3. Fundamentals of a Program

Basics

- Computers ~~[illegible]~~ are very dumb machines.
  - They only do what they are told to do

- The basic operations of a computer will form what is known as the computer's instruction set. So, we need to tell the computer how it's going to do, we do it in the form of a program at an even lower level. Each computer has what's called an instruction set and instruction set is basically on the CPU (processor)

- To solve a program using a computer, you must provide a solution to the problem by sending instructions to the instruction sets.
  - a computer program sends the instructions necessary to solve a specific problem.

- The approach or method that is used to solve the program is known as an algorithm.
  - So, if we to create a program that tests if a number is odd or even.
    - The method that is used to test if the number is even or odd is the algorithm.

- To write a program, you need to write the instructions necessary to implement the algorithm.
  - These instructions would be expressed in the statements of a particular computer language, such as Java, C++, Objective C, or C.

Terminology

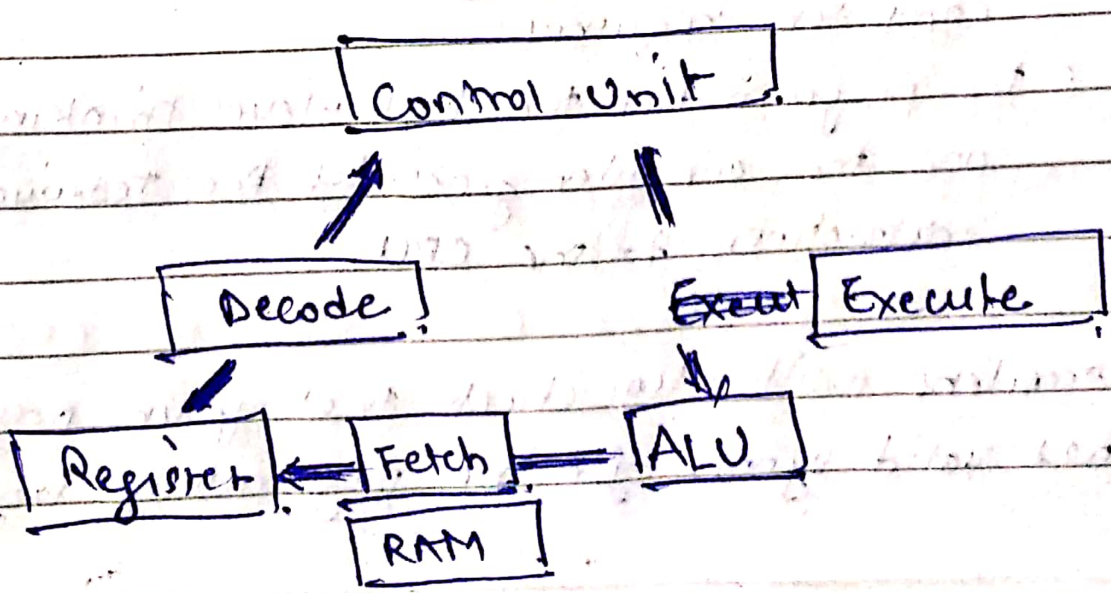
- CPU (central processing unit)
  - does most of the computing work
  - Instructions are executed here.

- RAM (random access memory)
  - stores the data of a program while it is running

- hard drive (permanent storage)
  - stores files that contain program source code, even while the computer is turned off.

- Operating system
  - developed to help make it more convenient to use computers.
  - a program that controls the entire operation of a computer.
    - All input and output
    - manage the computer's resources and handles the execution of programs.
    - Windows, Unix, Android, etc.

- Fetch / Execute cycle (life of a CPU)
  - fetches an instruction from memory (using registers) and executes it (loop).
  - A gigahertz CPU can do this about a billion times a second.



Higher Level Programming Language:

- High-level programming language makes it easier to write a programs.
  - Opposite of assembly language.
  - C is a higher level programming language that describes actions in a more abstract form.
  - The instructions (statements) of a program look more like problem solving steps.
  - do not have to worry about the precise steps to particular CPU would have to take to accomplish a particular task
    - total = x + v2, mv ax, 5, mv cx 4, etc.....

- Compilers
  - a program that translates the higher-level language source code into the detailed set of machine language instructions the computer requires.
  - The program does the high-level thinking and the compiler generates the tedious instructions to the CPU.

- Compilers will also check that your program has valid syntax for the programming language

Scanned with CamScanner

That you are compiling.
- finds errors and it reports them to you and doesn't produce an executable until you fix them.

- high-level languages are easier to learn and much easier to program in than are machine languages.

Writing a program

- The act of writing a C Program can be broken down into multiple steps.

1. Define a program objectives
2. Design a the program
3. Write the code.
4. compile
5. Run the Program
6. Test and debug the program
7. Maintain and modify the program

Steps in writing a program.

1. Define the Program objectives.
   - Understand the requirements of the program.
   - Get a clear idea of what you want the program to acicomplish

2. Design
   - Decide how the program will meet the above requirements.
   - What should the user interface be like?
   - How should the program be organized?

3. Write the code.
   - Start implementation, translate the design in the syntax of C.
   - you need to use a text editor to create what is called a source code file.

4. Compile
   - translate the source code into machine code (executable code).
   - consists of detailed instructions to the CPU expressed in a numeric code.

5. Run the Program
- The executable file is a program you can run

6. Test and Debug
- Just because a program is running, does not mean it works as intended.
- Need to test, to see that your program does what it is supposed to do (may find bugs).
  - Debugging is the process of finding and fixing program errors.
  - Making mistakes is a natural part of learning.

7. Maintain and Modify the Program.
- Programs are released and used by many people.
- have to continue to fix new bugs or add new features.

- For the above steps, you may have to jump around steps and repeat steps.
  - E.g. when you are writing code, you might find that your plan was impractical.

- Many new programmers ignore step 1 and 2 and go directly to writing code.
  - A big mistake for larger programs, may be ok for very simple programs.

- The larger and more complex the program is, the more planning it requires.
- Should develop the hebit of planning before coding.

- Also, while you are coding, you always want to work in small steps and constantly test. (divide and conquer)