**Chapter 1: INTRODUCTION & VIRTUAL MACHINE**

**Topic – 1: Welcome to Assembly Language**

**Introduction**

* Recommended to run in latest version of **Microsoft Visual Studio**.
* Check installation & setup steps at [www.asmirvine.com](http://www.asmirvine.com).
* We are going to learn **MASM** (macro assembler).

**Other Assembly Languages**

* **TASM** (turbo assembler) [For Windows]
* **NASM** [For Linux] {syntax similar to **MASM**}
* **MASM32** (a variant) [For Windows]
* **GAS** (GNU assembler) [For Linux]

**Fact!**

**🡪 Donald Knuth is known as father of asymptotic-notations.**

**Topic – 2: Questions You Might Ask**

**Common Terms**

* **Assembler:** Converts assembly codes to machine code.
* **Linker:** Combines all related assembly programs into single **executable** program.
* **Debugger:** Allows programmer to examine **registers** & **memories**.

**What programs can MASM create?**

* **32-Bit Protected Mode:** For 32-bit Windows versions.
* **64-Bit Modes**
* **16-Bit Real Address Mode:** For 32-bit Windows versions & embedded systems.

**Language Relevancies**

* Assembly code to machine code – **One-to-one mnemonics**
* High-level code to assembly/machine code – **One-to-many relation**
* **One-to-many** as **one** instruction in high-level language will unfold into **multiple** assembly instructions.

**Topic – 3: C++ to Assembly Conversion**

**C++**

***int Y;***

***int X = (Y + 4) \* 3;***

**NASM**

***mov eax, Y ; move Y to the EAX register***

***add eax, 4 ; add 4 to the EAX register***

***mov ebx, 3 ; move 3 to the EBX register***

***imul ebx ; multiply EAX by EBX***

***mov [X], eax ; move EAX to X***

**Topic – 4: Assembly Languages Rules**

**Unrestricted Memory Use**

* Java **restricts** programmers from accessing certain memory addresses.
* To address this problem, **JNI** is imported for writing **C** code in that memory space.
* **JNI:** Java native interface
* Assembly on the other hand has **no restriction** on accessing any memory address.

**Assembly Language Applications**

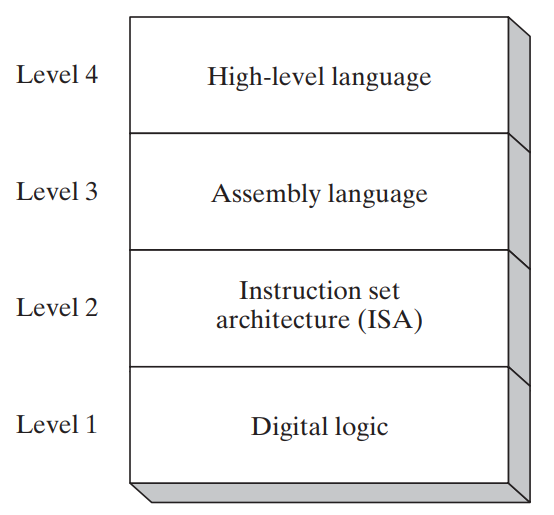
* Benefit of using **OOP languages** is that they can contain **millions** of lines of codes.

**Topic – 5: Virtual Machine Concept**

**Virtual Machines**

* Java uses **JRE** to convert its **Java** native code into **Java byte code**.
* And this **Java byte code** is at immediate lower level of abstraction to **Java** code.
* So, when executing, this **Java byte code** which works at lower level is executed by **JVM**.
* JVM is a virtual machine.
* Means user is seeing something, but there is something else happening in hardware.
* And this **JVM** implemented on various processor architecture is what makes Java **platform independent**.

**Virtual Machine Levels**



* In **ISA (level-2)** the computer chip manufacturers embed **microprograms**.
* **Microprogram:** A program which executes **machine code**.
* **Microcode Interpreter:** Converts **machine code** into **digital circuit-level operations**.
* Assembly codes are easily translated to machine code.