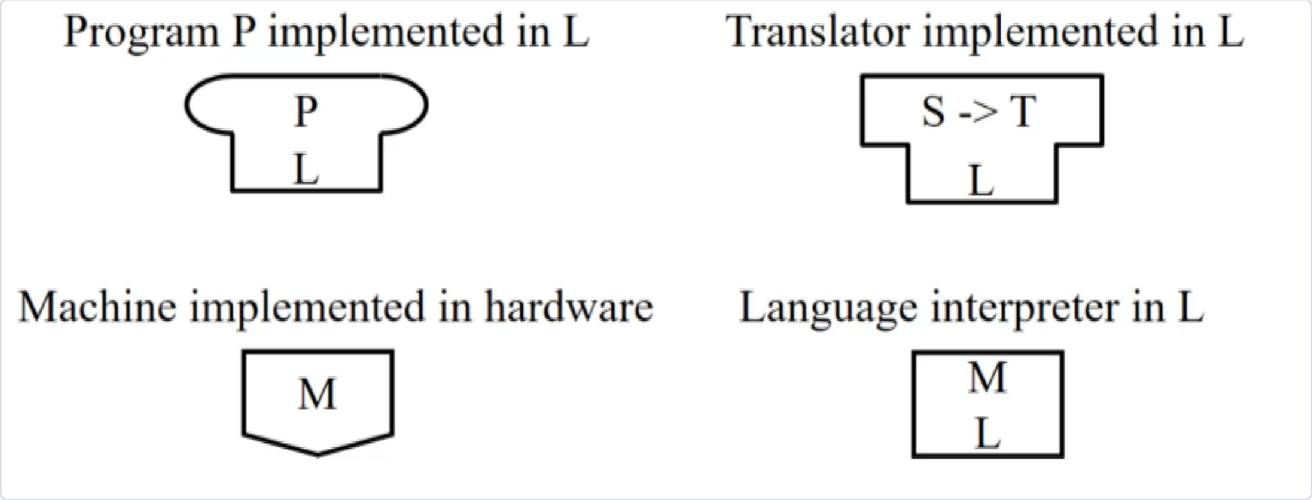
What are Tombstone Diagrams

Tombstone diagrams are used to illustrate the connection between program, compiler and the machines to run the program.
Below can be seen the different tombstone "versions" and their meaning.



Use

Tombstone diagrams are a way of illustrating the process or lifecycle of a program going from a certain language to another that is executable by the target machine. For example, if you have a program *X*, written in Language *Y*, on a machine that uses language *Z*. You would not be able to run the said program on the machine as the language used by the actual machine, is not the same as what was written for the program. This is where Compilers come in. A compiler is needed to translate the program from Language *Y* into language *Z*.

Cross-compilation

Cross-compilation is used when a program is written in language *X* for a target language *Y* on a machine using language *Z*. By cross-compilation, a compiler is used to compile the language into the language of the target machine rather the language of the same machine. A compiler used for cross compilation therefore uses language *Z* going from language *X* to *Y*. This is illustrated in the picture below:

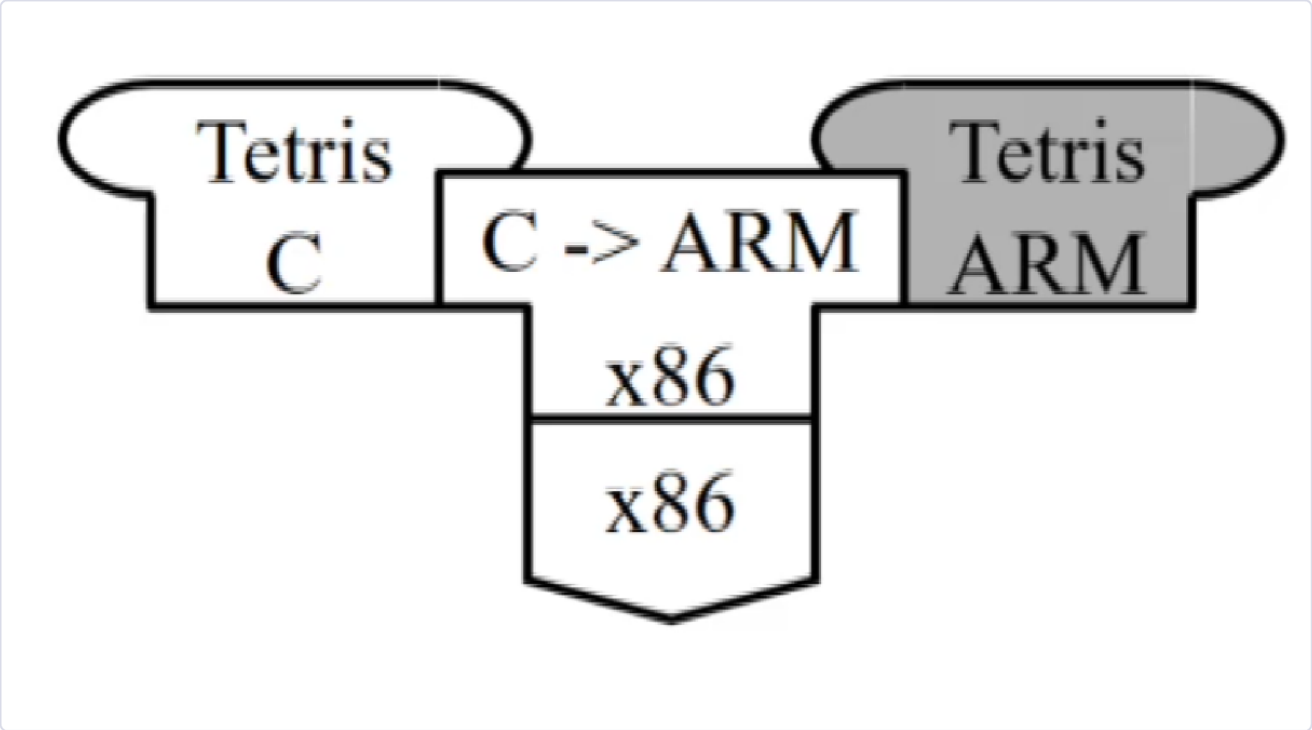


Illustration of Cross compilation

As seen on the illustration, Tetris is here written in the language C and the target is an ARM machine. And the machine used at development is an x86 machine. Using cross-compilation then translate the Tetris program from C to ARM using a compiler build for x86 machines.

Two Stage Compilation

Two Step Compilation is quite self-explainatory. A compiler translate a program from language X into Y, and another compiler translate the output of the first compiler from language Y to Z. This is used for Java, which translates from Java to JVM to Machine Code.