Chapter 1 | Introduction

Fetch, Decode, Execute and Write • Computer Language • Computer Organization • Divide and Conquer

CS102 Computer Environment

“千里之行始于足下”~老子
“A thousand miles journey starts with a single step”~Lao Tzu

The chief end of the cs102 is to demystify the computer. For most computer users, the computer is an amazing piece of machinery. It is capable of several different tasks (e.g., accounting, word processing, web browsing, internet chatting, online games, smart AI such as chess playing etc). There is scarcely a machine more versatile than the computer. Most gadgets or inventions we know are dedicated to a single task (e.g., fridge, washing machine, cars etc). Almost anything that is multi-functional has a computer somewhere inside it. However, while we amaze at the achievements of the computer, cs102 seeks to make its operations simple to understand and uncover the secrets to its versatility.

Fetch, Decode, Execute, Write

The chief insight into the secrets of the computing is hinted at in the quote given above from the Chinese Philosopher Lao Tzu and it is this: a complex task is composed of smaller and simpler sub-tasks. Take for example, a thousand mile journey on foot essentially consists of steps. How many steps? That depends on the length of your stride.

Let us illustrate the above insight with a detailed scenario. We need to buy a hard disk from Sim Lim square. However, we don’t have time to go to Sim Lim ourselves. So we ask our younger brother (or a favourite relative) to help us. Assuming that he has never been to Sim Lim before, we need to write down a list of instructions on how to go about the task. What would you write?

Perhaps the instruction list may resemble the following:

1. Go downstairs and wait at the bus stop across the road
2. Take bus 86
3. Get down the bus after 20 bus stops.
4. You will see Sim Lim square on the right, across the road. Walk there.
5. Go to the fifth floor and look for a shop called Videopro
6. Buy a hard disk …..
7.

Because your brother (or your relative) has never been to Sim Lim and perhaps unfamiliar with computer hardware, we need to give detailed instructions that consist of several basic steps, where each step is easily understood by him.

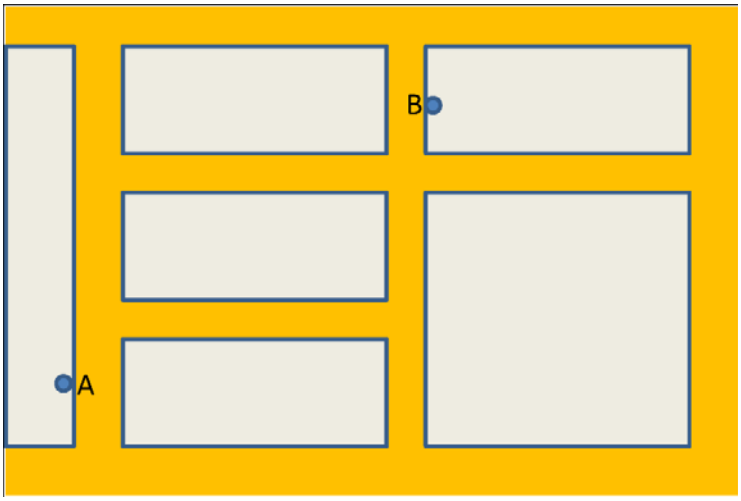


Figure 1:

Using only commands "Turn Left, Turn Right, Go Up, Go Down, Go Left, Go Right" Direct a person to go from A to B on the Map

However, the important thing is what he does with each step. We argue that the way he completes the task is by doing a "Fetch-Decode-Execute-Write" loop. In other words, what he does with every step in the list above consist of 4 stages:

- Fetch – he reads the next instruction to be executed. We call this *fetch*.
- Decode – the list is written in English. So he needs to *decode* the alphabets and words to decipher the meaning of the instruction. We may not be so conscious of this stage in real life because we do it so intuitively. But we have to realize that this intuitive response is due to years of using English as a medium.
- Execute – Now we know what the instruction means, carry out the instruction. *Execute* it.
- Writeback – The *writeback* stage refers to the change in the environment or state after the execution of the instruction. For example, if step 1 above is executed, instead of being at home, he should now be physically at the bus stop across the road.

Let us illustrate this with another example. Suppose we need to direct a person to go from A to B as shown in Figure 1. In this contrived example, the only commands we can issue to the person would be "Turn left", "Turn Right", "Go Up", "Go Down", "Go left", "Go right". Assuming that the person was facing right at the beginning. One way for him to go from A to B might be:

1. Turn Left
2. Go Up
3. Turn Right
4. Go Right
5. Turn Left
6. Go Up

Again, how a person would follow these commands will follow the fetch-decode-execute-write pattern for every step listed above. But notice that we have included in the latter illustration an additional constraint—the list of commands that can be used (no other commands can be issued).

Similarly, every computer comes with its own set of instructions. The instructions found in the set form the basic steps that can be performed by the computer. Any program that is to be executed by the computer has to be a sequence of these steps. The computer follows this sequence of instructions and what the computer does with each instruction is fetch, decode, execute and write. [1] We call this the fetch-decode-execute-write cycle because this is what the computer does with **every** instruction in the program.

Computer Language

So we have established that we can instruct the computer to perform the tasks by giving the computer a sequence of instructions. When we communicate with each other in English, we use words that are made up of alphabets. What are the alphabets for a computer then? Two alphabets – 0 and 1. Everything the computer understands need to be stated using 0s and 1s.

Digital versus Analog

The computers, as we have it, are electrical devices. Therefore, we need some way of representing 0s and 1s so that the computer can understand. One way to represent 0s and 1s (also known as logic levels) is to measure the voltage levels. Let's say a high voltage of 5V would be 1 and a low voltage of 0V would be 0. In an ideal situation, we would like all electrical circuits to be always either 5V or 0V. However, we do not live in a black and white world. Rather, we know that between black and white, there is a range of shades of grey. There is bound to be fluctuations and variations in the electrical current.

Figure 2 illustrates the problem we have. Electrical current are bound to vary slightly over time due to physical reasons. The real voltage that we measure is represented by the curvy blue line that fluctuates over time. We call this an **analog** signal because it is continuous – the value of the voltage can be anything between 0 and 5. What we need for a computer ideally is a **digital** signal – a signal that is discrete as opposite to continuous i.e., it should always be either 5 or 0 and nothing else in between. Therefore in practice, we set what is known as voltage levels. For example, we can consider all voltages between 0V and 0.8V to be 0 and all voltages between 2.7V and 5V to be 1. Anything else would result in error and it is the job of the electrical engineer to make sure that the fluctuations due to noise would not exceed the acceptable range defined above.

[1] Almost all computers have an **instruction set**, where each "instruction" is a basic kind of execution "steps" supported by the computer. A*ny* program that runs on the computer is a sequence of basic instructions.

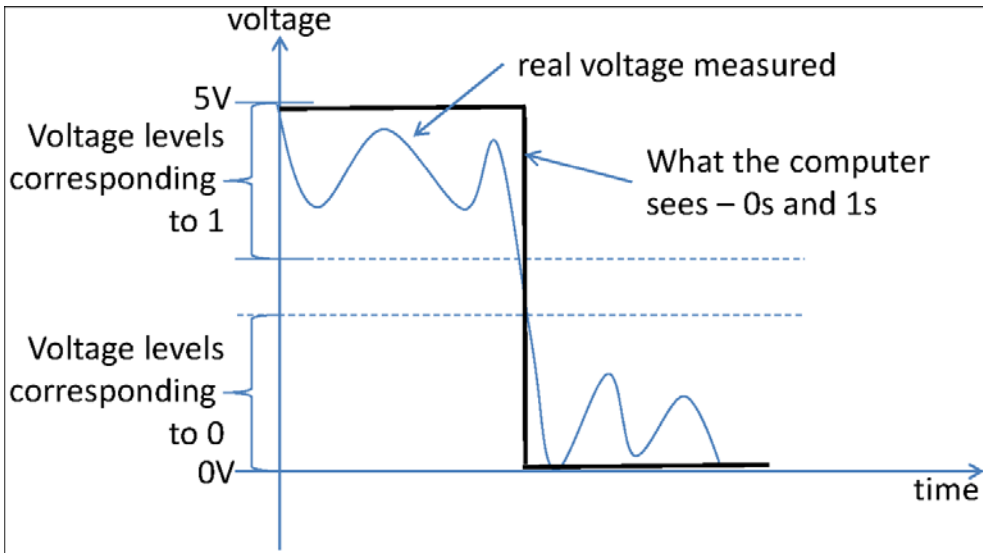


Figure 2: How analog voltages are converted into digital signals

Encoding

So far, we have understood that the language of computers to consist of only 2 alphabets: 0 and 1s. English, on the other hand, is made up of 26 alphabets. We will use the alphabets in English to form words. And the words will be used to form sentences. And sentences then can be used to form paragraphs. However, in English, we have a way to distinguish between words. Consider the following:

iamaboy

with

I am a boy

The spaces between words serve to differentiate between the words. The reason we need the 'spaces' in English is because the words are made up of different length. We can think of the space as an additional/implicit alphabet to help us to make out the words. However, computers are made up of 0s and 1s **only**. So given a string of 0s and 1s, how does the computer manage to read it? Every word in the computer has a fixed length.

Consider this:

1010101111001101000000100100011

The above string of 0s and 1s would have 2 words, each word consisting of 16 alphabets. It is time to introduce the notion of "bits" – a short form for binary digits. So in this example, a word consists of 16 bits.

In summary, the language the computer speaks is made up of bits – 0s and 1s. Instructions need to be encoded into 0s and 1s so that the computer can fetch these instructions (made up of 0s and 1s) , decode them (understand them), execute the instruction and update the state.

Computer Organization

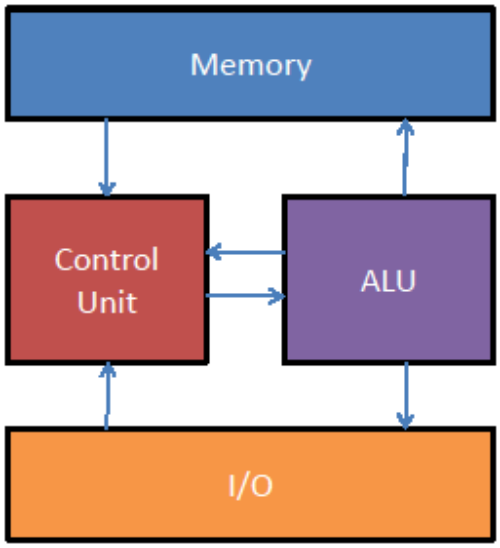


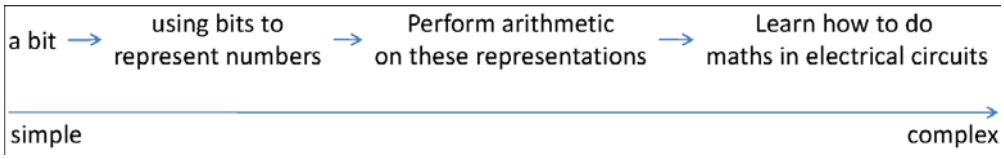
Figure 3: Basic Computer Organization

We are now in a position to introduce how the computer is organized. We have understood so far that how the computer works is fetch-decode-execute-writeback and that everything the computer understands need to be encoded in 0s and 1s. In order to help the computer do this, the computer is made of the following major components:

- Memory
  - A storage place for the instructions of the programs and the data. (Think of data like the state of the program.)
- Control Unit
  - A piece of circuit that controls that reads the instruction from Memory and informs the ALU(see below) what to do.
  - Every instruction consists of opcode and operand. The opcode is a binary representation of the desired operation e.g., add, sub, mul, read from memory, write to memory etc.
  - The Control Unit decodes the instruction into opcode and operands, then reads in the appropriate operands (either from I/O or memory) and pass the binary values to be operated on to the ALU.
- ALU (Arithmetic-Logic Unit)
  - A piece of circuit that can perform basic maths like add, subtract, multiplication etc. What operation it performs depends on the control unit. May write new data back into Memory or I/O.

Divide and Conquer

The purpose of CS102 is to learn how the computer works. As we have tried to show so far, a key strategy we are taking to learn this is by doing divide and conquer. We understand the complex by understanding the simpler things that make up the complex thing. For example, a strategy taken in this module is the following:



We learn incrementally. By understanding the simple parts, we will get to understand the complex whole. Incidentally, divide and conquer is also how the computer works. Everything the computer does consist of very simple and basic steps. And it is by combining these basic steps we get a computer to execute programs that we consider wonderful – like word processing, games, web browsing etc. A thousand miles journey starts with a single step.