

**Computer Architecture
Using a Thorough, Concise,
Step-by-Step Approach:
A Course for Newcomers to the
Information Security Field**

By

**Jayme M. Speva
Lewis University**

September 4, 2006

(This page is intentionally blank)

Table of Contents:

	Introduction	4
Chapter 1	History of Computers	5
Chapter 2	Data Representation and Numbering Systems	20
Chapter 3	Computer Hardware	32
Chapter 4	Operating System Concepts	41
Chapter 5	Digital Electronics	49
Chapter 6	Disk Operating System	55
Chapter 7	Linux	63
Chapter 8	Security Topics	72
Chapter 9	Conclusion	77

Introduction

The Master of Science in Information Security (MSInfoSec) at Lewis University is a cooperative program developed between the Management Information Systems Department and the Mathematics and Computer Science Department. The MSInfoSec program consists of two foundation courses, six core courses, a capstone experience, two one-hour seminars, and four remaining courses in one of two concentrations: the managerial concentration and the technical concentration.

The nature of the MSInfoSec program appeals to individuals with diverse educational backgrounds and work experiences. Likewise, it is a diverse subject that draws from many different disciplines or skill sets, such as investigations, programming, networking, managerial, and technical backgrounds. The foundation courses lay the groundwork for future study in the major by presenting theories of computer hardware and software operations. The purpose of this paper is to provide a textbook for the foundation course Computer Organization (68-500). The problem with the current choices of text is that they are either too general of an introduction to data processing, or too technical. In fact, to achieve the goal of this foundation course, an instructor could easily use multiple books to disseminate the required skill set. This work provides a single text which the instructor may use to present all the concepts that are relevant to an understanding of information security.

The text that is currently used is William Stallings *Computer Organization and Architecture*. Although it is an excellent text for someone with a computer or electronics background, it is not a beginner's text, designed for a person with no computer background or prior work experience. Therefore, the goal of this paper is to fill the void and bridge the gap somewhere between an introductory to advanced technical text. This text is organized in a manner that a person with no prior computer experience can easily understand. Topics are organized using a high-altitude overview to convey fundamental concepts, before each topic is discussed in depth. The topics are also organized so that they build upon one another for a better understanding. This paper's goal is to provide every student in the MSInfoSec program with a required skill set that will ensure success in the program, and a foundation upon which the remaining courses in the major can build.

Chapter 1: History of Computers

Man's fascination has always driven people to look for ways to invent the better mousetrap. As far as computers are concerned, most introduction to data processing texts include such innovations as the abacus, Pascal's calculating machine (1642), and Charles Babbage's Difference and Analytical Engines (1822) as some of the most important developments in history. However, there have been many innovations over hundreds of years which have contributed to the development of calculating and thinking machines. In the history of the world, there are few other inventions that have had as great an impact as the computer. If you compare the pace of computer innovation with any other field you find that the pace of innovation and improvement in computing surpasses most other fields. It is truly amazing that a one-thousand dollar server that can now sit on a desk has the same computing power as a massive mainframe that cost a million dollars 30 years ago. If the automotive industry had made the same strides in price performance as the computer, cars would only cost a few dollars. The goal of this chapter is to highlight and explore the major computer inventions and developments during the past 100 years. It will conclude with a summary of the major events in the history of computer security during the past 25 years.

Most new technology has been introduced to address a major historical event or problem. For example, the 1890 Government census required a much quicker means of counting and tabulating, and meeting this need ushered in the computer revolution. In 1790, it took the United States Census Bureau over eight months to complete the first census. By 1860, the population increased almost tenfold since 1790, from 3.8 million to 31.8 million. [1] In 1887, the Census Bureau completed the eleventh census seven years after it began. The processes used to manually tabulate the data took so long as the population increased that by the time the data was collected and tallied it was outdated already. The Census Bureau needed some form of new technology to tabulate census data in a more timely fashion.

The Census Bureau's solution was to have a competition to find a new method to tabulate the census. A man by the name of Herman Hollerith entered and won this competition. With his victory, not only did Hollerith make it possible to complete the census in a reasonable time frame, but the methods that he invented became the foundation of data processing today. Herman Hollerith is now known as the father of information processing. Hollerith's contributions to the computer age were numerous. Even though he did not invent the computer, his ideas were revolutionary and his influence is still seen today in modern machines.

In 1882, Hollerith was on the staff of Mechanical Engineering at MIT. He was inspired by the punch card system that railroad conductors used on trains. In this system conductors would punch holes into passenger tickets to identify the passenger status. Likewise, Hollerith's prototype consisted of a paper card that used combinations of round punched holes to record census statistical data. This invention was the first known advancement in storage media. This tabulating card was modeled around 1887 US paper

currency as shown in Figure 1. Following the tabulating card, Hollerith also invented a series of mechanized machines that could read and sense the presence or absence of holes in the punched cards [2]. This was made possible by a small wire that passed through the holes into a cup of mercury beneath the card completing an electrical circuit. This device triggered mechanical counters and sorter bins that were able to tabulate the appropriate data.

1	2	3	4	CM	UM	Jp	Ch	Oc	In	20	50	80	Dv	Un	3	4	3	4	A	E	L	a	g
5	6	7	8	CL	UL	O	Mi	Qd	Mo	25	55	85	Wd	CY	1	2	1	2	B	F	M	b	h
1	2	3	4	CS	US	Mb	B	M	0	30	60	0	2	Mr	0	15	0	15	C	G	N	c	i
5	6	7	8	No	Hd	Wf	W	F	5	35	65	1	3	Sg	5	10	5	10	D	H	O	d	k
1	2	3	4	Fh	Ff	Fm	7	1	10	40	70	90	4	0	1	3	0	2	St	I	P	e	l
5	6	7	8	Hh	Hf	Hm	8	2	15	45	75	95	100	Un	2	4	1	3	4	K	Un	f	m
1	2	3	4	X	Un	Ft	9	3	i	c	X	R	L	E	A	6	0	US	Ir	Sc	US	Ir	Sc
5	6	7	8	Ot	En	Mt	10	4	k	d	Y	S	M	F	B	10	1	Gr	En	Wa	Gr	En	Wa
1	2	3	4	W	R	OK	11	5	l	e	Z	T	N	G	C	15	2	Sv	FC	EC	Sv	FC	EC
5	6	7	8	7	4	1	12	6	m	f	NG	U	O	H	D	Un	3	Nw	Bo	Hu	Nw	Bo	Hu
1	2	3	4	8	5	2	Oc	0	n	g	a	V	P	I	AL	Na	4	Dk	Fr	It	Dk	Fr	It
5	6	7	8	9	6	3	0	p	o	h	b	W	Q	X	Un	Pa	5	Ru	Ot	Un	Ru	Ot	Un

Figure 1. 1890 Census punch card [3]



Figure 2. The 1890 Hollerith tabulator & sorter box [4]

The Hollerith 1890 Tabulator and Sorter Box was an electrical tabulating machine with a large number of clock-like counters that accumulated the results. By means of switches, operators could instruct the machine to examine each card for certain characteristics, such as profession, marital status, number of children, etc. Hollerith's apparatus for compiling statistics eventually evolved into a complete system of machines. This machine greatly automated the 1890 census but it was a pre-programmed machine, meaning it was hardwired and could only be used for census data and nothing else. Furthermore, the machine could only count.

During his technological reign, Hollerith not only refined and improved his original 1890 machine, but he built a company around it. In 1896, Hollerith founded the Tabulating Machine Company [5]. During this time, he invented many other models that

automated his processes more, including a keypunch machine to punch the holes in the cards that operated from a keyboard. These included automatic card-feeding mechanisms, sorters, and tabulators that could do addition and not just count. In 1906, he invented the Type I Tabulator. This machine took the first steps toward programming concepts. It introduced a wiring panel that could program the machine, allowing one machine to do different jobs, as shown in Figure 3. This was a major advancement over the 1890 machine that could only be used for the census. Herman Hollerith's work laid the foundation of the modern information processing industry.

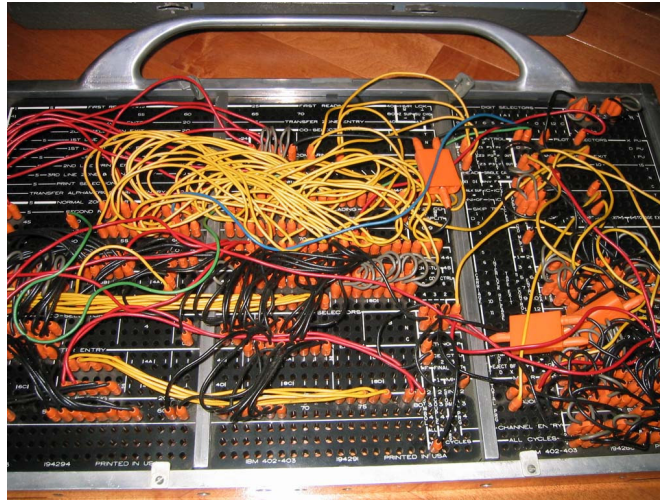


Figure 3. Hardwired plug board [5]

In 1911, Hollerith's Tabulating Machine Company merged with two other companies, the Computing Scale Corporation and the International Time Recording Company. The latter was renamed to Computer Tabulating Recording Company (CTR). CTR manufactured many products including employee time keeping systems, clocks, weighing scales, meat slicers, and punch card equipment. As the business evolved, the company focused mainly on punch cards and punch card equipment. In 1924, CTR changed its name to International Business Machines Corporation, otherwise known as IBM [6].

At the onset of its founding, IBM took "Hollerith machines" to the next level. The punch card era was a large growth period because many companies were racing to capitalize on this new frontier. In the early 1900's Hollerith cards used round holes that supported 45 columns, where each column represented a single character or data value. In 1924, Remington Rand Corporation invented a technique to double the capacity of the card but they never capitalized on this. In 1929, IBM perfected the rectangular hole and standardized the 80 column card. Many other formats appeared sporadically but the 80 column format remained the standard for a whole century.

The IBM punch card was in the center of data processing. There was a whole suite of machines used to perform different tasks: Tabulators, Sorters, Key Punches, Calculators, Interpreters, Reproducers, Collators, and Interpreters. These early machines were big, slow mechanical devices consisting mostly of motors, gears, and relays. As

time progressed they improved and became faster with more functionality. Many of these vintage machines were outdated but were used into the 1980s and some even until the year 2000.

In 1939, Dr. Jan Atanasoff, a physics professor with graduate student Clifford Berry, designed and built the first digital computer, the Atanasoff-Berry-Computer (ABC). This invention started the transition into electronic digital computers. Atanasoff received a research grant of \$650 to build a prototype of a computer he had designed for solving linear equations [7]. The ABC computer was approximately the size of a large desk, and used approximately 270 vacuum tubes. This digital approach was a significant technical advancement over the mechanical, electromechanical, or analog based machines of this time. This prototype had many limitations; it was slow, not programmable, and could not do general purpose computing. Its only purpose was to calculate linear equations. Another noticeable design characteristic of the early computers is that they were an enclosed system, meaning the main processor did all the work. It not only did calculation but it interfaced and controlled everything else on the computer. This simply meant that a processor that was very fast was slowed down waiting on much slower input output I/O devices. This limitation was improved upon in later machines.

During World War II British scientist Alan Turing designed the Colossus computer, a programmable electronic digital computer that was used to decrypt encoded German messages. The Colossus was used as early as February 1944, and nine improved versions of the machine were made over time. Due to the war efforts this computer's existence was kept secret until the 1970s. At this time in history, WWII was the catalyst that drove the growth in the computer industry. From breaking encryption codes to calculating ballistic trajectory tables faster, these needs called for quicker ways to perform mathematical calculation.

In 1945, Dr. John von Neumann wrote a paper describing the stored program concept. His breakthrough idea was that memory holds both data and stored programs. This stored-program concept was so completely revolutionary to computer architecture that every computer made from this point on should be called a "von Neumann machine". His idea was one of self-replicating, eliminating the lengthy job set-up process. The stored program model simply meant that to run a payroll process you load a payroll program into memory and execute the instructions from memory. The computer essentially replicated itself into a payroll machine. By loading a different program, the machine would again replicate itself to that particular task. In 1946, von Neumann and his colleagues began designing a new stored program computer known as the IAS computer at Princeton Institute for Advanced Studies. The IAS was completed in 1952 and was the prototype for all modern computers. The von Neumann IAS computer had twenty-one hardware machine instructions. Using this machine's instructions a programmer could write a program that the hardware could understand and execute. This machine language code was very primitive, but at the time it was light-years ahead of hardwiring panel boards.

On December 22 in 1947, William Shockley, John Bardeen, and Walter Brattain invented the transistor at Bell Labs. The transistor started the second generation of computer architecture, as it replaced the power hungry vacuum tubes found in earlier models. Transistors mounted on printed circuit boards were much smaller in size,

demanded less power, and increased reliability. Second generation computers were still expensive due to design costs and were primarily used by universities, large corporations, and government installations. Few could afford to purchase these expensive machines, so most were leased from the manufacturer. Second generation computers were prevalent in the 1950's into the 1960's.

In the northeast, the ENIAC (Electronic Numerical Integrator And Computer) was conceived and designed by Dr. John William Mauchly and J. Presper Eckert of the University of Pennsylvania. It was completed in 1946 at a cost of \$486,000. The ENIAC was one of the first large scale, general purpose, electronic, digital computers we know today. Furthermore it was the first machine to use binary math. While earlier computers had been built with some of these properties, the ENIAC was capable of being reprogrammed to solve a full range of computing problems. ENIAC was also originally designed to calculate ballistic equations for the military, and was used in the design of the hydrogen bomb. The ENIAC was huge, measuring over 100 feet long, it encompassed a room 30 feet by 50 feet, and weighed 30 tons. It used 17,468 vacuum tubes, 7,200 crystal diodes, 1,500 relays, 70,000 resistors, 10,000 capacitors, approximately 5 million solder joints [8]. Programming the ENIAC consisted of physically hardwiring into plug boards using jumper wires. The first time the ENIAC was powered on, lights dimmed in an entire section of Philadelphia.



Figure 4. ENIAC [9]

After completing the ENIAC, Dr. John William Mauchly and J. Presper Eckert moved out of the university circle and started a corporation called the Eckert- Mauchly Computer Corporation. In 1950, the Remington Rand Corporation purchased this company. Remington Rand, with the help of these two men, launched the first commercially available digital computer the UNIVAC I (Universal Automatic Computer). The first UNIVAC installation was for the U.S. Government Census Bureau

in 1951. The first commercial customer to purchase a UNIVAC was the Prudential Insurance Company. In 1952, the UNIVAC I became famous in successfully predicting that Dwight D. Eisenhower would win the 1952 presidential election. It was the first mass-produced computer, with a total of 46 systems built and delivered at a cost of approximately one million dollars. Comparable in size to the ENIAC, this machine was 25 feet by 50 feet in length, contained 5,600 tubes, 18,000 crystal diodes, and 300 relays. It had an internal storage capacity of 12,000 characters [10].

The major improvement in the UNIVAC computer was that it performed general purpose computing with large amounts of input and output. It utilized magnetic tape for input devices and typewriters for output, and was the first computer to use the concepts of buffer memory. In Figure 5, one can notice that the UNIVAC was an open system, instead of being closed like earlier computers. The main processor was in one frame and the I/O devices (peripherals) were now in other frames spread out in the room. By using buffer memory concepts, I/O devices buffer their requests letting them move bigger pieces of data between an I/O device and the main processor. This greatly freed up the processor to perform calculation, instead of it waiting on slow I/O devices. For this reason, this was a vast improvement in the system throughput or amount of work it could perform.



Figure 5. UNIVAC I system [11]

Thanks to Hollerith, the ABC Computer, ENIAC, and UNIVAC I, truly remarkable and groundbreaking concepts such as the vacuum tube, the computer revolution was off and running. This foundation was equally significant to the development of the computer as Neal Armstrong's first steps on the moon were to the space program. Although these early designs and concepts were refined over many years, it is important to note that many of these early engineering concepts and ideas are still in use today.

It is important to step back and take a look at the complete picture to understand from a historical perspective, the incredible fury of activity taking place at this time. Through the new developments, there were many startup companies competing against one another to capture this incredible new market. It was the computer architecture gold rush, and everyone had gold fever. Most of these early companies have long come and gone. Punch card machines were the center of modern data processing at this time because they were faster than manual methods. While advanced for their age, they had many deficiencies such as requiring set-up time. The card machines and early computers were programmed via “hardwiring” patch panels, and job scheduling was done on a daily or weekly basis. If Friday was payroll day, the computer operators would wire up the machines to perform payroll. They would run the payroll jobs to print out checks and then would power off the machine to set-up the next series of jobs. Even with the implementation of the first generation computers, many of these card processes and set-up steps were required. By the end of WWII, IBM was the world-leading punch-card company. Even though many of these early computers were more like sophisticated electronic business calculators, they were ten times faster than punch card processing. Specifically, this speed improvement was forcing a drastic shift in technology.

In 1953, the first widely used mass-produced computer system, the IBM model 650, was introduced. Originally, IBM planned to produce 50 systems, but the 650 was so successful that IBM eventually manufactured more than 2,000 machines. The first shipments were in 1954 and IBM ended production in 1962. The 650 performed 78,000 additions or subtractions per minute 5,000 multiplications per minute 3,700 divisions per minute, and 138,000 logical decisions per minute [12]. IBM was a tremendous powerhouse at this time, and they dominated this market into the next decade. IBM marketed many machines in the 1950’s, all first generation vacuum tube machines, called the 6xx and 7xx series products.

In the late 1950’s and early 1960’s, the transistor began to change influence change in the computer world. The second-generation transistorized machines appeared. In 1959, IBM shipped the first transistorized mainframe, the 7090, and a smaller version, the 1401. These machines used ferrite magnetic core memory; they had 4, 8, or 16 K memory capacity, which could store 16,000 characters of information. IBM shipped 12,000 7090 machines, making it the most successful machine in history at the time. These first transistorized computers were more or less clones of their vacuum tube counter parts. In 1960, the IBM model 650 was replaced by IBM transistorized version 1620.

In 1959, Jack Kilby of Texas Instruments invented the integrated circuit, which had two transistorized devices on it. This invention laid the foundation for high-speed, large capacity memory computers. The integrated circuit sparked yet another era of explosive growth with third generation computers. Simultaneously occurring in the 1960s, explosive programming improvements were being devised, and many new programming languages were in development. Other developments such as, Digital Equipment Corporation’s launch of the first “mini-computer” the PDP1 a smaller less expensive version of a mainframe, also had great impact. In 1961, Burroughs announces the B5000, the first dual processor and virtual memory computer in incorporating memory stack technology. In 1962, the Sperry Rand UNIVAC 1107 was the first to use

general purpose register concepts. On a side note, it is interesting to see the historical design patterns repeat themselves. Original first generation, vacuum tube computers evolved into smaller, faster second generation transistorized machines. Now second generation transistorized machines are made smaller, faster, and more reliable using third generation, integrated circuits, while incorporating new design concepts.

In the past, computers were not only big and expensive, but they were difficult to use. Even while utilizing the von Neumann model of stored-program concepts, machines of this day were programmed in machine language, not a human language. Another issue with machine language is that it was solely machine dependent. If one wrote software to run on a UNIVAC, it was not compatible with an IBM or vice-versa. In fact software written for one IBM model was not compatible with a different IBM model. Therefore a reoccurring theme in the computer industry, one that is still true today, is the paradigm of “first”, “fastest”, or “cheapest”. This is a condition that if one is trying to market a product to someone one has to meet two of these three criteria to make the sale. Be the “first” to market with the product, have the “fastest” product, or be the “cheapest”. Nowhere in this equation does being the “best” factor in to making a sale. IBM gained massive dominance in the computer market, simply because they were the “first” in the market. This fact gave IBM a tactical time advantage over competitors, because by the time the competition could develop a similar product, IBM’s next announcement would be out. Once customers made massive investments to write software to run on IBM, they were locked into IBM products. While they could have switched to another vendor, it would cost a company too much time and money to rewrite thousands of programs they have already developed to run on IBM’s platform.

With these early computer systems, application development was an extremely tedious task. Programmers had to speak to the computer hardware in a language that the hardware could understand, called machine language. Machine language is the set of hardware instructions that the processor hardware is designed to execute, such as, ADD, SUBTRACT, MULTIPLY, DIVIDE, MOVE, LOAD, JUMP, etc [13]. As a programmer, one had to keep track of all input and output functions, contents of memory locations, and general purpose registers. Registers acted as scratch pads where one could store a number into a register, then ADD that register to a different register to get a total. Machine language used direct addressing methods, if the main program branched to another location, as a programmer, one had to keep track of all these locations in order to return back to the main program. Programmers had to have extensive knowledge of the hardware in order to write simple software applications. In fact, just running jobs on early computers was difficult. The IBM 1401 required that one load a monitor deck of cards. Operators would put the deck of cards in the card reader and then press the load button on the main console. The 1401 would load the monitor program into main storage and this program would give the machine “smarts”. Once it loaded, it would come to a wait code that the operator would read out from console lights in binary, this was like booting up a modern PC. Once the monitor program loaded, the monitor would know where to start executing instructions. The operator would then load the program from a card deck into memory and hit the execute button to run the program.

Until the development of the operating system, writing programs and executing jobs on early computer systems was difficult. An operating system is a software

application that manages hardware resources and is an interface between the hardware and the computer operator. The operating system humanizes the computer, making it much easier for operators to instruct the system to perform a given task. Compiler developments led to the birth of many high-level programming languages, making it easier to develop software applications. By the 1960s, about 200 different programming languages were developed. The major ones were: FORTRAN (1957), COBOL (1960), and BASIC (1965).

In 1964, a major pivotal point in development emerges the IBM System 360 family of computers. IBM named it the 360 because of the 360 degrees in a circle, a circle that would figuratively encircle the globe. Unlike products of the past, designed to perform a specific task, and lacking compatibility with other systems, the System 360 family from the smallest processor to the largest, all ran the same hardware instruction set, and utilized forty common peripherals devices. This was a major deviation from the past. Now a company could start out with one model 360 and develop applications. If the company outgrew this model they simply could roll in a new faster processing unit, connect all their existing IO peripherals, without the time and expense of having to rewrite application programs. The system 360 was a tremendous gamble for IBM, it was the most expensive computer project in history at this time. IBM invested \$5 billion dollars in this project, and it was a huge product deviation from their other money generating products. In the 1960's IBM's research and development budget was larger than the Governments. They hired 60,000 new employees, built new plants to manufacture integrated circuits. Tom Watson Jr. and Tom Watson Sr. the founding family members of IBM argued intensely over this gamble [14]. If the system 360 failed IBM would have gone bankrupt. Tom Watson Jr. saw the future in electronic transistorized computers and went against his father's wishes in this undertaking. \$5 billion in 1964 dollars is equal to about \$30 billion in 2005 dollars. The only project that was larger than System 360 during this time was the Apollo Space program.

Furthermore, the system 360 introduced microcode technology, a low level layer of computer code that interfaces with the computer hardware. This microcode layer exists between the hardware and the operating system. This innovation achieves a common denominator between hardware and application software. For example, if the smallest processor in the 360 used blue hardware, it would come with blue microcode. This layer of blue microcode would be written so the blue hardware would understand it. If the application program that resides above this microcode layer executes an instruction, the blue microcode would convert this instruction so the blue hardware could understand and execute it. If we install a faster processor, that uses green hardware, it would come with green microcode. The same application instruction executes, and the green microcode would convert this instruction so the green hardware could understand and execute it. This microcode layer makes the application independent of model, making hardware transparent to the user applications. The same application could now run on any model in the system 360 family, without any conversion time or expense. Another major breakthrough was the concept of emulation. By now IBM had a large customer base, many customers had large capital investments in applications running on old hardware. It was a difficult sell to move customers into the newer 360, because this required rewriting all application software. Emulation software ran on the newer 360 processors and tricked the hardware to think it was something it wasn't, a different machine. This enabled a 1401

customer to install a System 360 and through 1401 emulation software run all their old 1401 applications this new platform with no software conversion. This gave IBM an even stronger foothold in the market place.

Other system 360 hardware developments were: twos complement arithmetic, floating point architecture, the 8-bit byte, 32-bit words, segmented and paged memory, and 16 general purpose registers [15]. Performance gains were achieved by using “smart peripherals”. In early computers the processor not only had to perform calculation, but also controlled the input output devices. As time evolved the IO devices became smarter by utilizing control units. Control units were standalone specialized processors that controlled the I/O devices. This offloaded work from the main processor to the control units thus enabling the main processor to devote all its time to calculation. Also control units utilized buffering to increase the size of data exchanged between main processor and control unit. Instead of only sending one character of data at a time to a device, now we could exchange pages of data at a time. This was a vast performance improvement. Watson’s gamble paid off big-time, even with numerous development problems, once the kinks were ironed out, IBM could not fill 360 orders fast enough. The system 360 and following system 370 completely redefined modern business computing. Customers embraced this technology, and for the next twenty years IBM totally dominated mainframe industry.



Figure 6. IBM system 360 [16]

In 1970, the fourth generation of computer emerged, built with computer chips that used large scale integration (LSI) technology. Just eleven years earlier, integrated circuits had 2 transistors on them. In 1970, they had 1000, increasing to 64,000 in 1980, then to 16 million in 1992. When IBM’s system 360 replacement was announced, the system 370, it utilized LSI circuits and introduced 32-bit technology. Microelectronic advancements resulted in massive performance gains like lower energy consumption, reduced heat generation, faster speeds, and much smaller in size. In 1971, Intel’s Dr Ted Hoff invented the first microprocessor the 4004 and it changed the landscape of the industry. The 1970’s, started a development explosion, Intel’s microprocessor made the

illusion of a computer on a chip, now an attainable possibility. From the microprocessor era emerged a new radical group of devoted computer hobbyists: Wozniak, Jobs, Allen, and Gates. These pioneers capitalized on the developments in microelectronics, operating systems, and programming to create and propel the industry into an era of unprecedented growth.

On a completely different technological front in 1975, the first home-based computer emerged the Altair. Sold in electronic magazines for \$397 dollars with features like no screen, no keyboard and the size of a modern microwave it was a hobbyist's dream. Twenty-one-year-old Bill Gates wrote a BASIC compiler to run on Intel's microprocessor. Wozniak and Jobs had their version of BASIC running on a Motorola microprocessor and the single-circuit board Apple 1 was born featuring a keyboard and hooked to a TV screen [17] In 1976, the Apple 2 came out, with a new developed program called a spreadsheet named VisiCalc. This gave the home computer legitimacy to do real business calculations, not just a toy or game console. Apple won the hearts of the school systems and the home market. Apple was prospering, and new microcomputer companies sprang starting another fury of gold fever. This was a very exciting time in the industry. These microprocessor computers along with Digital Equipment Corporation DEC mini-computers started a computer revolution that hit so fast and forcefully that it unseated the mainframe from its dominant hold.

In the late 1970's, the mainframe computer growth started to flat line off its double-digit, growth of early years. This emerging small computer phenomenon caused IBM to look at this smaller computer market, and they realized they were missing out on this new opportunity. What happened next is one of the greatest business success stories and one of the greatest business failures to ever occur. IBM developed the Personal Computer (PC) in 1981. They approached the PC in a totally different light, abandoning conventional development cycles. Instead of building every component for the product themselves, they shopped around and assembled the computer with readily available parts: Tandem diskette drives, Zenith power supplies, Epson printers, and an Intel 16-bit 8088 microprocessor. This allowed IBM to complete the product in nine months.

The part about how IBM chose and acquired the operating system has a few different versions. One story is that IBM was in contact with Bill Gates startup company Microsoft, because they wanted his BASIC, FORTRAN, and COBOL compilers [18]. After failing to acquire CP/M, the most popular operating system at the time, they went back to Gates and had him come up with one. Another version is that Gate's mother overheard an IBM executive talk about the PC in development. Gates and Allen approached IBM with a proposal that they would license their Operating System to IBM. IBM liked this idea, so Gates and Allen shifted to panic mode to come up with an operating system because they did not have one at this time. They purchased Quick and Dirty Operating System QDOS from a Seattle company for \$50,000. They improved it and called it MS-DOS. The rest of the story is history. IBM's oversight gave Microsoft billions in revenue that IBM could have enjoyed. Now, almost 25 years later, Microsoft is IBM biggest competitor.

The failure on IBM's part was in licensing the software from Microsoft. This gave Microsoft the right to market the software it to anyone, and they did. The IBM Personal Computer was so successful that it was named the man of the year in 1982 by *Time*

Magazine. This caused the clone market to open up and compete against IBM using the same operating system MS-DOS that Microsoft was happy to sell to anyone. These PC clone companies were much cheaper than the IBM PC, and they sold millions of them. This mistake may have been difficult for IBM to avoid. In hindsight, IBM should have agreed to an exclusive license agreement for MS-DOS or purchase it outright from Microsoft. There are many reasons why this may not have happened. One is simply massive oversight IBM was so successful with mainframes they did not look at the PC seriously. Government was suing IBM for being a monopoly in the mainframe market and this divestiture may have been taken as a way to show the government they competed fairly using a competitor's product. At this time, IBM felt invincible and may have even feared the PC cutting into its higher profit mainframe margins. This was the same arrogance General Motors exhibited in the 1970's, when the Japanese started selling smaller cars. Both IBM and GM made massive tactical errors, which cost them dearly.

In the last quarter of the century, gold fever continued to move everyone west, finally stopping in California's Silicon Valley. Intel's microprocessor started the personal computer revolution, causing new technologies to continually play leap frog. Microelectronic advancements progressed to very large scale integration (VLSI), into ultra large scale integration (ULSI) enabling more circuits to be placed on smaller chips. Not only did size of these devices decrease, but the sheer horsepower increased, too. The 16-bit PC revolution has now progressed into 64-bit processor. Man's appetite for smaller, faster computers simply cannot be suppressed.

Much more could be said about the history of the computer to provide a more complete picture. It is important to mention just some of the many companies that contributed to the development of Computer Architecture like: GE, Remington Rand, RCA, NCR, Control Data, Memorex, Burroughs, Univac, Amdahl, Fairchild, Data General, Prime, Wang, Four Phase, Sperry, Motorola, Zilog, AMD, Intel, IBM, DEC, and Ampex. However, progress great progress has also been made in the areas of networking and protocols, and that is where we turn our attention now.

In the early days of information security, security was implemented only in the physical sense. For example a "secure" computer system was enclosed in a concrete building with strong locks and doors. Perhaps military or government locations would have armed guards to control personnel entering the center, but that was essentially the extent of secure operations. If you had security clearance to the physical system, you could run any job. There were no safeguards built into the early hardware or operating systems to control access levels or access rights. There have been methods of encryption used for top secret data dating back to the early 1900's., but these methods were crude, consisting of ciphers that did nothing more than substitute characters to scramble the plaintext message.

Needless to say, lot has changed in the last 25 years in regards to information security. The catalysts that drove this were (1) improvement in telephone communication systems that enabled remote connections, (2) standardized communication protocols that enabled different manufactures computers to communicate with one another [19], (3) the growth of the personal computer, and (4) the Internet. One of the first users of the early computer systems was the government. Each of the four branches of military had early

installations of mainframe systems. However, one problem the military encountered was that each branch of military selected a different vendor's computer. This was done as a precautionary measure. The military did not want to risk national security by placing all its eggs in one basket by using one vendor. They feared that if a vendor went out of business or had reliability issues with their system, they did not want this affecting all branches of the military, especially during the Cold War. Therefore, by solving one problem they created another. At this time, there were no standards in data communication protocols, which is a set of rules governing communication within and between computing endpoints.

The birth of the Internet started when the United States Space Program was born. In the late 1950s and early 1960s, the United States got caught up in the space race with Russia. In response to the fear of Russia launching a nuclear warhead at the United States, one of the immediate reactions was the creation of the Advanced Research Projects Agency (ARPA), within the Department of Defense. ARPA's mission was to apply state-of-the-art technology within the United States for defense. The initial emphases of ARPA activities were on space, ballistic missiles and nuclear test monitoring. However, communications became a major focus of ARPA activities. ARPA started a project called ARPANET; its goal was to link computer systems together between different geographical locations. The reasoning behind this was that if one location was attacked, the other locations could be used. In 1965, computers in Berkeley and MIT were linked over a low speed dial-up telephone line to become the first wide-area network ever created. After this, advancement continued and in 1971, 23 computers were networked together. In October of 1972, ARPANET went public at the First International Conference on Computers and Communication, held in Washington DC, ARPA scientists demonstrated the system in operation, successfully linking computers together from 40 different locations. This stimulated further research in the scientific community throughout the western world. Soon, other networks would appear along with more innovation. E-mail was also invented in 1972, TCP/IP in 1974 and in 1984 the number of hosts topped 1,000 for the first time.

In 1984, the U.S. military portion of the ARPANET was broken off into a separate network, the MILNET. The National Science Foundation (NSF) became involved in Internet research in the mid-1980s and basically took over as the governing branch of ARPANET. Likewise, work was progressing to develop the Internet at this time. The U.S. Department of Defense decided the network was developed enough for its initial purposes, and decided to stop further funding of the core Internet backbone. The NSF agreed to provide the backbone for the US Internet service, and provided five supercomputers to service traffic. A protocol called TCP/IP was accepted as the standard protocol for routing movement of data from place to place. Fiber Optic cable was developed and was being used to handle the expansion of growing traffic. IBM entered the personal computer market in the early 1980's and provided credibility and acceptance to a "computer in every home theory". Shortly after, home PC users were introduced to e-mail.

The emphasis on computer security has evolved with these technological advancements. In October 1967, a Task Force was organized by the ARPA to study and recommend appropriate computer security safeguards that would protect classified

information in multi-access, resource-sharing computer systems. The report of the Task Force was published by The Rand Corporation in February 1970. It was called Security Controls for Computer Systems. This report expanded information security beyond the physical level to thinking in terms of securing data, users, and infrastructure. Operating systems were also improving in the security realm. The first operating system to address security issues was called Multiplexed Information and Computer Service (MULTICS). MULTICS was also notable for its early emphasis on security by design, because it was the first operating system to be designed as a secure system from the ground up. It also incorporated user password protection and authentication into multiple user security levels.

Along with the increased growth in computers, the Internet, and communication devices, there is also the same increase in computer crime. Security has never been as important as it is today. Criminals will always go places to seek out their victims. Today, that place is the internet. From hacking into computer system to steal personal and financial information, to stealing government or corporate secrets the computer security professional will become the growth profession of the future.

The purpose of this chapter was to give you an appreciation for how the current state of the art in computer technology was achieved. Computer Science is a field that emphasizes problem solving. The industry has attained its current level through a series of missteps and fixes. There are numerous problems still facing the industry today, particularly in terms of the security of systems. Your challenge will be to advance the state of the art by applying your talents and creativity to solving these problems.

References:

[1] United States Census. In *Wikipedia* [Web]. Retrieved September 3, 2006, from http://en.wikipedia.org/wiki/United_States_Census

[2] Herman Hollerith. In *Wikipedia* [Web]. Retrieved September 3, 2006, from http://en.wikipedia.org/wiki/Herman_Hollerith

[3] Figure 1 Hollerith 1890 Census Tabulator. Retrieved September 3, 2006, Web site: <http://www.columbia.edu/acis/history/census-tabulator.html>

[4] Figure 2 Hollerith Tabulator & Sorter Box. Retrieved September 6, 2006, from IBM Archives > Exhibits > Antique attic, vol. 1 > Artifacts list for vol. 1 > Web site: http://www-03.ibm.com/ibm/history/exhibits/attic/attic_071.html

[5] Figure 3 Hardwired Plug Board. Retrieved September 6, 2006, from IBM Archives > Exhibits > Antique attic, vol. 1 > Artifacts list for vol. 1 > Web site: http://www-03.ibm.com/ibm/history/exhibits/attic/attic_054.html

[6] Art of Compiling Statistics; Apparatus for Compiling Statistics. Retrieved September 3, 2006, from www.museum.nist.gov Web site: <http://museum.nist.gov/panels/conveyor/hollerithbio.htm>

- [7] ABC (Atanasoff-Berry Computer) 1940. Retrieved September 4, 2006, from www.Computer Museum.il Web site: <http://www.computermuseum.li/Testpage/ABC-Computer-1940.htm>
- [8] ENIAC. Retrieved September 4, 2006, from www.wikipedia.com Web site: <http://en.wikipedia.org/wiki/ENIAC>
- [9] Figure 4 ENAIAC Photo Timeline of Computer History. Retrieved Sept 6, 2006, from The Computer History Museum Web site: <http://www.computerhistory.org/timeline/?category=cmptr>
- [10] UNIVAC I. In *Wikipedia* [Web]. Retrieved September 4, 2006, from http://en.wikipedia.org/wiki/UNIVAC_I
- [11] Figure 5 UNIVAC I System. Photo courtesy of: The Smithsonian
- [12] 650 Feeds & speeds. Retrieved September 4, 2006, from www.IBM.com Web site: http://www-03.ibm.com/ibm/history/exhibits/650/650_fs1.html
- [13] Stalling, William (1999). *Computer Organization and Architecture*. Upper Saddle River, New Jersey: Prentice Hall.
- [14] Thomas J. Watson and Peter Petre, (2000). *Father, Son & Co.: My Life at IBM and Beyond*.
- [15] IBM System/360. In *Wikipedia* [Web]. Retrieved September 1, 2006, from http://en.wikipedia.org/wiki/System_360
- [16] Figure 6 IBM System/360 Model 75. Retrieved September 6, 2006, from IBM Archives > Exhibits > Antique attic, vol. 1 > Artifacts list for vol. 1 > Web site: http://www-03.ibm.com/ibm/history/exhibits/attic/attic_054.html
- [17] Apple Computer. In *Wikipedia* [Web]. Retrieved September 4, 2006, from http://en.wikipedia.org/wiki/Apple_computer
- [18] Ferguson, Morris, Charles H, Charles R (1993). *Computer Wars, How the West Can Win In a Post-IBM World*. New York, New York: Random House, Inc.
- [19] Communications protocol. In *Wikipedia* [Web]. Retrieved November 12, 2006, from http://en.wikipedia.org/wiki/Communications_protocol
- [20] Computer security. In *Wikipedia* [Web]. Retrieved Nov 6, 2006, from http://en.wikipedia.org/wiki/Computer_security

Chapter 2: Data Representation

Throughout history, man has used numerous methods to communicate with another, such as smoke signals, sign language, Morse code, and verbal languages. If two people wanted to communicate with one another, they could also use a non-verbal form of communication, by tapping once for yes and twice for no. Although this non-verbal communication is somewhat strange and different, it could be effectively used. The struggle that early computer developers faced was how to get humans to communicate effectively with computers. Man and computers speak in two entirely different languages; Americans use English as their main language, and base-ten decimal as their numbering system. Computers work off the principal of a switch, which has only two states, either on or off, such as a simple light switch. The struggle is how to represent all characters and number symbols with two states. Fortunately, the state of on or off is base two, and fits in perfectly with the binary numbering system.

In 1890, when Herman Hollerith invented the punch card, the first obstacle he had to overcome was getting the holes in the card to represent certain characters and numbers. He did this by using a hole or multiple holes in a column to represent one character of data, called the 80 column card. The 80 column card is comprised of 80 columns and 12 rows. The 12 rows were broken into two different sections called zone and numeric rows. The top three rows of the card are the zone rows and the bottom 9 rows are the numeric rows as shown in Figure 1. If one wanted to represent a number they would punch one hole in the column in the numeric row that corresponds to the number. For example; a one hole = number 1, a two hole = number 2, etc. To represent a character punch two holes in the same column, a zone punch with a numeric punch. For example; a 12 zone with a one hole equals the letter A, a 12 zone with a two hole equals a letter B, and so on. If you look close at Figure 2, you see that a 12 zone hole with a numeric hole represents letters A to I. An 11 zone hole with a numeric hole represents letter J to R. The remaining letters S to Z, are represented with a 0 zone hole with a numeric hole. Special character symbols were represented by 3 holes in the same column.

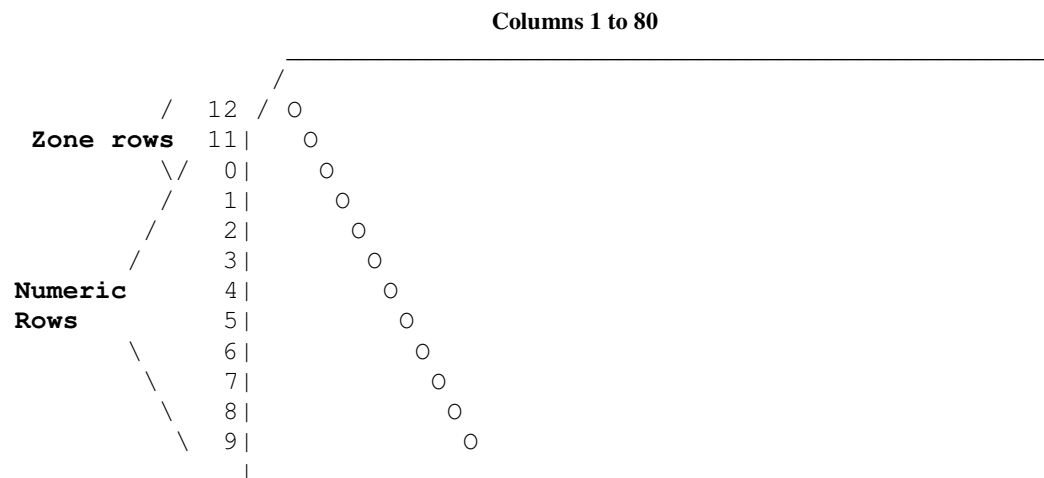


Figure 1. 80 column punch card layout

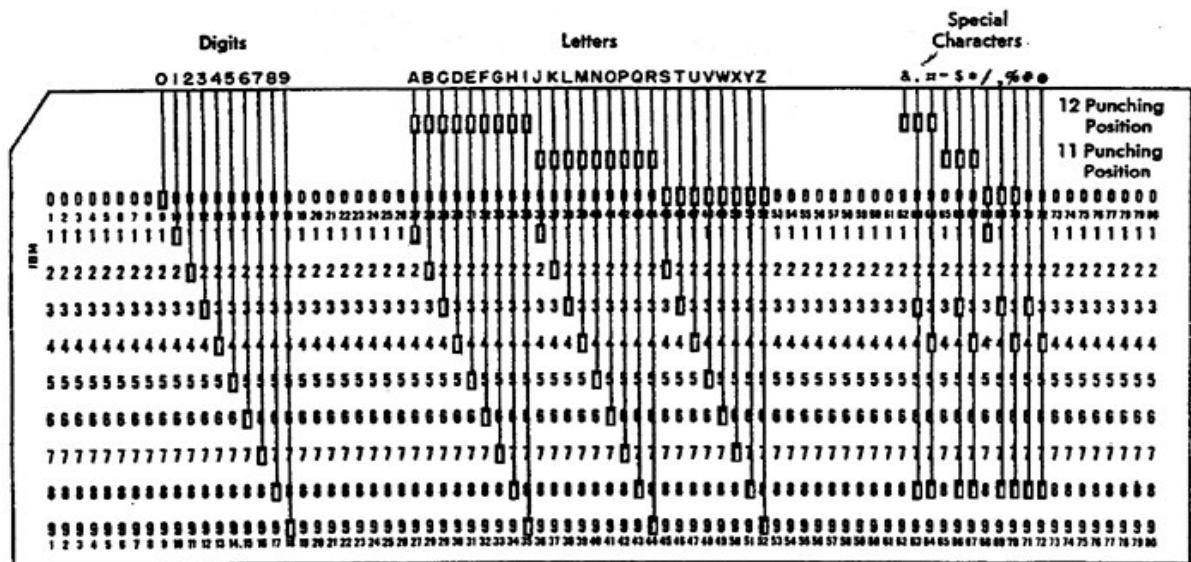


Figure 2. Punching Positions in Card

Figure 2. 80 Column punch card [1]

In mathematics, many notational systems are used to represent numbers. A numbering system is defined by the base it uses. The decimal system is the most common number system in use today. It uses ten different symbols to represent numbers and is therefore called a base-10 system. In computer applications, binary (base 2), and hexadecimal (base 16), are the most commonly used numbering systems. It is very important to understand binary and hexadecimal in order to gain a deeper understanding how computers function at their inter-core.

Decimal

In order to understand a different numbering system, it is important to review the numbering system we use the most decimal. We use decimal so frequently, that we sometimes forget how it fundamentally works. Once one understands the principle of decimal, this same principle applies to binary, and hexadecimal. Decimal utilizes ten symbols 0,1,2,3,4,5,6,7,8,9. The position of a symbol or placement of a symbol denotes the value of that symbol in terms of exponential values of the base. That is, the symbol placement determines positional value.

A numbering system that uses positional representation means that the value represented by a digit depends on its position in the representation of the number. For example, the 6 in 461 represents the value of sixty, while the 6 in 6927 represents six thousand. The value represented by a digit in a position representation system is the digits own value multiplied by a power of the base. It can be expressed as:

$$d_x d_{x-1} \dots d_2 d_1 d_0$$

where the d 's are digits, the subscript represents the exponential power to raise to the base number. Representing the digit d_i simply means to multiple the radix raised to the i th power.

In decimal, we call the rightmost position the ones (10^0) place, it also is called the least significant digit (LSD). The next (from the right) position is the tens (10^1), the next the hundreds (10^2), and so on. We can express the decimal number 7903 as:

$$\begin{aligned} &(7 \times 10^3) + (9 \times 10^2) + (0 \times 10^1) + (3 \times 10^0) = \\ &(7 \times 1000) + (9 \times 100) + (0 \times 10) + (3 \times 1) = \\ &7000 + 900 + 0 + 3 = 7903 \end{aligned}$$

Another way to observe this would be to examine how we count. We start counting in the Least Significant Digit (LSD) position d_0 , so d_0 digit placement value is expressed as 10^0 . Note that any number raised to the zero power equals one. So d_0 placement value is one, or the one's field. In decimal ten we count, zero, one, two, three, four, five, six, seven, eight, and nine. When you run out of numeric symbols, you have to carry over to the next digit position d_1 . d_1 digit placement value is expressed as 10^1 or ($10 \times 1 = 10$) ten raised to the first power equals ten. So d_1 placement value is tens. Once again, counting zero to nine, until carrying to the next digit position d_2 . Every time carrying to the next digit position the digit placement value increases exponentially as illustrated in Table 1.

Table 1. Represents decimal digit placement values

MSD					LSD	
d_x	d_4	d_3	d_2	d_1	d_0	Digit Position
10^x	10^4	10^3	10^2	10^1	10^0	Placement Value
	10,000	1,000	100	10	1	Positional Value
0	0	0	0	0	0	Symbols
1	0	0	0	0	0	
2	2	2	2	2	2	
3	3	3	3	3	3	
4	4	4	4	4	4	
5	5	5	5	5	5	
6	6	6	6	6	6	
7	7	7	7	7	7	
8	8	8	8	8	8	
9	9	9	9	9	9	

Example problem 1:

In the number 357. The Least Significant Digit (LSD) $d_0 = 7 \times 10^0 = 7 \times 1 = 7$, The 5 digit $d_1 = 5 \times 10^1 = 5 \times 10 = 50$. The Most Significant Digit (MSD), 3 number $d_2 = 3 \times 10^2 = 3 \times 100 = 300$. The sum of these positional values is $7 + 50 + 300 = 357$.

Binary

The binary system works like the decimal system, however, with only two symbols which can be used to represent numerical values: 0 and 1 [2]. Similar to decimal, the position of a symbol or placement of a symbol denotes the value of that symbol. The symbols placement determines positional values. Symbol placement and positional values can be seen in Table 2.

Table 2. Binary placement

d_x	d_4	d_3	d_2	d_1	d_0	Digit position
2^x	2^4	2^3	2^2	2^1	2^0	Placement value
	2x2x2x2	2x2x2	2x2	2x1	2x0	
	16	8	4	2	1	Positional value
0	0	0	0	0	0	Symbols
1	1	1	1	1	1	

To understand binary simply count, examine Table 3 closely to see what is taking place. Start counting at zero (0000), one (0001), two (0010) notice the carry because there are no more symbols to use, three (0011), four (0100) carry's to the next column, etc. (Reference Table 3 below) [3]. Binary maybe unusual, but it is the foundation of how digital electronics works. Two states can be on of off, light or no light, hole, or no hole, and voltage or no voltage. The two states "0" and "1" can be represented a voltage of 0 volts for zero value, and 5 volts for one value, this is how transistors and switches function.

Table 3. Counting in binary

Decimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

One nice thing about binary is that it is easy to convert binary to decimal. Although binary uses two digits, each column is worth twice the one before it. In the binary system, the columns are worth 1, 2, 4, 8, 16, 32, 64, 128, 256, etc. To convert a number from binary to decimal, simply write it in expanded notation. For example, the binary number 101101 can be rewritten in expanded notation as $(1 \times 32) + (0 \times 16) + (1 \times 8) + (1 \times 4) + (0 \times 2) + (1 \times 1)$. By simplifying this expression, the binary number 101101 is equal to the decimal number 45 as illustrated in Table 4.

Table 4. Converting binary 101101

32	16	8	4	2	1	Position value
1	0	1	1	0	1	Binary Number
32×1	16×0	8×1	4×1	2×0	1×1	Expression
32	0	8	4	0	1	Sum

$$32 + 0 + 8 + 4 + 0 + 1 = 45_{10}$$

$$101101_2 = 45_{10}$$

Easy way to convert binary to decimal is to simply list the positional values over the digits starting with the least significant digit. The positional values in binary are easy to remember because they double. Starting with the LSD the positional values are: 1,2,4,8,16,32,64,128, etc. It is much easier to see in a table format refer to Table 5.

Table 5. Binary breakdown of 10101010

128	64	32	16	8	4	2	1	Positional Value
1	0	1	0	1	0	1	0	Binary

Now add the positional values for every column with a 1 bit.

$$10101010_2 = 128 + 32 + 8 + 2 = 170_{10}$$

Bits and Bytes

As a rule, the smallest unit of data in a computer is a bit. We seen in binary notation a bit can be a “0” or a “1”, on or off, etc. Since a single bit can only represent two different states, there had to be a method to represent the complete character set of the alphabet; special characters, and numbers. To do this a method of grouping a number of bits together was devised. In Hollerith’s punch card methodology, he devised his standard by using a series of holes in a column to represent numbers and character data. Modern computers, they do the exact same thing by grouping a number of bits together. In a 16-bit processor configuration we group 8 bits together to form what is called a byte. Grouping two bytes together or 16 bits forms a word. The bits of a word are numbered from 0 to 15 starting with the rightmost, least significant bit. A longword consists of four contiguous bytes, or 32 bits, and a quadword consists of eight contiguous bytes, or 64 bits. These concepts are illustrated in Figure 3.

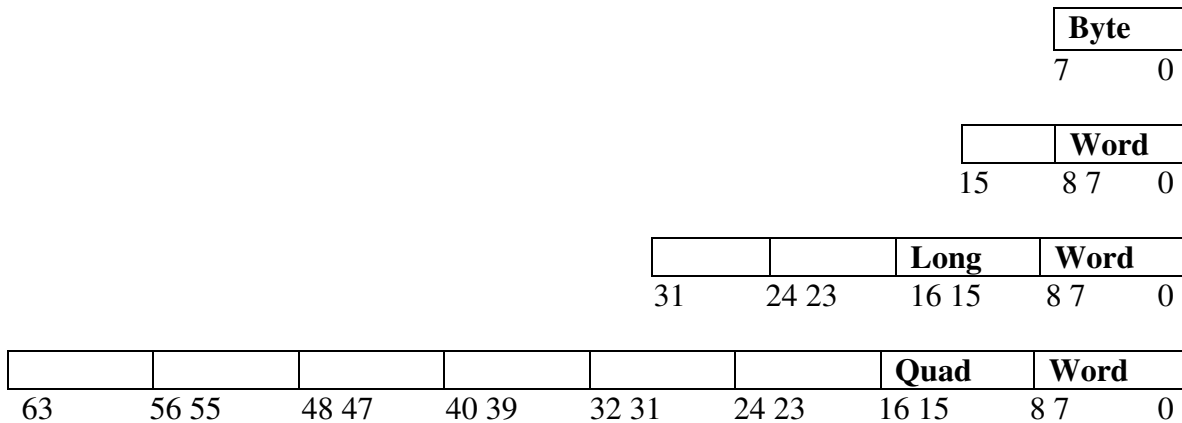


Figure 3. Units of Memory

The above example is true for the 16-bit processors used in the first personal computer. Even though the relationship that 8 bits form a byte is always true. The number of bits to form a word may not always be the same. The size of a word generally corresponds to the size of the processor. In a 32-bit processor, the word length is comprised of 32 bits, 16 bits would be called a half word. Likewise, in a 64-bit processor a word is 64 bits long.

Data Sizes and Precision

Understanding binary is imperative to understanding processor architecture. If you examine the fundamental differences is between; 8-bit, 16-bit, 32-bit, or 64-bit processor architectures, you first have to understand the numeric values of these word lengths. Refer to Table 6.

Table 6. Common binary values

Size	Binary Value	Decimal Value
8 bits	1111 1111	255
16 bits	1111 1111 1111 1111	65,535
32 bits	1111 1111 1111 1111 1111 1111 1111 1111	4,294,967,295

Upon examining the above table, it is apparent that the maximum value that can be expressed with 8 bits is decimal 255. Likewise the maximum value expressed in 16 bits is 65,535 and 4,294,967,295 with 32 bits. This is important for two reasons; first, computers use memory locations to store programs and data. The internal processor addressing mechanism uses this word length to address these memory locations. Therefore an 8-bit processor can only address 256 memory locations, and a 16-bit can access 65,536 locations. This illustrates one area of major confusion that appears with numbering systems. In decimal, zero is not usually referenced, but in binary, zero is always reference, because in terms of memory addressing, location zero is an addressable location. So in converting the binary number 1111 to decimal, we know it equals decimal fifteen, but it's important to keep in mind and in terms of addressable memory it's sixteen locations, 0 to 15.

In terms of computing performance addressing more memory equates to faster processing. Secondly, a character of data is represented with 8 bits, thus a 8 bit processor can move one character of data at a time. A 16 bit processor can move two 8 bit characters of data simultaneously, and a 32 bit processor moves four characters at one time. Larger word size increases performance drastically.

Representation of Character Data in Memory

The fundamental problem is how to group one and zeros together to represent the English alphabet, special characters and numbers. The two most common methods are ASCII (American Standard Code for Information Interchange) [4], and EBCDIC (Extended Binary Coded Decimal Code) [5]. ASCII was first introduced in 1967. At this time computers were based on eight-bit architecture, the ASCII coding method uses seven bits patterns to represent data. Using only 7 bits, ASCII can express only 127 characters. EBCDIC was developed by IBM in 1964 and uses 8 bits, therefore it can represent 255 characters. ASCII is very commonly used today. Refer to the decimal, binary, hexadecimal, and ASCII codes in Tables 7 and 8.

Hexadecimal

Hexadecimal, or hex, is another common numbering system used in data processing. Hexadecimal is base 16 and is written using the symbols 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E, and F. All the positional values and general rules used with decimal and binary apply to hexadecimal. Refer to Table 9.

The greatest use of hexadecimal is to humanize the language of communicating long binary words. Everything in the computer is represented in binary because that is all digital circuits understand. Yet as humans it is difficult if not impossible to communicate in binary. If someone called on the telephone and said they had a 1111 1111 1100 1010 error on their system you would have to be a genius to understand that. But if we converted it to hex, it becomes simple to speak as FFCA.

In converting binary to hexadecimal you group the binary number into four bits and start right to left convert to hex as show in example problem 2 below. Using conversion Table 10 makes it easy to convert binary to hex.

Example problem 2:

$$1000\ 1000\ 0100\ 0011_2 = 8843_{16}$$

$$1100\ 1001\ 0011\ 1101_2 = C93D_{16}$$

$$1100\ 0101\ 0011\ 1100_2 = C53C_{16}$$

Table 7. ASCII Codes

Decimal 0-63															
Dec	Binary	Hex	ASCII	Dec	Binary	Hex	ASCII	Dec	Binary	Hex	ASCII	Dec	Binary	Hex	ASCII
0	0000 0000	0	NUL	16	0001 0000	10	DLE	32	0010 0000	20	space	48	0011 0000	30	0
1	0000 0001	1	SOH	17	0001 0001	11	DC1	33	0010 0001	21	!	49	0011 0001	31	1
2	0000 0010	2	STX	18	0001 0010	12	DC2	34	0010 0010	22	"	50	0011 0010	32	2
3	0000 0011	3	ETX	19	0001 0011	13	DC3	35	0010 0011	23	#	51	0011 0011	33	3
4	0000 0100	4	EOT	20	0001 0100	14	DC4	36	0010 0100	24	\$	52	0011 0100	34	4
5	0000 0101	5	ENQ	21	0001 0101	15	NAK	37	0010 0101	25	%	53	0011 0101	35	5
6	0000 0110	6	ACK	22	0001 0110	16	SYN	38	0010 0110	26	&	54	0011 0110	36	6
7	0000 0111	7	BEL	23	0001 0111	17	ETB	39	0010 0111	27	'	55	0011 0111	37	7
8	0000 1000	8	BS	24	0001 1000	18	CAN	40	0010 1000	28	(56	0011 1000	38	8
9	0000 1001	9	HT	25	0001 1001	19	EM	41	0010 1001	29)	57	0011 1001	39	9
10	0000 1010	0A	LF	26	0001 1010	1A	SUB	42	0010 1010	2A	*	58	0011 1010	3A	:
11	0000 1011	0B	VT	27	0001 1011	1B	ESC	43	0010 1011	2B	+	59	0011 1011	3B	;
12	0000 1100	0C	FF	28	0001 1100	1C	FS	44	0010 1100	2C	,	60	0011 1100	3C	<
13	0000 1101	0D	CR	29	0001 1101	1D	GS	45	0010 1101	2D	-	61	0011 1101	3D	=
14	0000 1110	0E	SO	30	0001 1110	1E	RS	46	0010 1110	2E	.	62	0011 1110	3E	>
15	0000 1111	0F	S1	31	0001 1111	1F	US	47	0010 1111	2F	/	63	0011 1111	3F	?
Decimal 64-127															
Dec	Binary	Hex	ASCII	Dec	Binary	Hex	ASCII	Dec	Binary	Hex	ASCII	Dec	Binary	Hex	ASCII
64	0100 0000	40	@	80	0101 0000	50	P	96	0110 0000	60	'	112	0111 0000	70	p
65	0100 0001	41	A	81	0101 0001	51	Q	97	0110 0001	61	a	113	0111 0001	71	q
66	0100 0010	42	B	82	0101 0010	52	R	98	0110 0010	62	b	114	0111 0010	72	r
67	0100 0011	43	C	83	0101 0011	53	S	99	0110 0011	63	c	115	0111 0011	73	s
68	0100 0100	44	D	84	0101 0100	54	T	100	0110 0100	64	d	116	0111 0100	74	t
69	0100 0101	45	E	85	0101 0101	55	U	101	0110 0101	65	e	117	0111 0101	75	u
70	0100 0110	46	F	86	0101 0110	56	V	102	0110 0110	66	f	118	0111 0110	76	v
71	0100 0111	47	G	87	0101 0111	57	W	103	0110 0111	67	g	119	0111 0111	77	w
72	0100 1000	48	H	88	0101 1000	58	X	104	0110 1000	68	h	120	0111 1000	78	x
73	0100 1001	49	I	89	0101 1001	59	Y	105	0110 1001	69	i	121	0111 1001	79	y
74	0100 1010	4A	J	90	0101 1010	5A	Z	106	0110 1010	6A	j	122	0111 1010	7A	z
75	0100 1011	4B	K	91	0101 1011	5B	[107	0110 1011	6B	k	123	0111 1011	7B	{
76	0100 1100	4C	L	92	0101 1100	5C	yen	108	0110 1100	6C	l	124	0111 1100	7C	
77	0100 1101	4D	M	93	0101 1101	5D]	109	0110 1101	6D	m	125	0111 1101	7D	}
78	0100 1110	4E	N	94	0101 1110	5E	^	110	0110 1110	6E	n	126	0111 1110	7E	r arr.
79	0100 1111	4F	O	95	0101 1111	5F	_	111	0110 1111	6F	o	127	0111 1111	7F	l arr.

Table 8. Decimal, Binary, and Hex from Decimal Values 128-255

<i>Decimal 128-191</i>											
Dec	Binary	Hex	Dec	Binary	Hex	Decimal	Binary	Hex	Dec	Binary	Hex
128	1000 0000	80	144	1001 0000	90	160	1010 0000	A0	176	1011 0000	B0
129	1000 0001	81	145	1001 0001	91	161	1010 0001	A1	177	1011 0001	B1
130	1000 0010	82	146	1001 0010	92	162	1010 0010	A2	178	1011 0010	B2
131	1000 0011	83	147	1001 0011	93	163	1010 0011	A3	179	1011 0011	B3
132	1000 0100	84	148	1001 0100	94	164	1010 0100	A4	180	1011 0100	B4
133	1000 0101	85	149	1001 0101	95	165	1010 0101	A5	181	1011 0101	B5
134	1000 0110	86	150	1001 0110	96	166	1010 0110	A6	182	1011 0110	B6
135	1000 0111	87	151	1001 0111	97	167	1010 0111	A7	183	1011 0111	B7
136	1000 1000	88	152	1001 1000	98	168	1010 1000	A8	184	1011 1000	B8
137	1000 1001	89	153	1001 1001	99	169	1010 1001	A9	185	1011 1001	B9
138	1000 1010	8A	154	1001 1010	9A	170	1010 1010	AA	186	1011 1010	BA
139	1000 1011	8B	155	1001 1011	9B	171	1010 1011	AB	187	1011 1011	BB
140	1000 1100	8C	156	1001 1100	9C	172	1010 1100	AC	188	1011 1100	BC
141	1000 1101	8D	157	1001 1101	9D	173	1010 1101	AD	189	1011 1101	BD
142	1000 1110	8E	158	1001 1110	9E	174	1010 1110	AE	190	1011 1110	BE
143	1000 1111	8F	159	1001 1111	9F	175	1010 1111	AF	191	1011 1111	BF
<i>Decimal 192-255</i>											
Dec	Binary	Hex	Decimal	Binary	Hex	Decimal	Binary	Hex	Decimal	Binary	Hex
192	1100 0000	C0	208	1101 0000	D0	224	1110 0000	E0	240	1111 0000	F0
193	1100 0001	C1	209	1101 0001	D1	225	1110 0001	E1	241	1111 0001	F1
194	1100 0010	C2	210	1101 0010	D2	226	1110 0010	E2	242	1111 0010	F2
195	1100 0011	C3	211	1101 0011	D3	227	1110 0011	E3	243	1111 0011	F3
196	1100 0100	C4	212	1101 0100	D4	228	1110 0100	E4	244	1111 0100	F4
197	1100 0101	C5	213	1101 0101	D5	229	1110 0101	E5	245	1111 0101	F5
198	1100 0110	C6	214	1101 0110	D6	230	1110 0110	E6	246	1111 0110	F6
199	1100 0111	C7	215	1101 0111	D7	231	1110 0111	E7	247	1111 0111	F7
200	1100 1000	C8	216	1101 1000	D8	232	1110 1000	E8	248	1111 1000	F8
201	1100 1001	C9	217	1101 1001	D9	233	1110 1001	E9	249	1111 1001	F9
202	1100 1010	CA	218	1101 1010	DA	234	1110 1010	EA	250	1111 1010	FA
203	1100 1011	CB	219	1101 1011	DB	235	1110 1011	EB	251	1111 1011	FB
204	1100 1100	CC	220	1101 1100	DC	236	1110 1100	EC	252	1111 1100	FC
205	1100 1101	CD	221	1101 1101	DD	237	1110 1101	ED	253	1111 1101	FD
206	1100 1110	CE	222	1101 1110	DE	238	1110 1110	EE	254	1111 1110	FE
207	1100 1111	CF	223	1101 1111	DF	239	1110 1111	EF	255	1111 1111	FF

Table 9 Hex breakdown

Decimal	Hex	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Table 10. Converting Hex to Decimal

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F.
0	000	001	002	003	004	005	006	007	008	009	010	011	012	013	014	015
1	016	017	018	019	020	021	022	023	024	025	026	027	028	029	030	031
2	032	033	034	035	036	037	038	039	040	041	042	043	044	045	046	047
3	048	049	050	051	052	053	054	055	056	057	058	059	060	061	062	063
4	064	065	066	067	068	069	070	071	072	073	074	075	076	077	078	079
5	080	081	082	083	084	085	086	087	088	089	090	091	092	093	094	095
6	096	097	098	099	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

If you're having trouble getting the hang of the above chart, here's a hint.
Hex 41 is equivalent to decimal 65 [6].

.Encrypting Data

It is important to point out that for every method created to represent data in a computer system, there are similar methods used to scramble or encrypt data. One simple method of encryption is called the Vernam cipher named after Gilbert Vernam whom created numerous methods to encrypt data [7]. Vernam's idea was to simply use a

previously prepared key named a one-time pad that is used in combination with the character data (plaintext) to produce a scrambled cipher text message [8]. Plaintext data can be seen and understood by anyone. To implement a Vernam cipher encryption operation, the pad values are added to the numeric values that represent the plaintext that needs to be encrypted. For example, in plaintext the sentence “Im dying here” is represented with the numeric characters of 9 13 4 25 9 14 7 8 5 18 5. Numeric values are determined from their positional values of the alphabet. In the above example I is the 9th letter in the alphabet. As shown in Table 11 the plaintext values are added to the values of the one-time pad. The one-time pad is “oh boy” which is repeated for the specific character length of the plaintext. Note that if the sum of the two values exceed 26, then 26 is subtracted from the total because there are only 26 letters in the alphabet and this keeps the numeric values in the 1 to 26 range. To decrypt the first character of cipher text, which is the letter A, you take its value of 1 and subtract the pad value of 18 which equals negative 17, since this is a negative number we must add 26 to it which equals 9 which equals the plaintext value of the letter I.

Table 11. Vernam cipher

Plaintext	I	M	D	Y	I	N	G	H	E	R	E
Plaintext Values	9	13	4	25	9	14	7	8	5	18	5
One-time Pad Text	O	H	B	O	Y	O	H	B	O	Y	O
One-time Pad values	18	8	2	18	25	18	8	2	18	25	18
Sum of plaintext and pad	27	21	6	43	34	32	15	10	23	43	23
After modulo subtraction	1			17	8	6				17	
Cipher text	A	U	F	Q	H	F	O	J	W	Q	W

References:

-
- [1] The IBM Key 026 Punch. Retrieved September 15, 2006, from Columbia education Web site: <http://www.columbia.edu/acis/history/026.html>
- [2] Introduction to Binary Numbers. Retrieved September 14, 2006, from Swanson Technologies Web site: <http://www.swansontec.com/sbinary.htm>
- [3] Greenfield, Joseph (1977). *Practical Digital Design Using IC's*. New York: John Wiley & Sons.
- [4] ASCII. Retrieved September 15, 2006, from Wikipedia Web site: <http://en.wikipedia.org/wiki/ASCII>

[5] EBCDIC. Retrieved September 15, 2006, from Wikipedia Web site:
<http://en.wikipedia.org/wiki/EBCDIC>

[6] ASCII Chart. Retrieved September 15, 2006, from Jim Price Web site:
<http://www.jimprice.com/jim-asc.htm>

[7] Gilbert Vernam. In *Wikipedia* [Web]. Retrieved Nov 7,2006, from
http://en.wikipedia.org/wiki/Vernam_cipher

[8] Encryption. In *Wikipedia* [Web]. Retrieved November 7,2006, from
<http://en.wikipedia.org/wiki/Encryption>

Chapter 3: Introduction to Hardware

An Overview

An interesting approach in examining how a computer works is to compare it to the human body. The central processing unit is much like the human brain because it can execute commands, make logical deductions, perform mathematical operations, and interface with input output devices. Computer input devices are the keyboard, mouse, and microphone, similar to the input devices of a human body the ears, eyes, and neurological senses. Following suit, the output devices of a computer include printers, displays, magnetic media, and speakers. While the output devices of a human body are vocal cords, arms, legs, hands, and feet. Generally speaking the computer requires a means of interconnecting the processing unit to external devices. This communication path is called a channel, or more commonly, a bus. Most computers use three buses: address, data, and control bus, much like the body's interconnection system of the nervous system, veins, and musculature structure.

From an operations perspective, humans and computers perform tasks in a similar manner. Even though it appears that we do many things at once, like walk, talk and chew bubble gum, humans can only do one task at a time. The reality is that at any increment of time we are only doing one thing at that instance. A computer with one processor can execute one command at a time. Likewise, processors today are so incredibly fast that it appears that they do many things simultaneously, but the truth is that they only do one thing at a time like humans. In examining how humans perform simple tasks, we can gain insight to the inner working of a processor. When one task is started and continued until it is complete, it is performed in a dedicated mode. Some tasks or processes are best performed this way. When working on a complex task, it is best to not get interrupted, and continue to work until it's completed. Dedicated processing is effective for some tasks, but there are also inefficiencies with this method. For example, in the process of doing laundry it may take an hour for the wash cycle and another hour for the dry cycle. By doing this in a dedicated fashion, during the two-hour duration one would be wasting time, waiting for the operation to complete, when in fact one could be doing something else. A method to overcome this inefficiency is to multitask. In multitasking, one has started multiple tasks and can interleave increments of time between tasks. For example, consider a busy mother with three demanding child. Although all three children want the mother's attention immediately, the mother cannot satisfy the needs of all children simultaneously, so she gives one child a slice of time to satisfy the first child needs, then a slice of time to the second and then the third.

Furthermore, computer processors employ these same methods of dedicated processing, called batch processing. In addition, multitasking can be performed with one of two methods. The first is polling, which uses equal increments of time to interleave one task to the next, and interrupt method, which assigns priorities to tasks. With the interrupt method, if a low-priority task is running, a higher priority task can then interrupt the lower priority task to begin processing. The computer assigns a maximum increment of time to a process. Once the current process has used up its time slice it becomes suspended, so the next process gets its time slice. In utilizing this methodology, the processor multiplexes between processes, so no process can hog the processor. Once its

maximum time allotment is reached, the processor moves to the next process, if there is one. This technique is used for processes and for exchanging data with I/O devices. This greatly improves the amount of work or throughput completed by the processor. The processor is the fastest component of a computer system; all other devices are extremely slow compared to it. This method improves processing efficiency because the processor can be utilized to its fullest capacity by remaining busy by servicing other processes or devices instead of sitting idle waiting on some slow device or a dedicate process to complete. These above methods are used at the hardware level to exchange data between the processor and hardware I/O devices. They are also employed by the operating system to schedule jobs and tasks.

At the hardware level, the interrupt driven method is used on hardware I/O devices. In this method, the computer architect determines what devices have a higher priority and which ones have a lower priority. Usually with I/O devices, the faster the device, the higher priority level assigned. Therefore, the processor devotes more time to faster devices than slower ones. The main objective is to keep the processor as busy as possible. A simple way to illustrate the interrupt driven method is by observing how humans process tasks. For example, if one is watching television, and the telephone or doorbell rings, one can momentarily suspend watching television to service the phone or doorbell request. When this request is completed, one can then go back to watching television.

Functional Units

Historically, the form, design and implementation of processors have changed dramatically in the last fifty years, but their fundamental operation has basically remained the same. Regardless of the number of millions of electronic components used in today's processors, the processor still performs four basic functions: process data, store data, move data, and control data [1]. The top-level structure of a computer is referenced below in Figure 1, where four basic parts or subsystems are illustrated: the central processing unit (CPU), input output subsystem (I/O), main memory, and the interconnection subsystem or bus. The CPU is the brain that processes the data, memory is where data and programs are stored, and the I/O subsystem is the vehicle to communicate to the outside world via input and output devices. The system bus provides a communication path between the CPU and memory, the CPU and the I/O subsystem also provides control functions.

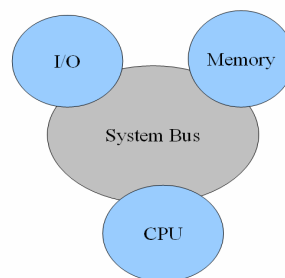


Figure 1. Top level structure

Looking inside the CPU, it is comprised of four functioning parts; the arithmetic and logic unit (ALU), registers, control unit, and an internal CPU bus. These are shown in Figure 2. The arithmetic/logic unit provides calculating ability and permits arithmetical and logical operations. The registers are temporary storage areas that hold data, keep track of instructions, and hold the location and results of these operations. The control section regulates the operations of the entire computer system, and the internal bus is the communication path between the subsystems.

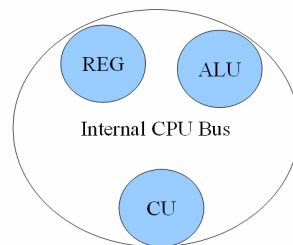


Figure 2. Overview of CPU

CPU Operation

Fundamentally, the operation of most CPUs, regardless of the physical form they take, is to execute a sequence of stored instructions in memory, called a program. Utilizing the concept of von Neumann's stored program model, the program and data are stored in memory. The CPU performs four basic operations or cycles: fetch from memory, decode instruction, execute instruction, and writeback [2]. See Figure 3 for a picture of this. The first cycle, fetch, involves retrieving the instruction from memory and placing it in a work register, called the accumulator register. An instruction is comprised of two parts; the opcode, which indicates what operation to perform, and operand, containing additional information that the opcode can use. For instance, if the opcode is an ADD instruction, the operand may contain numeric data used with the instruction. Depending on the opcode, the operand can be numerical or character data, or an address value to use with the operation. The decode cycle breaks the instruction down to interpret the operation to perform. The execute cycle performs the instruction. The final cycle, writeback, simply writes back the results of the execution to memory. After the execution of the instruction and writeback of the resulting data, the entire process usually repeats until termination of the program.

Although it may seem straightforward, there is extensive circuitry to perform and carry out the above four operations. For example, in the register functional unit of the CPU, there are many registers. Some are used as scratch pad areas for the programmer to temporarily store information, while some are specialized registers that keep track of internal operations. These include the program counter (PC), which contains the address of the next instruction to fetch, the (MAR) memory address register contains the next memory address to read or write to. Other registers are used to buffer data between memory and the I/O subsystem, and the program status word (PSW) which is a control register that keeps track of current conditions of the processor events using condition flags, such as machine exceptions, or arithmetic overflow or underflow.

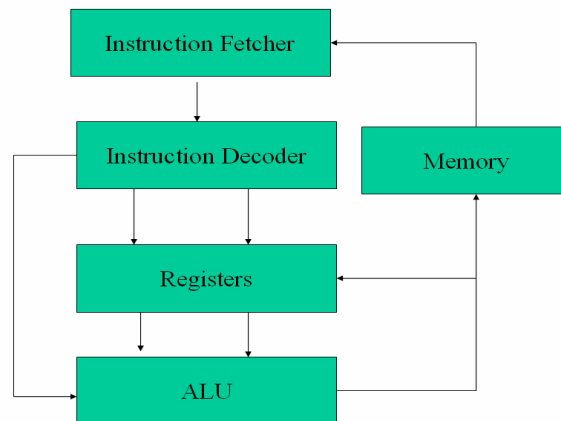


Figure 3. Inside the CPU

The Instruction

Conceptually, in designing a processor, the computer architect design team first determines the operational goals of the processor [3]. Once the operational goals are specified, the specifications and complexity of the instruction set would then be determined. This has direct impact on the internal structure of the hardware, and the amount of electronic circuits required. For example, early processors design was highly influenced by mathematicians, so all the processors were designed to perform precision mathematics [4]. Precision numeric floating point operations greatly complicated the hardware complexity. The main function of the processor is to interpret the instruction and perform the operation. If one wanted to design a simple set of instructions, they first have to determine the rules they want the machine to follow, such as instruction word size and integer or floating point precision. This process is similar to designing a verbal language. First, design the language semantics or protocols rules, and then determine what words to use. In the example shown in Table 1, the pseudo-instruction length is 16-bits, with the four most significant bits used for the op code and the other 12-bits for the operand fields. In using 4-bits for the opcode field, our design can only have a maximum 16 instructions. Table 2 illustrates a simple set of pseudo-instructions.

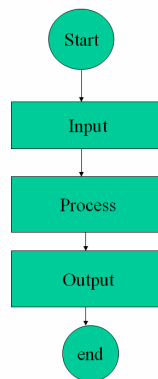
Table 1. Example instruction format

0	3 4	15
Opcode	Operand(s)	

Table 2. Sample instructions

Opcode	Operand 1	Operand 2	Instruction
0001	0000	Numeric value	ADD Immediate
0010	Register Value	Address value	LOAD contents addr into reg
0011	Register Value	Numeric value	MOV (Move)
0100		Address	JMP to address
1111		Address	STORE acc to addr

Once an instruction is fetched from memory, it is decoded, which determines what the instructional operation the processor will perform. The instruction set protocol rules determine if the operand provides additional information. For instance, the opcode operation may require a numeric value, memory address, or register value, contained within the instruction. Once the decode step is complete, the instruction is then executed. For example, if the decode step determined the operation was ADD, the execute cycle would ADD the numeric value contained in the operand field to the number that is currently in the accumulator register. This sum would then be written back into memory during the writeback step. This entire process will repeat until the program is terminated.

**Figure 4. Program flow**

In describing the operational characteristics of a program, all programs basically input data from a source, an input device or a file, manipulate this data in some fashion, and then output the data. A simple flowchart is shown in Figure 4. Fundamentally this is the basic premise for all programs. Table 3 illustrates what the binary instruction look like residing in main memory.

Table 3. Memory contents

Memory address	Instruction in memory	Instruction Comment
0000	0010 1010 0000 0100	Load contents from addr '0100 ₂ ' to reg A
0001	0001 1010 0000 0001	Add Immediate value '1' to reg A
0002	0100 0000 0000 0001	JUMP to memory address '0001'
0003	0000 0000 0000 0000	Contents of memory
0004	0000 0000 0000 0000	Contents of memory

The above program executes very quickly, by simply loading the value (zero) from memory address 0004 into register A, then increments the A register by 1 in an endless loop. This type of program is known as processor intensive, because it uses no I/O operations. One deficiency in the early computer designs was that they were inefficient performing I/O operations, because the processor controls all the I/O operations. If the program was generating printed output and the printer was out of paper, the processor would sit idle waiting for the print operation to complete, even if it took hours for an operator to load paper. As the computer progressed as a machine, this weakness was overcome. One way this was improved upon was due to the invention of control units. Control units are specialized processors external to the main processor that controls the I/O devices, which would offload all the I/O work from the main processor to the control unit. The main processor and the control unit would simply pass data back and forth.

Memory

In modern day computer terminology, the word “memory” has numerous meanings. In the von Neumann model, memory is one of the core fundamental components, closely coupled to the processor. Early memory was referred to as core, because it was made from ferromagnetic core, which was magnetized to retain a bit value. There are many terms used to describe memory such as: main memory, random access memory (RAM), read only memory (ROM), primary storage, and internal storage [5]. External or secondary storage refers to tape or disk storage external to the processor unit. In contemporary terms, memory is commonly referred to as form of solid-state storage. Since memory is electronic and resides so close to the processor, access times to write and retrieve data from memory are very speedy. One way to increase system performance is to increase the size of memory. One thing is for certain, that if it was not for memory, the computer could never retain a value, or store and execute a program. In fact, a computer would be reduced to the computing power of a simple calculator, without solid-state memory.

Memory is where programs and data are stored. The program instructions are retrieved from memory, decoded, and executed by the hardware. Memory used to be a very expensive component and therefore, early processors had little of it to utilize. Programs had to be written very efficiently in order to fit inside the amount of memory a particular processor had. If a processor had 16 K of main memory, programs had to be 16 K or smaller in order to run, this memory limitation was very constraining. In early machines, one revolutionary method to overcome the memory size limitation is called virtual memory (VM). Virtual memory looks at secondary storage, such as a disk drive, as main memory. If a processor only had 16 K of memory and a program was larger than 16 K in size, virtual memory broke up the larger program into smaller portions called pages. If the page size was 4 K, the machine would load the first 4 K page into memory. Then as the program executed it would load or swap additional pages when required. Simply speaking, the virtual memory concept artificially increases the apparent amount of main memory in the computer. Conceptually, virtual memory is like reading a book. If the reader is currently reading page one that would be known as the active page. (For this example pretend the reader can only retain one page at a time.) Now the reader can jump

to another page, swapping in that information. One could sequentially read from page to page, or could randomly jump to any page next.

The Bus

Specifically a bus is a set of wires that send and receive data. Most commonly, there are three buses: address, control, and data bus. The bus is used for transferring data among the components of a computer system. The easiest way to understand what a bus is and does is to think of it as a highway. A 8-bit wide bus is like a single lane road, it can only handle low traffic. A 16-bit bus is like a double lane road, a 32-bit bus is a four-lane highway, and a 64-bit bus is an eight lane superhighway. The more lanes a highway has, the more traffic it can handle. The same is true of a computer bus, the wider the bus, the more signals it can handle at any given interval. Overall this means, faster computing and greater throughput. The address bus works exactly like the data bus, except it only handles address traffic, and the data bus handles only data traffic. The control bus is for control functions. For instance the data and address bus can perform multiple functions depending on the status of the control bus. Furthermore, data and address buses are bidirectional, depending on the active control line at the time which determines which direction the data is moving. Control lines perform functions such as: Data In, Data Out, Memory Read, Memory Write, Address In, and Address out. The address bus plays another extremely important role, depending on the design of the address bus that determines the amount of memory that can be addressed or used. In the original PC's of 1981 they used an Intel 8086 chip that had a 20-bit address bus, which could only address a maximum of 1 megabytes of storage, verse a newer AMD Athlon 64 chip that uses 36 bit addressing, which can address 64 gigabytes of main storage.

On a side note the width of the processor, address bus, and data bus do not have to be the same size. For instance, a computer system could be designed using a 32 bit processor, because it will be used for mathematically intensive calculation, but the in order to keep costs down the design team chose a 16 bit I/O bus because 16 bit I/O device are less expensive that 32 bit I/O devices.

Clock

Another element that plays an important role is the clock, or oscillator. The clock is the main time reference that all events are coordinated or synchronized from. Everything from bus transfers to instruction fetch, decode cycles are synchronized using the clock cycle. This is the main pulse or heartbeat of the machine. Clock speed is something to be aware of when shopping for a new computer. In the early days of the PC, performance comparison of one computer to another could be determined by comparing the clock speed. Early PC clocks where 4.77 megahertz (MHz), so a newer model running at 10 MHz was twice as fast. System speed expressed in megahertz equals millions of cycles or instructions executed per second. A 4-gigahertz (GHz) system is running 4 billion cycles or instructions per second. What was once an easy number to quickly look at when shopping for machines, is not as accurate an indication any more, because we also have to consider the bus size. For example, a 2 GHz 64-bit machine, is twice as fast as a 2 GHz 32-bit machine.

Storage Devices

Historically, the original storage device was the punch card. This had many limitations because it could only store 80 characters of data, and was terribly slow. The two most prevalent types of external or secondary storage today are magnetic disk and tape storage. These devices have been used for over fifty years and since then many improvements have been made in tape and disk technology. Original tape devices used 12-inch reels of magnetic tape, and access time was slow because the data had to be read sequentially off the reel of tape. Years ago, this was a minor trade-off because the reels held much more data over cards. Disk storage was a massive improvement over tape, because it not only offered increased data storage capacity, but increased data accessing speed by utilizing random access methods. Basically, data storage methods are similar to audio or video storage used today. Tape operates much like videotape; one may have to view a whole videotape just to find a small segment at the end of the tape. Consequently, disk drives access data much like a photograph. If one needed to access the fourth song of a record, direct access allows the ability to skip directly to the fourth song. It's interesting to look back at the original technology of early tape and disk devices. Even though many improvements have been made and new technology such as optical drives, have been accepted, the original design methodologies are still currently used.

CISC vs. RISC

Early computers were referred to as, Complex Instruction Set Computers (CISC) and utilized very powerful and complex instruction sets. These machines were designed this way because main memory was so expensive. In having instructions set that combined operations yielded more computing power while using minimal memory resources. For example, many early processors were designed to do precision floating-point mathematics, which made the instruction set design and associated hardware more complex. This increased processor overhead even if floating point was not being utilized. In the 1970's the trend moved from CISC processors to Reduced Instruction Set Computers (RISC) [6]. Upon analysis of processor instruction set mix, it was discovered that most processors only used a small percentage of their instructions. By designing simpler processors, this greatly reduced the cost and increased the speed of the processor hardware. The trend caught on, and most processors today employ the RISC model.

Today

If we fast-forward to the technologies used today, not only have microprocessors become much faster, current computer systems incorporate many microprocessors in one system. In addition, microprocessors used for the main processor are also used in control units, attachment cards, and I/O devices. All devices in a computer system are now more intelligent and self-reliant, buses are wider and internal with all components in a closer proximity. By keeping everything close, speeds are increased dramatically when transferring data. Processor technology has evolved from machines that took up rooms to the size of a silicon chip smaller than a thumbnail. Overall, the ultimate goal is to manufacture a mainframe the size of a human cell.

References:

- [1] Stalling, William (1999). Computer Organization and Architecture. Upper Saddle River, New Jersey: Prentice Hall.
- [2] Wikipedia . Retrieved September 21, 2006, from Central Processing Unit Web site:
http://en.wikipedia.org/wiki/Central_processing_unit
- [3] Wikipedia. Retrieved September 21, 2006, from Instruction Set Web site:
http://en.wikipedia.org/wiki/Instruction_set
- [4] Wikipedia. Retrieved September 21, 2006, from Computer Architecture Web site:
http://en.wikipedia.org/wiki/Computer_architecture
- [5] Mueller, Scott (2004). Upgrading and Repairing PCs. Indianapolis, Indiana: Que.
- [6] Wikipedia. Retrieved September 21, 2006, from CPU Design Web site:
http://en.wikipedia.org/wiki/CPU_design

Chapter 4: Introduction to Operating Systems

The textbook definition of an operating system (OS) is the software that controls hardware resources such as memory, central processing unit (CPU) time, disk space, and peripheral devices [1]. The operating system is the foundation on which applications run under control of, as well as acting as an interface between the user and the hardware by allowing shared hardware and data among users. However one chooses to define them, operating systems are an integral part of the computer system and must be understood to some degree by all computer users.

In the beginning the first computers had no operating systems. Programmers simply input hand-coded machine instructions directly into memory in order to execute them. In the 1950s, the focus in design of operating systems was to simplify the job stream. The transition from one job to the next was difficult, and a great deal of time was required to make this transition. Computers from this era only performed dedicated processing, meaning the current job had total and dedicated control of the machine. This started the beginning of batch processing operating systems in which jobs were gathered in groups or batches. As each batch was processed, control was returned to the operating system and the operator could initiate the next batch process. In these uniprogramming environments, memory is divided into two parts, one for the operating system and the other for the program. These operating systems were crude at best, and only one job could be run at a time. If an urgent processing request was required, the current job would have to be terminated so the request could be initiated and executed. In examining Figure 1, one can see that memory has the operating system kernel and one user program loaded. The kernel is the core of the operating system which is loaded into memory. If the program is small, additional memory goes unused as seen in this example. After this application executes and terminates, a new application could then be loaded and executed.

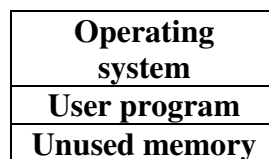


Figure 1. Single-user, dedicated-storage allocation

The next generation of operating systems overcame the deficiency of dedicated processing by utilizing multiprogramming. This ingenious methodology simply utilizes storage management techniques to trick the system into appearing to run multiple jobs simultaneously. One multiprogramming technique is called partitioning which is illustrated in Figure 2.

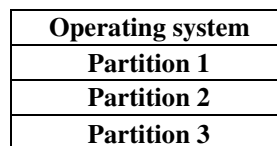


Figure 2. Fixed-partition multiprogramming

Partitioning allows memory to be carved up into smaller pieces called a partition. This enables a unique application to be loaded into each partition. This works because the operating system rapidly switches system resources between each partition for an increment of time. During this time, the active application has control over all system resources and appears to be a dedicated process running. In partitioning, the operating system acts as the traffic cop or gatekeeper. It opens the gate for each individual partition to have access to the hardware for a period of time, as illustrated in Figure 3. Multiprogramming operating systems are complex because they require all hardware registers and current processes status to be stored and recalled as they switch from one partition to the next. Registers are temporary high-speed memory internal to the CPU. In spite of this complexity, faster and more powerful hardware made this switching seamless, appearing nonexistent. To the user, it appears that all jobs are running simultaneously and gives the illusion that each job has exclusive access to the machine. This method greatly increased performance, by simply having multiple applications resident in memory, which simultaneously reduced job setup time. Partitioning can be implemented by using static fixed sized partitioning, so when a process is loaded into memory, it is placed in the smallest available partition that will hold it. The other more flexible scheme is variable-size partitions, when a process is loaded into memory it is dynamically allocated the exact amount of memory the process requires.

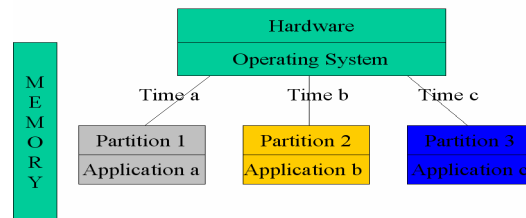


Figure 3. Memory partitioning

Partitioning led the way for more innovation such as, timesharing, which enabled users to interface via terminals to retrieve interactive data. Real-time processing also emerged, which provides immediate responses. Private industry and the military utilized real-time processing used in many manufacturing plant control functions, to monitor thousands of things concurrently. Partitioning increased system flexibility; many partitions could now run simultaneously with different applications such as interactive, batch, and production. With the proliferation of magnetic disk drives, many deficiencies in punch cards and sequential-access magnetic tape processing were overcome. The biggest issue with sequential-access devices like tape is that the current operation would have to complete before the tape could be rewound and another operation started. Random-access disks did not have this problem because the heads could rapidly move to different areas on the platters. A process could read from one area of a disk and another process could quickly write to another area of the same disk. Likewise disk drives were incorporated as buffer areas for I/O operations. This system process was called spooling. Spooling overcame I/O dependencies when outputting to slow printing devices. Spooling

simply gave the illusion that a file is sent to a slow printer when instead the file was written to a high-speed disk, which could be sent to a printer at a later time.

Process management

The above descriptions of operating systems are generalizations at best. There are numerous complexities with all aspects of operating systems. It has taken many years of development and refinement to get to where we are today. Process management is one area that is highly complex. A process is a program in execution, also known as a job or task, and is a unit of work in a system. There have been numerous developments in order to handle job scheduling. It was easy on dedicated systems because only one process was active at a time. The computer operator had total control over job scheduling. Job scheduling takes on a whole new dimension when one examines the many active processes running inside today's modern computers. How does the operating system know which one to execute, and in what order? There have been many mathematical algorithms devised to quantify different methods employed. We will explore some of these methods.

In addition when dealing with scheduling, jobs can be preemptively or non-preemptively scheduled. A task is non-preemptive when the task has the CPU allocated to it and is executing, the task cannot be interrupted until it is completed. Preemptive means the task can be interrupted before completion. Non-preemptive is used in extremely critical applications such as in a real-time environment. Priority methodologies have been used where jobs could be prioritized in the schedule based on importance. Some criteria to consider in process scheduling include (1) fairness, so that all processes are treated equally so that no one process can over-utilize resources; (2) maximizing throughput, by scheduling processes to keep the CPU always utilized; (3) balancing system resource use, trying not to over utilize some I/O while under utilizing other I/O; (4) avoiding indefinite postponement of processes; (5) avoiding process deadlocks and system bottlenecks [2].

One of the simplest methods used in scheduling is First-In-First-Out (FIFO), shown in Figure 4. This system executes jobs in the order they arrive in the job queue. FIFO is simple to implement but has drawbacks such as a long job would make short jobs in the queues wait. One method created to overcome this is Round Robin (RR) Scheduling. In Round Robin, processes come out of the job queue the same as FIFO, but they are given a limited amount of CPU time. Once this time slice is completed, the process goes back into the queue. See Figure 5 for an illustration of this. Shortest-Job-First (SJF) and Shortest-Remaining-Time (SRT) are two other ways to increase system output by having the shortest jobs run first.

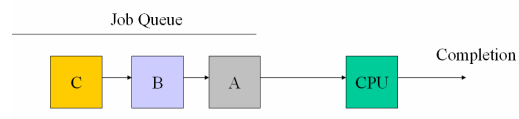


Figure 4. FIFO job scheduling

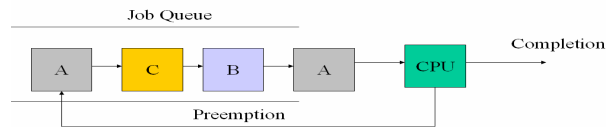


Figure 5. Round-robin scheduling

In short, we've reviewed only a few of the scheduling mechanisms and terminology. There is no definitive conclusion to draw in determining what methods are better than others. There is no one-size-fits all answer. In some applications, one method may work better than another to address a particular system performance issue. The truth of the matter is that most modern scheduling mechanisms are hybrids, incorporating the best features of multiple methods.

Storage Management

Storage management is another complex topic about which much could be written. To gain a basic understanding of storage management, we will touch on key topics. In dedicated processing, an application is loaded in contiguous memory space. This is simple because the only requirement is that the program size must fit the storage size. A flaw with this is that memory is expensive and as applications grew if they exceeded system memory size they could not load and execute.

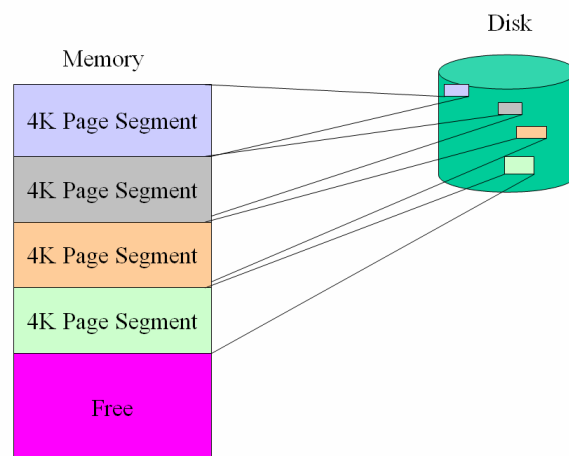


Figure 6. Virtual storage

Virtual storage solved this problem. Virtual storage provided the ability to address storage space larger than what is actually available in main memory. This virtualization theory takes the storage capacity of a hard drive and makes it appear as main memory by carving up memory and disk drive storage into smaller chunks called pages or segments. This is shown in Figure 6. Now processes would load into main memory (now call real storage), one page at a time. As the process executed, it would pull in additional pages as needed from disk or virtual storage. Dynamic address translation tables and additional hardware mechanisms kept track of all the virtual address pages. This unlimited artificial memory was transparent to the user, and it freed the programmer from worrying about

machine limitations in order to concentrate on application development. It is important to note that there is much more to virtual storage techniques. In fact, virtual storage management is very similar to process management. Many strategies are used to optimize page replacement such as, First-In-First-Out (FIFO), Least-Recently-Used (LRU), Least-Frequently-Used (LFU), and Not-Used-Recently (NUR) page replacement. Illustrated in Table 1 are the different methods of storage organization.

Table 1. Storage organizations [2]

Real	Real		Virtual		
Single user dedicated system	Real storage multiprogramming systems		Virtual storage multiprogramming systems		
	Fixed partition	Variable partition	Pure paging	Pure segmentation	Combined paging segmentation

Popular Operating Systems

OS360 & VM

In the 1960's, the massive development in hardware drove similar developments in operating systems. One of the most significant events was the announcement of the IBM system 360 hardware and operating system (OS360). This product was revolutionary and way ahead of its time. The OS360 was the first operating system that was truly upward compatible within the 360 processor series. Likewise the most radical innovation born out of this era was the virtual machine (VM). VM was introduced on the system 360 Model 67 processor [3]. The virtual machine concept was similar to partitioning, but instead of having a unique area in memory for different applications to reside in, virtual machine took partitioning to the next level. VM carved up areas in memory in order to have discrete operating systems executing. This essentially creates a number of sandboxes in memory for operating systems to play. These operating systems are completely isolated from one another, and each operating system can peacefully and concurrently coexist. One could be rebooting without affecting another. Through software virtualization, the user was given the appearance to have more than one physical hardware system installed. VM has evolved over many decades and has many new and improved features, it is still very popular today and used extensively. Today, it is popularly known as Logical Partitioning (LPAR). Some reasons to carve up a physical system are to isolate production and development systems, and isolate different time zones or customer regions. It is effectively used in a test environment it is common to install a new version of the OS or application software in a test LPAR before it putting it in production. LPAR is robust in an educational environment, where a college could carve up one physical system into a thousand logical systems, so students could be assigned their own virtual system. This is also extensively used in server consolidation, by implementing one small mainframe with LPAR, which can replace a room full of server racks.

Timesharing Systems

On a different development front in the 1960's, researchers and universities created a number of timesharing systems. The goal of timesharing was that users could share applications and data concurrently. One of the first such timesharing systems was developed at Massachusetts Institute of Technology. It was called the Compatible Time Sharing System (CTSS). This early CTSS evolved into a next-generation operating system called MULTIC (Multiplexed Information and Computing Service) [4]. This was a cooperative project led by MIT, Bell Laboratories, General Electric, and later, Honeywell. Although Multics was not successful and never reached a commercial product it had a major impact due to many new concepts such as high availability, dynamic linking, dynamic hardware reconfiguration, and use of daemon processes.

UNIX

In 1969, Bell Laboratories dropped out of the Multics project and moved on to develop Unix. Ken Thompson, and Dennis Richie developed Unix jointly. Richie had worked on the Multics project and Unix is highly influenced by Multics. The name Unix is merely a pun toward Multics. Unix evolved through a number of different rewritten versions running on different hardware. Version 3 was written in a high level language also developed by Richie called C [5]. By writing Unix in a high level language, this made it portable to different hardware platforms. Unix became widely used by Bell Laboratories and at many universities. Unix development bounced around Bell Labs and AT&T through the 1970's. Unix development was also emerging on the educational front at University of California at Berkeley, University of Illinois, Harvard, and Purdue. The strength in Unix came from an extremely efficient, powerful core called the kernel. On top of the kernel were tightly coupled modules that provided features and functionality to Unix such as text editor (vi), send mail and the Internet protocol (TCP/IP).

To illustrate today's popularity of Unix and Unix variants, Unix can be run on all hardware platforms from PC desktops, midrange systems, and the most powerful mainframes. Many variants are offered from many vendors such as AT&T, SUN systems, HP, Berkeley, and IBM just to name a few. The Internet is built off of Unix-based servers. Currently in September 2006 of the almost 97 million websites that comprise the Internet, almost 70% of the hosts are running Unix and Unix variants operating systems [6].

MS-DOS

Microsoft's Disk Operating System (DOS) was a single user, single task, operating system that was born in the beginning of the IBM Personal Computer craze. IBM outsourced to Microsoft to supply DOS, based on the 16-bit architecture of the PC instead of acquiring another popular 8-bit operating system at the time called CP/M. The original PC was based on Intel's 8088 microprocessor, capable of addressing 1 meg of memory. Of this 1 meg of memory, DOS supported 640 K for itself and applications [7]. This was a significant advancement at this time because previous systems only supported 64 K of memory. DOS used a Command Line Interface (CLI), meaning everything was done using text commands entered via the keyboard. It was named Disk Operating System because one of its major features allowed the use of floppy and hard disk drive

file management support. Even though it was not packed full of bells and whistles, it was a functional operating system. DOS gained huge success simply because it was the first one for the personal computer. DOS version 1.0 was released in August 1981, and there were at least ten other versions released until the last commercial version (7.0) was released in 1995.

Windows

Windows started out as a kinder gentler form of DOS. Window 1.0 was released in 1985, as a graphical interface (GUI) built on top of DOS. The premise was that users could load numerous applications concurrently into memory and then switch between them in the windows environment. This GUI revolutionized how the user launched applications one simply used a pointing device to maneuver in this environment. Windows has evolved and many new products and releases have been announced. Windows evolution and growth is similar to the early mainframe operating systems. As microprocessor technology caught up to the robustness of a mainframe processors, Windows grew to support these features. Windows has monopolized the desktop market share, and the Network Operating Systems such as NT.x, Windows 200x, running on today's powerful servers has cut into the midrange market share.

Linux

The Linux movement started in 1991 when student Linus Torvalds wrote the kernel from a Unix variant Minix. This powerful, well-written kernel was released and freely distributed under the GNU general public license and its source code is available to everyone [8]. This started a new revolution of worldwide collaboration into Linux operating systems and its many variants. Perhaps it was the anti-Microsoft sentiment that started this movement, but it has gained much momentum and has been embraced by many including major computer vendors. Its strengths include no cost, similar to Unix, and the kernel is so small and efficient it can run on any hardware. In less than fifteen years, Linux has grown from a hobbyist project to becoming a viable alternative for proprietary Microsoft and Unix operating systems.

Security Concerns

The term "secure operating system" maybe a misnomer, because what could be considered secure in one instance may not be in another. In 1983, the Department of Defense published a standard called the *Trusted Computer System Evaluation Criteria* (TCSEC). The TCSEC, frequently referred to as the Orange Book, is a standard that sets basic requirements for assessing the effectiveness of computer security controls [9]. The TCSEC standard has evolved over the years and is now replaced by the development of the *Common Criteria* international standard published in 2005 [10]. *Common Criteria* does not provide a list of product security requirements or features that products must contain. Instead, it describes a framework in which computer system users can specify their security requirements.

Some key items to consider when evaluating an operating system based on security implications include (1) mandatory security policy which enforces access control rules based directly on an individual's clearance, authorization for the information and the confidentiality level of the information being sought; (2) accountability which requires identification, authentication, and auditing mechanisms to recognize, verify, and audit the user activities; (3) assurance that the computer system must contain hardware/software mechanisms that can be independently evaluated to provide sufficient assurance that the system enforces the above requirements.

References

[1] Silberschatz, A, & Galvin, P (2004). *Operating System Concepts*. John Wiley & Sons.

[2] Deitel, H (1984). *An Introduction to Operating Systems*. Addison- Wesley Publishing Company.

[3] Virtual machine. Retrieved September 24, 2006, from Wikipedia Web site: http://en.wikipedia.org/wiki/Virtual_machine

[4] Multics. Retrieved September 24, 2006, from Wikipedia Web site: <http://en.wikipedia.org/wiki/Multics>

[5] UNIX. Retrieved September 24, 2006, from Wikipedia Web site: <http://en.wikipedia.org/wiki/UNIX>

[6] September 2006 Web Server Survey, Retrieved September 24, 2006, from Netcraft Web site: http://news.netcraft.com/archives/web_server_survey.html

[7] MS-DOS. Retrieved September 24, 2006, from Wikipedia Web site: <http://en.wikipedia.org/wiki/MS-DOS>

[8] Linux. Retrieved September 24, 2006, from Wikipedia Web site: <http://en.wikipedia.org/wiki/Linux>

[9] Trusted Computer System Evaluation Criteria. In *Wikipedia* [Web]. Retrieved November 12, 2006, from <http://en.wikipedia.org/wiki/TCSEC>

[10] Common Criteria. In *Wikipedia* [Web]. Retrieved November 12, 2006, from http://en.wikipedia.org/wiki/Common_Criteria

Chapter 5: Digital Electronics

The term “digital electronics” usually refers to the physical circuits of a computer system such as electronic gates, switches, relays, or transistors. However, these components comprise only one part of digital electronics for it is a subject that has a much broader meaning. It is a methodology of transforming logical statements into boolean expressions, whose meaning, in its simplest form, can be represented simply by the absence or presence of an electrical signal: an on or off. Boolean algebra deals with logic rather than calculating numeric values, and was developed by George Boole in the mid 1800s [1]. It is based on the idea that logical propositions are either true or false. “True” propositions correspond to the digital logic value or level of 1, while “false” proposition correspond to 0. These two logic states of 0 or 1 fit perfectly into the binary numbering system. In electronic logic, the logic level is represented by a voltage level. A digital circuit takes one or more logic level inputs and produces a single output voltage using a basic building block called a logic gate. There are seven fundamental logic gates used today. In combining or cascading these basic gates, one can construct complex digital circuitry used in today’s electronic devices.

Basics

Digital electronics is a conglomeration of many disciplines such as elementary logic, boolean algebra, and logic gates. We started learning elementary logic in grade school with simple mathematical word problems. In the language of logic, one could examine simple sentences to determine if they are logically true or false statements. We unconsciously use elementary logic daily. By examining a simple statement such as “Sue wears a hat and gloves when it’s cold. Today is cold”. One can conclude that Sue is wearing her hat and gloves. In fact, many of us use boolean expression without realizing it. If you use any search engines you can refine your searches by utilizing boolean operators. For example, consider using Google’s search engine to find a pair of bowling shoes. If one searches on “shoes” this produces 389,000,000 hits, while searching on “bowling” produces 97,000,000 hits. But if you search on “bowling shoes” this produces 8,370,000 hits. Google’s boolean default is “AND” which means search engine joins words using the “AND” operator. You can use other operators such as the “OR” operator or “NOT” operator to exclude words in your searches.

Boolean algebra

There are several basic boolean operations, which may be used alone or in combinations: logical inversion (NOT gate), logical multiplication (AND gate), logical addition (OR gate) [2]. Boolean algebra is expressed by using variables and operators, similar to mathematics.. Variables have values of 1 or 0, where 1 equals the logical state of “true” and 0 equals the logical state of “false”. The basic logical operations are AND, OR, and NOT, which are symbolically represented by a dot (\cdot), plus (+), and over bar ($\bar{\quad}$). Example 1 below illustrates boolean expression

Example 1 – Basic Logic Operations

$$A \text{ AND } B = A \cdot B$$

$$A \text{ OR } B = A + B$$

$$\text{NOT } A = \bar{A} \quad (A = \text{NOT } \bar{A})$$

Truth table

Truth tables are mathematical tables used in logical expressions to show all possible combinations of values that logical variables may have [3]. Truth tables are used to reduce basic boolean operations to simple correlations of inputs to outputs. To help understand Boolean or logic expression, one must understand truth tables. A truth table is a simple way to illustrate and express all the variable options of a logic expression. Shown in Table 1 below is a truth table for the boolean expression $A \text{ AND } B = C$.

Table 1. Truth table $A \text{ AND } B = C$

Input		Output
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

You read a truth table by rows, starting from the top row and reading left to right. The rows are read like this:

If A is 0 AND B is 0 then C is 0.

If A is 0 AND B is 1 then C is 0.

If A is 1 AND B is 0 then C is 0.

If A is 1 AND B is 1 then C is 1.

Furthermore, one could think of these statements in terms of true and false. For example, if A is true, AND B is true, then C is true. Essentially, 1 represents a true value, and 0 represents a false value. In digital electronics, truth tables are shown using ones and zeros. A zero represents no voltage and a one represents voltage. Also notice that A and B are referred to as the input side, and C is the output side. This is how a digital logic gate works. The input side of a gate can have one or many inputs, but only one output. In this example, if input A has voltage (logic state 1) AND input B has voltage (logic state 1), then output C has voltage (logic state 1).

Electronic Gates

An electronic gate is the fundamental building block of digital electronics logic circuits [4]. Gates are constructed from transistorized electronic circuits. However one does not need to understand the electronics behind the gate in order to implement them. Gates are simply used like building blocks, assembled and interconnected together to produce an output based on a boolean expression. There are seven basic gates used in digital electronics: AND, OR, NOT, NAND, NOR, XOR (exclusive OR) and XNOR (Exclusive NOR) [5]. Gates are defined in three ways: graphic symbol, algebraic notation, and truth tables. Tables 2 and 3 shown below have all seven basic gates illustrated. The first gate shown is the AND gate that can have two or more inputs. Its output is the logical AND of the inputs. Therefore, all inputs must be 1 in order for the

output to be a 1. With the OR gate, if one or more inputs are 1, it produces an output of 1. The inverter's function is to change a 1 input to a 0 output, or a 0 input to a 1 output. Hence, it inverts or complements the signal. An inverter has only one input and one output, and its symbol is a small triangle with a small circle on the output. The NAND gate takes the AND of its inputs, and then inverts the output. The symbol is an AND gate with the inversion circle on the output. The NOR gate takes the OR of its inputs and then inverts it. The symbol is an OR gate with the inversion circle on the output. The exclusive OR (XOR) is shown in Table 3. The logic behind the XOR gate is this: if either A OR B is 1, but NOT both, C is 1. Its symbol is an OR gate with an extra curved line in front. The exclusive NOR (XNOR) shown in Table 3 is similar to the exclusive NOR but with the output inverted. Its symbol is an XOR gate with the inverter circle on the output. Its logic is A or B equal NOT C, as shown in the truth table.

Implementation

To implement logical expressions into digital circuits, you simply substitute hardware gates for the given boolean expression [7]. For example to implement the expression $Q = (A + B)C + \bar{A}BC$, you simply substitute the logical operands for the digital gates. In this case $(A + B)C$ means input A is OR'ed (+) to input B, its output is the AND'ed with C, this result is OR'ed to the output of the expression $\bar{A}BC$, NOT A AND B AND C. The circuit is shown below in Figure 1.

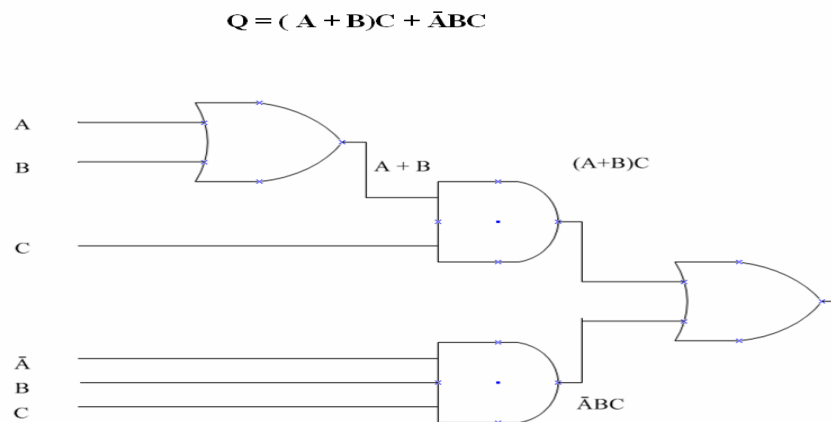


Figure 1. Implementation of an expression

Binary adder

In digital electronics, gates can be connected together to implement expressions from truth tables or boolean operations. Let's take it a step farther to see how binary addition works inside the hardware. Binary addition can be represented with the truth table as shown Table 4 below. Binary addition differs from boolean algebra in that the end result includes a carry. When adding two bits together they produce a result and a carry bit.

Table 3. Simple logic gates [6]

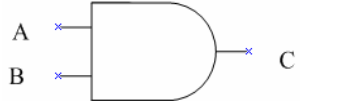
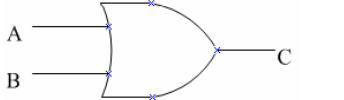
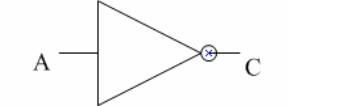
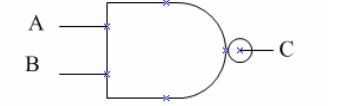
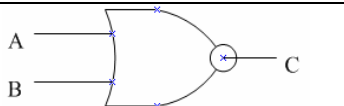
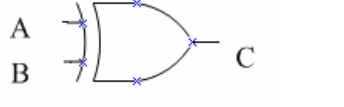
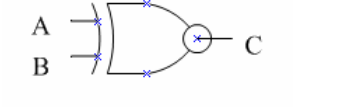
Name	Symbol	Algebraic Function	Truth Table															
AND		$C = A \cdot B$ or $C = AB$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	C	0	0	0	0	1	0	1	0	0	1	1	1
A	B	C																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$C = A + B$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	C	0	0	0	0	1	1	1	0	1	1	1	1
A	B	C																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
NOT		$A = \bar{C}$	<table border="1"> <thead> <tr> <th>A</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	C	0	1	1	0									
A	C																	
0	1																	
1	0																	
NAND		$C = (\overline{AB})$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	C	0	0	1	0	1	1	1	0	1	1	1	0
A	B	C																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$C = \overline{(A + B)}$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	C	0	0	1	0	1	0	1	0	0	1	1	0
A	B	C																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
XOR		$A + B = C$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	C	0	0	0	0	1	1	1	0	1	1	1	0
A	B	C																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
XNOR		$A + B = \bar{C}$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	C	0	0	1	0	1	0	1	0	0	1	1	1
A	B	C																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

Table 4. Single-bit adder

Input		Output	
A	B	Result	Carry
1	1	0	1
1	0	1	0
0	1	1	0
0	0	0	0

Input A is the first operand, B is the second operand, C is the carry bit, and R is the result. The truth table is read left to right. A plus B equals result R, with the Carry C. The rows read like this:

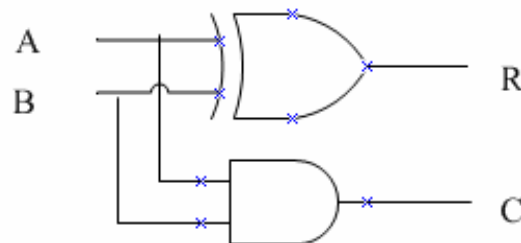
If A is 1 plus B is 1 then R is 0 and C is 1.

If A is 1 plus B is 0 then R is 1 and C is 0.

If A is 0 plus B is 1 then R is 1 and C is 0.

If A is 0 plus B is 0 then R is 0 and C is 0.

If you examine the truth table closely, you can implement a 1-bit adder by using 2 gates an exclusive OR and an AND Gate as shown in Figure 2 below.

**Figure 2. Logic gates for 1-bit adder**

In order to create a wider adder, it simply requires the use of more logic gates. This can be accomplished by cascading adders so that the carry bit from one adder is provided as the input to the next adder, and so on. This requires many logic gates to implement the 16 or 32 bit adder that is commonly used today. Today, a computer engineer does not have to worry about designing circuits using discrete components, because discrete circuits are placed on integrated circuit chips. One integrated circuit chip may have all the circuit functionality to perform a 16-bit add. Thus, an engineer can quickly implement circuits using a “black-box” approach by implementing integrated circuits.

References

[1] Boolean algebra. In *wikipedia* [Web]. Retrieved Oct. 2006, from http://en.wikipedia.org/wiki/Boolean_algebra

[2] Digital Circuit. In *wikipedia* [Web]. Retrieved Oct. 2006, from http://en.wikipedia.org/wiki/Digital_electronics

[3] Truth table. In *wikipedia* [Web]. Retrieved Oct. 2006, from http://en.wikipedia.org/wiki/Truth_tables

[4] Logic gate. In *wikipedia* [Web]. Retrieved Oct. 2006, from http://en.wikipedia.org/wiki/Logic_gate

[5] Brain, Marshall How Stuff Works. Retrieved October 16, 2006, from How Boolean Logic Works Web site: <http://computer.howstuffworks.com/boolean1.htm>

[6] Greenfield, Joseph D. (1977). *Practical Digital Design Using IC's*. Canada: John Wiley & Sons, Inc.

[7] Stallings, William (1999). *Computer Organization and Architecture: Design for Performance*. Upper Saddle River, New Jersey: Prentice Hall.

Chapter 6: Introduction to DOS

In order to gain a fundamental understanding of any operating system such as MS-DOS or PC-DOS, one must have a basic understanding of the hardware. This enables one to know how the operating system and hardware interface with each other. Hardware and software are two different entities that are so closely intertwined, it can be difficult to distinguish between them at times. On one end of the spectrum, we have application software and on the other, we have hardware. The gray area between the two is called the Basic Input Output System (BIOS). It is also the layer between the physical hardware and the operating system. BIOS is special programming code in that it does not get loaded into memory like application programs, but instead it is permanently written or burned into Read-Only Memory (ROM) chip that resides on the system board. (Code that is burned onto memory chips is also referred to as firmware.) BIOS is a core component and is the reason that different operating systems can be loaded onto the same hardware platform. As long as the operating system is designed to be BIOS compatible, it will run. Application programs don't need to know specifics about the hardware because they interface with the operating system, which interfaces with BIOS, which in turn interfaces directly to the hardware. A PC system can be described as a series of four layers, with hardware, and software, as seen in Figure 1. Using this layered approach, applications are transparent to the hardware platform it runs on.

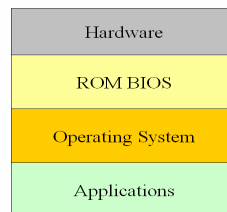


Figure 6. Four layers of a PC

DOS is a single-user, single-task operating system that consists of two primary components; the input output system and the shell. The I/O system is program code that is loaded into low memory on startup and controls the interaction of the I/O devices on the system. The shell is the computer user's interface. Its primary purpose is to interpret commands and translate them into actions. To explore how an operating system works, let's start from the beginning. In the PC realm, this start-up process is called booting on other platforms it is also known, as an Initial Program Load (IPL). Booting is the chain of events that occur from when power is first applied to the final step, when the operating system is completely loaded.

How DOS Loads and Starts (Simplified Version)

Step 1. Power on, after power supplies stabilize, hardware sends Power_Good_Signal to system board / microprocessor.

Step 2. Microprocessor reads and executes ROM BIOS code. ROM BIOS starts Power On Self Tests (POST). POST tests the basic functionality of the PC to determine critical components are functioning so the OS can load. Memory and video are critical and tested. If a POST test fails it reports the error using beep codes, because the video system may not be started yet. Successful completion of POST is indicated by a single beep.

Step 3. BIOS searches for a boot record at cylinder 0, head 0, sector 1 on the default boot device and then alternate boot devices such as; diskette, disk, CD, DVD, or network drive. The Master Boot Record (MBR) is loaded into memory. This is also known as the bootstrap loader.

Step 4. Core OS files loaded into memory and executed. (IO.SYS, IBMBIO.COM) All initialization code loaded and executed. The DOS file system is now active.

Step 5. CONFIG.SYS file loaded and executed. This file defines devices and loads software drivers associated with them.

Step 6. COMMAND.COM is loaded. This is the user shell. If AUTOEXEC.BAT file is present COMMAND.COM loads and executes it. AUTOEXEC.BAT is a startup file you can specify what applications you want automatically started on boot. This is when you see the Command Line Interface (CLI), otherwise known as the **A:\>** or **C:\>** Prompt. The system is now ready to accept commands.

Command Essentials

Once DOS completes booting, it sits at the command prompt (A:\>) waiting for commands to execute. There is a specific set of rules or syntax commands must follow. The problem is that there are many different commands. Some have no options while others have many options. It is simply a matter of becoming familiar with commonly used commands. Generally, DOS command begins with a keyword or name of command, and can be followed by parameters or arguments, then command line switches. Commands generally follow the format of:

KEYWORD Drive, File, Switches

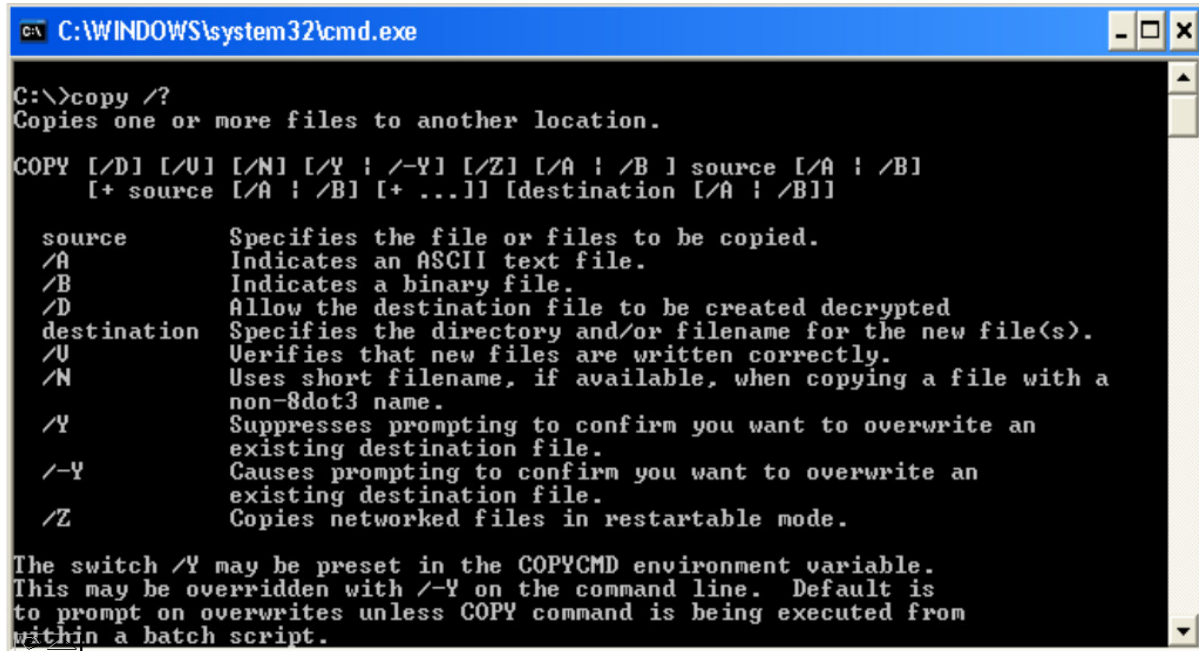
Or

KEYWORD SourceFiles, File Switches

To understand all the commands and options one can either consult a command reference book or use the DOS built in help feature. To invoke help one can simply type the command followed by “/?” . (**KEYWORD** /?) or type “help” before the keyword. Figure 2 shows a help screen example for the COPY command. All commands follow a specifically defined format notation.

- You must enter any words shown in capital letters (keywords are represented this way).
- You must supply any items shown in italic letters
- Items in square brackets [] are optional.
- Items separated by a bar (|) means you can enter one of the other (Example OFF | ON)
- An Ellipsis (...) indicates you can repeat as many time as you want.
- You must include all punctuation, except square brackets and vertical bars.

File switches are entered by using the forward slash (/). A switch is a parameter that turns on the optional function of a command. For example the DIR command lists files in the current directory. Now, if you enter DIR /S this switch will show all files not only in the current directory but in all subdirectories.



```

C:\WINDOWS\system32\cmd.exe

C:\>copy /?
Copies one or more files to another location.

COPY [/D] [/U] [/N] [/Y | /-Y] [/Z] [/A | /B ] source [/A | /B]
      [+ source [/A | /B] [+ ...]] [destination [/A | /B]]

source       Specifies the file or files to be copied.
/A           Indicates an ASCII text file.
/B           Indicates a binary file.
/D           Allow the destination file to be created decrypted
destination Specifies the directory and/or filename for the new file(s).
/U           Verifies that new files are written correctly.
/N           Uses short filename, if available, when copying a file with a
             non-8dot3 name.
/Y           Suppresses prompting to confirm you want to overwrite an
             existing destination file.
/-Y          Causes prompting to confirm you want to overwrite an
             existing destination file.
/Z           Copies networked files in restartable mode.

The switch /Y may be preset in the COPYCMD environment variable.
This may be overridden with /-Y on the command line. Default is
to prompt on overwrites unless COPY command is being executed from
within a batch script.

```

Figure 2. Copy /? Command

The Heart of DOS

The main propose of DOS is to interface with a disk hence the name Disk Operating System. The disk can be either a floppy diskette or a hard drive. In order to understand DOS's role and file system basics, one must first understand the hardware. A disk is simply a circular platter made of either metal or plastic that is covered with a magnetic material. Diskettes are plastic single-platter devices and hard drives can have multiple platters. In either case, there is a read write head assembly that is used to access areas on the disk. The circular platter is organized into a number of concentric rings, called tracks. Tracks are divided into smaller pieces called sectors. (See Figure 3.) The number of tracks and sectors a platter has determines the amount of data the device can store. The original diskettes were designed to store the same amount of data that a box of punch-cards could hold.

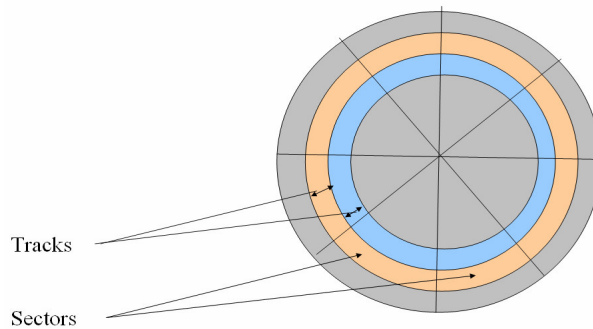


Figure 3. Diskette layout

Here is closer look into how data is organized on a platter. First, data is read or written one block at a time. The size of the data block usually is the same size as a sector. Early sector sizes were 80 bytes, the same as an 80 character punch-card. Currently common sector sizes for diskettes are 512, or 1024 bytes. DOS looks at data on a disk not in terms of tracks and sectors, but in a continuous array of logical sectors. Logical sectors are numbered starting with the outermost track with sector 0, continuing sequentially to the innermost track. Data is organized basically just like a written book is structured. In a book, the table of contents points to the specific chapter where the data is located. (Table 1 shows the layout of data on a diskette.)

Table 1. Diskette sector layout [2]

Logical Sector Number on diskette	Use	
0	Reserved for bootstrap loader	
1-6	FAT 1	File allocation Tables (FATs)
7-12	FAT 2	
13-29	Directory	
30-2001	Data Area	

The file allocation table (FAT) is the master index of all files on the disk similar to a table of contents in a book. There are two copies of the FAT table because it is so important [5]. The directory is a table that contains information about each file such as file name, size, date and time of creation, data and time when the file was last written, and any special attributes. The rest of the disk is for the data area. When a file is created or saved, three things occur: 1) An entry is made into the FAT table to indicate where the file is stored in the data area of the disk. 2) A directory entry is made with the file name, size, and link to the FAT table, and other information. 3) The data is written to the data area. One item to observe is that a file usually has a minimum size it cannot be one logical sector in size. Diskettes use the term, allocation unit, which assigns a number of logical sectors to a file. Hard drives use the term clusters to group a number of logical

sectors together for the minimum file size. The number of logical sectors in a cluster or allocation unit can vary depending on the format density of the media.

DOS introduced multi-level hierarchical directories which allowed for better organization of files, and multiple directories that allowed many more files to be stored on the hard disk. The top layer of the directory structure is called the root, and supports a number of sub directories, below it. This allows for greater flexibility of file naming over a flat directory structure. See Figure 4. Under the root directory, there can be multiple levels of sub directory. Sub directories simply help keep your files organized. Think of a directory as a kitchen drawer, one could organize towels, silverware, and junk in different drawers. Similarly one could create different directories for different regions, or for different files types such as; text, program, or data.

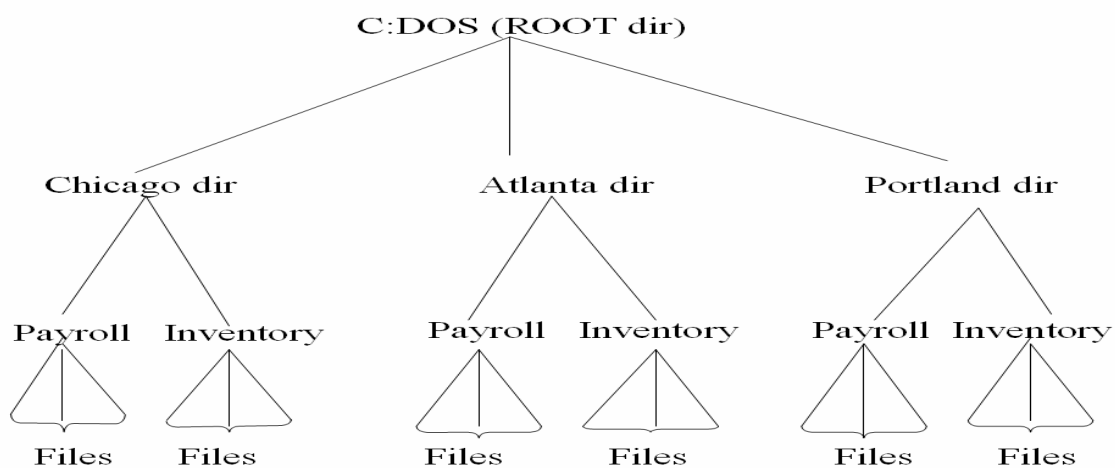


Figure 4. Directory structure

Path Concepts

DOS identifies the drive component by the drive letter, followed by a colon, and then a backslash. (A:\) DOS continues this convention to distinguish subdirectories. For example in the path expression; A:\DOS\GAMES\PACMAN, this tells DOS to look for files starting on the A drive, moving into the DOS directory, moving into the GAMES directory, into the PCMAN directory. A path expression gives DOS directions to directories. DOS provide the command MKDIR or MD (make directory) to create new directories. The Change directory command (CD) provides a way for one to change from one directory to another. If your currently at the A:\ > prompt you are at the root directory. Issuing a CD \GAMES command will move you into the A:\GAMES> directory. Delete directories with the command RMDIR which literally stands for “remove directory”. Its syntax is the same as the MKDIR command which can be used to crate a directory.

File System Basics

File naming conventions follow the eight-dot-three pattern. The first part of the name can be eight characters or less, followed by a period, and then a three character file extension. (See Figure 5.) From a security point of view, it is important to understand what happens with file deletion. When a file is deleted two things occur the FAT table entry is zeroed out, and the first character of the directory entry filename is changed to a special character. (Note nothing is done to the physical file.) By removing the entry in the FAT and directory tables, the operating system knows it can use that free space. But with the physical file never being touched, it is easy to recover. To recover the file all that needs to be done is to relink the FAT table to the data area where the file is stored, and change the directory entry filename back to a legal character.

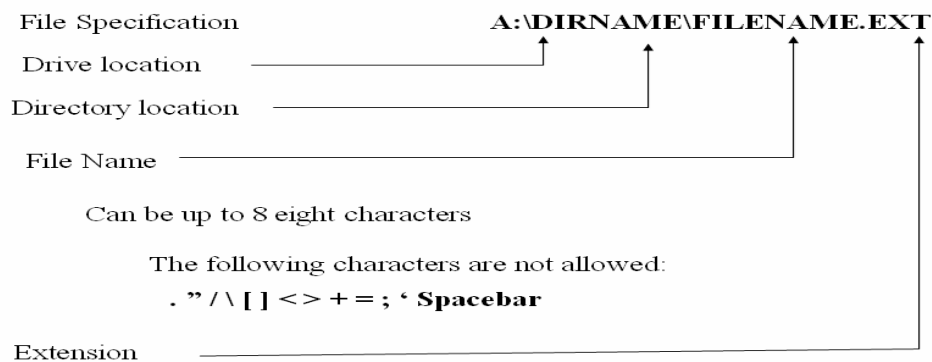


Figure 5. File naming specifications

Common Commands

Listed below are some of the more commonly used DOS commands [6].

DATE -Enters or displays date.

TIME - Enters or displays time.

DIR - Displays a list of files and subdirectories in a directory.

CLS - Clears your screen.

CD – Change Directory

VER – Displays version number

TREE – Display all directory paths

MKDIR – Creates a subdirectory

RMDIR - Removes a subdirectory

COPY - Copies files

SORT – Sorts files

ERASE – Deletes files

FORMAT – Formats diskettes / disks

RENAME – Renames files

VER – Displays version of DOS

Redirecting Commands

DOS defaults to accepting standard input from the keyboard and standard output to the video display. DOS also has the ability to redirect this data by using a process called piping.

Three symbols are used to accomplish piping: |, <, >. Think of the greater than symbol (>) as the sent to, the less than symbol (<) as the receive from, and the vertical bar (|) is the pipe symbol allows commands to be chained together.

<code>DIR > PRN</code>	//redirects a DIR command to the printer
<code>DIR > myfile.txt</code>	//redirects a DIR command to a file
<code>myfile.txt < input.txt</code>	//redirects file input.txt to myfile.txt
<code>myfile.txt SORT sorted.txt</code>	//myfile.txt gets sorted then piped to sorted.txt file

Global Filename Characters

Two special characters, ? and * can be used as wildcards. The question mark indicates that any character can occupy that position, and the * character indicates that any character can occupy that position and all remaining positions in the filename or extension.

Reserved Device Names

Reserved device names are used for common input output devices. They are useful with redirect and piping commands. A list of reserved device names appears below in Table 2.

Table 2. Device names

Reserved Name	Device
CON	Console keyboard Screen
AUX or COM1	1 st Com Port
COM2	2 nd Com Port
LPT1 or PRN	1 st Parallel Printer
LPT2 or LPT3	2 nd & 3 rd Parallel Printer
NUL	Nonexistent

The Merits of DOS

Although DOS has fallen out as a main stream operating system, there is still has merit in learning it. Windows was originally built off DOS, and therefore many of the same qualities carry over to a Windows environment. Furthermore, the operating system, Linux is gaining popularity today. Linux has improved on many of DOS's short comings. For example, Linux supports multi-users, multi-tasking, and improved security features. Interesting enough, many of the DOS commands are similar to Linux commands.

References:

[1] Mueller's, Scott (1999). *Upgrading and Repairing PCs Eleventh Edition*. Indianapolis, Indiana: QUE.

[2] Patterson, Tim (1983,June). An Inside Look at MS-DOS. *Byte Magazine*, Retrieved Sept 25,2006, from <http://www.patersonstech.com/Dos/Byte/InsideDos.htm>

[3] Conventional memory. In *Wikipedia* [Web]. Retrieved Sept 25,2006, from http://en.wikipedia.org/wiki/Conventional_memory

[4] Wyatt, Allen, & Tiley, Edward (1993). *Using MS-DOS 6.2*.Que Corporation.

[5] File Allocation Table. File Allocation Table. In *Wikipedia* [Web]. Retrieved Oct 1, 2006, from <http://en.wikipedia.org/wiki/FAT32>

[6] Gookin, Dan (1998). *DOS for Dummies 3rd edition*. Foster City, CA: IDG Books Worldwide, Inc.

Chapter 7: Introduction to Linux

The Revolution

On the surface, Linux is a new, revolutionary open source Unix-based operating system. It is extremely popular today, and is quickly becoming a household name. Linux is similar to the Internet in that it is not an overnight success. Like the Internet, Linux too is also more than thirty years in the making. The story begins in 1969 with the birth of Unix, AT&T's flagship operating system began. Unix not only became popular in the AT&T realm, but in the world of education as well, due to the fact it was written in the high level language C. AT&T permitted free academic access to the source code and by 1976, it was being taught in operating system classes at the university level. As university popularity caught on to Unix, AT&T pulled the source code from universities texts because it revealed too much of Bell Lab's proprietary code.

The start of collaborative programming arrived with the academic variant of Unix known as Berkeley Systems Distribution (BSD), started at the University of California. As the BSD collaborative project reached worldwide popularity and proportions, so did the lawsuits between AT&T and the university, over intellectual property rights to the code during the late 1970's and into the 1980's [1]. In the midst of this legal mess, Richard Stallman of Massachusetts Institute of Technology wrote a portable operating system that was free from the constraints of intellectual property rights. The operating system was named GNU meaning, "GNU's Not Unix", and was licensed under the GNU General Public License (GLP) agreement. This started the birth of the free software movement. Around 1990, the GNU project consisted of a suite of programming tools such as editors, compilers, and file utilities that could be freely compiled and used on any Unix system. The GNU movement had developed everything, except the kernel.

At this time, Unix was gaining popularity on the commercial front with corporate customers. Sun, Hewlett-Packard, and IBM offered proprietary Unix-based operating systems. In 1991, and Finnish university student Linus Torvalds enters the picture [2]. Torvalds started a hobbyist project to teach himself the C programming language and operating system concepts. He originally used Minux, a system similar to Unix, that was used for teaching operating system design. His search for a Minux replacement is where the Linux kernel was born. He published the kernel's source code on the Internet for public collaboration and by adapting GNU libraries to Linux kernel, created a full function free GNU/ Linux operating system.

Why Learn Linux?

With almost 70% of the servers on the Internet being Unix or a Linux variants, this makes the knowledge of Linux important, and offers more flexibility for security professionals [3]. In addition to requiring Linux knowledge to secure Linux servers, Linux provides the security professional with an unlimited number of security tools and utilities they can utilize. In fact, with the Linux kernel being so small in size, distributions fit right onto a CD and some can even be modified to fit on a USB memory stick. This enables the Linux operating system to self boot, without requiring installation on the hard

drive. This offers flexibility when a system won't start, since one can simply boot off a stand alone CD. Many security tools such as network detection, penetration testers, and forensic tools are simply booted from the CD in order to execute.

The Linux operating system has evolved into a true multi-user and multi-tasking environment. Today, it has blended between hardcore command-line Unix type, and the point-and-click GUI type. It can be used in a simple text form, (small and powerful operating systems utilizing only a command line interface), or used with one of the many window-like GUI interfaces, such as KDE, or GNOME. Linux is truly a reliable, powerful operating system, which makes it a considerable option today. It is also affordable, with its strongest selling point being the cost.

Distributions

Linux comes in many different flavors, called distributions, or “distro” for short. In fact, currently there are more flavors of Linux than of ice cream. Technically, Linux is only the core, or kernel, of the operating system, which is by managed by creator Linus Torvalds and team. Linux distributions are prepackaged collections of software, generally comprising the kernel, and other software such as: compilers, shells, utilities, graphical user interface (GUI), and other applications. Thus, a Linux distribution includes the Linux kernel, plus bundled software that comprises the operating system. A distribution can be comprised of strictly open source (free-of-charge) applications, or commercial products, such as Red Hat and SuSE. A distributor (such as Red Hat or SuSE) charges for their distribution because they bundle it with service, custom software, and support features.

A Google search in (October 2006), provides an indication of the popularity of the different distributions of Linux. Table 1 shows the number of hits each distribution produced.

Table 1. Distribution popularity based on Google search

Distribution	Google hits
Linux (generic)	806,000,000
Red Hat	163,000,000
SuSE	60,800,000
Gentoo	30,000,000
Mandrake	23,600,000
Slackware	21,800,000
Knoppix	15,800,000
Debian	9,870,000
Turbolinux	3,610,000
Windows	104,000,000
DOS	697,000,000

New Linux variants that are increasing in popularity are the live versions. Live distributions boot and run completely from CD, not requiring hard drive installation. Knoppix is an example of a live distro [4], which has the operating system and numerous applications compressed and loaded onto a CD. One simply boots the CD and within minutes, has a working desktop at their fingertips. This is excellent for someone who desires to learn Linux. There are also many security tools that are built off a live distro. In utilizing a live distro, a security professional can boot the CD and easily have hundreds of security applications available. Also Knoppix is useful in recovering operating systems that are corrupt and cannot boot. One can simply perform an alternate boot from the Knoppix CD and try to mount and recover the corrupt file system.

File system

Linux uses a hierarchical file system, similar to Unix and Windows. If you picture a tree upside down, it looks like the file structure with the roots at the top and the branches hanging downward. The top directory is fittingly called the “root” and is represented by “/”. This tree is just like a family tree, with the directory above your current directory, referred to as the “parent”, while the directories below are known as the “child” directories. Linux treats everything as a text file and uses a directory structure to keep all system files and devices organized. (See Figure 1). Due to the fact that Linux supports many additional features, this directory structure is much more complex to support additional kernel modules and multi-users.

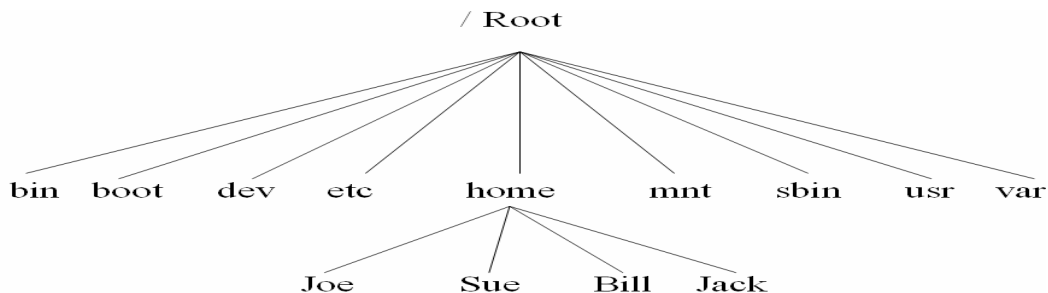


Figure 1. Basic Linux directory structure

Directory contents include:

- /bin Common commands.
- /boot Files used at boot time, LILO, or GRUB.
- /dev Files that represent devices on the system.
- /etc Administrative files and scripts.
- /home Directory for each user on the system.
- /mnt Provides mount points for external and removable file systems.
- /sbin Administrative commands and process control daemons.
- /usr Contains local software, libraries, games, etc.
- /var Logs.

Another concept used with the file system, is that of relative versus explicit paths. When you provide a pathname to a command or file, including a “./” in front of the command, it means explicit path. This explicitly lets you define the starting point. If you begin, a pathname without a “./” indicates that your path starts in the current directory and is referred to as a “relative pathname”.

File and user permissions

The word “root” in Linux can mean two things. The root user is the administrator of the system. The root directory is the top of the tree. All users have a log-in name and password in the system. Users have a defined environment in which to work. Unique home directories are assigned to all users, which protects and secures users from one another. The “root” user has the highest system permissions available. Another name for the root user is the super-user because the root user can read and write in any directory, along with being able to change system characteristics. There are three entities concerning file permissions or authorities; owner, group, and other. Every file has an owner, the creator of the file. Users belong to one or more groups of users that are defined in the system, which comes with predefined groups, such as “system” or “guest”. The designation of other allows anyone to access specific files.

Each directory and file has permissions set individually for the file owner, group, or other. These individual permissions are, read, write, and execute. The read permission allows users to view the contents of the file write permissions allow modifications and deletion of a file, and the execute permission allows the program file to run or execute. To display file permissions from a command line, use the list command demonstrated below.

```
ls -l
-rw-r--r-- 1 joey system 3894 oct 6 15:20 chap7
```

If you look at the first ten characters, you have a dash (-) followed by nine more characters. The first character describes the file type, a dash (-) indicates a regular file, a d indicates a directory, and a l means it is a link. The next nine characters indicate file permissions. They are broken down into groups of three, as shown in Figure 2. In the above example file chap7, the owner has read and write permissions, but restricts group and others to read-only permissions.

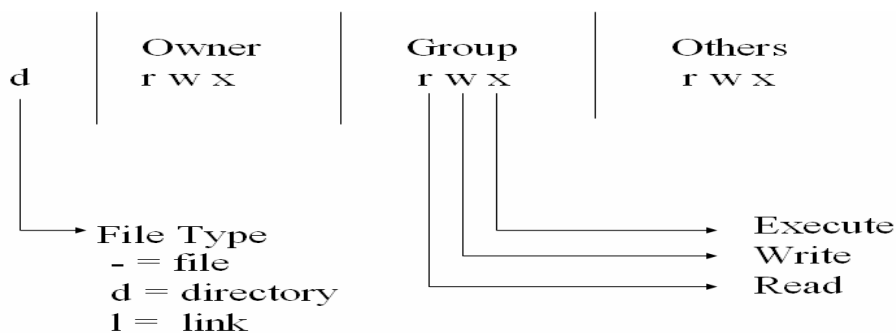


Figure 2. File permissions

Universal naming conventions

Linux uses a logical method to name partitions and devices. Most desktop computers use an IDE interface for the hard drives and CD-ROM drives. Linux refers to the primary master and primary slave IDE devices as /dev/hda and /dev/hdb, respectively (hda = hard drive a). On the secondary IDE interface, the master is called /dev/hdc and the slave is /dev/hdd. If the computer contained other IDE interfaces they would continue to be named sequentially such as, hde,hdf,hdg, etc. SCSI devices follow the naming convention of /dev/sda, /dev/sdb, etc (sda = SCSI device a) in the order of appearance on the SCSI chain. Partitions are named after the disk on which they are located,. The primary or extended on a master IDE drive is named /dev/hda1 through /dev/hd4, when present. Logical partitions are named /dev/hda5, dev/hda6 etc in the order of their appearance in the logical partition table. SCSI devices follow this same naming convention as seen in Table 2.

Table 2. Universal naming conventions [5]

Devices	Linux name	
floppy disk drive	/dev/fd0	
second floppy disk drive	/dev/fd1	
parallel ports	/dev/lp0 /dev/lp1	
serial ports	/dev/ttyS0 /dev/ttyS1	
	IDE	SCSI
First disk drive (master,IDE-0) - primary partition - extended partition with three logical partitions	/dev/hda /dev/hda1 /dev/hda5 /dev/hda6 /dev/hda7	/dev/sda /dev/sda1 /dev/sda5 /dev/sda6 /dev/sda7
second fixed disk drive - primary partition - second primary partition - extended partition with two logical partitions	/dev/hdb /dev/hdb1 /dev/hdb2 /dev/hdb5 /dev/hdb6	/dev/sdb /dev/sdb1 /dev/sdb2 /dev/sdb5 /dev/sdb6

Summary of Linux

Table 3 summarizes some of the most popular and helpful Linux commands. A demonstration of using some of these commands follows the table. After experimenting with these commands for a few days, you will become proficient in using them. You may even begin to prefer using the command line to a graphical user interface, since many “experts” feel they can be more productive and get things done more quickly by simply typing commands.

Table 3. Summary of Linux commands [6]

Command	What it does
<code>cd <path></code>	Change the current directory.
<code>mkdir</code>	Create a new subdirectory.
<code>rmdir</code>	Delete an empty subdirectory.
<code>rm <filename></code>	Removes a file.
<code>cp <filename></code>	Copies file.
<code>mv <filename> <pathname></code>	Moves file to the directory.
<code>ls -l</code>	List files in current directory long format
<code>pwd</code>	Print current directory (where am I)
<code>chmod</code>	Changes file permissions
<code>man <command></code>	Online help
<code>cat</code>	Display the contents of a file.
<code>shutdown -h now</code>	Shutdown system now.
<code>reboot</code>	Reboot.
<code>ps</code>	Show processes
<code>Kill <pid></code>	Terminate process
<code>df</code>	Disk free space status

Examples of Linux Commands

```
linux:/home/jayme # ls /*This lists files in directory*/
.
..
.adobe
anger.tar.gz
.bash_history
.bashrc
bin
brutus-aet2.zip
crack5.0.tar.gz
.DCOPserver_linux_0
.DCOPserver_linux_:0
default.pwl
Desktop
.dmrc
Documents
.dvipsrc
.emacs
.kde
.kermitrc
klaxon.tar.gz
LB1.jpg
LB2.jpg
LB3.jpg
LB6.jpg
LB7.jpg
LB8.jpg
LBSurfer.jpg
linux30.jpg
linux3.jpg
linux42.jpg
linux9.jpg
linux6.jpg
.macromedia
.mailcap
.profile
psad-1.4.3.tar.gz
public_html
.qt
.recently-used
samba-latest.tar.gz
samba-tng-0.4.99
samba-tng0.4.99.tar.gz
sara-6.0.7
sara-6.0.7.tgz
satan-1.1.1
satan-1.1.1.tar.gz
scanlogd-2.2.5.tar.gz
shadow
.shel
.ssh
.sversionrc
```

```
.exrc .mcpop .thumbnails
.fonts .mime.types .urlview
.fonts.cache-1 Misc .viminfo
.fonts.conf .mozilla .Xauthority
.gconf .muttrc .xcoralrc
.gconfd netcat-0.7.1 .xemacs
.gnome netcat-0.7.1.tar.gz .xim.template
.gnome2 nmap-3.93-1.src.rpm .xinitrc.template
.gnome2_private OpenOffice.org1.1 .xsession-errors
.ICEauthority opie-2.4.tar.gz .xtalkrc
john-1.6 portsentry-1.2.tar.gz
john-1.6.tar.gz portsentry_beta
linux:/home/jayme #
```

```
linux:/home/jayme # ifconfig /* Displays network interface */
```

```
eth0 Link encap:Ethernet HWaddr 00:C0:4F:0D:43:F5
inet addr:192.168.1.104 Bcast:192.168.1.255 Mask:255.255.255.0
inet6 addr: fe80::2c0:4fff:fe0d:43f5/64 Scope:Link
UP BROADCAST NOTRAILERS RUNNING MULTICAST MTU:1500 Metric:1
RX packets:1118 errors:0 dropped:0 overruns:0 frame:0
TX packets:903 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:1347711 (1.2 Mb) TX bytes:132002 (128.9 Kb)
Interrupt:11 Base address:0xdc00
```

```
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:74 errors:0 dropped:0 overruns:0 frame:0
TX packets:74 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:5231 (5.1 Kb) TX bytes:5231 (5.1 Kb)
```

```
linux:/home/jayme # ifconfig > textfile.txt /* Pipes the ifconfig into a text file */
```

```
linux:/home/jayme # ls /* Lists files notice new textfile.txt */
. .kde .profile
.. .kermrc psad-1.4.3.tar.gz
.adobe klaxon.tar.gz public_html
anger.tar.gz LB1.jpg .qt
.bash_history LB2.jpg .recently-used
.bashrc LB3.jpg samba-latest.tar.gz
bin LB6.jpg samba-tng-0.4.99
brutus-aet2.zip LB7.jpg samba-tng-0.4.99.tar.gz
crack5.0.tar.gz LB8.jpg sara-6.0.7
.DCOPserver_linux__0 LBSurfer.jpg sara-6.0.7.tgz
.DCOPserver_linux_:0 linux30.jpg satan-1.1.1
default.pwl linux3.jpg satan-1.1.1.tar.gz
Desktop linux42.jpg scanlogd-2.2.5.tar.gz
.dmrc linux9.jpg shadow
Documents linux6.jpg .skel
.dvipsrc .macromedia .ssh
.emacs .mailcap .sversionrc
.exrc .mcpop textfile.txt
.fonts .mime.types .thumbnails
.fonts.cache-1 .Misc .urlview
.fonts.conf .mozilla .viminfo
.gconf .muttrc .Xauthority
.gconfd netcat-0.7.1 .xcoralrc
.gnome netcat-0.7.1.tar.gz .xemacs
.gnome2 nmap-3.93-1.src.rpm .xim.template
.gnome2_private OpenOffice.org1.1 .xinitrc.template
.ICEauthority opie-2.4.tar.gz .xsession-errors
john-1.6 portsentry-1.2.tar.gz .xtalkrc
john-1.6.tar.gz portsentry_beta
linux:/home/jayme #
```

```
linux:/home/jayme # cp textfile.txt iptextfile.txt
```

```
linux:/home/jayme # ls /* Copies textfile.txt to iptextfile.txt */
/* Lists files notice new file iptextfile.txt */
```

```

.                               john-1.6.tar.gz                portsentry_beta
..                              .kde                          .profile
.adobe                          .kermrc                       psad-1.4.3.tar.gz
anger.tar.gz                    klaxon.tar.gz                 public_html
.bash_history                   LB1.jpg                       .qt
.bashrc                          LB2.jpg                       .recently-used
bin                              LB3.jpg                       samba-latest.tar.gz
brutus-aet2.zip                 LB6.jpg                       samba-tng-0.4.99
crack5.0.tar.gz                 LB7.jpg                       samba-tng-0.4.99.tar.gz
.DCOPserver_linux__0           LB8.jpg                       sara-6.0.7
.DCOPserver_linux_:0           LBSurfer.jpg                  sara-6.0.7.tgz
default.pwl                     linux30.jpg                    satan-1.1.1
Desktop                         linux3.jpg                     satan-1.1.1.tar.gz
.dmrc                           linux42.jpg                    scanlogd-2.2.5.tar.gz
Documents                       linux9.jpg                     shadow
.dvipsrc                        linux6.jpg                     .skel
.emacs                          .macromedia                   .ssh
.exrc                            .mailcap                       .sversionrc
.fonts                          .mcpop                         textfile.txt
.fonts.cache-1                 .mime.types                    .thumbnails
.fonts.conf                     Misc                            .urlview
.gconf                          .mozilla                       .viminfo
.gconfd                         .muttrc                        .Xauthority
.gnome                          netcat-0.7.1                  .xcoralrc
.gnome2                         netcat-0.7.1.tar.gz           .xemacs
.gnome2_private                 nmap-3.93-1.src.rpm           .xim.template
.ICEauthority                   OpenOffice.org1.1             .xinitrc.template
iptextfile.txt                 opie-2.4.tar.gz               .xsession-errors
john-1.6                        portsentry-1.2.tar.gz         .xtalkrc

```

```

linux:/home/jayme # rm textfile.txt          /* removes textfile.txt */
linux:/home/jayme # ls                      /* lists files textfile.txt is removed*/

```

```

.                               john-1.6.tar.gz                portsentry_beta
..                              .kde                          .profile
.adobe                          .kermrc                       psad-1.4.3.tar.gz
anger.tar.gz                    klaxon.tar.gz                 public_html
.bash_history                   LB1.jpg                       .qt
.bashrc                          LB2.jpg                       .recently-used
bin                              LB3.jpg                       samba-latest.tar.gz
brutus-aet2.zip                 LB6.jpg                       samba-tng-0.4.99
crack5.0.tar.gz                 LB7.jpg                       samba-tng-0.4.99.tar.gz
.DCOPserver_linux__0           LB8.jpg                       sara-6.0.7
.DCOPserver_linux_:0           LBSurfer.jpg                  sara-6.0.7.tgz
default.pwl                     linux30.jpg                    satan-1.1.1
Desktop                         linux3.jpg                     satan-1.1.1.tar.gz
.dmrc                           linux42.jpg                    scanlogd-2.2.5.tar.gz
Documents                       linux9.jpg                     shadow
.dvipsrc                        linux6.jpg                     .skel
.emacs                          .macromedia                   .ssh
.exrc                            .mailcap                       .sversionrc
.fonts                          .mcpop                         .thumbnails
.fonts.cache-1                 .mime.types                    .urlview
.fonts.conf                     Misc                            .viminfo
.gconf                          .mozilla                       .Xauthority
.gconfd                         .muttrc                        .xcoralrc
.gnome                          netcat-0.7.1                  .xemacs
.gnome2                         netcat-0.7.1.tar.gz           .xim.template
.gnome2_private                 nmap-3.93-1.src.rpm           .xinitrc.template
.ICEauthority                   OpenOffice.org1.1             .xsession-errors
iptextfile.txt                 opie-2.4.tar.gz               .xtalkrc
john-1.6                        portsentry-1.2.tar.gz

```

```

linux:/home/jayme # cat iptextfile.txt      /* displays contents of iptextfile.txt */
eth0      Link encap:Ethernet HWaddr 00:C0:4F:0D:43:F5
          inet addr:192.168.1.104 Bcast:192.168.1.255 Mask:255.255.255.0
          inet6 addr: fe80::2c0:4fff:fe0d:43f5/64 Scope:Link
          UP BROADCAST NOTRAILERS RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:1123 errors:0 dropped:0 overruns:0 frame:0
          TX packets:909 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1348023 (1.2 Mb) TX bytes:132398 (129.2 Kb)

```

```
Interrupt:11 Base address:0xdc00

lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        UP LOOPBACK RUNNING  MTU:16436  Metric:1
        RX packets:74 errors:0 dropped:0 overruns:0 frame:0
        TX packets:74 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:5231 (5.1 Kb)  TX bytes:5231 (5.1 Kb)

linux:/home/jayme #
```

References

- [1] BSD. In wikipedia [Web]. Retrieved Oct. 2006, from [http://en.wikipedia.org/wiki/BSD_\(operating_system\)](http://en.wikipedia.org/wiki/BSD_(operating_system))

- [2]Linux. In *wikipedia* [Web]. Retrieved Oct. 2006, from <http://en.wikipedia.org/wiki/Linux>

- [3] October 2006 Web Server Survey . Retrieved September 24, 2006, from Netcraft Web site: GNU. In *wikipedia* [Web]. Retrieved Oct. 2006, from <http://en.wikipedia.org/wiki/GNU>

- [4]Live Distro. In *wikipedia* [Web]. Retrieved Oct. 2006, from <http://en.wikipedia.org/wiki/LiveDistro>

- [5]Sheer, Paul Rute Users Tutorial and Exposition. Retrieved October 16, 2006, from Rute User's Tutorial and Exposition Web site: <http://members.toast.net/art.ross/rute/>

- [6]Siever, Ellen (1999). *Linux in a Nutshell*. Sebastopol,CA: O'Reilly.

Chapter 8: Introduction to Security Topics

Now that we have gained a wide array of knowledge in the first seven chapters, let's take it one step further and apply this knowledge to identify and secure a computer system. In order to effectively secure a system, we must first understand the vulnerabilities common today in order to safeguard against them. Although threats come in many shapes and sizes, the common ones can be classified into the following categories:

Virus is simply a software program that is comprised of two parts, a propagation mechanism and a payload. The propagation mechanism is the delivery mechanism that spreads or propagates the payload, and the payload is the malicious code that executes and does damage, such as deleting hard drive data. A virus can propagate itself in the form of an e-mail attachment.

Worms are very similar to viruses, but the main difference between them is that a worm has more emphasis on the propagation mechanism. A worm basically, worms its way through a network at a very rapid pace infecting all computers in its path.

Trojan horse is a harmless looking piece of software that can deliver a malicious payload. A Trojan can be delivered in the form of an e-mail attachment, and once it is opened, can install any threat on the system.

Exploits are software flaws or bugs that can be exploited to overtake a computer system. One popular exploit was Microsoft's Internet Explorer web browser buffer overflow. Hackers would write data beyond the buffer size, causing an overflow condition. The system would then terminate normal processing and start execution of the overflowed code. Code could be fed into the overflow execution to take over the system for malicious reasons.

Backdoors are alternate hidden doors into a system that are virtually undetected by the computer user. An unauthorized person could use a backdoor to gain access to the system remotely to steal information or utilize this computer to launch an attack.

Spyware are bugs or cookies that monitor and data mine your internet activity. They can become planted on your system by downloading software from questionable sites. Although many are harmless, they can pose an unauthorized invasion of your privacy.

Social Engineering such as e-mail fraud, web hoaxes, and phishing scams can be as simple as someone calling your house and stating that they're from your bank then asking for your account information. Many people would not trust this person unless they personally knew the caller. However, many are fall victim to fake e-mails that link them to a hacker's server that looks and feels like a bank's server. The only difference is that it is a trap to capture your account number and passwords, then to steal all your money [1].

Although the above list of threats is not comprehensive, it provides an adequate starting point. The old saying that a chain is only as strong as its weakest link is the golden rule of computer security. It is illogical to spend an excessive amount of money securing a network, while having a weak password policy in the organization. One must formulate an effective, comprehensive, best-security practice that has no weak link. Of course, the first problem is determining what the best practices for security are and how to implement them. Ask a group of 10 security experts the same question and you will probably receive 10 different answers. In general, security “best practices” are a consensus of approaches, architectures, and solutions that protect your network, systems, and data [2].

Security threats can be external or internal in nature. The most overlooked threats are the internal one, when employees have access to confidential data not required for their job responsibilities, or something as silly as a posted note with their passwords written on it. Most security professionals have their guard up against external attacks and overlook or give too much trust to employees. Stealing company information can occur in low-tech or no-tech situations. The steps suggested below will help improve the overall security of a desktop computer system.

1. Set good passwords

Make passwords at least 8 characters long, using numbers, letters, and symbols. Pass phrases are encouraged such as “That’s one \$mall \$tep for man, one giant leap for mankind” Notice the symbol substitution for the letter “s”. Do not use words found in the dictionary. Change your password monthly or at least every other month.

2. Keep your software updated

For a Windows-based system, simply go to Microsoft’s web site (<http://www.microsoft.com/downloads>) and download the service packs and the latest security updates. This process can be automated by using automatic updates which downloads recommended updates and installs them on a schedule set by the user. Microsoft security patches come out on the second Tuesday of every month [3].

3. Run anti-virus software.

Install and run antivirus software on a weekly basis. It is also important to update your antivirus software with the last virus definitions. Symantec and McAfee are popular commercial products, while Grisoft’s AVG and ALWIL avast offer free home editions.

4. Run anti-spyware programs

Anti-spyware programs scan the system similar to a virus scan program, only they identify and remove spyware. Popular anti-spyware scanners include Lavasoft’s Ad-Aware, Spybot Search & destroy, Spy Sweeper and Ewido.

5. Run a personal firewall

A firewall is designed to prevent unauthorized actions on a system. Firewalls can be hardware or software. There are many choices in software firewalls, Microsoft has bundled one directly into their XP operating system. Symantec's Norton's Personal Firewall and ZoneAlarm Pro are a few of the many commercial software firewalls [4].

6. Do not use peer-to-peer file sharing software

Installing programs such as KaZaA, Gnutella, or BitTorrent are only going to invite trouble. First of all you are downloading from untrusted sites, so the probability of downloading a threat are high, and you are giving others access to your share folder and files. Neither action is recommended.

7. Don't open e-mail or attachments from unknown sources

Computer users should never open unexpected attachments. Many viruses arrive as executable files that are harmless until they are opened. Many common types of file attachments such as jpeg's or zip's are used for spreading viruses.

8. Be cautious of unsolicited messages.

Even though you may recognize the name of the sender, scam artists can use spoofing tactics to get personal information from you. Never give out your ID, password, credit card or social security number in response to an unsolicited request. It is almost a known fact that banks, credit card companies, ISP's, and big commercial websites such as eBay, or Amazon won't be soliciting information from you. You can download a free anti-phishing toolbar to protect against this at <http://toolbar.netcraft.com/>.

9. Create a Process History

First, baseline your system by benchmarking and documenting all the processes that are running on the computer before trouble starts. This will give you a point of reference when something does happen. You can easily print all the current processes running with one command on Microsoft's XP operating system [5] as shown in Figure 1. Simply follow these commands:

Open the Command prompt window:

Start → Run,

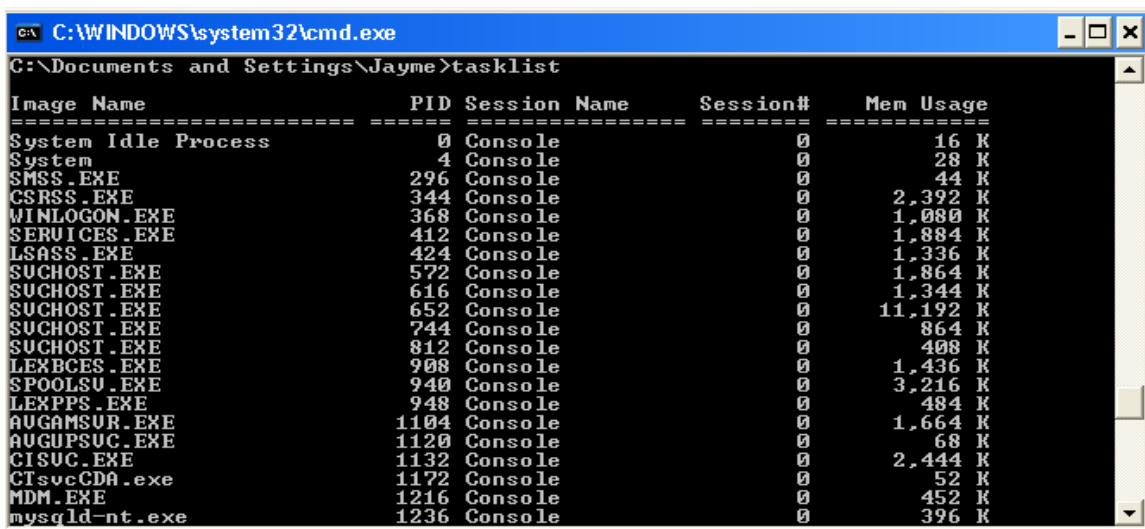
type **cmd** in the text field of the dialog box, and then OK to run.

To see list of active tasks running on your system:

type **tasklist /nh** (at the command prompt) (reference figure 1 for output).

or to redirect to a printer: **Tasklist /nh |sort >prn:**

or to redirect to a file: **Tasklist /nh |sort >ProcessFile.txt**



```

C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\Jayme>tasklist

Image Name                    PID Session Name        Session#    Mem Usage
-----
System Idle Process           0     Console             0             16 K
System                         4     Console             0             28 K
SMSS .EXE                     296   Console             0             44 K
CSRSS .EXE                    344   Console             0            2,392 K
WINLOGON .EXE                 368   Console             0            1,080 K
SERVICES .EXE                 412   Console             0            1,884 K
LSASS .EXE                    424   Console             0            1,336 K
SUCHOST .EXE                  572   Console             0            1,864 K
SUCHOST .EXE                  616   Console             0            1,344 K
SUCHOST .EXE                  652   Console             0           11,192 K
SUCHOST .EXE                  744   Console             0             864 K
SUCHOST .EXE                  812   Console             0             408 K
LEXBCE .EXE                   908   Console             0            1,436 K
SPOOLSU .EXE                  940   Console             0            3,216 K
LEXPPS .EXE                   948   Console             0             484 K
AUGMSUR .EXE                 1104   Console             0            1,664 K
AUGUPSUC .EXE                1120   Console             0              68 K
CISUC .EXE                   1132   Console             0            2,444 K
CTsvcCDA .exe                1172   Console             0              52 K
MDM .EXE                     1216   Console             0             452 K
mysqld-nt .exe               1236   Console             0             396 K

```

Figure 1. Listing processes with Tasklist /nh

10. Limit access to your system

Turn off file and print sharing.

Do not sign on as root or administrator except when absolutely necessary.

Turn off all unneeded services [6].

In summary, securing a computer can be accomplished through three processes: preventative measures, detection, and incident response [7]. Still, information security is an evolving science. There is no one-size-fits all solution. One cannot even rely on one vendor of software or hardware to provide a solution. You must sleep with one eye open, have a response plan, a contingency plan and maybe even another fail safe back-up plan. Although you maybe protected today, you never know what tomorrow will bring.

References

[1] Volonino, L, & Robinson, S (2004). *Principles and Practice of Information Security*. Upper Saddle River New Jersey: Prentice Hall.

[2] InfoWorld. Retrieved October 17, 2006, from Security best practices Web site: http://www.infoworld.com/article/02/03/15/020318opsecurity_1.html?Template=/storypages/printfriendly.htm

[3] MicroSoft Download center. Retrieved October 17, 2006, from Download Center Web site: <http://www.microsoft.com/downloads/Search.aspx?displaylang=en>

[4] Skoudis, Ed, & Zelter, Lenny (2004). *Malware Fighting Malicious Code*. New Jersey: Prentice Hall.

[5] Tittel, Ed (2005). *PC Magazine Fighting Spyware, Viruses, and Malware*. Indianapolis,IN: Wiley Publishing, Inc.

[6] SANS InfoSec Reading Room - Best Practices . Retrieved October 17, 2006, from SANS Institute Web site:
http://www.sans.org/reading_room/whitepapers/bestprac/?portal=ad0c4de7cea371f43cca2e453b840584

[7] Computer Insecurity. In *Wikipedia* [Web]. Retrieved Oct 17,2006, from http://en.wikipedia.org/wiki/Computer_insecurity

Chapter 9: Conclusion

Purpose

The goal of this paper was to provide a foundation for the course, Computer Organization (68-500), as conveyed in the title, “*Computer Architecture, Using a Thorough, Concise, Step-by-Step Approach: A Course for Newcomers to the Information Security Field*”. This is a text unlike any other conventional computer science texts because it is specifically tailored to non-technical students pursuing a career in computer information security. It represents a fresh approach by using comprehensive overviews to convey fundamental concepts. The chapters are organized in a manner that is easy to follow, and presented so that topics build upon one another, to solidify and reinforce previously learned concepts. Despite the fact that it is tailored to non-technical students, it also benefits anyone by teaching a foundation skill set to help students succeed in an information security career. This work provides a single text that the instructor may use to present all the concepts that are relevant to an understanding of information security.

Methodology summary

This paper incorporates and accentuates the strengths of the skill set that the current Computer Organization (68-500) course strives to convey. Starting with Chapter 1, it incorporates topics such as historical hardware evolution, hardware terminologies, and technological advancements. Chapter 2 concentrates on how data is represented to hardware and software components and how to convert between decimal, hexadecimal, binary, and character data. Chapter 3 takes an in-depth view into concepts introduced in Chapter 1, by looking closely at what really happens inside the CPU and at the register level. Chapter 4 introduces an overview of the operating system and how it interfaces with the hardware. Chapter 5 covers many digital electronic concepts used in the current course. The remaining chapters 6, 7, and 8 are designed to overcome the weaknesses in the current course. Although Linux (chapter 7) was introduced in the current course, it was not covered thoroughly enough to be beneficial. Chapter 6, DOS is introduced as a learning tool to those who are not well-versed in command line interface operating systems. It also provides a natural stepping stone into Linux, yet is valuable to the security field by introducing concepts that apply specifically to security. Furthermore, Chapter 6 concepts can be applied to Windows operating systems. In chapter 7, Linux, covers details of the structure, common commands, and commands used in security. Chapter 8 is a short introduction into security issues that provides a step-by-step “best practices” for a windows system.

Achieved the purpose

I feel that this text, along with the accompanying course will achieve its purpose of level setting all new students in the information security program. It will provide a foundational skill set to students with a non-technical background or no prior work-related experience in the computer field. After completing all the courses in this information security program except one, I know this text will provide an excellent

springboard into the following courses in the major. This will provide students with an easier transition into advanced courses. The accompanying Table 1 Skills Matrix illustrates the core skill set and how these skills can be applied to the corresponding classes.

Table 1. Skills matrix

Skills matrix							
Skills	Courses						
Courses	Data Networks 68-510	Operating Systems 68-515	Intrusion Detection 68-520	Encryption 68-525	Securing Windows 68-560	Securing Linux 68-561	Securing Database 68-563
General HW terminology	X	X	X		X	X	
General SW terminology	X	X	X	X	X	X	X
Operating System Concepts	X	X	X		X	X	X
Hierarchical Directory Structures	X	X	X		X	X	X
Numbering systems	X	X	X	X	X	X	
Boot Process		X			X		
Process / Job scheduling		X	X		X	X	X
Boolean logic	X	X		X		X	X
Common commands	X	X	X		X	X	X
Command Line Interface	X	X	X		X	X	X
Security Concepts	X	X	X	X	X	X	X

The final test to the effectiveness of this text will be to employ it in 68-500 and evaluate student and instructor attitudes toward it in a formal evaluation. The course will be designed to explore the various key concepts conveyed in each chapter. Afterwards, students and instructors will rate the course and the text through the standard course evaluation forms, and provide written suggestions for improvements or additions to the course material.