How Computers Work
Dr. John P. Abraham

Computers are used all over the world, by people of many languages, and for specialized and general purposes as we discussed in the previous chapter. How is that possible? Does a computer understand many languages and many disciplines? Absolutely not! It is the software incorporated into the hardware that makes computers so useful. At the hardware level it only recognizes switches that are turned off and on. By the very nature of transistors, they can act as switches and amplifiers. As more transistors are packed into a CPUs, it becomes very powerful.

One switch that provides two states, on or off (1,0), is called a bit (short for binary digit). We could use these two states to mean anything, stop/go, sit/stand, red/blue, yes/no, in/out, etc. Two switches will provide us 4 states, three switches 8 states, four switches 16 states, and so on. The basic unit of switches used in a computer is 8 switches (byte) or multiples of 8 (word). Eight switches provide us 256 states or it can represent 256 items. It can easily represent 26 English characters (alphabet), 10 symbols used for numbers, etc. Most languages have lesser than 256 characters in their alphabet. Some languages have more than 256 characters, so we need multiples of 8 bits. English alphabet (actually it is the modified Roman alphabet) uses the ASCII coding scheme, where 01000001 represents “A”. The Unicode provides for more than 256 characters in an alphabet. More about it can be found in the next chapter.

When we have someone to do something for us, we give him/her instructions and data to accomplish the task. For instance if you ask someone to make a telephone call, the instruction (opcode) is “call” and you give a telephone number, that would be the data (operand). When we give instructions to a computer, it is called a program. The program is stored in a portion of the memory and the data is stored in another part of the memory. Such machines that stores program in its memory are called Von Neumann machines. As long as the computer is given the beginning location of the program it can run the program as needed. This also is explained in the next two chapters. The number of instructions available within a computer (CPU) is small, 32, 64 or 128. Each instruction is given a unique binary pattern. In the following chapters you will be asked to come up with bit patterns for different instructions.

A computer system has a central processing unit, and the input and output units. The CPU fetches one instruction at a time, decodes it, and if data is required it fetches that data, executes the instruction, and puts the results back in the instructed location. Here are the steps:

1. Figure out where the next instruction is in the memory (instruction address calculation).
2. Fetch the instruction
3. Figure out what the fetched code means (decoding the instruction).
4. If it requires an operand
    a. Figure out where the operand is kept (operand address calculation).
    b. Fetch the operand
    c. If there are multiple operands repeat a and b.
5. Execute the instruction (such as add, subtract, multiply, jump, loop, etc)
6. Figure out the location to store the result (operand address calculation)
7. store the result, if there are multiple results repeat 6 and 7.
8. Go back to 1 to do the next instruction, or End if it is the last instruction.

These steps need to be modified for today's sophisticated computers. For instance when a memory address is calculated, it may not be an absolute address. It could be a register address, relative address or virtual address. The next instruction is located in a special memory called a register known as the program counter (PC). Registers are the fastest memory inside the CPU. We will be studying about various types of storage locations and some techniques used to speed up memory access in the next few weeks.

As mentioned above an instruction should contain the opcode and the operand(s). It should also indicate where to put the result unless it is implied. Similarly, it should also indicate where the next instruction is found, unless it is implied. The operand could be directly supplied, as in the example of the telephone call instruction discussed earlier. It is equally possible for this example to give a telephone book for the user to lookup the number. We will have a lecture on addressing modes and formats.

In the previous paragraphs it was mentioned that instructions and data were stored in different parts of the memory. Memory can be considered as boxes or group of boxes in which we can place either data or instructions. How big are these boxes? It varies from one byte to several bytes (word); word is a term for the natural unit of data. Size of word depends upon the width of the buses and size of the registers in the CPU. Amount of data transferred between the CPU and the cache memory is a word. It is possible the size of a word to be smaller than the data bus to keep downward compatibility. It is also possible for a word to be twice the size of the dat bus; in which case the data will be transferred in two chunks. The CPU (central processing unit) fetches each instruction, from the memory, decodes it, fetches data if need be and executes the instruction, and stores the result in the memory.

The following example of adding two numbers are taken from William Stalling’s Computer Organization and Architecture. Suppose two variables, X and Y, are stored in memory locations 513 and 514 respectively. We want to do the operation X = X+Y, meaning add the contents of 513 and 514 together and store the result in 513. This can be accomplished by two different computers as follow:

1. Load X into accumulator
2. Add Y to the contents of the accumulator
3. Store the contents of the accumulator into X

OR

1. Load X and Y into registers R1 and R2
2. Add R1 and R2 placing the result in R1
3. Store R1 to X

The first computer only has one usable register called the accumulator (AC) therefore this computer will work much slower than the second machine. Main Memory based operations are very slow. The second machine has at least two usable registers, perhaps many more (as many as 128 or 256). We will look at the Accumulator machine in more detail in next chapter.

We work with all types of operands such as small numbers, large numbers, decimals, characters, yes/no, etc. We refer to these as data types. Many of you, when programming, used data types.

Please be thinking about the repertoire of instructions that you might need in a computer, particularly the one you will be asked to design within the next couple of weeks. You need instructions to move the data from memory to CPU and back, and to move data from memory to other devices such as storage, video or printer. You would also need instructions for arithmetic and logic operations and for branching. How about for video and audio? What else can you think about?

Other thinking points: How big of a memory will you need? If you put something in memory how would you know where it was placed? How big of a number can you assign to a memory address?