**What is assembly language?**

Is a a language that produces exatly one machine instruction. In other words is one-to-one corrspondece between machine instructions and statements in the assembly program. If each line in the assembly language program contains exatly one statment and each machine word contains exactly one machine instruction

**What is an assembler?**

When the source language is essentially a symbolic repressentation for a numerical machine language the translater is called an assembler else when translates high level language is called a compiler .

**What is a macro?**

A symbol, name, or key that represents a list of commands, actions, or keystrokes. Many programs allow you to create macros so that you can enter a single character or word to perform a whole series of actions. Suppose, for example, that you are editing a file and want to indent every third line five spaces. If your word processor supports macros, you can create one that consists of the following keystrokes:

Macros are used to make a sequence of computing instructions available to the programmer as a single program statement, making the programming task less tedious and less error-prone.[1][2] (Thus, they are called "macros" because a big block of code can be expanded from a small sequence of characters).

**What are assembler directives?**

Assembly directives, also called pseudo-opcodes, pseudo-operations or pseudo-ops, are instructions that are executed by an assembler at assembly time, not by a CPU at run time. The names of pseudo-ops often start with a dot to distinguish them from machine instructions. Pseudo-ops can make the assembly of the program dependent on parameters input by a programmer, so that one program can be assembled different ways, perhaps for different applications. Or, a pseudo-op can be used to manipulate presentation of a program to make it easier to read and maintain. Another common use of pseudo-ops is to reserve storage areas for run-time data and optionally initialize their contents to known value

<http://en.wikipedia.org/wiki/Assembly_language>

**What are text and data sections?**

.data and .text are both directives to the assembler. .data tells the assembler that the upcoming section is considered data. .text tells the assembler that the upcoming section is considered assembly language instructions.

pag 38-39 text on the mips book

<http://www.cs.umd.edu/class/sum2003/cmsc311/Notes/Mips/dataseg.html>

**How are decision statements implemented?**

In its most basic form, a decision is some sort of branch within the code that switches between two possible execution paths based on some condition. Normally (though not always), conditional instruction sequences are implemented with the conditional jump instructions

[**http://www.plantation-productions.com/Webster/www.artofasm.com/Linux/HTML/LowLevelControlStructsa2.html**](http://www.plantation-productions.com/Webster/www.artofasm.com/Linux/HTML/LowLevelControlStructsa2.html)

**How are loops implemented?**

Program loops consist of three components: an optional initialization component, a loop termination test, and the body of the loop. The order with which these components are assembled can dramatically change the way the loop operates. Three permutations of these components appear over and over again. Because of their frequency, these loop structures are given special names in high-level languages: WHILE loops, REPEAT..UNTIL loops (do..while in C/C++), and infinite loops (e.g., FOREVER..ENDFOR in HLA).

**How are arrays implemented in memory? How are the accessed in assembly code?**

**To access an array in assembly language, we use a pointer. A pointer is simply a register or variable that contains a memory address.**

**The value in the pointer is computed as shown in the previous sections by adding the base address of the array and the offset of the desired element.**

[**http://www.cs.uwm.edu/classes/cs315/Bacon/Lecture/HTML/ch12s04.html**](http://www.cs.uwm.edu/classes/cs315/Bacon/Lecture/HTML/ch12s04.html)

Part of the computation can be done using offset addressing mode, but note that the offset in offset addressing mode is in bytes, and does not account for the size of an element. For example, if working with words in MAL, we must manually multiply the offset by 4.

[**https://courses.engr.illinois.edu/ece390/books/artofasm/CH05/CH05-2.html**](https://courses.engr.illinois.edu/ece390/books/artofasm/CH05/CH05-2.html)