Indiana University
Multi-campus Modular
Computing Curriculum

Introduction to Computing

Module # 3: The Essence of Computation

Goals

- Recognize the simplest tasks computers can perform
- Become familiar with the essential characteristics and capabilities of processing units
- Recognize the characteristics of analog and digital information storing techniques
- Describe the fetch / execute cycle
- Classify various types of encoding schemes
- Recognize the essential machine language commands

Prerequisite Skills

- comfort with keyboard and mouse
- Experience with the STAIR process

1 Teacher, why are we learning this?

That is actually a very good question. We have thought very carefully about which information to include in this section. We think that there are certain basic concepts that are so central to computing that we will not be able to teach you properly unless you know them. Having said that, many of these concepts refer to processes that are hidden deep in the structure of computers. Many people use computers for years quite happily without knowing how to convert binary numbers, or what a register is. If you can do so, great. But remember, we want to be able to use computers as problem solving tools. We really need to understand their nature if we hope to use them to their potential.

There is another factor at work here. Arthur C. Clarke once wrote that "Any sufficiently advanced technology is indistinguishable from magic." Anybody who has taken a computer science course ought to know just a little bit about how the magic works. We won't divulge all the secrets of the universe, but we will explain enough so that we can give you reasonably clear answers. There is a mystique about computers. Some of it is deserved. Computers give us awesome new capabilities. Computers are also vastly over-rated and sometimes feared for the wrong reasons. We can only begin to explore the real potentials and limitations of computers when we dig a little bit into the science that created them. We need to know something about what they do and how they do it.

Finally, a little theoretical background never hurt anyone. This is, after all, a university science course. Presumably, you are here for more than a how-to course. You can buy a book on Excel or Wordperfect, and learn them without the struggles of taking a class. We are assuming that you are here because you want to go just a little bit deeper than that. We promise not to overwhelm you with the math or science, but we respect you enough to not completely hide it from you either.

2 What Is a Computer?

Anyone who has lived in a modern society within the last few years is aware of what a computer is. We all have seen them, and we have all used them. Sometimes we are aware we are using a computer, and sometimes we are not. Even though we know what a computer looks like, and we might know something about what it does, there are some puzzling things about the nature of this machine that make understanding it a little more elusive than other machines. In this section, we will examine what actually happens deep inside the computer and see how it really works. You will discover an interesting paradox. Although computers are almost completely universal, capable of doing all kinds of complex processes, they can only really do a very select number of tasks. These simple tasks are combined in complex ways to make the computer capable of very complicated jobs. We are also used to seeing computers deal with every kind of information from words to numbers to pictures and music. We will see that computers can actually deal only with a very limited kind of information, but it can manipulate that information in very complex ways so it can be interpreted as text, music, or whatever.

We will define a computer in this way: A computer is a universal information manipulator. We will see in our discussions today exactly what that means.

3 Information

Computers are designed to work with information. This is fundamentally different from most machines. Mechanical devices typically deal with physical entities only. Information is more conceptual. Numbers, words, and instructions are good examples of information.

Information is also referred to as data. Incidentally data is a plural noun. One piece of information is referred to as a datum. Generally, computers work with large amounts of information at a time, so you will encounter the term data more frequently than datum.

3.1 Digital and Analog Information Representation

Information can be stored and manipulated in a number of ways. The two most important major ways of storing information are referred to as digital and analog. The consumer electronics market has made the terms digital and analog very familiar to people, but we are often unaware of what the distinction really is.

3.1.1 Analog Information Encoding

We have had 'information machines' for many years before computers were common. Among the most common such machines are traditional watches and clocks. These machines are used to measure time, an abstract numerical. They use purely mechanical means to represent information. A tiny motor controls gears which manipulate the hands. The only reason the position of the hands has any informational meaning is because we are trained to interpret it as such. This type of information handling device is referred to as an analog device. The term 'analog' is used because we use an analogy to describe the data. In the case of the watch, we actually have several analogies working at once. The motion of the second hand is analogous to the passage of seconds in a minute. The motion of the minute hand around the dial is an analogy to minutes in an hour, and the hour hand represents hours in a half-day. The process of learning how to read an analog watch comes down to understanding the analogies and what they represent. We have encountered many other analog devices: Dial and liquid thermometers are good examples, so are any of the dials on your dashboard such as the speedometer and fuel gauge. A slide rule is analog. Record players (remember those?) use analog technology. Nearly any display which features a needle or a dial uses analog technology.

Analog information is mechanical. It usually offers nearly infinite precision, but limited accuracy. Here is an example: The dial thermometer outside Joyce's apartment registers 74 degrees F. By looking at the back of the thermometer, she sees it is simply a coil of some sort of metal. When the temperature changes, the metal expands and contracts. The needle is attached to the outside of the coil, so when the length of the coil changes due to a change in temperature, the location of the needle on the dial changes. If she looks carefully enough, (perhaps with a magnifying glass!) she will see the dial fluctuate even with the most minor changes in temperature. As the temperature changes from 50 degrees to 74 degrees, the dial will touch EVERY single intermediate spot in that interval. There is a continuous motion with no jumps or gaps. This is an example of the precision of analog instruments.

At the same time, Joyce recognizes that the accuracy of this device may be suspect. By squeezing the coil, she can change the apparent temperature reading. She can only assume that nobody has changed the shape of the coil manually, thus changing the accuracy of the reading. She also has to trust that the dial was calibrated properly at the factory. If the painting machine were a bit off, or the coil was installed improperly, the machine would not show the proper reading. This problem of accuracy is very common with analog information devices.

3.1.2 Digital Information Storage

A digital device records information as a series of numbers. These numbers are then translated to represent another entity. Digital instruments do not have as much precision as their analog counterpoints, but they tend to be much more accurate. If Joyce looks at a digital thermometer next to an analog thermometer, she will readily see the difference. A digital thermometer would have a readout that says in numbers what the temperature is. If the display says 74.3 degrees, it does not matter how closely she looks at the thermometer, she will not get a closer approximation of the temperature. A digital device offers discrete values. There are no intermediate values on a digital instrument, although the values given may be very close together. The digital thermometer is more likely to be accurate, because the instrumentation is unlikely to change by changes in the physical environment (except of course temperature, but that's the point of a thermometer!)

3.1.3 Comparison of Digital and Analog

The obvious example is watches. Some of us wear digital watches, some wear analog. The digital watches have numbers displayed on them, the analog ones use hands.

Q. 1 So what about those watches with an LCD panel that shows little hands? Are they digital or analog? What about a sundial or an hourglass? What if you had a computer program that showed a sundial or hourglass on the screen?

(Answer on page 3-17.)

Recording technology also provides us with some very straightforward examples. Thomas Edison Pioneered a form of analog sound recording in 1877. Remember that sounds are simply waves. To record a sound, a membrane in a microphone is used to copy that wave onto some surface. (Foil, in Edison's case) To replay the sound, a needle is forced through the groove created by the recording process. This needle is attached to another membrane in a speaker. When the speaker membrane vibrates, the original sound wave is recreated. The process is entirely analog. No numbers are involved, the process is completely mechanical, and there is infinite precision, but very limited accuracy and much room for error in the sound recording and reproduction process.

Compact disk technology uses digital means to record and play sounds. The sound waves are read by a computer which analyzes each instance of the sound, and assigns it a numerical value. Many of these numerical values are stored each second. When the music is played back, it goes through another computer, which retranslates the numbers into the sounds that the numbers represent. As anyone who listens to CDs can attest, digital recordings seem much more accurate than analog recordings. Since they are recorded at such frequent tiny intervals, the lack of precision is not a problem, and we find digitally recorded music more accurate.

Q. 2 Have you ever seen the little markings on CDs that look like this: AAD, ADD, DDD? What do those marks represent? Which one would sound the best?

(Answer on page 3-17.)

(Reference– See http://www.tc.umn.edu/nlhome/g496/eric0139/Papers/paper.html for a very nice discussion of digital audio as it compares to analog sound reproduction.)

3.1.4 Important characteristics of a Digital Computer

We tend to think of computers as digital devices. They do not have to be so, but most of them are. The computers we use will all be digital in nature (Note that Digital is also a brand name of a certain type of computer. Most computer users refer to the company's initials DEC to avoid confusion. When we speak of digital technology in this course, we are not referring to the corporation, but the information storage technique.) The digital nature of computers is important because it gives them many of their characteristics. Remember that digital devices have limited precision, but extreme accuracy. Computers have the same properties. Digital devices manipulate numbers. Computers do this. Computers are able to make the numbers represent various other kinds of information.

3.2 Binary Storage

While we say that computers store numbers, we are technically not being accurate. A computer is still essentially a machine. It deals with electronic impulses. **The only thing today's computers really understand is fluctuations in electronic voltages.** The computer can recognize two values, high and low. These values are also sometimes referred to as on or off, true or false, yes or no. The term binary is often used to describe this kind of behavior as well.

Any mechanical device that exhibits this yes/no behavior is referred to as a switch. We are already comfortable with the notion of switches for lights and other devices. A computer is essentially a huge number of switches. A computer science might refer to them as binary switches, indicating they allow only two possible values apiece.

One huge advantage of this type of switch is the capability for self - correction. Voltage is actually an analog property, but forcing the circuitry to accept it as one of two values makes the computer a digital system, and minimizes the possibility of mistakes due to external changes in voltage. This characteristic is the main reason computers use binary notation. It is very easy to build self-correcting circuits using binary switches. What that really means to us is that computers can be highly accurate digital devices, even though they still use a form of analog signaling deep within their structure.

It doesn't seem possible that a switch - based system could hold enough information to be usable, but it can. Imagine your instructor has developed a code for your class: When you come into the lecture room, if the lights are on, the class will meet. If the lights are off, the class will be cancelled that day. (NOTE: this is ONLY an example!) With one switch, your instructor can send you two possible messages: On = class today, Off = no class. Imagine you have a class with lab and lecture components. This one-switch system cannot tell you about both the lab and lecture sessions. However, if the lecture room has two banks of lights, for example one in the front of the room and one in the back, the number of messages possible could be doubled. For example, let's say the front lights represent whether the lecture will meet, and the back lights represent the likelihood of a lab session. We now could send four messages:

front switch	back switch	meaning
OFF	OFF	No class, no lab
OFF	ON	No class, but lab
ON	OFF	Class, but no lab
ON	ON	Both will meet

NOTE: Put drawings of light switches in here. It would be more effective, I think... -ajh

Q. 3 If we add another switch, how many messages can we send?

(Answer on page 3-17.)

This seems like a very small capability, and it is. But these on/off impulses can be combined in a simple scheme to represent numbers. Once we can represent numbers, we have a digital machine. As you have seen, digital machines can do many interesting things with great accuracy. Everything computers do is based on the idea of on/off impulses representing numbers.

If we take the table above and agree to some conventions, we can make it more general. Let's say we represent a switch with a 1 if it is on and a 0 if it is off. Let's also agree to define a bank of switches as a series of digits written together. For example, 1000 means we have four switches, and only the first one is on. All the rest are off. Using these conventions, we can re-write the above table like this:

Value	Message Number
00	0
01	1
10	2
11	3

Q. 4 Why did we start with message 0 rather than message 1?

(Answer on page 3-17.)

Using this kind of scheme, we could arbitrarily assign values to different combinations of switches. As long as we are willing to keep adding switches, we can store any number of messages simply by using switches.

We can get numbers from on-off signals if we review some simple mathematics. (Don't worry, we're not going to do anything fancy here!) We are used to thinking of numbers in base 10. This is so natural to us that we sometimes forget that base 10 is an arbitrary way of representing numbers. People did mathematical calculations for thousands of years using other bases. Base 10 is simple enough that we rarely think about needing to use other bases, but it is not the only way to do math.

Let's review what it means to represent numbers in base 10.

If I have a number, say 345, what does that number really mean? It means I have 3 times 100 plus 4 times 10 plus 5 times one. That seems pretty obvious, but how did we get the 100, 10, and one? These values are all based on powers of 10. 100 is 10 squared (in computing, we often refer to exponents with the caret symbol(^), so 10 to the second power is 10^2.) 10 is 10^1, and 1 is 10^0. (remember, any number raised to the zero power is 1) While we're discussing math notation on computers, you should know that multiplication is usually denoted by an asterisk (*). This avoids confusion with the X or x symbols. The second sentence in this paragraph could be re-written like this: It means I have (3*100) + (4*10) + (5*1).

The number 345 can be summarized like this:

3*10^2 +..	4*10^1 +..	5*10^0
3*100 +..	4 * 10 +..	5 * 1
300 +..	40 +..	5

In other words, the values of the digits are all based on powers of 10. With one digit, we can describe up to 10 different values. If we have two digits in base 10, we have 10 * 10 possible values we can describe (100). If we have three digits in base 10, we can describe 1000 (or 10 * 10 * 10) different values. Each additional digit increases the number of values we can describe by a factor of our base (10 in our normal counting system). We can describe any number using base 10, but we can also describe any number using any other base.

We don't have to base a numbering system on 10. (The Sumerians based their system on 60. We still have 60 minutes in an hour.) We simply use 10 because we have 10 fingers on our hands. It makes sense to us. Remember that the computer tends to see things in terms of on or off, yes or no. The most natural number for the computer to count with is two. Computers can represent any number, but they represent these numbers in terms of two.

The binary system is base two. It works just like base 10, but rather than using powers of 10 as its foundation, it uses powers of two. The rightmost digit in binary represents 2^0, or the ones digit, anything raised to the zero power equals one.) The next digit to the left represents 2^1 or the twos digit. The next digit to the left represents 2^2, or the fours digit. Each successive digit doubles the number of possible values. (Sound familiar?) Examine the following table to see how numbers are stored in binary notation.

Decimal Value	Binary Value	2^3	2^2	2^1	2^0
		8s	4s	2s	1s
0	0	0	0	0	0
1	1	0	0	0	1
2	10	0	0	1	0
3	11	0	0	1	1
4	100	0	1	0	0
5	101	0	1	0	1
6	110	0	1	1	0
7	111	0	1	1	1
8	1000	1	0	0	0

Q. 5 From this chart, determine what the binary values for the numbers 9 through 15 would be. *(Answer on page 3-17.)*

3.3 Binary and Other Kinds of Information

As you have seen, it is not difficult to make computers store numbers using switch technology. The only numbers we have seen, however, are whole numbers. Whole numbers, as you remember, are those numbers we count with, starting with zero (0,1,2,3...). Binary notation makes it relatively easy to store and work with whole numbers, but these are not the only kind of information we want to work with. Computer scientists have developed ways of making computers recognize other kinds of values.

3.3.1 Integers

Integers are the whole numbers and the negative numbers. (-3, -2, -1, 0, 1, 2, 3...) As you can imagine, it is not too difficult to make computers deal with negative numbers. Computer scientists simply add one more switch to a number which tells us if the number is positive or negative. You usually don't have to worry about this as a computer user, except you might need to specify in some way that you are working with an integer. (It makes a huge difference in some kinds of programs, and no difference at all in others. We will illustrate as you learn new kinds of applications.)

3.3.2 Approximation of Real Numbers

Real numbers cause us a few more problems. These are the numbers that can be represented by fractions or decimal values. (1/3, 2.357, and so on..) The binary notation system does not seem to have room for the decimal point. Computer scientists have worked around this problem too. When a computer is dealing with a real number, it works in something like scientific notation. The number 731.456 is stored in two pieces. The computer stores the value 731456 in binary notation, and in a separate place stores a 3 (again in binary notation), meaning that the decimal place belongs three places from the left. As a user, you don't generally have to know how this works, but you do need to know that computers interpret real numbers differently than integers. (Although at the deepest level they are both stored as a series of 1 and 0 impulses.)

Real Numbers and Error. There are certain kinds of real numbers that are very difficult for the computer to store properly. These are the numbers with endless decimal values. One classic example is the repeating decimal, like you get when you divide 1 by 3. 1/3 = 0.33333.... It goes on forever. Irrational numbers like pi have a similar characteristic. They have an infinite number of digits to the right of the decimal point. Computers have a finite (although often huge) capacity for information. You could fill all the memory of the largest computer in the world with the digits of pi, and still not have a complete solution to pi. (There are a number of interesting research projects out there doing exactly that kind of work.) You could completely fill your computer's capacity with this kind of number, but you probably don't need that close an approximation. 0.3333333 is probably close enough to 1/3 for most calculations, and 3.14159265 is probably close enough to pi for most calculations. Pi is an infinitely precise value, but digital computers do not have infinite precision. The phenomenon is referred to as round - off error, and is important to keep in mind when you are doing very precise calculations. Most of the time it is not a problem, because humans rarely need to be that exact, and there are other complex schemes for dealing with numbers that do not cause this error. You should, however, be aware that round-off error exists.

3.3.3 Text Characters

So far, we have seen how computers translate 1s and 0s into numbers, but we want to use computers for a lot more than math. Computers can also deal with a lot of other kinds of information, but perhaps the most important type of information is text. Ultimately, text characters are also stored in the computer as binary impulses, but they have intermediate values as numbers. Imagine two children keeping a secret code. They might agree to encode their messages by representing each letter in the alphabet with its corresponding number, so A becomes 1, B becomes 2, and so on. The message "HI THERE" would become this: 8 9 20 8 5 18 5. This code would be somewhat effective at keeping secrets, but it has another benefit. The children could communicate by tapping on the walls, or banging on a drum. 8 taps would represent an "H", and 20 taps a "T".

This would be fine for simple communication, but eventually the children might want to send more complex messages with punctuation marks, capital and lowercase letters, and some special symbols for spaces, the end of a message, and so on.

Early computer scientists developed codes along exactly the same lines. The most popular encoding technique is called ASCII (American Standard Code for Information Interchange) (Yea, it should be ASCFII, but nobody asked my opinion...) In this scheme, capital A is represented by a 65, B by 66 and so on... Lowercase a is represented by 97, lowercase b by 98. Most punctuation and other special characters are assigned to the values below 65. These seem like completely random values until you look at them in binary notation.

Character	Binary value	Decimal Value
(space)	0100000	32
A	1000001	65
B	1000010	66
a	1100001	97
b	1100010	98

Notice that "A" and "a" have completely different (but numerically related) values. Some programs can tell that "A" and "a" mean nearly the same thing, and some programs cannot. The programs that treat them as completely different letters are referred to as "case-sensitive."

It would be silly for you to memorize how these codes are stored. That's the computer's job. The important thing for you to recognize is that letters are translated to numbers, and ultimately to 1s and 0s.

Here's a trap: The character '4' has an ASCII value of 52(which is binary 110100). This is fine when we are thinking of '4' as a character inside a text document. However, if we want the computer to do math on the value 4, it needs to be stored as 4 in binary (100). If we try to do math on the numeric figures, 4+4 in binary is 100 + 100 =1000 or eight. If we tried to do the math on the ASCII representation of four, we would get

110100 + 110100 = 1101000 or "4" + "4" = 104(!), but if we are thinking of characters, 104 = lowercase "h" ?!??!

Don't get hung up on the details here. This is just an illustration of how important it is to keep numbers and text defined properly. In some programs, the numeric value 4 is VERY different than the character '4' (Notice the quotes). When we get to those types of programs, we will refer back to this discussion.

3.3.4 The Translation Trap

Computers can only deal with on/off values, which are thought of as ones and zeroes. To get past that limitation, computer scientists have dreamed up a bunch of translation schemes so that other kinds of information can be stored as well. Here's the problem: How does the computer know what kind of value it is dealing with? If you could look at the computer's memory, you could only see ones and zeroes. It would be impossible to tell if those values were parts of characters or numbers. This is the key fact to remember: **computers track different kinds of information in different ways!** Sometimes you will run into problems because of this fact.

4 Manipulation

We said that computers were universal information manipulators. We have begun to look at information. Now we need to examine what we mean by the term 'manipulation'.

4.1 Registers and Memory

We know now what kind of stuff the computer works with. Now we are thinking what the computer does with the information. Mainly, information just sits there doing very little. The information sits in a series of switches who just hold things. This type of switches is referred to as memory. **Memory just holds things.** You can get stuff from memory, and put it there. You can't really change anything directly in memory.

You might think of memory as a whole bunch of tiny mailboxes. Each mailbox has an address and contents. If you want to put something in a mailbox, you need to know the mailbox address and the contents to put in it. Most of the time as a computer user, this process is hidden from you, but you ought to know it is happening. Each mailbox can only hold very specific amounts of information. For example, if you are typing a letter using some kind of word processing program, each address of memory might contain one character. Since you know that computers can hold long documents, you can see that there are many such addresses in a typical computer, and they could be difficult to deal with, but your word processing program will handle it for you. You don't have to deal directly with all these addresses.

Memory is good for storing things, but computers do more than simply storing information. They need to be able to do things to information. Computers use special place in memory to hold values while they work on them. These special places are called registers. You can think of registers as "information garages". Think of taking your car to the body shop. When you take your car in to have some work done, they usually do not take it directly to the service bay. Instead, they put it in a parking space. When the mechanic comes in, he might be told "Fix the bumper on the car in space 32." He does not go out to space 32 to do all the work, because it might be cold out, and all his tools are in the service bay. It makes much more sense to bring the car into the garage and work on it there. Then when the work is done, he might take it back out to space 32. Registers work in pretty much the same way. If you tell a computer to do something to a number in memory, it will go to the memory address, copy the value of that address to a register, and do whatever you want to the value now in the register. It might then copy the value from the register back to the memory area.

4.2 The Elemental Commands

Just as the simple on/off choices of a switch can be combined to make nearly any kind of information, the seemingly endless variation of things a computer can do really boil down to a small number of tasks that can be combined in very complex ways. These codes are stored in the computer just like everything else as ones and zeroes. A certain set of the most basic commands are built into a computer chip. Each of these commands is represented by a command number. When a computer is expecting a command, it looks at the number it is given and does something based on what that number is. This set of very basic, machine-specific commands is sometimes referred to as a machine language.

load The load command means "Go to a spot in memory, and look at the value in that spot. Copy that value to the register."

store The store command is the opposite of the load command. It says "Take the value of the register, and copy it to a particular spot in memory."

add The add command tells the computer to take the values of two registers, and add them up. All this addition happens in binary. All the other arithmetic operations (subtraction, multiplication, and division) can be derived from addition.

test The test command simply compares to values. It notes whether the two values are the same or different. It is almost always used in conjunction with the jump command below.

jump The jump command tells the computer "Move to another command in the list of instructions" This is useful if the computer's behavior should change based on circumstances. For example, different parts of a quiz program should activate based on whether an answer is correct or not.

halt This command tells the computer that the list of commands is finished. Without such a command, the computer will keep trying to execute commands until it reaches the end of the memory.

4.3 The Fetch / Execute Cycle

A list of instructions to a computer is called a program or software. Such a list is still simply information, stored in ones and zeroes. The way that computers process information is sometimes called the fetch / execute cycle, because programs fetch a command, execute it, and fetch the next command, until they are told to stop.

5 Universality

Programs are why computers are so flexible, because one computer can run many different programs, each one changing the way the computer behaves. This flexibility is why programs are considered universal. The term universal seems a little odd in this context. We think of computing as a pretty exact science, so the term 'universal' seems way too big and imprecise. How can anything be truly universal?

5.1 Computers Are Nearly Universal

A truly universal machine could take any kind of stuff and do anything with it. You could feed it a paper clip and have a goat come out. You could tell it your name and have it tell your birthday. Universality refers to the kind of materials a machine works with, and the kinds of operations it can do on those materials. At first glance, computers are not universal at all. We have established that the only kind of material they can work with is information in the on/off format, and they can only do a handful of operations to that information. Rather than being universal, these characteristics imply that computers are very limited in their capabilities.

In a sense, they are very limited. But modern computers are capable of beginning to overcome this limitation by sheer size and speed. The earliest computers could do very basic operations only on ones and zeroes. Later machines had more capability. They still could do only rudimentary tasks, but they had made enough advances in memory capacity and speed that these operations could be combined into more complex operations. For example, early computers could add. By repeated addition, they could also multiply, but the multiplication was slow because the addition was slow. As the computers became faster, repeated addition became quick enough to really be thought of (by human users) as multiplication. The process did not change, but the speed of the computer made it seem more powerful.

Likewise, increases in the memory capacity of computers has dramatically improved the variety of information that can be stored in them. Integers are one of the most simple forms of information, so even the earliest computers were able to work with them efficiently. More complex types of information that can be distilled down to integers (such as sound files, graphics, and videos) are not much more complex for the computer to work with, but take up so much space that the early computers could not handle them. Today's $20.00 electronic address book might have several times the memory capacity of the first computers. Increased memory capacity means the ability to deal with larger and more complicated types of information.

5.2 The Limitations Are In The Human Mind

As the speed and size of computing technology continues to improve, we are seeing fewer and fewer technical limitations on universality. We can already make computers do many kinds of operations by combining the ones we have. We also know we can encode many kinds of data into the binary format the computer needs. The hardware is not the limiting factor in computing. The limiting factor is human imagination.

If we can imagine how to translate an operation into the core operations, we can get a computer to do it. This process is the art of programming and using a computer. Likewise, if we can imagine how any value can be represented digitally, we can teach a computer how to store and manipulate that value.

Computers can do nearly anything with information, but they have to be taught (by humans, for now) how to store and manipulate that information. They can only do what we teach them to do. Programmers obviously do a lot of this teaching, but users do, too. You teach a computer when you type a document into it or play a game on it. Just as a paintbrush can be used to create a picture of anything the artist can imagine, a computer can do the same in the hands of a skilled user. Only the paint is different. Rather than oils and acryllics, a computer user paints with information and procedures.

6 Laboratory Assignment

You will do some assignments on a special machine language called ABNIAC. Your instructor will give you details.

7 Vocabulary and Important Concepts

Computer In general, a universal information manipulator

Universal Capable of doing any kind of operation to any kind of information. Today's computers are not quite universal, but they come closer to being universal than any other machine yet invented.

Information The stuff a computer works with. A computer can work with any kind of data that can eventually be translated into binary (one/zero) format.

Manipulator A machine that processes something. A computer is unique because it can process information, not just physical quantities like most machines.

Data Another term for information. Some computer scientists refer to all the stuff a computer deals with as data, and the stuff that has some meaning as information. Note that "data" is a plural noun. The singular form is "datum."

Digital An information coding scheme based on numeric representation of values. Digital information is characterized by limited precision, but strong accuracy.

Analog An information coding scheme based on physical analogies between a value and some physical object. Analog information is characterized by infinite precision but limited accuracy.

Precision How small and close together the units of measure are. A teaspoon is more precise than a dump truck. A scalpel is more precise than a chain saw.

Accuracy How faithfully a recording scheme captures the original. How close something is to perfection. A photocopy is more accurate than a handrwitten transcript.

Binary Relating to the value two. The binary system is a numbering system in base two. Anything that has only two possible values is thought of as binary. A light switch a good example of a binary device.

Base 10 The numbering system we are used to using. The digits all represent powers of 10. Each digit can hold one of 10 possible values (including zero), and 9 is the highest possible value in each digit.

Base 2 Also referred to as the binary system. The ditgits are represent powers of 2. Each digit can hold one of 2 possible values, zero or one. Any number can be represented in binary with a long enough string of zeroes and ones.

Binary System Another term for the base 2 numbering system

Integers In Mathematics, the whole numbers and the negative numbers. Decimal values and fractions are NOT integers. In computing, integers are stored with a relatively simple scheme in binary notation

Real Numbers In Mathematics, the numbers that can be expressed as a fraction or decimal value. In computing, these are stored differently than integers. Some programs require you to specify whether you are working with real numbers or integers.

Round-Off Error A term applied to a number of related problems common in digital computers. Any digital storage technique has limited precision. Some real numbers have infinitely precise values. Round-Off error is the generic term applied to any problem that relates to this characteristic.

Case-Sensitive In most text recognition schemes, lowercase and uppercase letters have completely different values. (EG “A” is not given the same value as “a”.) Some programs are designed to correct automatically for this feature, and some do not. This property is called “case-sensitivity” DOS is a non-case-sensitive operating system, meaning that the command “Foo” could be entered in as “foo”, “Foo”,“fOO”, or “fOo”, and the command would work the same. Unix is a case-sensitive operating system, meaning that if the command were called “foo”, only “foo” would work. “Foo”, “fOO”, and all the other variations would not. At this point, just be aware what case-sensitivity is, so you will recognize it when you see it.

Memory A bank of switches designed to hold information. Memory does nothing but hold stuff.

Address Each little piece of memory has a numerical address that lets the computer find and put stuff there. You generally don’t have to deal with these addresses directly as the user.

Register A number of special places in the processor that allow the computer to manipulate information. Information is brought to a register, examined, maybe changed, and maybe sent back to memory. (Depending on what program the computer is running.)

Program A list of instructions for the computer to follow. In it's deepest level, the computer only has a few built-in instructions it can follow.

machine language The small number of built-in instructions that are designed as a part of a computer chip. Machine language programming is very tedious and difficult.

8 Review Questions

Q. 6 Does a sundial work with a physical entity or information? Is it digital or analog? Is it a computer by our definition?

(Answer on page 3-17.)

Q. 7 Is a pocket calculator a computer?

(Answer on page 3-17.)

Q. 8 Is a cassette tape digital or analog?

(Answer on page 3-18.)

Q. 9 If a standard light switch is a binary digital device, what about a dimmer switch? (You know, the kind with a dial that you can turn low for a romantic dinner or very bright for surgery on the dining room table.)

(Answer on page 3-18.)

Q. 10 From our discussion, you might notice that even a binary switch is really an analog device that can give a digital value. Is there any such thing as a TRULY digital machine in a physical sense?

(Answer on page 3-18.)

Q. 11 How many possible signals could we send with 4 binary signals? What is the highest number we could represent with these four switches? What if we have eight switches?

(Answer on page 3-18.)

Q. 12 Does addition and subtraction work the same way in binary as it does in other bases? Use the binary representations for 1(01), 2(10), and 3(11) to check.

(Answer on page 3-18.)

Q. 13 Are other bases possible besides base 10 and base 2? If so, why would you ever care?

(Answer on page 3-18.)

Q. 14 Given a computer with memory locations 1 to 10, and registers A B and C, what do you think the following "machine language" code might do?

Load value from memory 1 to register A Load value from memory 3 to register B Add Register A and Register B, putting response in C Store value in register C to memory 7

(Answer on page 3-19.)

Answers to Questions

1. The watch with an LCD panel that shows little hands is really a digital device. It has all the characteristics of a digital computer, except the display mimics an analog computer. The sundial and hourglass are analog devices, but there is no reason digital devices could not be made to act like them. Such machines would still be digital, even though they might look like analog devices.

2. The D and A represents Digital or Analog. The symbol represents how the recording was recorded, mixed, and mastered. In my CD library, my newer albums are marked DDD, meaning they were all recorded digitally, mixed digitally, and mastered digitally. "The Grateful Dead Greatest Hits" is marked ADD, meaning it was recorded with analog means, but mixed and mastered digitally. (Digital recording was not practical when much of this group's best music was recorded!) Music lovers have intense fights about which sounds better, but most would agree that DDD music sounds much like live performances.

3. Eight. You might be tempted to say six, but that is not the case. Every time we add a switch, we DOUBLE the number of messages we send. Think of it this way: With two switches, we have four messages. If we add another switch but keep it turned off, we have the same four messages we had before adding the new switch. If we turn the new switch on, we have another four brand new messages, totaling eight. Each new switch doubles the number of messages we can send.

4. Traditionally in computing, we start any counting scheme with the value zero. You will see a numbering scheme later on in this discussion called binary representation. After you understand how binary notation works, come back and look at this table again, and see if you then understand why the first message is given the value 0.

5.

Decimal	Binary
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

6. A sundial does deal with information, namely the location of the sun in the sky. (OK, the rotation of the earth in relationship to the sun, if you want to be picky!) It deals with that information in an analog fashion. The shadow of the stick is an analogy to the position of the sun. It doesn't really manipulate the information, just reflects it. It is universal to a degree, in that it could report the position of any light source, but that would give meaningless data in terms of the intended purpose, telling time. It is not a computer by out definition.

7. This comes much closer. It clearly manipulates information. That information is digital in form, and stored using binary techniques. Clearly the information is manipulated, as new values are calculated internally. It even has a rudimentery form of programming, as the

user enters values and operations on the keyboard. Computer scientists might argue about its universality. Sure, a pocket calculator can handle numbers pretty well, but what about text, sound, and graphics? It has a level of universality, but not what we expect from a full-powered modern computer.

8. Actually this is a trick question. Standard cassette tapes are still analog, but digital tape is used in recording studios, and will be practical for the consumer market very soon. Current cassette tapes use an analog recording technique storing sound waves as magnetic impulses.

9. A dimmer switch would be more like an anolog device. It has infinite precision within a range, but limited accuracy.

10. A truly digital machine would represent numbers which are a very abstract idea. Even the symbols like 1,2, and 3 are not truly digital, they are analogs to the CONCEPT of a number. However, any machine that can work with numbers (even if that numeric representation is analog in its heart) might be considered digital.

11. With four switches, we could send a total of 2ˆ4, or 16 messages. The largest number we could represent with four switches is 15. Fifteen in binary is 1111. Sixteen would require one more digit! (10000). Don't forget that 0 is a value, so we have 16 values in all.

    With eight switches, we have 2ˆ8 possible values, which works out to 256. The largest number we can represent in eight digits of binary is 255. (11111111)

12. In Base 10, $1 + 2 = 3$, so in binary, we would expect $01 + 10$ to equal 11. Lining up the values in traditional arithmetic fashion gives us:

```
  01
 +10
====
  11
```

    which works out. In fact, we can even use carrying. Remember, in binary, the only legal values for a digit are 0 and 1, so $1 + 1 = 10$. Examine the following problems for more clarification (All are in base 2)

```
  11     11    11
 +11    + 1   - 1
====    ===   ===
 110    100    10
```

    Multiplication and division work, too. (Trust me, you don't want to see it, it isn't pretty!) It's OK to trust that the computer can handle these functions in binary, since you will only see the results in base 10, a system you can understand more readily.

13. You can store any number in any base. It turns out that there are a couple of other bases that are useful in computer science. Occaisionaly you will find a piece of information that is stored in three switches. This three switch machine is capable of recording 8 different pieces of information (0-7). One digit of base 8 can do the same. When computer scientists need

to refer to the information in an eight switch configuration, they know it will be the same as one digit in base 16, which is referred to as hexidecimal. Just so you can say you've seen them, here are the numbers 0 through 16 in a number of bases:

Base 10	Base 2	Base 8	Base 16
Decimal	Binary	Octal	Hexidecimal
0	0000	000	00
1	0001	001	01
2	0010	002	02
3	0011	003	03
4	0100	004	04
5	0101	005	05
6	0110	006	06
7	0111	007	07
8	1000	010	08
9	1001	011	09
10	1010	012	0A
11	1011	013	0B
12	1100	014	0C
13	1101	015	0D
14	1110	016	0E
15	1111	017	0F
16	10000	020	10

14. It would add up the values in memory cells 1 and 3, and place the results in memory cell 7.