2.1 Prologue:

Interface: the [place](http://dictionary.cambridge.org/us/dictionary/english/place) where two [systems](http://dictionary.cambridge.org/us/dictionary/english/system) come together and have an [effect](http://dictionary.cambridge.org/us/dictionary/english/effect) on each other, or a [connection](http://dictionary.cambridge.org/us/dictionary/english/connection) between two [computers](http://dictionary.cambridge.org/us/dictionary/english/computer) or between a [person](http://dictionary.cambridge.org/us/dictionary/english/person) and a [computer](http://dictionary.cambridge.org/us/dictionary/english/computer)

Mnemonic: something such as a poem or a sentence that you use to help you remember a rule, a name etc

Computer program can be stored in the computer memory itself in the form of encoded CPU instructions

* Programers wrote programs, technicians punched the programs into *punch cards*/ perforated tape.
* Programs were written in machine code using octal or hexadecimal instruction codes and memory addresses. ( coding: writing programs and encrypting texts). Run programs faster by developing efficient algorithms

**2.2.** Use Assembly lang. instead of numerically coded CPU, dependent on parti CPU ( a special program, assembler, converts the text of a program written in assembly lang. Into the machine code expected by the CPU), porting a program to a different type of machine would require rewriting the code (assembly lang.)

* High-level programing languages were developed for writing programs in a more abstract, machine-independent way. These programs obey very formal syntax rules of the lang. ( unambiguous, computer can interpret correctly), and follow styles ( easier to read, not mandatory)
* The text of a program in a parti programming language is referred to as source code, or simply the source. The source code is stored in a file, called the source file.
* To be able to be used in a computer, high-level programing language has to be converted to CPU instructions, ( an aproach with C++ called a compiler-> analyze the source’s code and generate into CPU, result saved in another file, object module ).

A large program may include several source files that are compiled into object modules separately.

Another program, **linker**, combine all the object modules into one executable program and saves it in executable file.

* editor-> source code -> compiler -> object code -> linker -> executable program -> program users
* Another approach, not compiler, intepreter. Difference: interpreter looks at what needs to be executed in high-level programing language and execute them; does not generate and store in object modules
* **Java also uses a compiler, but, as we will explain shortly, the Java compiler does not generate object code for a parti CPU**
* **Java is differental it uses a mixed compiler-plus-interpreter approach.**
* Java compiler compiles the program into bytecode ( ><machine w this machine lang. does not exist) , bytecode is then interpreted by Java interpreter for that parti CPU. A program in bytecode is not object code (b/c its platform-independent, same version of bytecode can be used w different types of computers), not source code ( not readable by humans)
* Java-enabled browser ( a browser that has a Java interpreter built into it) can run little Java programs, *applets*, (miniature applications)
* editor-> source code-> compiler -> bytecode-> interpreter
* JIT ( Just In Time)-> speed up applets: combine 2 features of compiler and interpreter. While interpreting bytecode, it also compiles it into executable code.
* Mordern software development systems combine an editor, a compiler, and other tools into one Intergrated Development (IDE), IDE has GUI (graphical user interface) - 1 mouse click on an icon will compie and run your program.
* Few programs are written on the first try w/o errors, or bugs

**Syntax Errors and Logic errors:**

Run-time errors: when the program is running. Some run-time errors cause exception, the program encounters fatal condition and is aborted with an error message. ( arithmetic exeption: division by a zero)-> may lead unexpected behavior/incorrect results.

* Cannot be solved by looking at the source code/ testing the program on different data -> debugger: slow in motion-> programmer can check the steps.

**2.3 Software Components and Packages**

You can reuse sources/ codes w some requirements p. 22

Individual software components are usually combined into packages. A package combines functions that deal w a parti set of stuctrures or objects: a graphics package that deals w graphic capabilities; a file package that helps to read and find files, etc.

**HW:**

#1: which are advantages of a high-level programming language, as opposed to a machine language?

1. It is easier to write a program. FALSE
2. It is easier to read and understand programs. TRUE
3. Programs run more effectively. FALSE
4. Programs can be ported more easily from one platform to another. TRUE

#2 Four commonly used programming languages besides Java:

Pascal and C, FORTRAN, C++, BASIC, Python

#4 Which program helps programmers enter and modify source code? D. Interpreter

#5 (MC) What is a debugger used for?

B. Running and Tracing programs in a controlled way

* Terminal : ls: lsit

Cd: change direction