


CSE

# Introduction to Operating Systems


Chapter 1 – 1.3  
Chapter 1.5 – 1.9



CSE



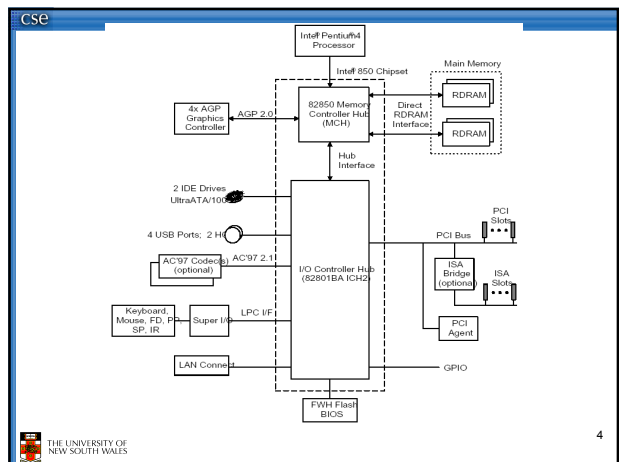
## Learning Outcomes

- High-level understand what is an operating system and the role it plays
- A high-level understanding of the structure of operating systems, applications, and the relationship between them.
- Some knowledge of the services provided by operating systems.
- Exposure to some details of major OS concepts.



CSE


## What is an Operating System?

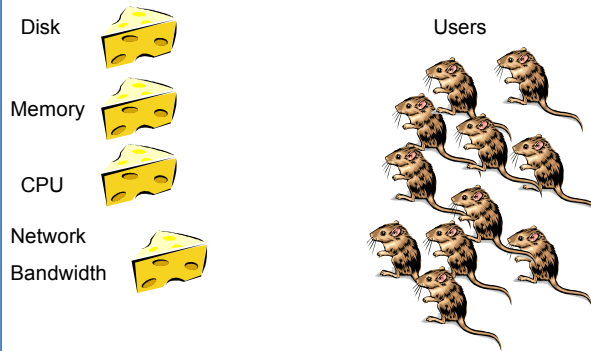
CSE

## Viewing the Operating System as an Abstract Machine

- Extends the basic hardware with added functionality
- Provides high-level abstractions
  - More programmer friendly
  - Common core for all applications
- It hides the details of the hardware
  - Makes application code portable



CSE




Disk

Memory

CPU

Network Bandwidth

Users



CSE

## Viewing the Operating System as a Resource Manager

- Responsible for allocating resources to users and processes
- Must ensure
  - No Starvation
  - Progress
  - Allocation is according to some desired policy
    - First-come, first-served; Fair share; Weighted fair share; limits (quotas), etc...
  - Overall, that the system is efficiently used

THE UNIVERSITY OF NEW SOUTH WALES 7

CSE

## Traditional View: the Operating System as the Privileged Component

The diagram illustrates the traditional view of the operating system. It is divided into two horizontal sections by a dashed line. The top section is labeled 'User Mode' and contains three boxes labeled 'Applications'. The bottom section is labeled 'Privileged Mode' and contains a single green box labeled 'Operating System'. Arrows labeled 'Requests (System Calls)' point from each 'Applications' box down to the 'Operating System' box. Below the 'Privileged Mode' section is a grey area labeled 'Hardware'.

THE UNIVERSITY OF NEW SOUTH WALES 8

CSE

## Kernel

- Portion of the operating system that is running in *privileged mode*
- Usually resident in main memory
- Contains fundamental functionality
  - Whatever is required to implement other services
  - Whatever is required to provide security
- Contains most-frequently used functions
- Also called the nucleus or supervisor

THE UNIVERSITY OF NEW SOUTH WALES 9

CSE

## The Operating System is Privileged

- Applications should not be able to interfere or bypass the operating system
  - OS can enforce the "extended machine"
  - OS can enforce its resource allocation policies
  - Prevent applications from interfering with each other
- Note: Some Embedded OSs have no privileged component, e.g. PalmOS, Mac OS 9
  - Can implement OS functionality, but cannot enforce it.
- Note: Some operating systems implement significant OS functionality in user-mode, e.g. User-mode Linux

THE UNIVERSITY OF NEW SOUTH WALES 10

CSE

## Structure of a Computer System

The diagram shows the structure of a computer system. It is divided into 'User Mode' and 'Kernel Mode' by a dashed line. In 'User Mode', there are two boxes, each containing 'Application' and 'System Libraries'. In 'Kernel Mode', there are two yellow ovals labeled 'Device' and a green box labeled 'Operating System'. The entire system is contained within a larger box labeled 'Memory'.

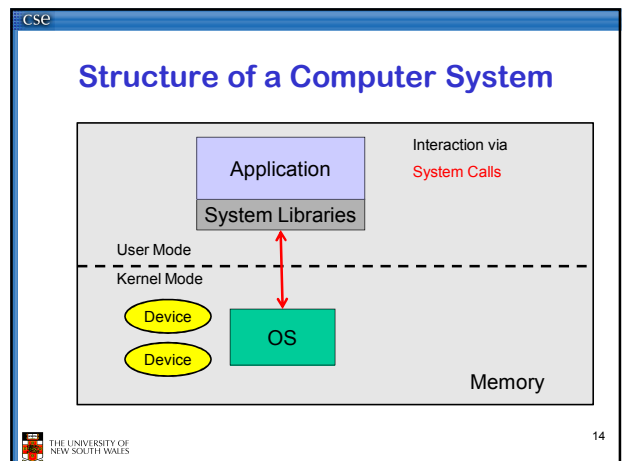
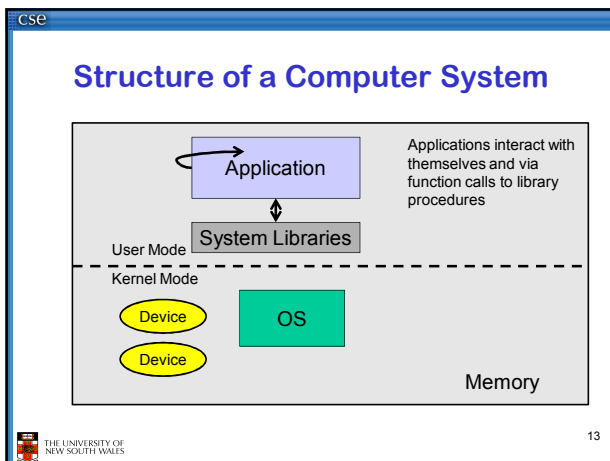
THE UNIVERSITY OF NEW SOUTH WALES 11

CSE

## Structure of a Computer System

This diagram is similar to the previous one but highlights the OS's interaction. In 'User Mode', there is one box containing 'Application' and 'System Libraries'. In 'Kernel Mode', there are two yellow ovals labeled 'Device' and a green box labeled 'OS'. Arrows show bidirectional communication between the 'OS' and the 'Device' ovals. A text box on the right states: 'OS interacts via load and store instructions to all memory, CPU and device registers, and interrupts'. The entire system is within 'Memory'.

THE UNIVERSITY OF NEW SOUTH WALES 12



- ### A note on System Libraries
- System libraries are just that, libraries of support functions (procedures, subroutines)
- Only a subset of library functions are actually systems calls
    - `strcmp()`, `memcpy()`, are pure library functions
      - manipulate memory within the application, or perform computation
    - `open()`, `close()`, `read()`, `write()` are system calls
      - they cross the user-kernel boundary, e.g. to read from disk device
  - System call functions are in the library for convenience
    - try `man syscalls` on Linux
- The University of New South Wales logo and the number 15 are in the bottom left corner.

- ### Operating System Objectives
- Convenience
    - Make the computer more convenient to use
  - Abstraction
    - Hardware-independent programming model
  - Efficiency
    - Allows the computer system to be used in an efficient manner
  - Ability to evolve
    - Permit effective development, testing, and introduction of new system functions without interfering with existing services
  - Protection
    - allow only authorised access to data, computation, services, etc.
- The University of New South Wales logo and the number 16 are in the bottom left corner.

- ### Services Provided by the Operating System
- Program execution
    - Load a program and its data
  - Access to I/O devices
    - Display, disk, network, printer, keyboard, camera, etc.
  - Controlled access to files
    - Access protection
  - System access
    - User authentication
- The University of New South Wales logo and the number 17 are in the bottom left corner.

- ### Services Provided by the Operating System
- Error detection and response
    - internal and external hardware errors
      - memory error
      - device failure
    - software errors
      - arithmetic overflow
      - access forbidden memory locations
    - operating system cannot grant request of application
- The University of New South Wales logo and the number 18 are in the bottom left corner.

CSE

## Services Provided by the Operating System

- Accounting
  - collect statistics
  - monitor performance
    - diagnose lack of it
  - used to anticipate future enhancements
  - used for billing users

THE UNIVERSITY OF NEW SOUTH WALES 19

CSE

## Operating System Software

- Fundamentally, OS functions the same way as ordinary computer software
  - It is a program that is executed (just like apps)
  - It has more privileges
- Operating system relinquishes control of the processor to execute other programs
  - Reestablishes control after
    - System calls
    - Interrupts (especially timer interrupts)

The diagram illustrates the software stack and execution modes. At the top is 'Application' (purple box) and 'System Libraries' (grey box). Below them is 'User Mode' (indicated by a dashed line). In 'Kernel Mode' (below the dashed line), there is 'OS' (green box) and 'Memory' (grey box). Two yellow ovals labeled 'Device' are shown at the bottom, connected to the OS and Memory.

THE UNIVERSITY OF NEW SOUTH WALES 20

CSE

## Major OS Concepts (Overview)

- Processes
- Concurrency and deadlocks
- Memory management
- Files
- Scheduling and resource management
- Information Security and Protection

THE UNIVERSITY OF NEW SOUTH WALES 21

CSE

## Processes

- A program in execution
- An instance of a program running on a computer
- The entity that can be assigned to and executed on a processor
- A unit of resource ownership

THE UNIVERSITY OF NEW SOUTH WALES 22

CSE

## Process

- Consist of three segments
  - Text
    - contains the code (instructions)
  - Data
    - Global variables
  - Stack
    - Activation records of procedure
    - Local variables
- Note:
  - data can dynamically grow up
  - The stack can dynamically grow down

The diagram shows a vertical stack of memory segments. From top to bottom: 'Stack' (purple box), 'Gap' (white box with a downward arrow above it and an upward arrow below it), 'Data' (blue box), and 'Text' (green box). The label 'Memory' is at the top right.

THE UNIVERSITY OF NEW SOUTH WALES 23

CSE

## Process

- Consists of three components
  - An executable program
    - text
  - Associated data needed by the program
    - Data and stack
  - Execution context of the program
    - All information the operating system needs to manage the process
      - Registers, program counter, stack pointer, etc...

THE UNIVERSITY OF NEW SOUTH WALES 24

CSE

## Multiple processes creates concurrency issues

(a) A potential deadlock. (b) an actual deadlock.

THE UNIVERSITY OF NEW SOUTH WALES 25

CSE

## Memory Management

- The view from thirty thousand feet
  - Process isolation
    - Prevent processes from accessing each others data
  - Automatic allocation and management
    - Don't want users to deal with physical memory directly
  - Protection and access control
    - Still want controlled sharing
  - Long-term storage
    - Virtual memory
    - File system

THE UNIVERSITY OF NEW SOUTH WALES 26

CSE

## Virtual Memory

- Allows programmers to address memory from a logical point of view
  - Gives apps the illusion of having RAM to themselves
  - Logical addresses are independent of other processes
  - Provides isolation of processes from each other
- Can overlap execution of one process while swapping in/out others.

THE UNIVERSITY OF NEW SOUTH WALES 27

CSE

## Virtual Memory Addressing

Figure 2.10 Virtual Memory Addressing

THE UNIVERSITY OF NEW SOUTH WALES 28

CSE

## File System

- Implements long-term store
- Information stored in named objects called files

THE UNIVERSITY OF NEW SOUTH WALES 29

CSE

## Example File System

THE UNIVERSITY OF NEW SOUTH WALES 30

CSE

## Information Protection and Security

- Access control
  - regulate user access to the system
  - Involves authentication
- Information flow control
  - regulate flow of data within the system and its delivery to users

THE UNIVERSITY OF NEW SOUTH WALES 31

CSE

## Scheduling and Resource Management

- Fairness
  - give equal and fair access to all processes
- Differential responsiveness
  - discriminate between different classes of jobs
- Efficiency
  - maximize throughput, minimize response time, and accommodate as many uses as possible

THE UNIVERSITY OF NEW SOUTH WALES 32

CSE

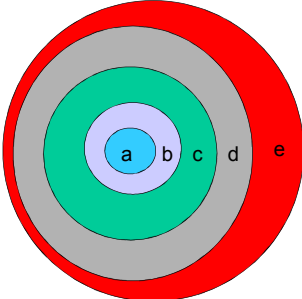
## Operating System Structure?

THE UNIVERSITY OF NEW SOUTH WALES 33

CSE

## Operating System Structure

- The layered approach
  - a) Processor allocation and multiprogramming
  - b) Memory Management
  - c) Devices
  - d) File system
  - e) Users
  - Each layer depends on the inner layers



THE UNIVERSITY OF NEW SOUTH WALES 34

CSE

## Operating System Structure

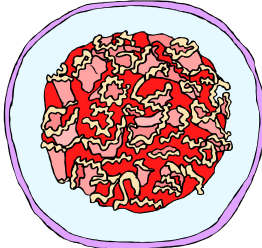
- In practice, layering is only a guide
  - Operating Systems have many interdependencies
    - Scheduling on virtual memory
    - Virtual memory on I/O to disk
    - VM on files (page to file)
    - Files on VM (memory mapped files)
    - And many more...

THE UNIVERSITY OF NEW SOUTH WALES 35

CSE

## The Monolithic Operating System Structure

- Also called the “spaghetti nest” approach
  - Everything is tangled up with everything else.
- Linux, Windows, ...



THE UNIVERSITY OF NEW SOUTH WALES 36

CSE

## The Monolithic Operating System Structure

- However, some reasonable structure usually prevails

The diagram illustrates the monolithic operating system structure with the following subsystems and their dependencies:

- File System** depends on: Memory Manager, Process Scheduler, Inter-Process Communication, Network Interface, and Library.
- Memory Manager** depends on: File System, Process Scheduler, and Initialization.
- Process Scheduler** depends on: File System, Memory Manager, Inter-Process Communication, and Initialization.
- Inter-Process Communication** depends on: File System, Memory Manager, Process Scheduler, and Library.
- Network Interface** depends on: File System, Process Scheduler, and Library.
- Initialization** depends on: File System, Memory Manager, Process Scheduler, and Library.
- Library** depends on: File System, Process Scheduler, Inter-Process Communication, Network Interface, and Initialization.

Legend: **Subsystem** — extracted dependency →

Bozeman, I. T., Holt, R. C., and Brewster, N. V. 1999. Linux as a case study: its extracted software architecture. In Proceedings of the 21st International Conference on Software Engineering (Los Angeles, California, United States, May 16 - 22, 1999). ICSE '99. ACM, New York, NY, 656-663. DOI: http://doi.acm.org/10.1145/324455.302891

37

THE UNIVERSITY OF NEW SOUTH WALES

CSE

## Computer Hardware Review

### Chapter 1.4

THE UNIVERSITY OF NEW SOUTH WALES

CSE

## Learning Outcomes

- Understand the basic components of computer hardware
  - CPU, buses, memory, devices controllers, DMA, Interrupts, hard disks
- Understand the concepts of memory hierarchy and caching, and how they affect performance.

39

THE UNIVERSITY OF NEW SOUTH WALES

CSE

## Operating Systems

- Exploit the hardware available
- Provide a set of high-level services that represent or are implemented by the hardware.
- Manages the hardware reliably and efficiently
- *Understanding operating systems requires a basic understanding of the underlying hardware*

40

THE UNIVERSITY OF NEW SOUTH WALES

CSE

## Basic Computer Elements

The diagram shows the basic computer elements connected to a central system bus:

- CPU** is connected to the system bus.
- disk controller** is connected to the system bus and manages two **disk** drives.
- printer controller** is connected to the system bus and manages a **printer**.
- tape-drive controller** is connected to the system bus and manages **tape drives**.
- memory controller** is connected to the system bus and manages **memory**.

41

THE UNIVERSITY OF NEW SOUTH WALES

CSE

## Basic Computer Elements

- **CPU**
  - Performs computations
  - Load data to/from memory via system bus
- **Device controllers**
  - Control operation of their particular device
  - Operate in parallel with CPU
  - Can also load/store to memory (Direct Memory Access, DMA)
  - Control register appear as memory locations to CPU
    - Or I/O ports
  - Signal the CPU with "interrupts"
- **Memory Controller**
  - Responsible for refreshing dynamic RAM
  - Arbitrating access between different devices and CPU

42

THE UNIVERSITY OF NEW SOUTH WALES

**The real world is logically similar, but more complex**

Intel X58 Express Chipset Block Diagram

43

**A Simple Model of CPU Computation**

- The fetch-execute cycle

Figure 1.2 Basic Instruction Cycle

44

**A Simple Model of CPU Computation**

- The fetch-execute cycle
  - Load memory contents from address in program counter (PC)
    - The instruction
  - Execute the instruction
  - Increment PC
  - Repeat

CPU Registers

PC: 0x0300
SP: 0xcbf3
Status
R1
↑
Rn

45

**A Simple Model of CPU Computation**

- Stack Pointer
- Status Register
  - Condition codes
    - Positive result
    - Zero result
    - Negative result
- General Purpose Registers
  - Holds operands of most instructions
  - Enables programmers (compiler) to minimise memory references.

CPU Registers

PC: 0x0300
SP: 0xcbf3
Status
R1
↑
Rn

46

**Privileged-mode Operation**

CPU Registers

- To protect operating system execution, two or more CPU modes of operation exist
  - Privileged mode (system-, kernel-mode)
    - All instructions and registers are available
  - User-mode
    - Uses 'safe' subset of the instruction set
      - E.g. no disable interrupts instruction
    - Only 'safe' registers are accessible

Interrupt Mask
Exception Type
MMU regs
Others
PC: 0x0300
SP: 0xcbf3
Status
R1
↑
Rn

47

**'Safe' registers and instructions**

- Registers and instructions are safe if
  - Only affect the state of the application itself
  - They cannot be used to uncontrollably interfere with
    - The operating system
    - Other applications
  - They cannot be used to violate a correctly implemented operating system.

48

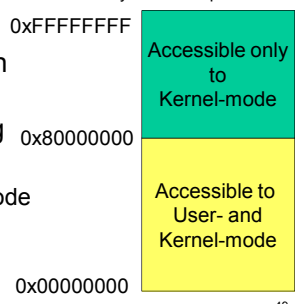


CSE

## Privileged-mode Operation

Memory Address Space

- The accessibility of addresses within an address space changes depending on operating mode
  - To protect kernel code and data



0xFFFFFFFF  
Accessible only to Kernel-mode  
0x80000000  
Accessible to User- and Kernel-mode  
0x00000000

THE UNIVERSITY OF NEW SOUTH WALES 49

CSE

## I/O and Interrupts

- I/O events (keyboard, mouse, incoming network packets) happen at unpredictable times
- How does the CPU know when to service an I/O event?

THE UNIVERSITY OF NEW SOUTH WALES 50

CSE

## Interrupts

- An interruption of the normal sequence of execution
- A suspension of processing caused by an event external to that processing, and performed in such a way that the processing can be resumed.
- Improves processing efficiency
  - Allows the processor to execute other instructions while an I/O operation is in progress
  - Avoids unnecessary completion checking (polling)

THE UNIVERSITY OF NEW SOUTH WALES 51

CSE

## Interrupt Cycle

- Processor checks for interrupts
- If no interrupts, fetch the next instruction
- If an interrupt is pending, divert to the interrupt handler

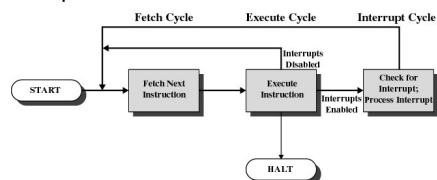


Figure 1.7. Instruction Cycle with Interrupts

THE UNIVERSITY OF NEW SOUTH WALES 52

CSE

## Classes of Interrupts

- Program *exceptions* (also called *synchronous interrupts*)
  - Arithmetic overflow
  - Division by zero
  - Executing an illegal/privileged instruction
  - Reference outside user's memory space.
- Asynchronous (external) events
  - Timer
  - I/O
  - Hardware or power failure

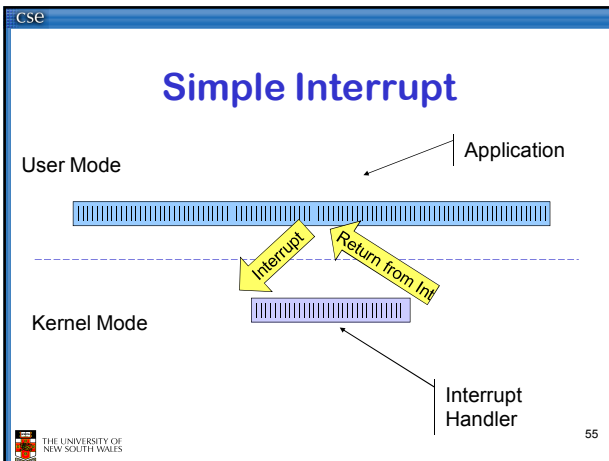
THE UNIVERSITY OF NEW SOUTH WALES 53

CSE

## Interrupt Handler

- A software routine that determines the nature of the interrupt and performs whatever actions are needed.
- Control is transferred to the handler by *hardware*.
- The handler is generally part of the operating system.

THE UNIVERSITY OF NEW SOUTH WALES 54



### Memory Hierarchy

- Going down the hierarchy
  - Decreasing cost per bit
  - Increasing capacity
  - Increasing access time
  - Decreasing frequency of access to the memory by the processor
    - Hopefully
    - Principle of locality!!!!

The pyramid diagram shows the memory hierarchy from top to bottom:
 

- Inboard Memory:** Registers, Cache, Main Memory
- Outboard Storage:** Magnetic Disk (CD-R/W, CD-R, DVD-R/W, DVD-R, M)
- Off-line Storage:** Magnetic Tape, MO, WORM

THE UNIVERSITY OF NEW SOUTH WALES **Figure 1.14. The Memory Hierarchy**

### Memory Hierarchy

- Rough (somewhat dated) approximation of 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

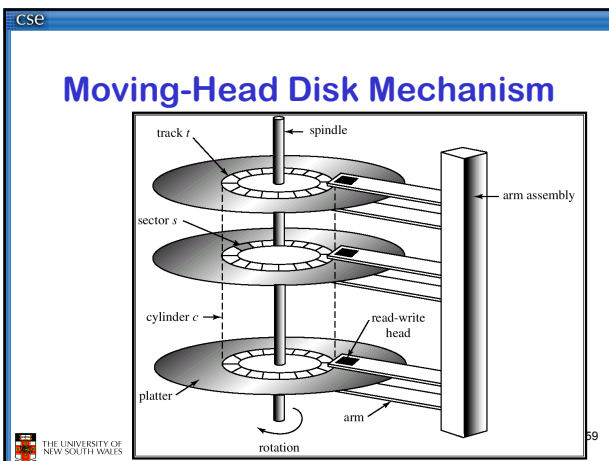
THE UNIVERSITY OF NEW SOUTH WALES 57

### Cache

The diagram shows a CPU with registers connected to a cache, which is connected to main memory. Word transfers occur between the CPU and cache, while block transfers occur between the cache and main memory.

- Cache is fast memory placed between the CPU and main memory
  - 1 to a few cycles access time compared to RAM access time of tens – hundreds of cycles
- Holds recently used data or instructions to save memory accesses.
- Matches slow RAM access time to CPU speed if high hit rate
- Is hardware maintained and (mostly) transparent to software
- Sizes range from few kB to several MB.
- Usually a hierarchy of caches (2–5 levels), on- and off-chip.
- Block transfers can achieve higher transfer bandwidth than single words.
  - Also assumes probability of using newly fetch data is higher than the probability of reuse ejected data.

THE UNIVERSITY OF NEW SOUTH WALES 58



### Example Disk Access Times

- Disk can read/write data relatively fast
  - 15,000 rpm drive - 80 MB/sec
  - 1 KB block is read in 12 microseconds
- Access time dominated by time to locate the head over data
  - Rotational latency
    - Half one rotation is 2 milliseconds
  - Seek time
    - Full inside to outside is 8 milliseconds
    - Track to track .5 milliseconds
- 2 milliseconds is 164KB in “lost bandwidth”

THE UNIVERSITY OF NEW SOUTH WALES 60

cse

## A Strategy: Avoid Waiting for Disk Access

- Keep a subset of the disk's data in memory
- ⇒ Main memory acts as a *cache* of disk contents

THE UNIVERSITY OF NEW SOUTH WALES 61

cse

## Two-level Memories and Hit Rates

- Given a two-level memory,
  - cache memory and main memory (RAM)
  - main memory and disk
- what is the effective access time?
- Answer: It depends on the hit rate in the first level.

THE UNIVERSITY OF NEW SOUTH WALES 62

cse

## Effective Access Time

$$T_{eff} = H \times T_1 + (1 - H) \times T_2$$

$T_1$  = access time of memory 1  
 $T_2$  = access time of memory 2  
 $H$  = hit rate in memory 1  
 $T_{eff}$  = effective access time of system

THE UNIVERSITY OF NEW SOUTH WALES 63

cse

## Example

- Cache memory access time 1ns
- Main memory access time 10ns
- Hit rate of 95%

$$T_{eff} = 0.95 \times 10^{-9} + (1 - 0.95) \times (10^{-9} + 10 \times 10^{-9}) = 1.5 \times 10^{-9}$$

THE UNIVERSITY OF NEW SOUTH WALES 64