Operating System Fundamentals

Robert Power & Robert Ford

Operating System Fundamentals
Robert Power & Robert Ford

School of Information Technology
College of the North Atlantic-Qatar

Table of Contents

Introduction 5

Unit 1: Computer Architecture Review

Why Review Computer Architecture? 6
A Map of Your Computer's Architecture 7
The CPU 10
Memory 11
Talking to Devices 12
Unit Summary 13
Key Terms 13
Review Questions 13

Unit 2: Operating System Principles

What is an Operating System? 14
Examples of Operating Systems 14
The Structure of Operating Systems 15
Interfacing with an Operating System 19
Managing System Resources 20
Unit Summary 23
Key Terms 23
Review Questions 24

Unit 3: Processes

Processes and Multitasking 25
Process 25
State Changing 28
Process Creation 32
Stopping a Process 34
Threads 35
Inter Process Communication 37
Process Scheduling Algorithms 47
Unit Summary 53
Key Terms 53
Review Questions 54

Unit 4: Memory Management

What is Memory Management? 55
The Memory Manager 55
Virtual Memory 58
Unit Summary 62
Key Terms 63
Review Questions 63

Unit 5: Input/Output

What are Input/Output Resources? 64
I/O Resources 65
I/O Management Software 68
Managing Magnetic Disks 70
System Clocks 74
Unit Summary 74
Key Terms 75
Review Questions 76

Unit 6: File Systems

Managing Files 77
Long Term Storage of Information 77
The File Manager 78
Data Storage Strategies 78
File Systems 81
The Master Boot Record and Partition Table 82
Interacting with File Systems 83
Unit Summary 85
Key Terms 86
Review Questions 87

Glossary 88

References 102

Introduction

This textbook is designed to give you an overview of what an operating system is, and how a modern operating system works. There are many different examples of operating systems available for computers. The most popular operating systems belong to the Microsoft Windows family (such as Windows 98, XP and Vista). Other examples are Unix and Linux, Mac OS X, and specialized operating systems for handheld devices like mobile phones.

An operating system is the software that controls (or operates) all of the parts of your computer. It manages all of your resources, and lets you interface with the computer. This textbook takes a look at the main problems that an operating system must be able to overcome, and the main functions that it must be able to perform. We begin by reviewing the architecture (or structure) of the physical parts of a computer, and how they communicate with each other. Then we take a look at fundamental operating system concepts, processes and process management, memory management, controlling input and output devices, and file system management.

Unit 1: Computer Architecture Review

Why Review Computer Architecture?

This textbook focuses on how operating systems work. *Operating systems* (or *system software*) are one of the two main types of software (the other is *application software*). However, we need to know some important things about the hardware inside of a computer in order to understand some of the critical functions of any operating system.

Computer architecture refers to the overall design of the physical parts of the computer. That is, it refers to:

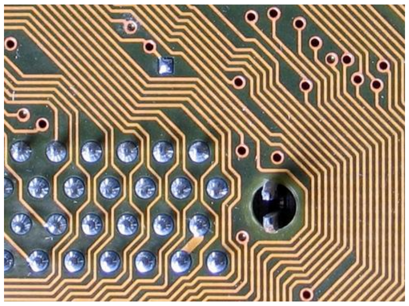
- what the main parts are;
- how they are physically connected to each other; and
- how they work together;

Although all of the parts of a computer are connected to each other by the *motherboard*, the operating system is essential in order to control how those parts talk to each other. Without the operating system, the parts of the computer would not be able to do anything that the user needs them to do! We need to know some basics about how the main parts of a computer are physically connected to each other before we can truly understand what an operating system does.

A motherboard

In this chapter we will look at a general map of a computer's architecture, showing the physical structure that allows all of the parts to exchange information and instructions. We will also take a brief look at some of the major components of a computer that an operating system must be concerned with, including the CPU, memory and how devices actually talk to each other.

A Map of Your Computer's Architecture


Lines (traces) on a motherboard are like roads in a city

You can think of the *motherboard* as a big city, and all of the parts of the computer as buildings throughout the city. Of course, there are roads to get between all of the buildings. These roads are those little metal lines (called *traces*) running all over the motherboard. These traces are part of what is called the *system bus*. There are several different busses on the motherboard, depending upon which devices they are connecting.

Some of the major components of your computer's architecture (the main buildings) that are controlled by the operating system include:

CPU – Central Processing Unit

- This is the brain of your computer. It performs all of the calculations.

RAM – Random Access Memory

- This is your system memory.
- This is like a desk, or a workspace, where your computer temporarily stores all of the information (data) and instructions (software or program code) that it is currently using.
- Most computers today have between 1 to 4 Gigabytes (GB) of RAM.

Graphics

- Many computers have a dedicated system bus and expansion card slot just for a video card.
- Many video cards include their own memory so that you do not need to use up all of the RAM to run your monitor.

I/O Busses

- Special busses (roads) connecting all of your input/output devices to your motherboard.
- The three main types of I/O busses are ISA, PCI and USB.

- ISA – Industry Standard Architecture
  - This was the industry standard in the 1980s and early 1990s.
  - It is now used to provide support for older and slower devices.
  - Common devices connected to the ISA bus might include an older modem, a joystick, a mouse, or a printer (using the older, wide-style printer port).

- PCI – Peripheral Component Interconnect
  - This is for newer and faster devices than ISA.
  - You can think of this like a wider road, with a faster speed limit!
  - Some common devices connected to the PCI bus include your network card, EIDE devices (hard disk, CD/DVD drive, etc).

- USB – Universal Serial Bus
  - Many new devices can connect to your computer using a USB port.
  - Examples include webcams, MP3 players, printers, PDAs, etc.

Figure 1.1 (below) is a diagram of the architecture of these main components (how they are all connected)

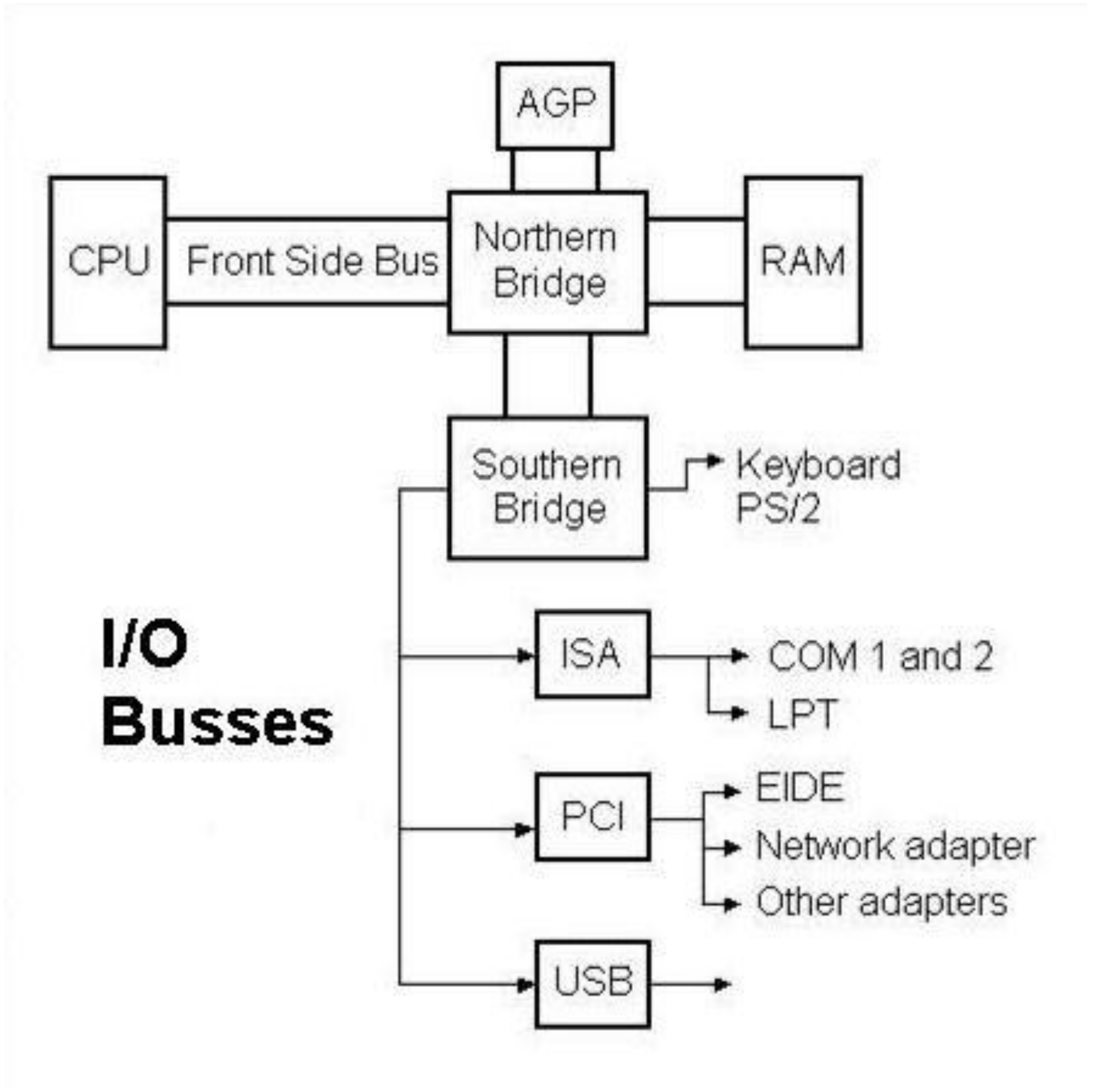


Figure 1.1
Diagram of system bus architecture

Busses and Bridges

In Figure 1.1 we see that the major components are all connected by different *busses*. The *Front Side Bus* provides the main connection between the *CPU*, *RAM*, the *graphics* card (*AGP – Accelerated Graphics Port*), and the *Northern* and *Southern Bridges*. The Front Side Bus looks wider than the *I/O busses* because it is wider and faster. It contains more wires (traces) for the transmission of data between the devices. In the comparison to a city, the Front Side Bus is like a major freeway with a fast speed limit. The smaller I/O busses are like smaller side streets. Some of the I/O busses are narrower and slower than others.

The two bridges in Figure 1.1 perform the same function inside your computer that would be performed by bridges or roundabouts in a city. They are major intersections where data from different devices cross paths. Of course, like any bridge or roundabout, there needs to be traffic laws to govern who goes first. If there were no rules (and no police to enforce the rules) then everyone would crash together. In computer terms, your data would become corrupted, and no information would ever reach its destination.

Figure 1.2 (below) compares the system bus architecture to a series of city streets with roundabouts:

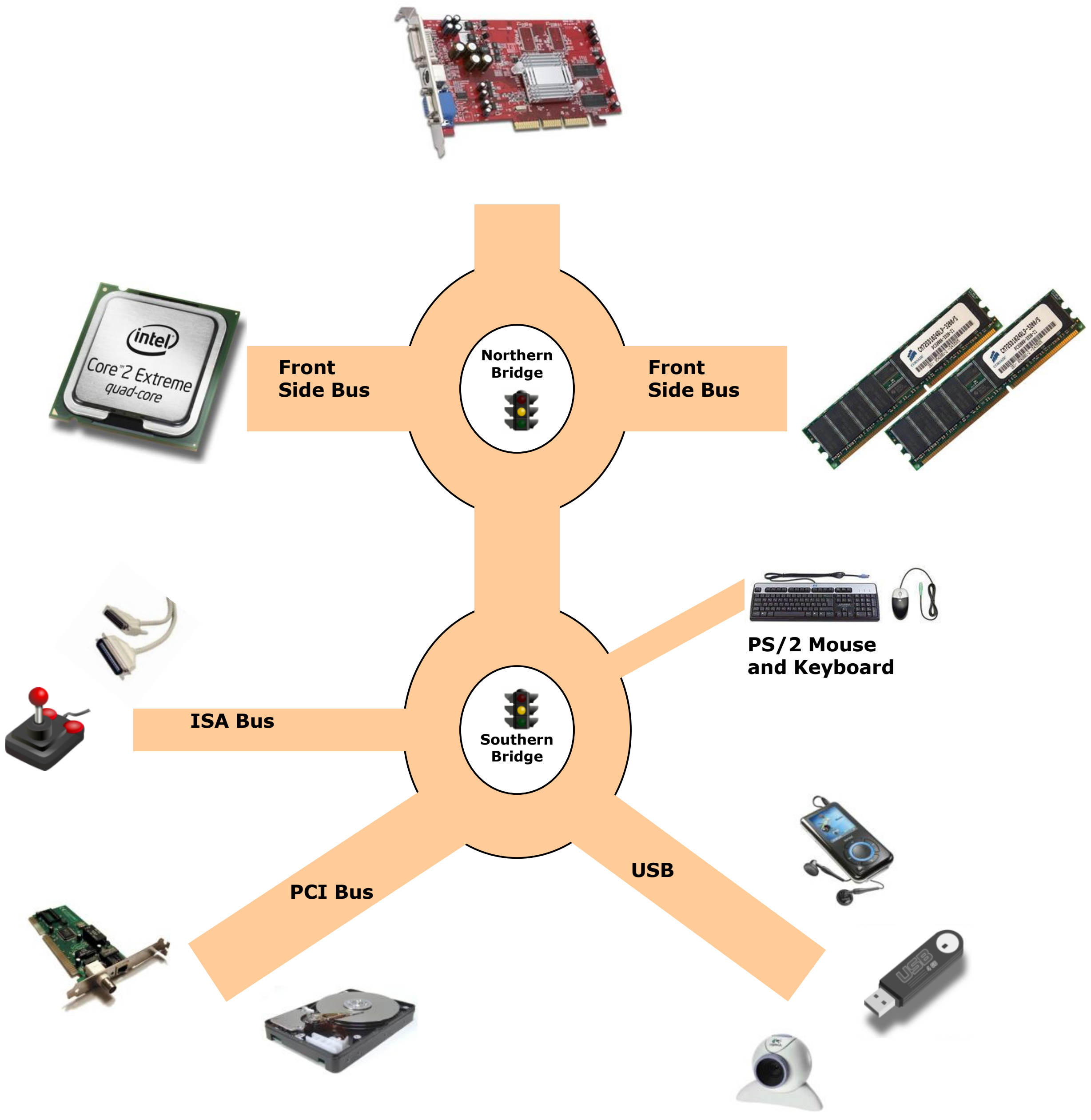


Figure 1.2
A different view of system bus architecture

The Central Processing Unit (CPU)

The *CPU* is the brain of your computer. It performs all of the calculations. Of course, in order to do its job, the CPU needs commands to perform, and data to work with. These instructions and data travel to and from the CPU on the system bus. The operating system provides rules for how that information gets back and forth, and how it will be used by the CPU.

Inside the CPU

Inside the CPU there are many important parts:

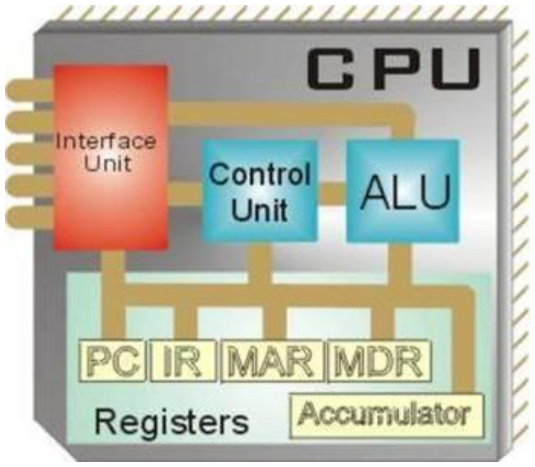


Figure 1.3
The parts inside a typical CPU

- The *Arithmetic Logic Unit* (*ALU*), which performs the calculations
- The *Control Unit*, which controls the flow of data inside the CPU
- The *Interface Unit*, or the *I/O Unit*, which acts like a gate for information entering and leaving the CPU
- *Registers*, which temporarily hold data and instructions waiting to be used
- The *Program Counter* (*PC Register*), which is a special register holding the address of the next instruction the CPU needs from the RAM

The Fetch—Decode—Execute Cycle

The CPU finds, interprets, and executes program code using a specific cycle, as follows:

1. The CPU looks in the PC register for the location of the next program instruction.
2. The CPU retrieves the next instruction from RAM, and places it in a register.
3. The CPU changes the PC register with the address in RAM for the next instruction.
4. The CPU performs the first instruction, and repeats the cycle until the power is lost.

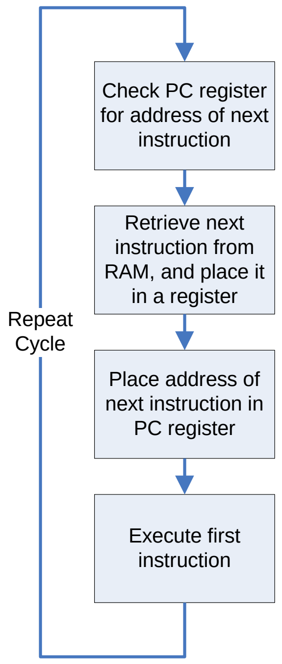
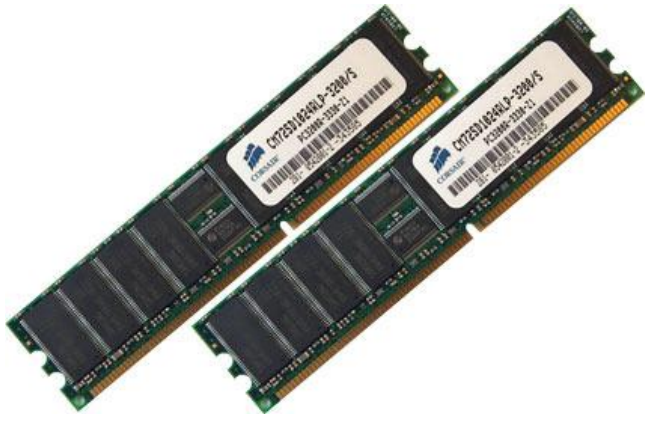

Check PC register for address of next instruction
Retrieve next instruction from RAM, and place it in a register
Place address of next instruction in PC register
Execute first instruction
Repeat Cycle

Figure 1.4
The Fetch—Decode—Execute Cycle

Memory

Memory is stored on Radom Access Memory (RAM) chips. A typical computer today has between 1 Gigabyte (GB) and 4 GB of RAM.



Memory is used to:

- Store data
- Store commands (instructions)
- Store system settings

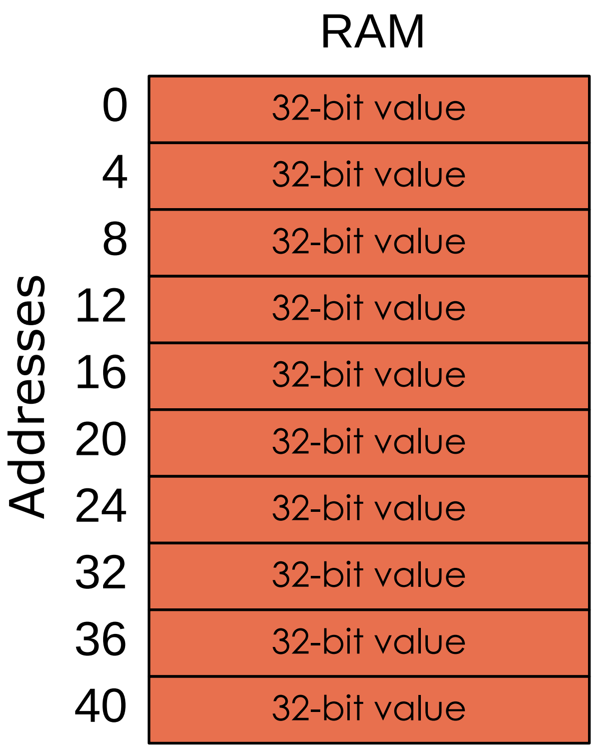


Figure 1.5
Address structure of RAM

Each RAM chip contains millions of address spaces. Each address space is the same size, and has its own unique identifying number (address). The operating system provides the rules for using these memory spaces, and controls storage and retrieval of information from RAM.

Fast fact – RAM is a device!

Without an operating system, a computer would not be able to use RAM chips. This is because your computer treats the RAM chips like a device that has been installed (just like a webcam, or a printer). When your computer first starts up, it can only use a small amount of RAM memory (1 Megabyte (MB)) that is built into the motherboard. Device drivers for RAM chips are included with the operating system, and must be loaded as part of the boot process in order for the RAM to work!

Problem: *If RAM needs an operating system to work, and an operating system needs RAM in order to work, how does your computer activate its RAM to load the operating system?*

Talking to Devices

Devices talk to each other and to the CPU. They need to communicate in order to share information, and in order to be told what to do! There are two types of devices that are controlled by information from the CPU:

- Programmed devices, and
- Interrupt-driven devices

Programmed Input/Output Devices

Programmed I/O devices need to be completely controlled by the CPU. That means the CPU must stop whatever task it is doing, and focus on the device until it has finished whatever it has been told to do. This wastes a lot of processing time!

Interrupt-Driven Devices

A more efficient way to control devices is by using an *interrupt controller*. The interrupt controlled keeps track of whichever devices need to talk to the CPU, and gives different priority to different devices. For example, the keyboard gets higher priority than a modem. When a device needs new instructions, or when it has finished a task, the interrupt controller issues an interrupt to the CPU (like raising your hand in class). The CPU stops whatever it is doing long enough to talk to the device. Although this is more difficult to program, it results in better computer performance.

Of course, the operating system provides all of the rules for communicating with both programmed and *interrupt-driven devices.*

Direct Memory Access

Sometimes devices may want to talk to each other without 'going through' the CPU. The *DMA Controller* controls access to the system bus, and RAM, and bypasses the CPU. The CPU does not need to get involved in the process, other than to set up the transfer. The CPU will get an interrupt when the transfer is complete.

Direct Memory Access is like adding police officers to a roundabout who will let traffic go through to other streets when the road is clear.

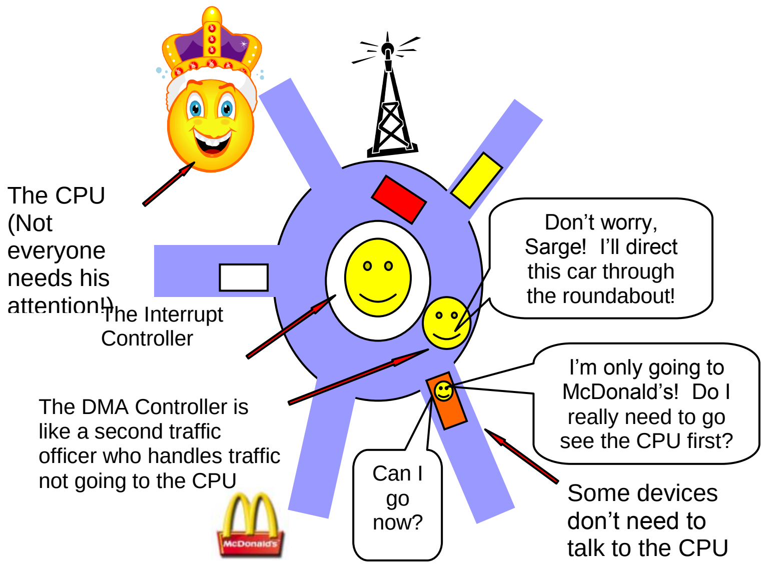

The CPU (Not everyone needs his attention!)
The Interrupt Controller
Don't worry, Sarge! I'll direct this car through the roundabout!
I'm only going to McDonald's! Do I really need to go see the CPU first?
The DMA Controller is like a second traffic officer who handles traffic not going to the CPU
McDonald's
Can I go now?
Some devices don't need to talk to the CPU

Figure 1.6
DMA is like an extra police office who guides cars through a busy intersection without bothering anyone back at the police station first

Unit Summary

A computer is a system of devices that are all connected together, just like buildings throughout a city. All of these devices are connected through the motherboard. A system of wires, called traces, provides a means for information to be exchanged between all of the devices. These wires are called busses, and they are like the roads throughout a city. Just like in a city, there must be traffic laws and police officers to enforce the flow of traffic, or it will all crash together and become useless. The rules for data transfer, and the control of devices installed in a computer, are provided and enforced by the operating system.

Key Terms

Address
AGP
ALU
Application software
Busses
Computer architecture
Control Unit (CU)
CPU
DMA
DMA Controller
Fetch—Decode—Execute cycle
Front Side Bus
Graphics
I/O Busses
Instructions
Interface (I/O) Unit
Interrupt
Interrupt Controller
Interrupt-driven I/O devices
ISA
Memory
Motherboard
Northern Bridge
Operating System
PCI
Program Counter
Programmed I/O devices
RAM
Registers
Southern Bridge
System bus
Traces
USB

Review Questions

1. Describe the difference between pre-emptive multitasking and cooperative multitasking.
2. What are registers?
3. List and briefly describe any three busses that are used to transport data.
4. Briefly describe the Fetch—Decode—Execute cycle.
5. Describe the difference between programmed I/O devices and interrupt-driven I/O devices.
6. What is the benefit of DMA?

Unit 2: Operating System Fundamentals

What is an Operating System?

You need two types of software in order to use your computer (or any other computerized device). These are *applications* and *system software*. Applications are the programs you use to do tasks, such as write a document, surf the web, or play games. System software runs the computer system for you. Another name for system software is an operating system. There are many different operating systems, but they all have a similar architecture (or structure). That is because they must all overcome the same problems and perform the same basic functions. An operating system must be able to:

- Manage system resources
    - CPU scheduling
    - Process management
    - Memory management
    - Input/Output device management
    - Storage device management (hard disks, CD/DVD drives, etc)
    - File System Management

- Simplify the development and use of applications

Examples of Operating Systems

A number of operating systems are available for personal computers. The most popular is Microsoft Windows, which is the operating system used on over ninety percent of the world's personal computer systems. Another popular operating system is *Mac OS X*, which is the operating system used for Apple Macintosh computers (like the Mac Book Pro laptop series). While IMB PCs (mostly Windows) and Mac computers are not directly compatible, it is possible to use virtualization to run one operating system on an incompatible computer.

Another group of widely used operating systems is based on *UNIX*. UNIX was a command line interface operating system developed for large scale computers and networks in the 1960s. The latest generation of operating systems derived from UNIX is called *Linux*. It is a free, open-source operating system that is supported by most computer platforms.

Special Purpose Operating Systems

Operating systems are not limited to just personal computers. Most electronic devices today use an operating system to manage their physical components and to make it easier to develop applications for use on the devices. Examples include the *Symbian, Blackberry, Palm* and *Windows Mobile* operating systems used for personal digital assistants (*PDA*s) and mobile phones. *Specialized operating systems* have even been developed to control computerized aircraft systems (*VxWorks, pSOS* and *QNX* are examples).

The Structure of Operating Systems

Layers

Accessing computer resources is divided into *layers*. The user represents one layer at one end of the system. Your computer's hardware represents the layer at the opposite end of the system. In order to use your hardware to do anything with the computer, you need software. Software forms the layers in between the user and the hardware and is divided up into application software and the operating system. The operating system must be able to manage resources from both the applications and hardware layers.

In the computer layer system the user interacts directly with software applications. The applications interact with both the user and the operating system. The operating system interacts with the applications and controls the hardware.

Each layer is isolated and only interacts directly with the layer below or above it. If you make changes to any one layer, they only directly affect the layer next to it. For example, if you install a new hardware device you do not need to change anything about the user or applications. However, you do need to make changes to the operating system. You need to install the device drivers that the operating system will use to control the new device. If you install a new software application you do not need to make any changes to your hardware. But you do need to make sure the application is supported by the operating system and the user will need to learn how to use the new application. If you change the operating system you need to make sure that both your applications and your hardware will work with the new operating system.

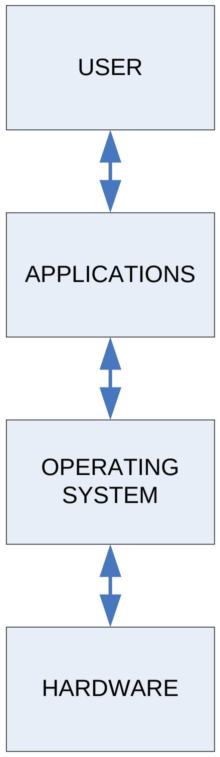


Figure 2.1
Layers in a computer system

Running Multiple Operating Systems

It is possible to install more than one operating system on a computer. You can do this by partitioning your hard disk(s) and installing different operating systems on different *partitions*. This can be very useful, because you may want to use different operating systems to perform different tasks. For example, you may have specialized applications that will only work with one operating system, making them incompatible with the rest of your software. When you turn your computer on, you are given a choice of which operating system to use. You can only run one operating system at a time. Figure 2.2 (right) shows the system of layers when multiple operating systems are installed on the same computer.

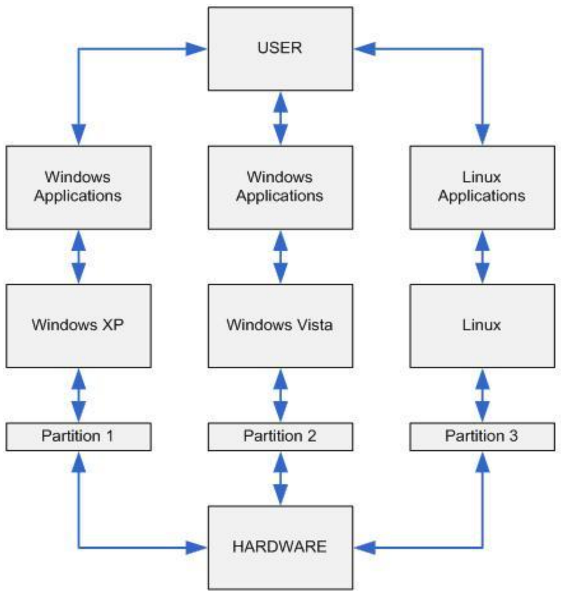


Figure 2.2
Layers in a computer with multiple partitions and operating systems

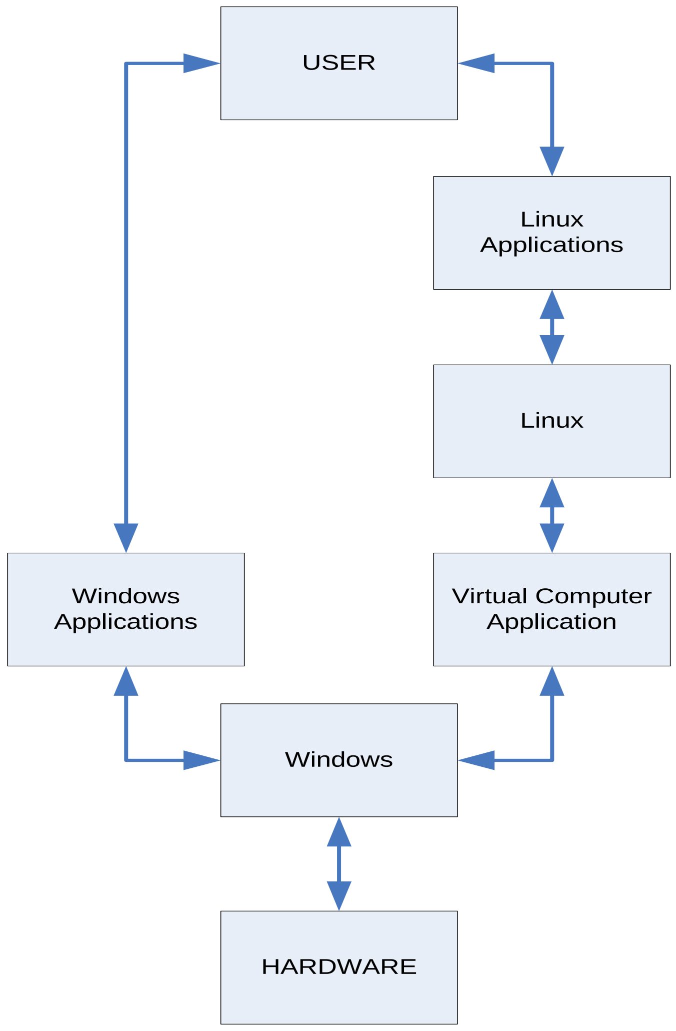


Figure 2.3
Layers with a virtual operating system

Running a Virtual Operating System

What happens if you want to work on applications in two operating systems at the same time? What about if you want to run an operating system that is not compatible with your computer's hardware? (For example, you cannot install the Mac OS X operating system on an IBM compatible PC). You can get around these problems by running a virtual computer.

A *virtual computer* is really an application within one operating system that lets you pretend you have a different operating system installed. Virtual computer applications like VMWare and Virtual PC act as translators. They convert instructions from the virtual operating system into instructions from the real operating system, which then controls your computer's hardware.

Figure 2.3 (left) shows the structure of layers when you run a virtual operating system within a Windows operating system. As far as Windows is concerned, it is simply running another application. Notice that the layers between the virtual computer application and the user are just like the layers for a single operating system (Figure 2.1).

Operating System Modes

A typical operating system has two *modes* of operation. These are like layers of operation within the operating system layer (Figure 2.1). The *User Mode* is concerned with the actual interface between the user and the system. It controls things like running applications and accessing files. The *Kernel Mode* is concerned with everything running in the background. It controls things like accessing system resources, controlling hardware functions and processing program instructions. The *Kernel* forms the core of the operating system, and it acts like a supervisor for everything that is happening in the computer. In the *client-server model* of an operating system, the User Mode is considered a client. That is, the User Mode accesses resources provided by the Kernel (the server). Figure 2.4 (below) shows what operating system functions are controlled by the User Mode and Kernel Mode.

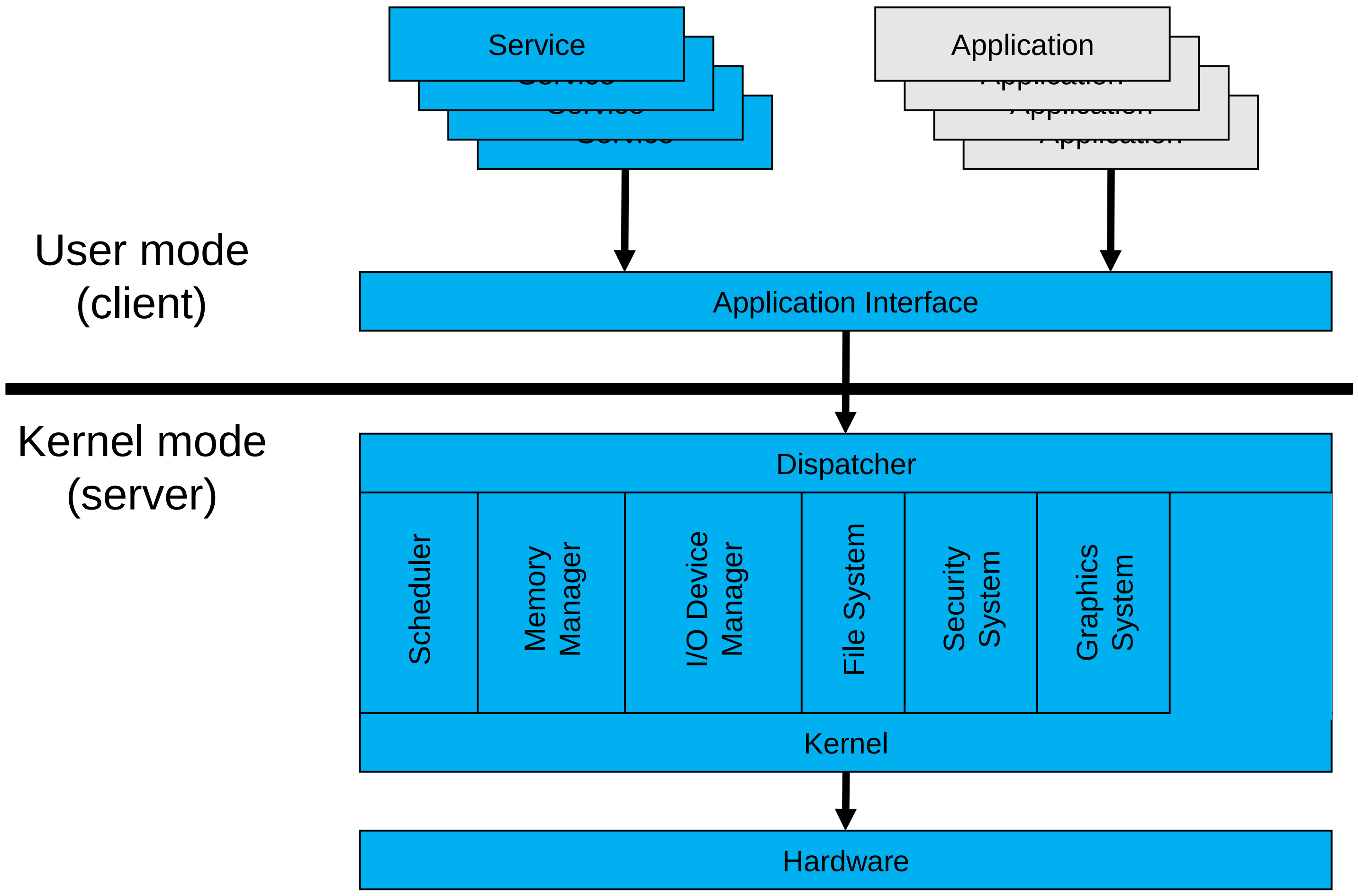


Figure 2.4
Typical structure in the Client (User Mode) – Server (Kernel Mode) model of an operating system

Starting an Operating System

Most personal computers have similar architecture and can use a variety of different operating systems. When a computer is first made, there is no operating system installed. Even after you have an operating system installed, you can remove it and install a different one. As we discussed earlier, you can even have multiple operating systems installed on the same personal computer. This raises the question—how does your computer start the operating system? If you have more than one operating system installed, how does your computer choose which operating system to use?

Your computer is designed to start in stages. In the first stage, you turn on the power supply to your computer. This sends electricity to the motherboard on a wire called the '*Voltage Good*' line. If the power supply is good, then the *BIOS* (*Basic Input/Output System*) chip takes over. At this stage the computer's CPU is operating in *Real Mode* (or real address mode), which means that it is only capable of using approximately 1 MB of memory built into the motherboard. RAM will be initialized later using device drivers from the operating system.

The BIOS chip contains basic instructions for starting up the rest of the computer system. The first thing that it will do is a *Power-On Self Test* (*POST*), which will check to make sure all your hardware is working properly. If the hardware is all working, BIOS will then look for a small sector at the very beginning of your primary hard disk called the *Master Boot Record* (MBR). The MBR contains a list, or map, of all of the partitions on your computer's hard disk (or disks). After the MBR is found the *Bootstrap Loader* follows basic instructions for starting up the rest of the computer, including the operating system. If multiple operating systems are installed, the user will be given a choice of which operating system to use.

Remember – RAM is a device!

In the first unit we said that without an operating system a computer would not be able to use RAM chips. Your computer treats RAM chips like a device that has been installed. When your computer first starts up, it can only use a small amount of RAM memory (1 MB) that is built into the motherboard. Device drivers for RAM chips are included with the operating system, and must be loaded as part of the boot process in order for the RAM to work!

Problem: *If RAM needs an operating system to work, and an operating system needs RAM in order to work, how does your computer activate its RAM to load the operating system?*

Solution: *Device drivers for RAM are loaded during the Early Kernel Initialization stage.*

The next stage is called *Early Kernel Initialization*. Remember that the Kernel is the core of the operating system, and it regulates all of the background functions of your computer. In the Early Kernel Initialization stage, a smaller core of the Kernel is activated. This core includes the device drivers needed to use your computer's RAM chips. Without the extra memory provided by RAM, it is not possible to run the more complicated code for the remainder of the operating system.

Once the Early Kernel Initialization is complete, the CPU switches to *Protected Mode*. The computer can now take advantage of the extended memory address system provided by RAM, and the operating system's Kernel is fully initialized. Only at this stage are the first User Mode processes initialized, and the user can begin interacting with the operating system, applications and hardware. Figure 2.5 (below) shows the stages in starting an operating system.

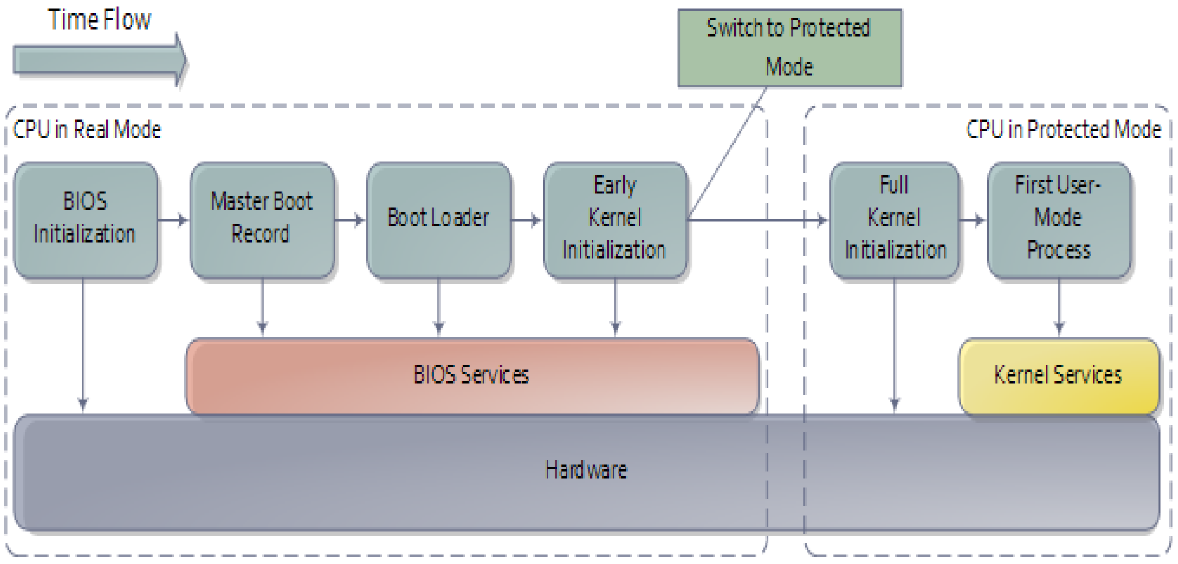


Figure 2.5
Stages in the startup of an operating system

Interfacing with an Operating System

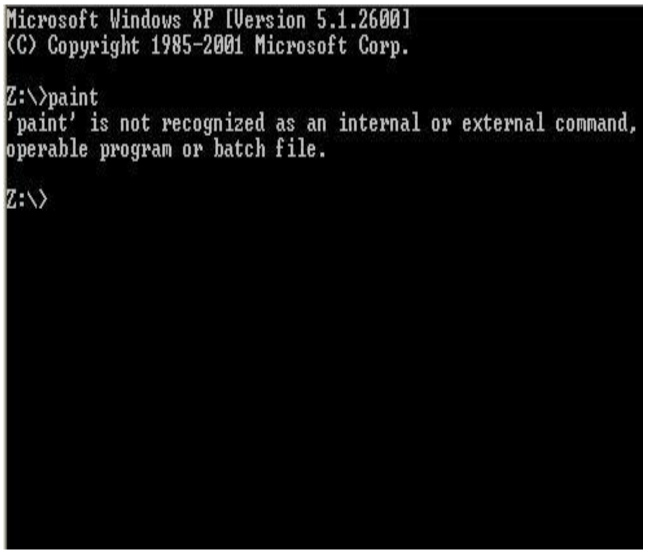
Types of User Interfaces

An operating system operates the functions of a computer. It also provides a way for users to interface with, or access, a computer's applications, resources and hardware. There are two main types of user interfaces for an operating system:

- Command Line Interface
- Graphical User Interface (GUI)

A *command line interface* uses typed commands to issue instructions to the computer. It can be more difficult to use because the user must type the precise commands and locations of files. *DOS* (*Disk Operating System*) and UNIX are examples of command line interface operating systems.

A *GUI* uses graphics (or pictures) and menus to help the user access resources and issue commands. Windows XP, Linux and Mac OS X are examples of GUI operating systems.



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

Z:\>paint
'paint' is not recognized as an internal or external command,
operable program or batch file.

Z:\>
```

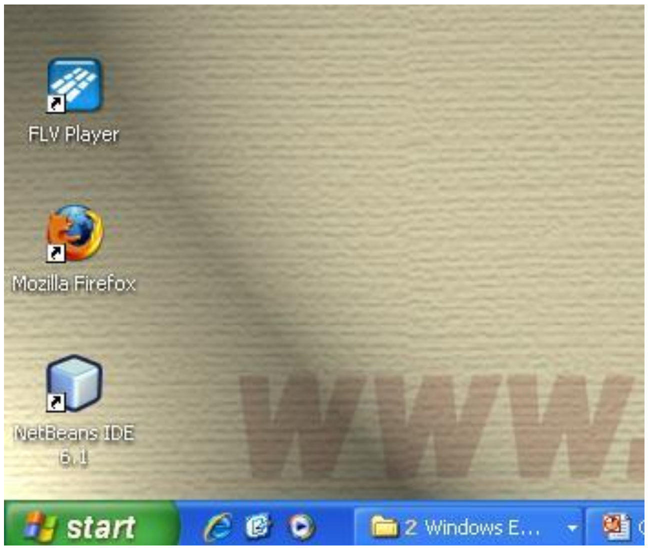


Figure 2.6
Examples of a command line and GUI interface

The Command Line Interpreter

Applications are accessed at the User Mode level. This means that they do not have the authority to directly access system resources that are controlled at the Kernel Mode level. When a user types a command (in a command line interface) or performs a task within an application (using a GUI), *processes* are initiated. Since those processes usually require access to system resources, the command line interpreter converts them into system actions (called *system calls*). Most interpreters execute applications to perform the system calls.