable programmer to minimize main-memory references by optimizing register use
  - user-visible
- ➜ Control and status registers
  - Used by processor to control operating of the processor
  - Used by operating-system routines to control the execution of programs
  - Sometimes not accessible by user (architecture dependent)

Slide 8

User-Visible Registers

- ➜ May be referenced by machine language instructions
- ➜ Available to all programs - application programs and system programs
- ➜ Types of registers
  - Data
  - Address
    - Index
    - Segment pointer
    - Stack pointer
  - Many architectures do not distinguish different types

Slide 9

Control and Status Registers

➜ Program Counter (PC)

- Contains the address of an instruction to be fetched

➜ Instruction Register (IR)

- Contains the instruction most recently fetched

➜ Processor Status Word (PSW)

- condition codes
- interrupt enable/disable
- supervisor/user mode

Slide 10

Control and Status Registers

➜ Condition Codes or Flags

- Bits set by the processor hardware as a result of operations
- Can be accessed by a program but not altered
- Examples
  - positive/negative result
  - zero
  - overflow

Slide 11

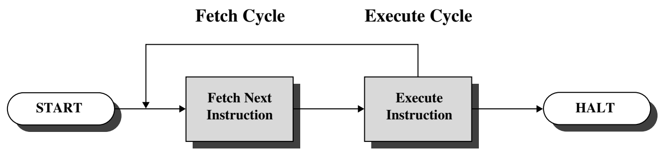
Instruction Fetch and Execute

➜ Program counter (PC) holds address of the instruction to be fetched next

➜ The processor fetches the instruction from memory

➜ Program counter is incremented after each fetch

➜ Overlapped on modern architectures (pipelining)



Slide 12

Instruction Register

➜ Fetched instruction is placed in the instruction register

➜ Types of instructions

- Processor-memory
  - transfer data between processor and memory
- Processor-I/O
  - data transferred to or from a peripheral device
- Data processing
  - arithmetic or logic operation on data
- Control
  - alter sequence of execution

Slide 13

INTERACTION BETWEEN PROCESSOR AND I/O DEVICES

➜ CPU much faster than I/O devices
  - waiting for I/O operation to finish is inefficient
  - not feasible for mouse, keyboard

➜ I/O module sends an interrupt to CPU to signal completion

➜ Interrupts normal sequence of execution

➜ Interrupts are also used to signal other events

Slide 14

CLASSES OF INTERRUPTS

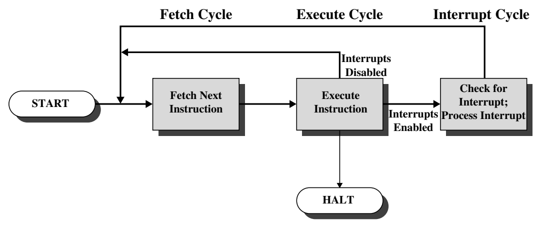
➜ Asynchronous (external) events
- I/O
- Timer
- Hardware failure

➜ Synchronous interrupts or program exceptions caused by program execution:
- arithmetic overflow
- division by zero
- execute illegal instruction
- reference outside user's memory space

Slide 15

INTERRUPT CYCLE

① Fetch next instruction
② Execute instruction
③ Check for interrupt
④ If no interrupts, fetch the next instruction
⑤ If an interrupt is pending, divert to the interrupt handler



Slide 16

INTERRUPT HANDLER

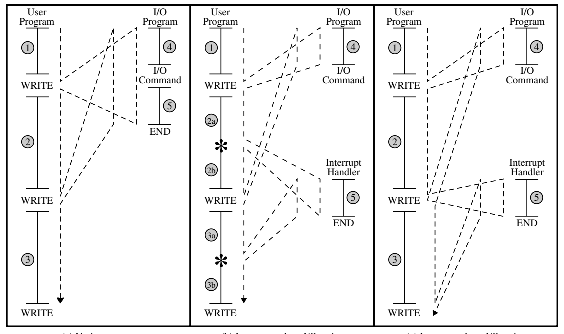
➜ A program that determines nature of the interrupt and performs whatever actions are needed

➜ Control is transferred to this program by the hardware

➜ Generally part of the operating system

Slide 17

Control Flow With and Without Interrupts

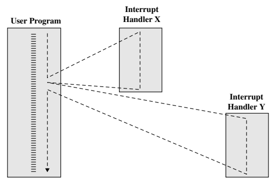


(a) No interrupts (b) Interrupts; short I/O wait (c) Interrupts; long I/O wait

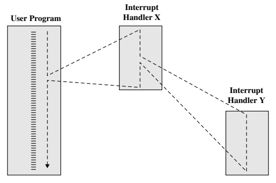
Slide 18

Multiple Interrupts

- ➜ Interrupt X occurs
- ➜ CPU disables all interrupts (only those with lower priority)
- ➜ Interrupt handler may enable interrupts
- ➜ Interrupt Y occurs
- ➜ Sequential or nested interrupt handling



(a) Sequential interrupt processing



(b) Nested interrupt processing

Slide 19

Multiple Interrupts

Sequential Order:

- ➜ Disable interrupts so processor can complete task
- ➜ Interrupts remain pending until the processor enables interrupts
- ➜ After interrupt handler routine completes, the processor checks for additional interrupts

Slide 20

Multiple Interrupts

Priorities:

- ➜ Higher priority interrupts cause lower-priority interrupts to wait
- ➜ Causes a lower-priority interrupt handler to be interrupted
- ➜ Example: when input arrives from communication line, it needs to be absorbed quickly to make room for more input

Slide 21

Memory

Sould be

- ➜ fast
- ➜ abundant
- ➜ cheap

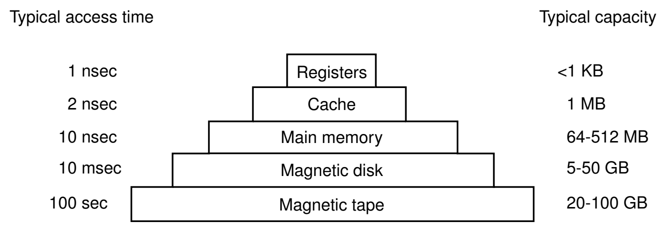
Unfortunately, that's not the reality...

Solution:

- combination of fast & expensive and slow & cheap memory

Slide 22

Memory Hierarchy



Typical access time		Typical capacity
1 nsec	Registers	<1 KB
2 nsec	Cache	1 MB
10 nsec	Main memory	64-512 MB
10 msec	Magnetic disk	5-50 GB
100 sec	Magnetic tape	20-100 GB

Slide 23

Going Down the Hierarchy

- ➜ Decreasing cost per bit
- ➜ Increasing capacity
- ➜ Increasing access time
- ➜ Decreasing frequency of access of the memory by the processor

Locality of reference is essential!

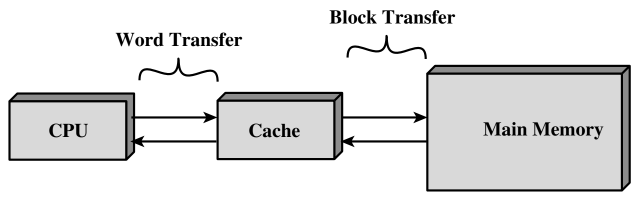
Slide 24

Disk Cache

- ➜ A portion of main memory used as a buffer to temporarily to hold data for the disk
- ➜ Disk writes are clustered
- ➜ Some data written out may be referenced again. The data are retrieved rapidly from the software cache instead of slowly from disk
- ➜ Mostly transparent to operating system

Slide 25

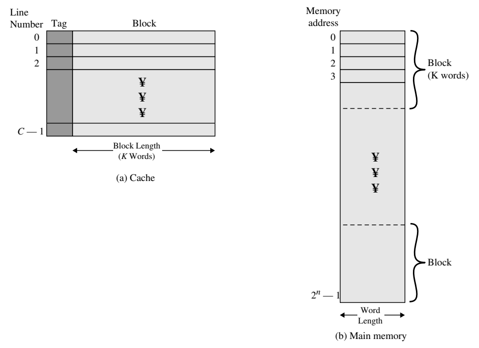
Cache Memory



- ➜ Contains a portion of main memory
- ➜ Processor first checks cache
- ➜ If not found in cache, the block of memory containing the needed information is moved to the cache
  replacing some other data

Slide 26

Cache/Main Memory System



Slide 27

Cache Design

- ➜ Cache size
  - small caches have a significant impact on performance
- ➜ Line size (block size)
  - the unit of data exchanged between cache and main memory
  - hit means the information was found in the cache
  - larger line size $\Rightarrow$ higher hit rate
    until probability of using newly fetched data becomes less than the probability of reusing data that has been moved out of cache

Slide 28

Cache Design

- ➜ Mapping function
  - determines which cache location the data will occupy
- ➜ Replacement algorithm
  - determines which line to replace
  - Least-Recently-Used (LRU) algorithm
- ➜ Write policy
  - When the memory write operation takes place
  - Can occur every time line is updated (write-through policy)
  - Can occur only when line is replaced (write-back policy)
    - Minimizes memory operations
    - Leaves memory in an obsolete state

Slide 29

Interaction between I/O Devices and Processor

➜ Controller (chip or set of chips) provides a simple interface to OS
  - often, embedded OS running on the controller

➜ Software that communicates with controller is called device driver

➜ Most drivers run in kernel mode

➜ To put new driver into kernel, system may have to
  - be relinked
  - be rebooted
  - dynamically load new driver

Slide 30

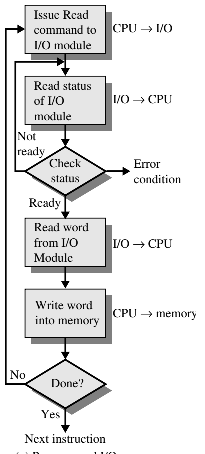
Programmed I/O (Polling)

➜ I/O module performs the action, not the processor

➜ Sets appropriate bits in the I/O status register

➜ No interrupts occur

➜ Processor checks status until operation is complete
  - Wastes CPU cycles



(a) Programmed I/O

Slide 31

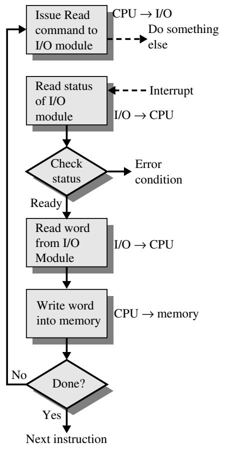
Interrupt-Driven I/O

➜ Processor is interrupted when I/O module ready to exchange data

➜ Processor is free to do other work

➜ No needless waiting

➜ Consumes a lot of processor time because every word read or written passes through the processor



(b) Interrupt-driven I/O

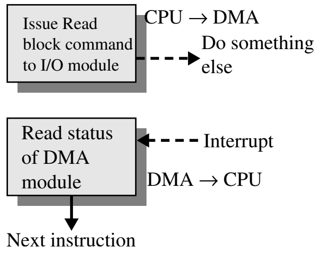
Slide 32

Direct Memory Access

➜ Transfers a block of data directly to or from memory

➜ An interrupt is sent when the task is complete

➜ The processor is only involved at the beginning and end of the transfer



Issue Read block command to I/O module
CPU → DMA
Do something else
Read status of DMA module
Interrupt
DMA → CPU
Next instruction