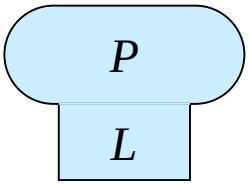
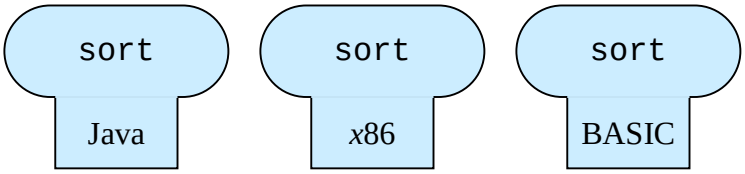
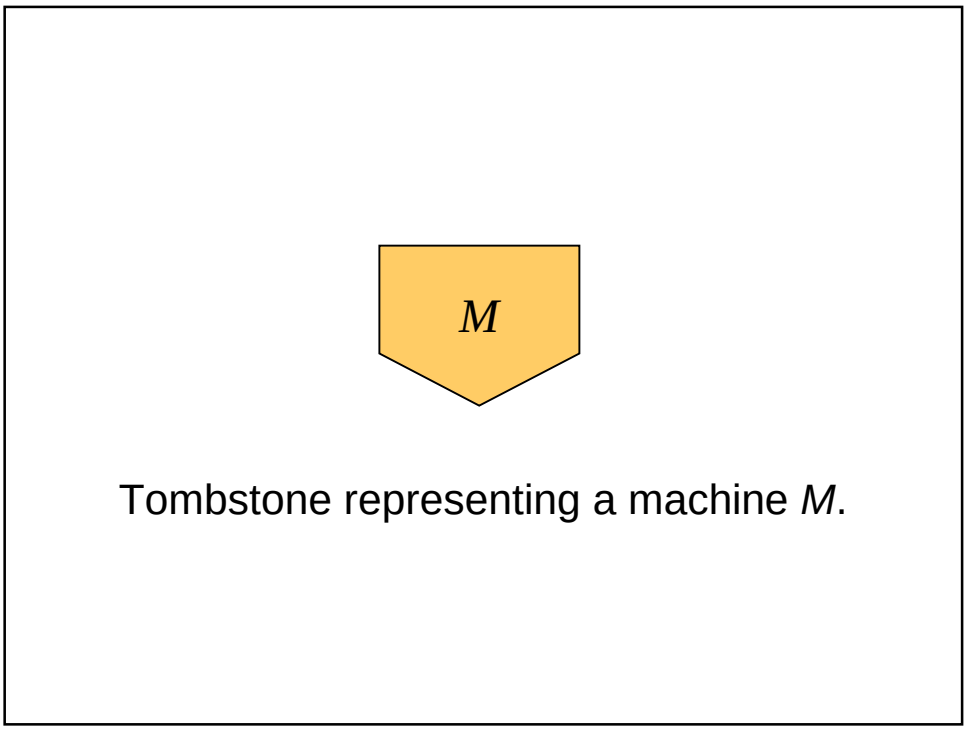
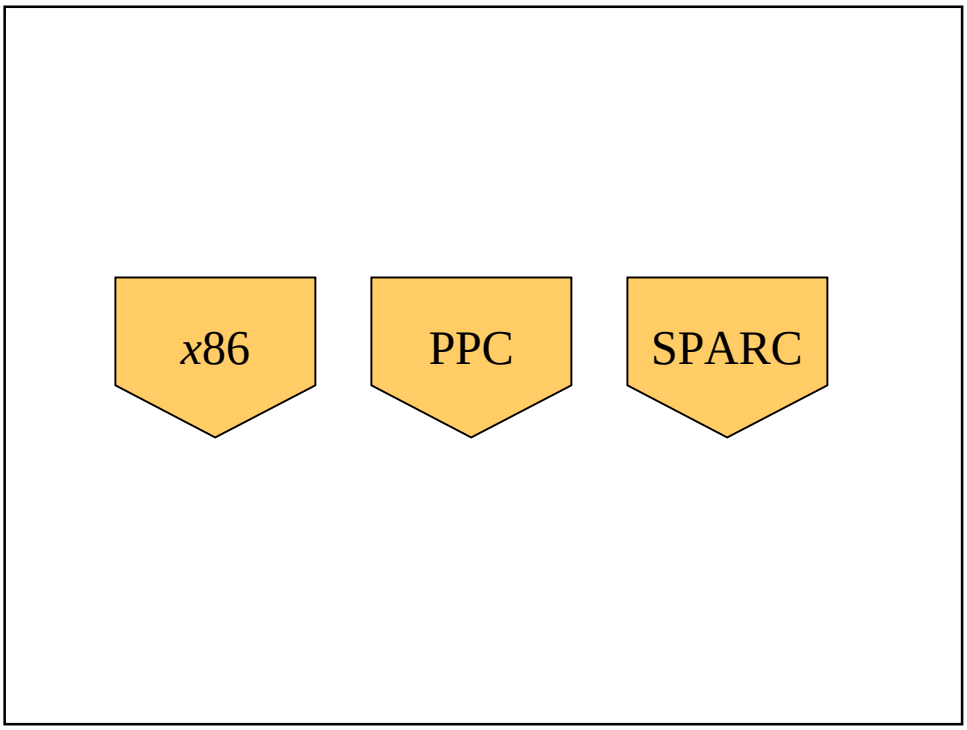


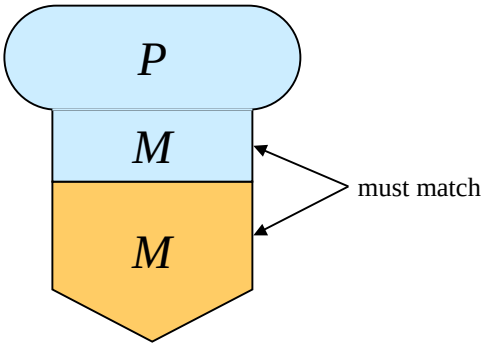
Tombstone representing a program *P* expressed in language *L*.



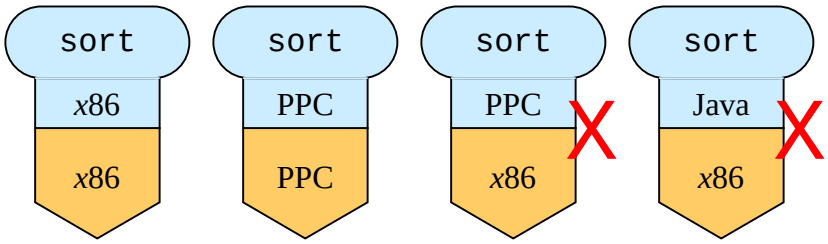


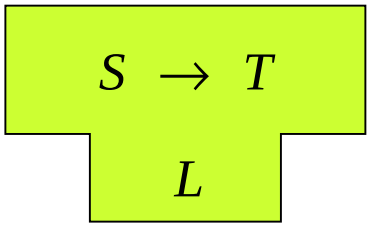
Tombstone representing a machine *M*.



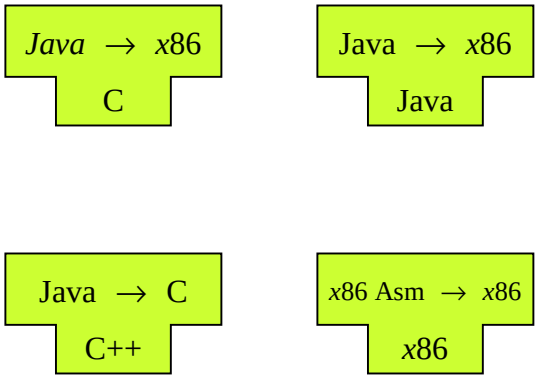


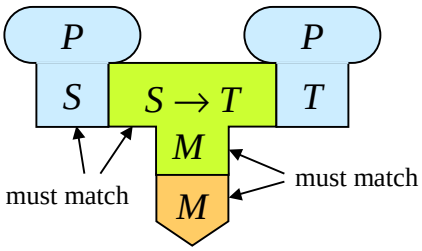
Running program *P* on machine *M*.





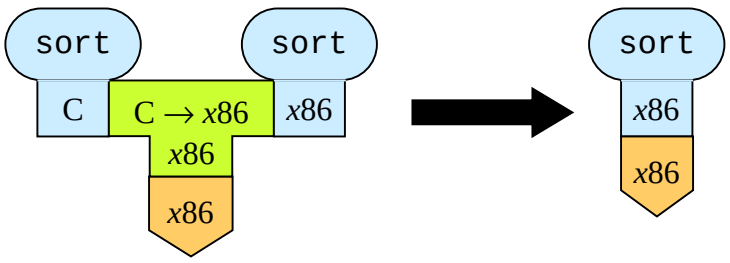
Tombstone representing an *S*-into-*T* translator expressed in language *L*.



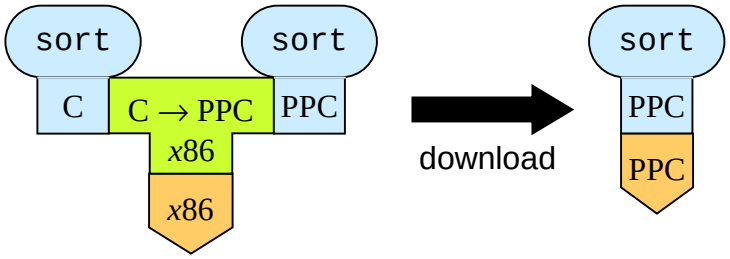


Translating a source program *P* expressed in language *S* to an object program expressed in language *T*, using an *S*-into-*T* translator running on machine *M*.

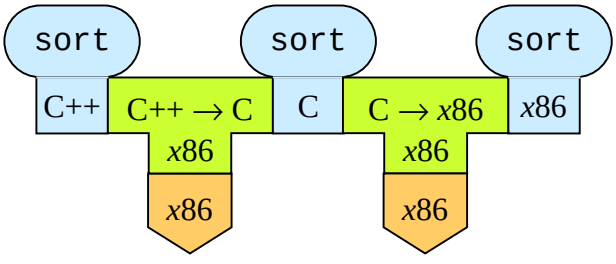
Compilation



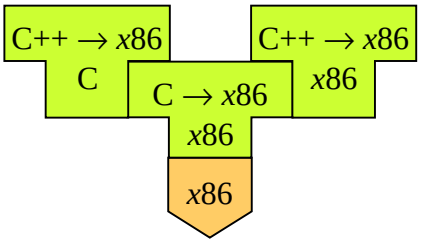
Cross-compilation

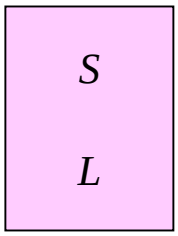


Two-stage compilation

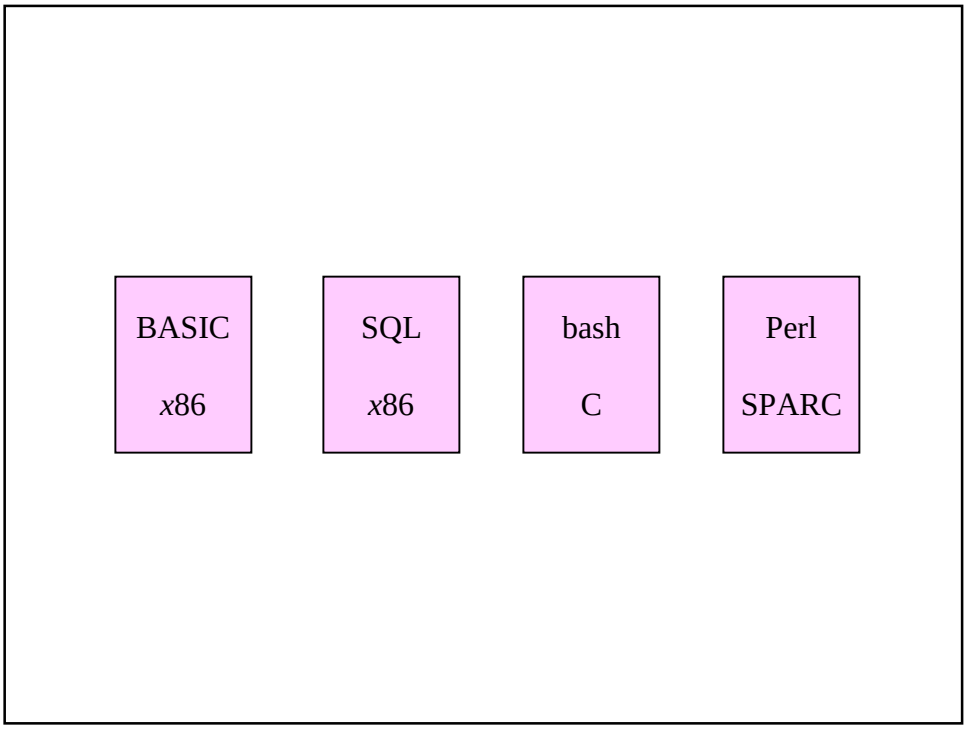


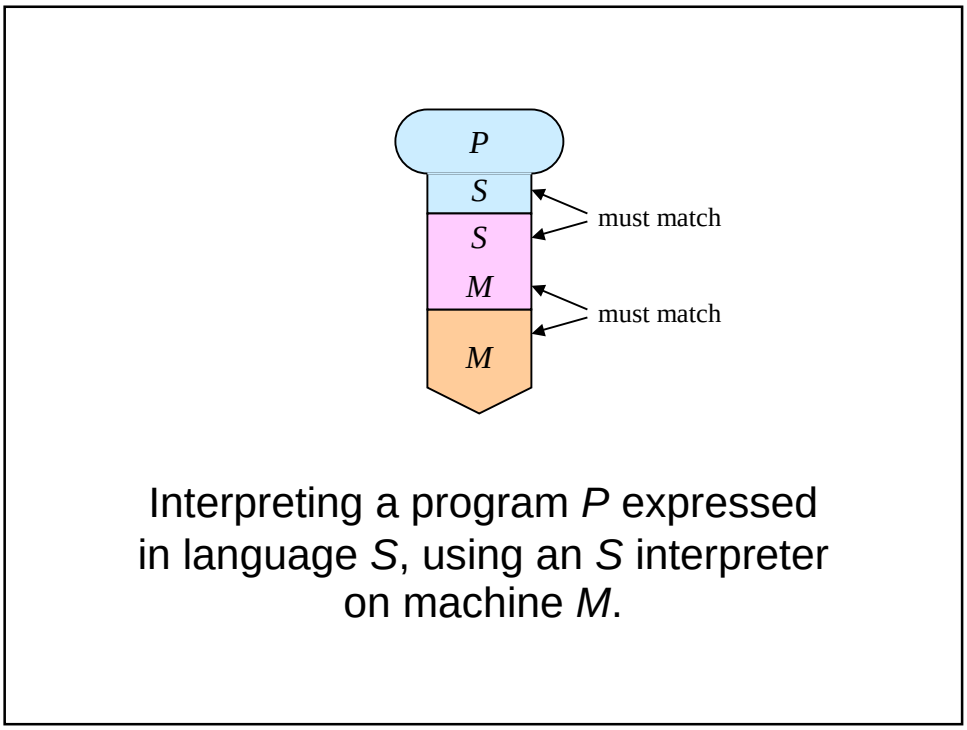
Compiling a compiler



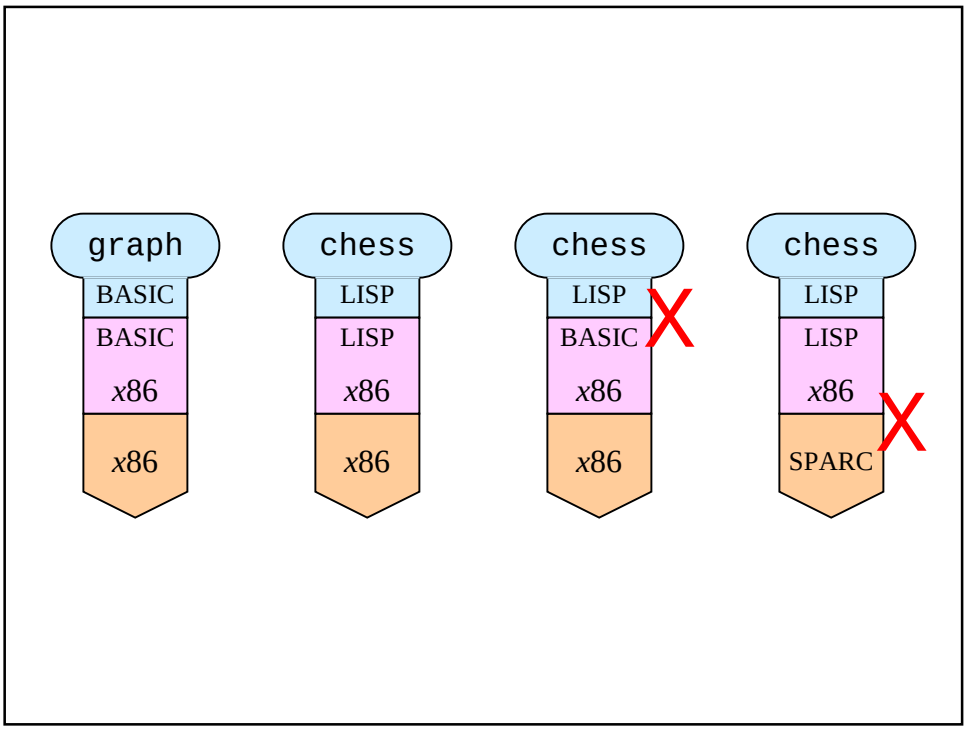


Tombstone representing an *S* interpreter expressed in language *L*.





Interpreting a program *P* expressed in language *S*, using an *S* interpreter on machine *M*.



Hardware emulation

We want:

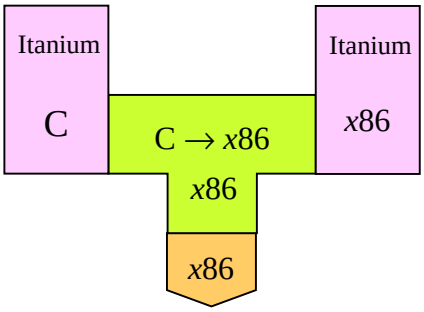
Itanium

We have:

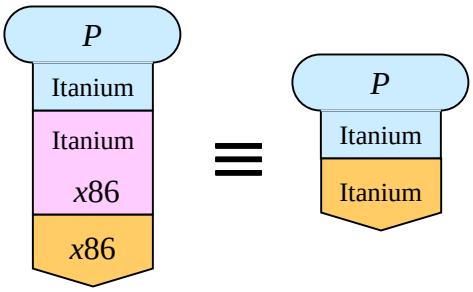
Itanium

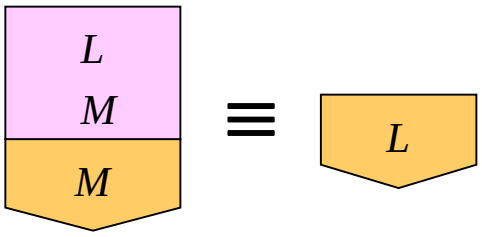
C

Hardware emultation (...)



Hardware emulation (...)

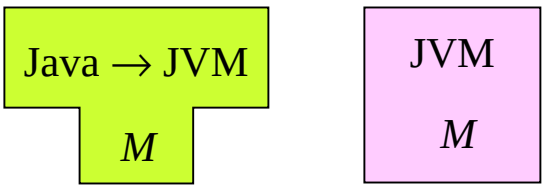




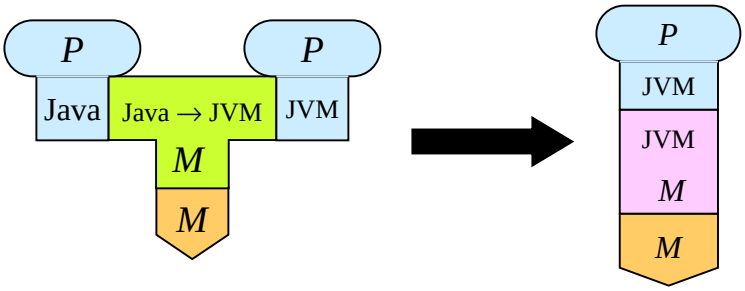
An abstract machine is functionally equivalent to a real machine.

Interpretative compilers

Java SDK components:

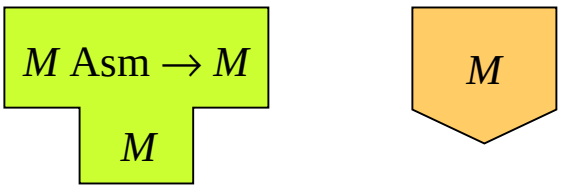


Interpretative compilers (...)



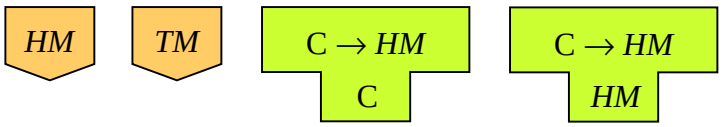
Exercise: Full bootstrap

How do we write a C language compiler for machine *M* if we only have the following components?



Exercise: Half bootstrap

How do we port a C compiler from machine *HM* to machine *TM* if we only have the following components?



Reference:

[WATT] pp. 28-48