Assembly is the final step in the translation to machine code. Instruction specifications are converted into the actual patterns recognized by the control unit of the computer, and these patterns are placed into a memory image. Some of the instruction specifications normally contain references to other items in the program; during assembly, these references are replaced by the addresses in the memory image of the referenced items.

In many respects, assembly is machine-dependent: The number and layout of fields within an instruction, the length of an instruction and the representation of a memory address are all items which vary from one computer to another. Nevertheless it is possible to derive a general model of the overall assembly process which
highlights its basic machine-independence. By studying the model, we can identify important interfaces and specify a collection of procedures which can be used to carry out an assembly. These procedures are machine-dependent only in their detailed operation; their functions and interconnections are independent of any particular computer or class of computers.

It is often convenient to split an assembly into several passes. The most common reason is to permit separate translation of modules in a large program. Each module is partially assembled, and the text placed in a file. When the program is to be executed, the files for all modules are combined in a final assembly pass. This final pass is usually termed "loading", "linkage editing" or "binding", but in reality is a completion of assembly.