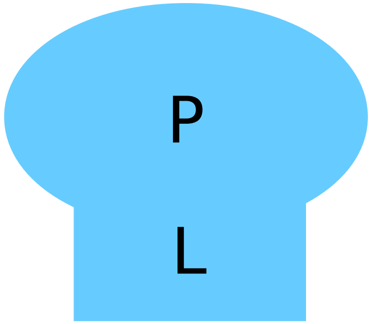
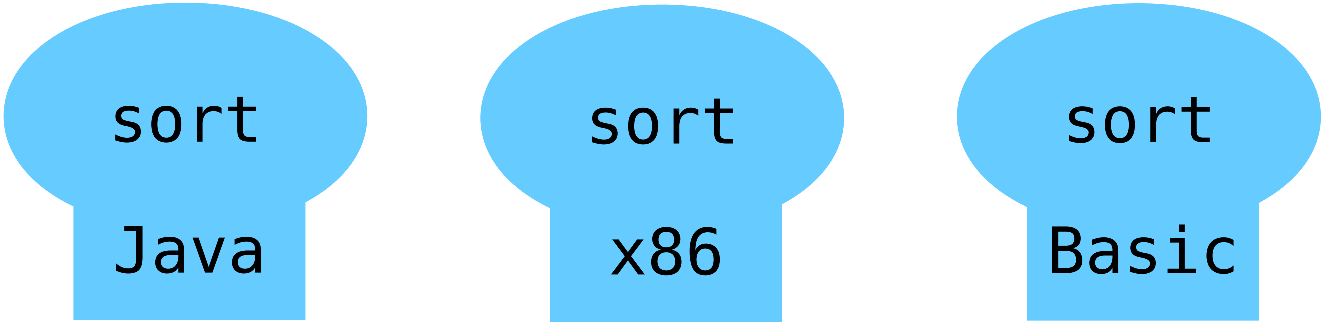
Tombstone Diagrams

Reference:

[WATT] pp. 28-48



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



sort

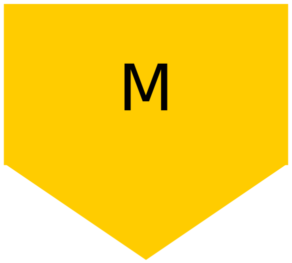
Java

sort

x86

sort

Basic

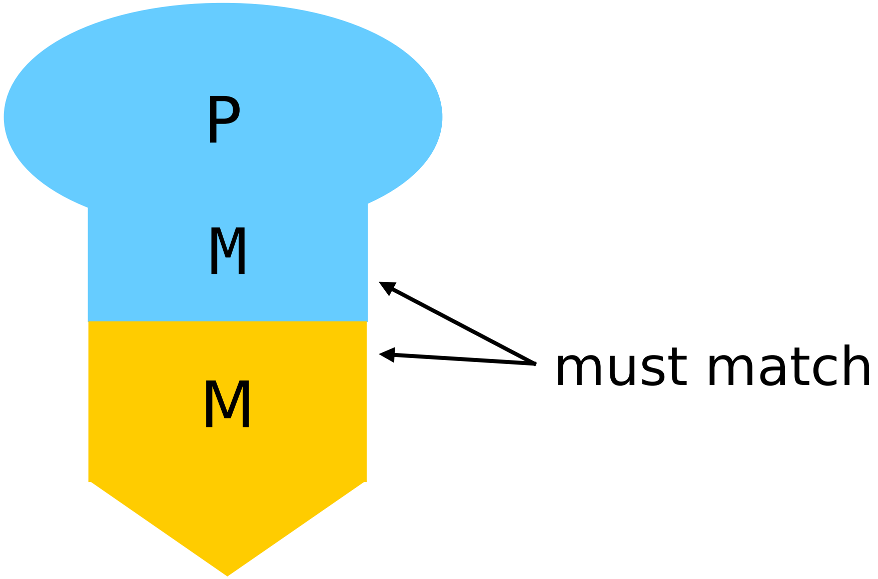


Tombstone representing a machine *M*.

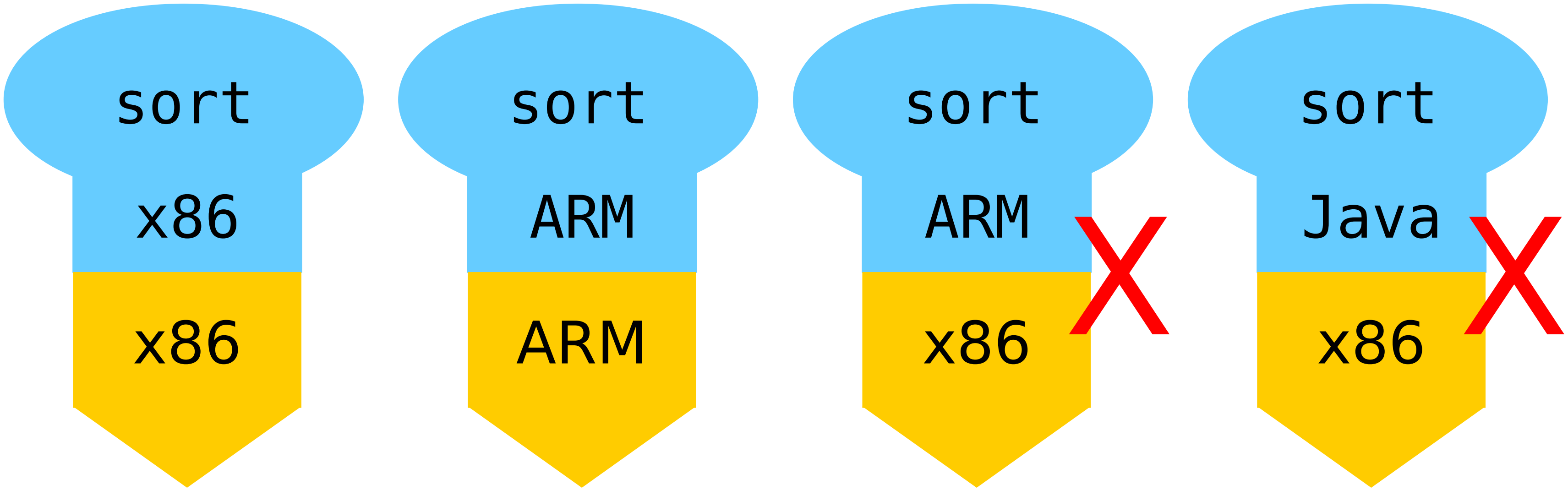
x86

ARM

Alpha



Running program *P* on machine *M*.

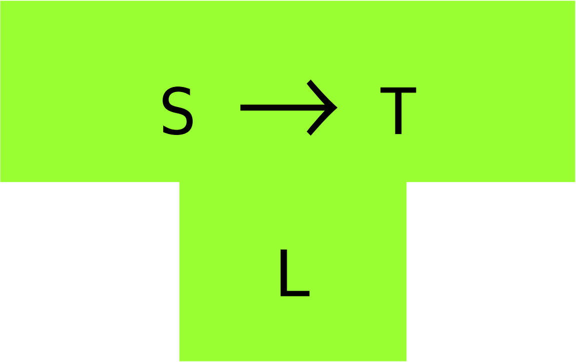


sort
x86
x86

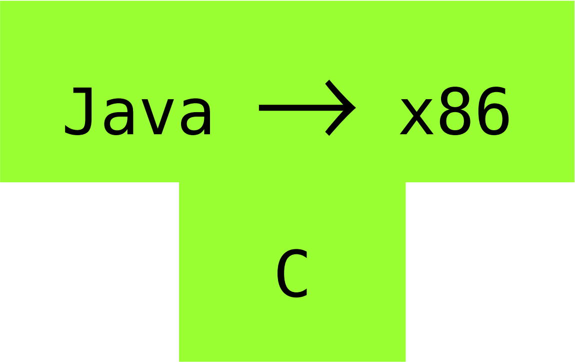
sort
ARM
ARM

sort
ARM
x86
X

sort
Java
x86
X

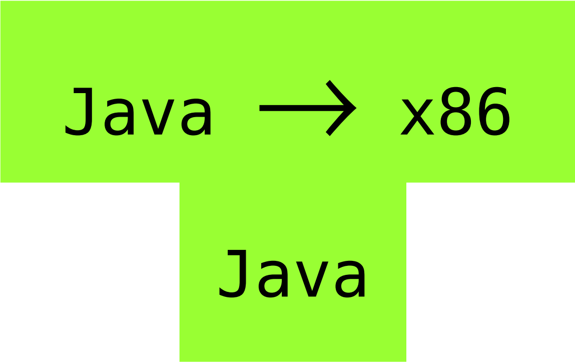


Tombstone representing an *S*-into-*T* translator expressed in language *L*.



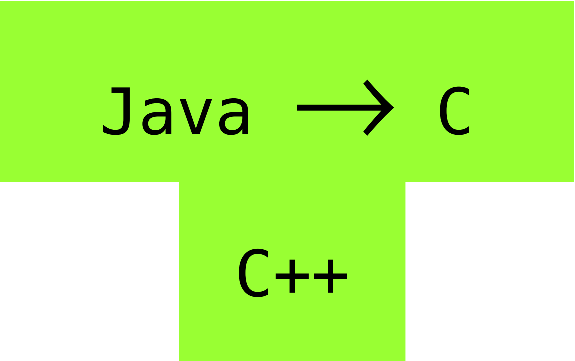
Java → x86

C



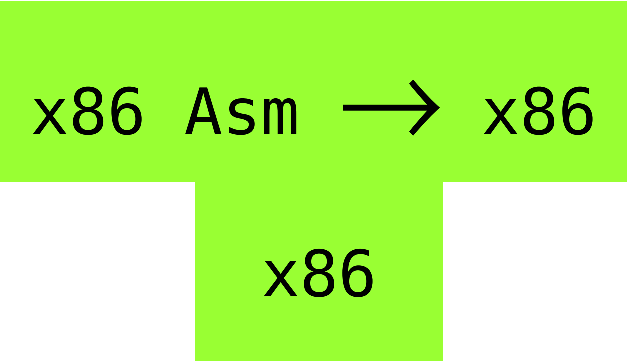
Java → x86

Java



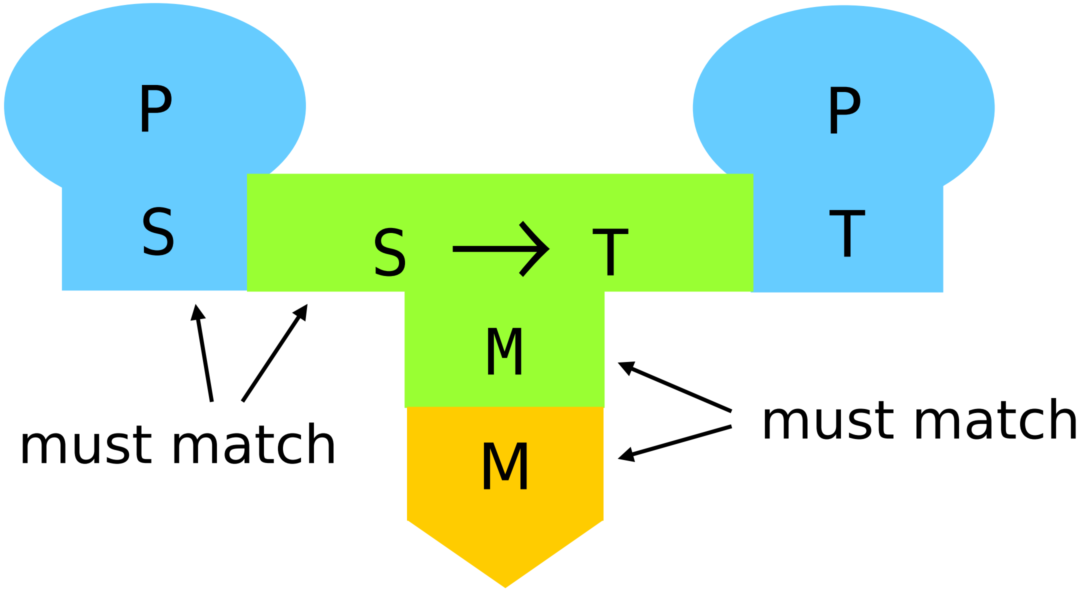
Java → C

C++



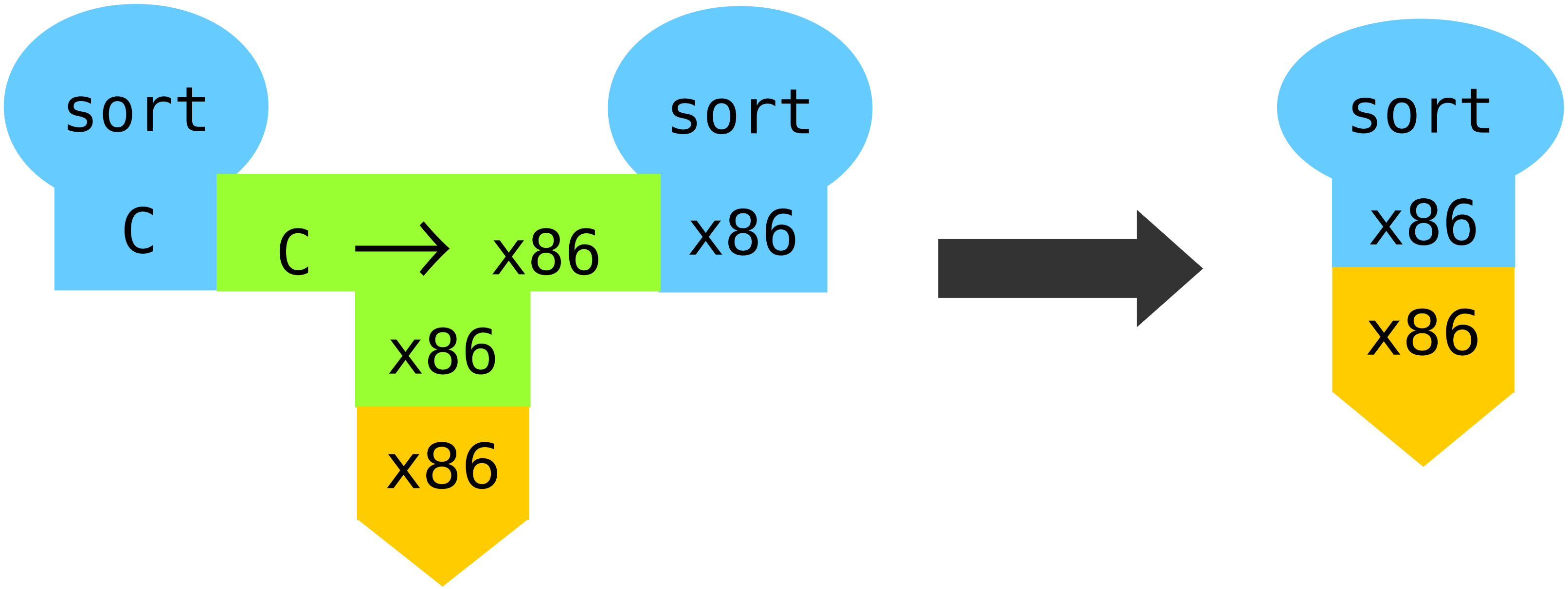
x86 Asm → x86

x86

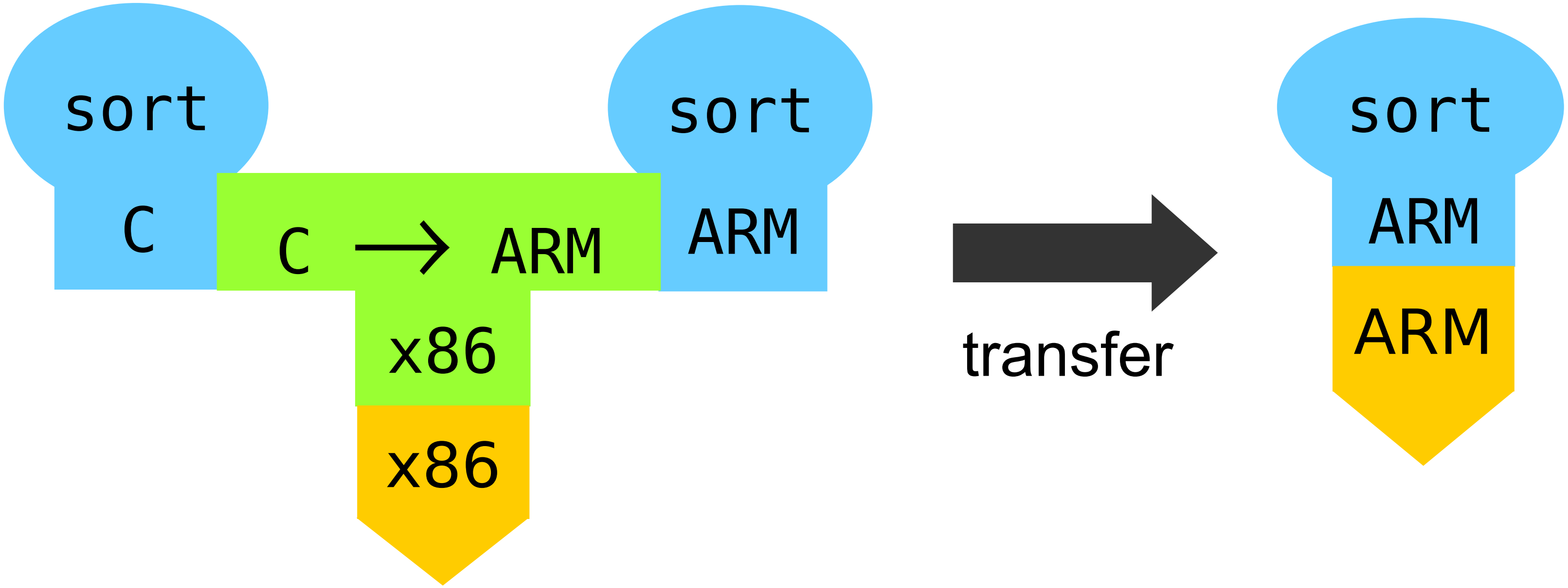


Translating a source program *P* expressed in language *S* to an object program expressed in language *T*, using an *S*-into-*T* translator running on machine *M*.

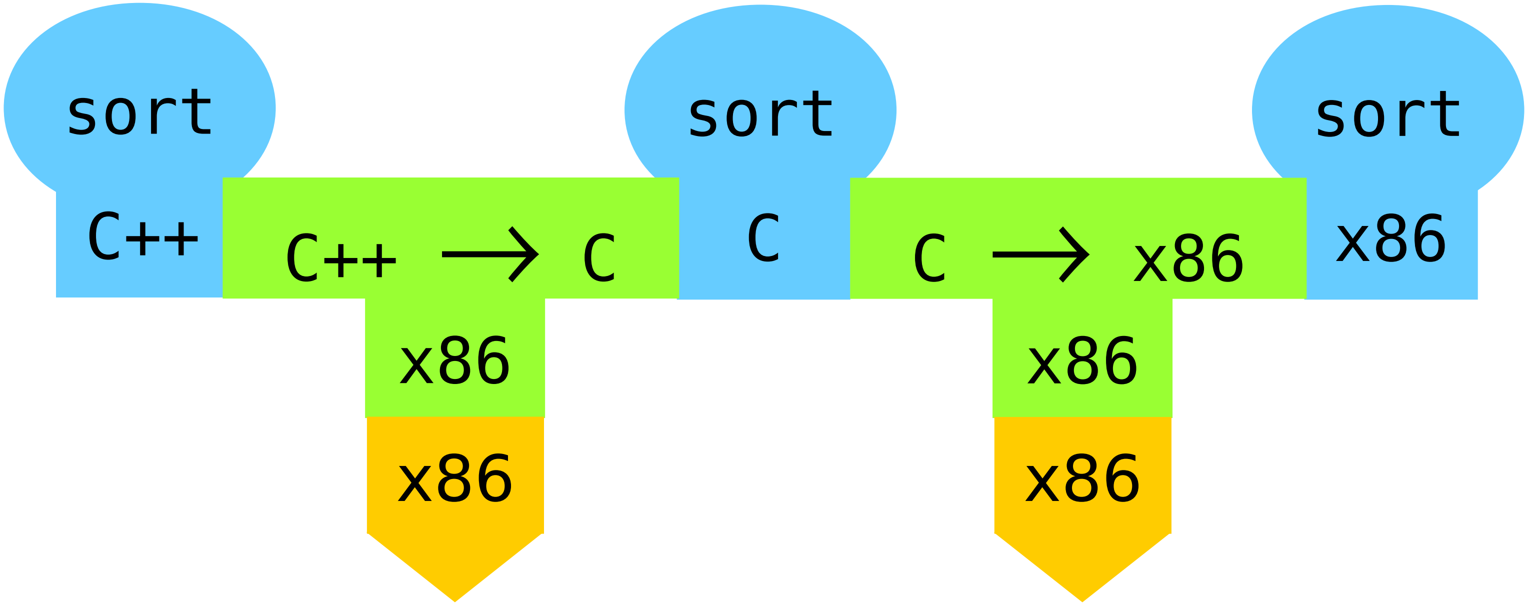
Compilation



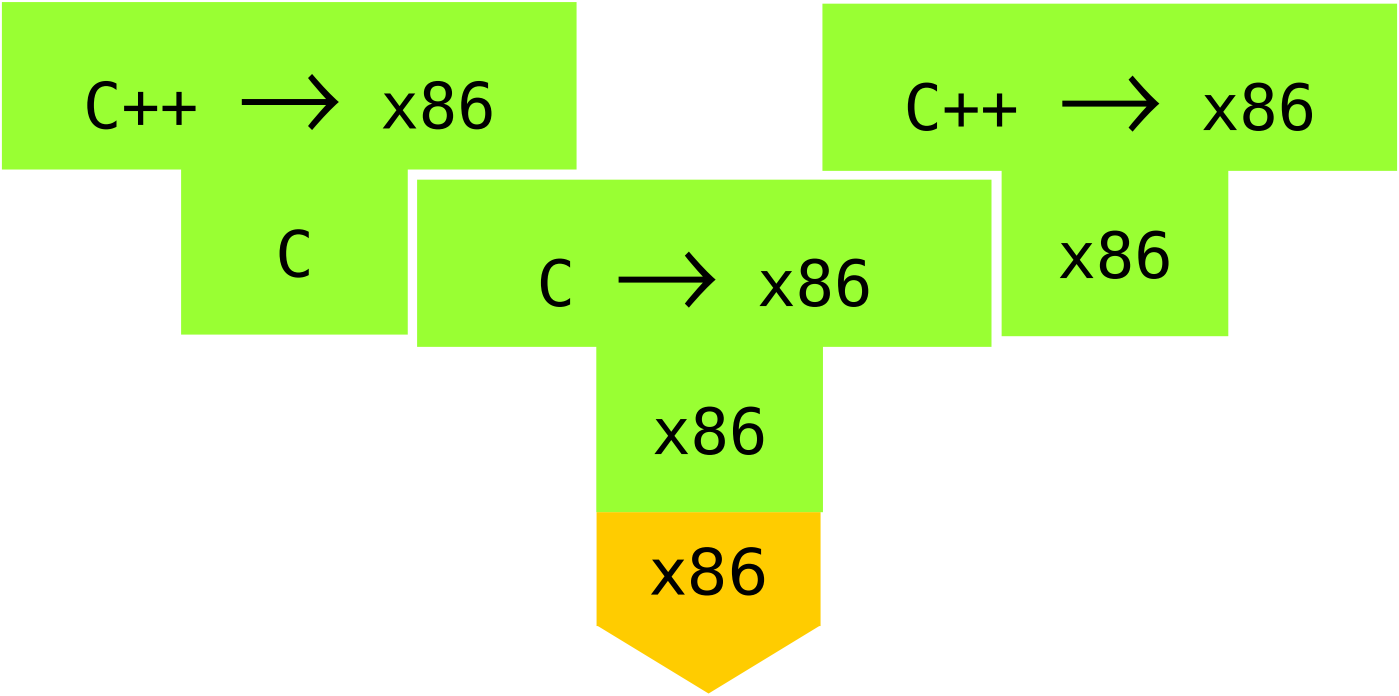
Cross-compilation

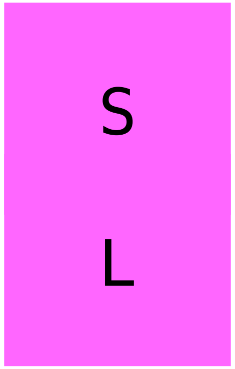


Two-stage compilation

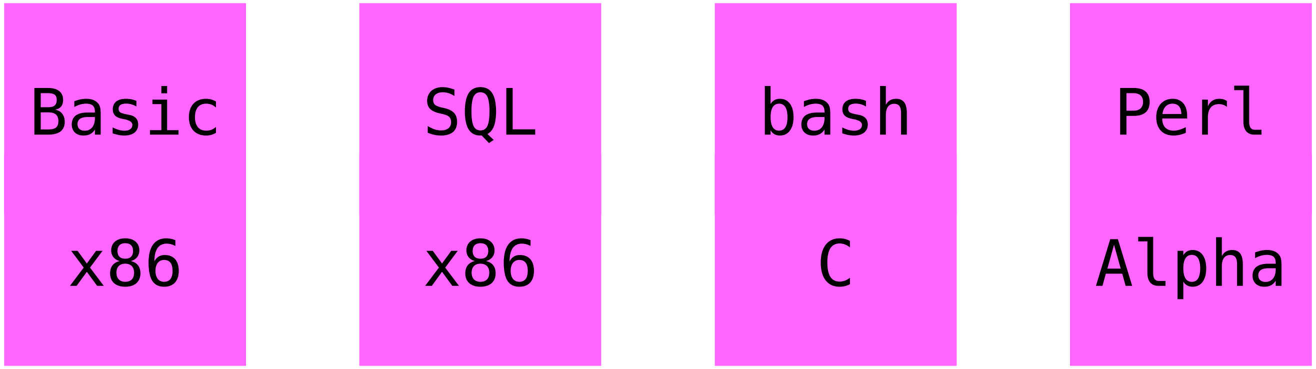


Compiling a compiler





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



Basic

x86

SQL

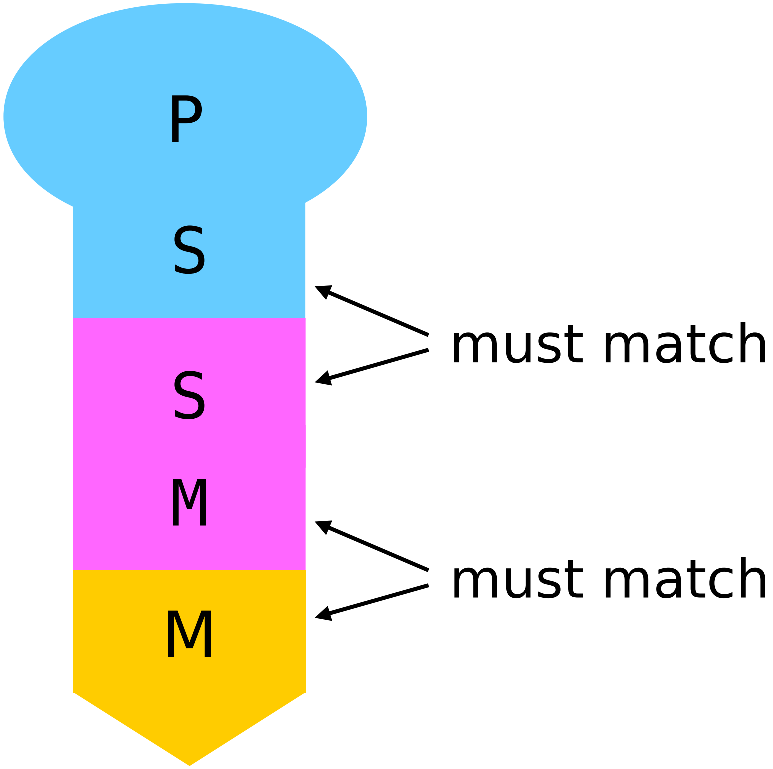
x86

bash

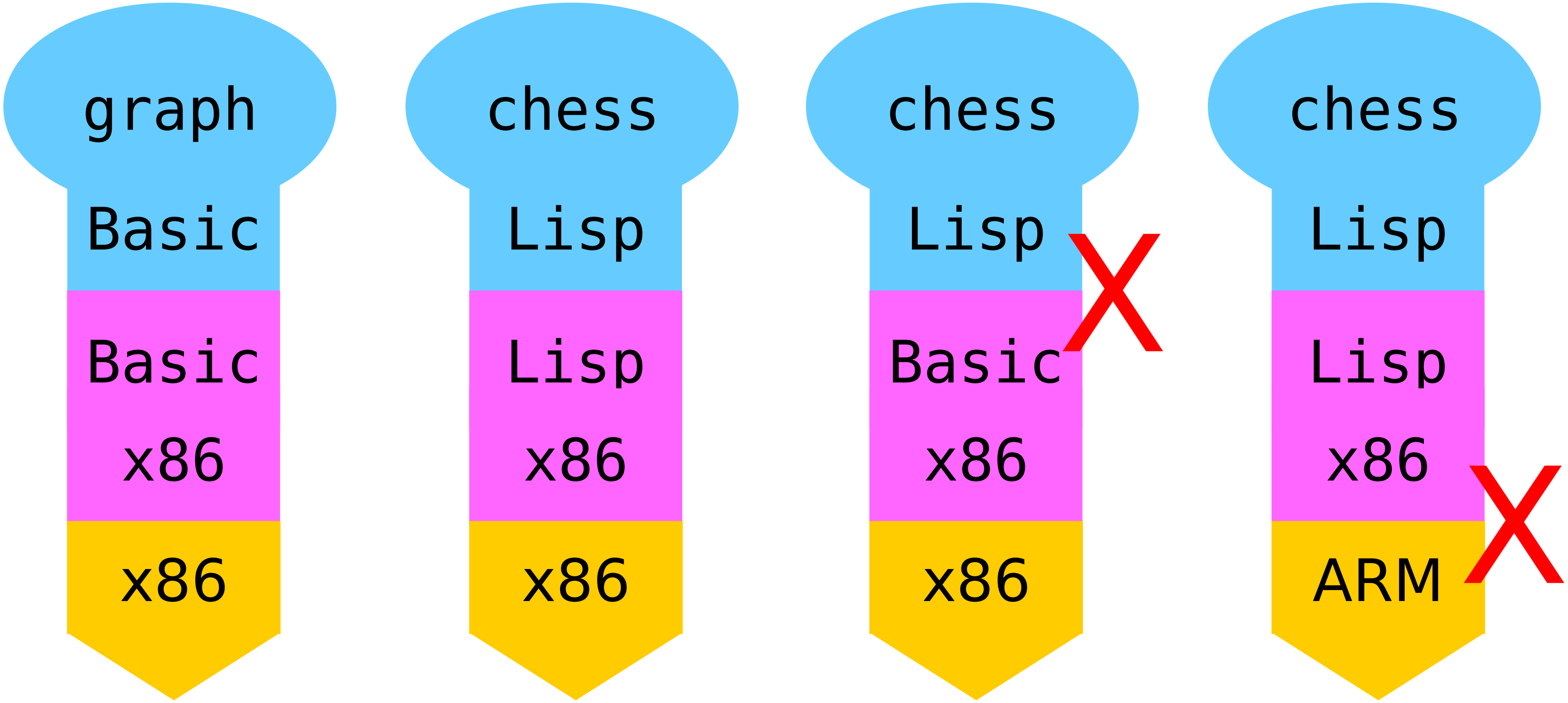
C

Perl

Alpha

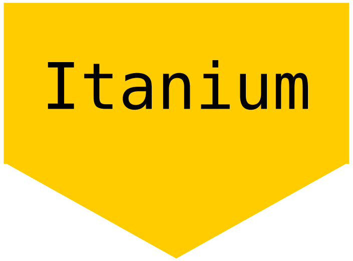


Interpreting a program *P* expressed in language *S*, using an *S* interpreter on machine *M*.

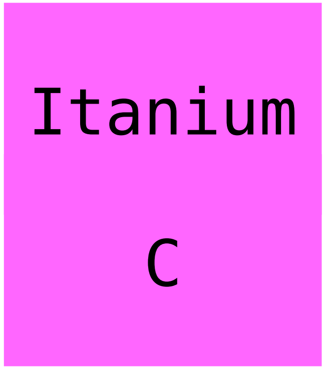

graph
Basic
Basic
x86
x86
chess
Lisp
Lisp
x86
x86
chess
Lisp
Basic
x86
x86
X
chess
Lisp
Lisp
x86
ARM
X

Hardware emulation

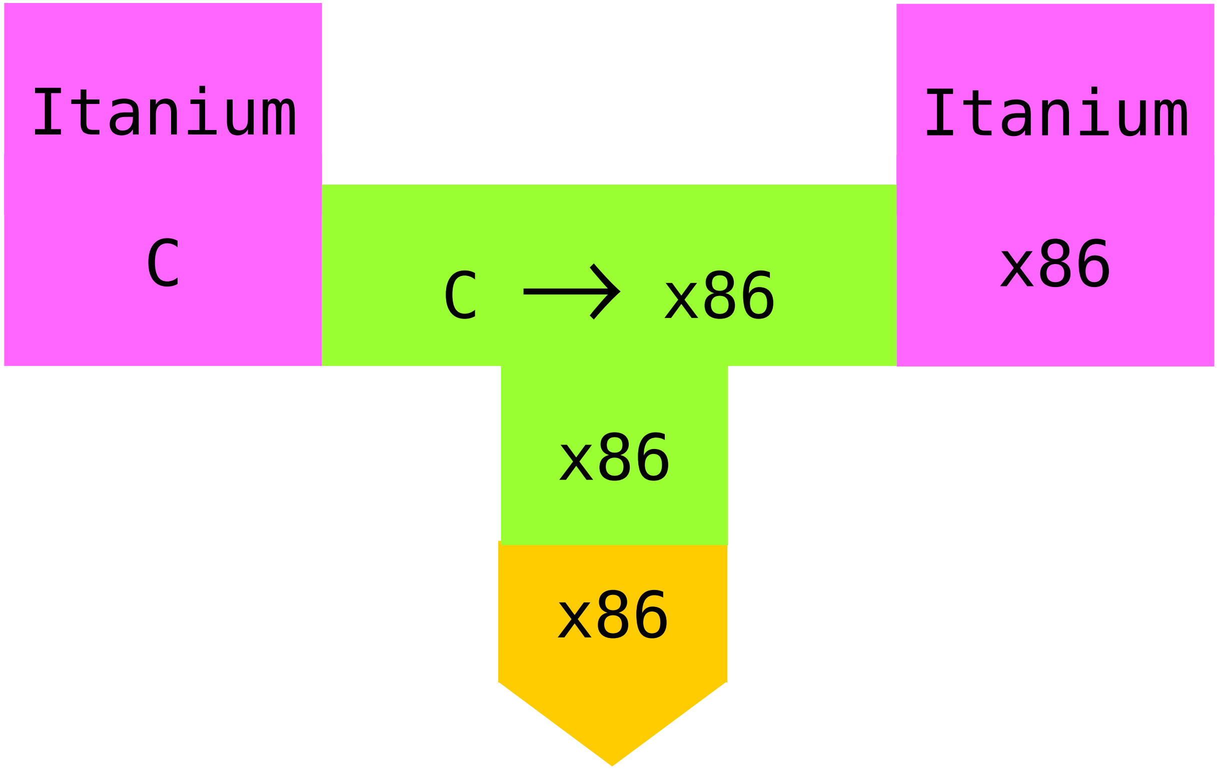
We want:



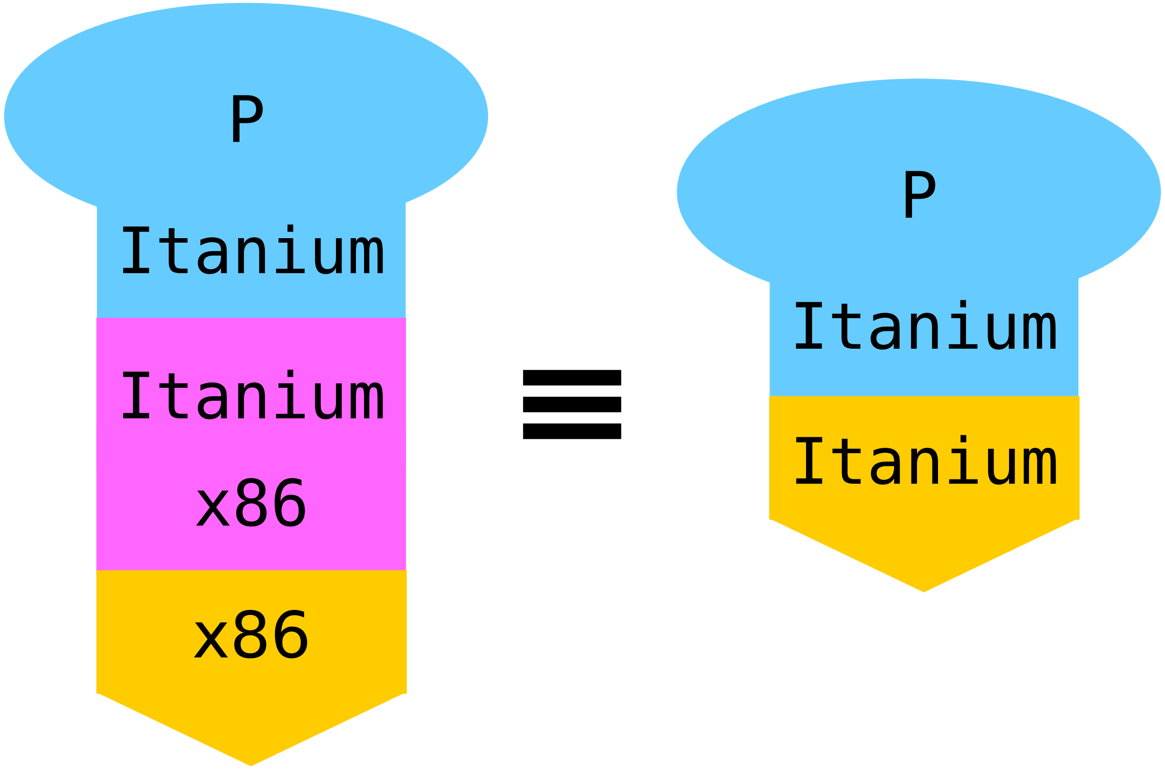
We have:

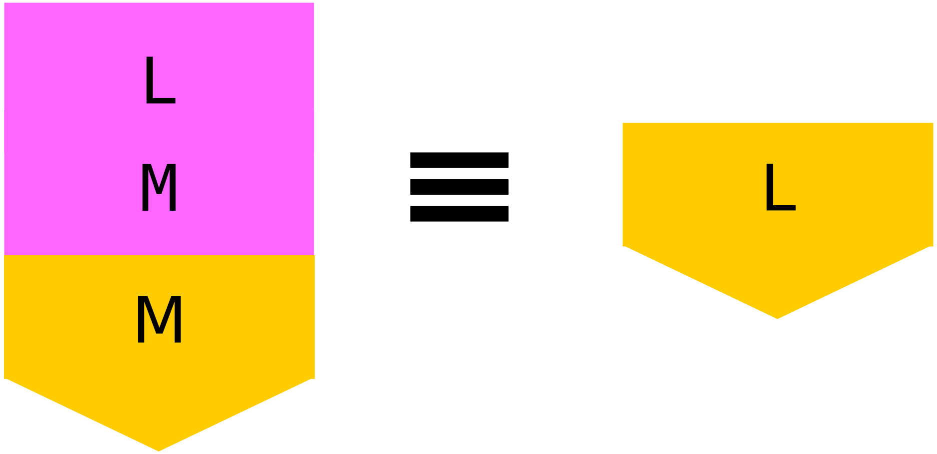


Hardware emultation (...)



Hardware emulation (...)

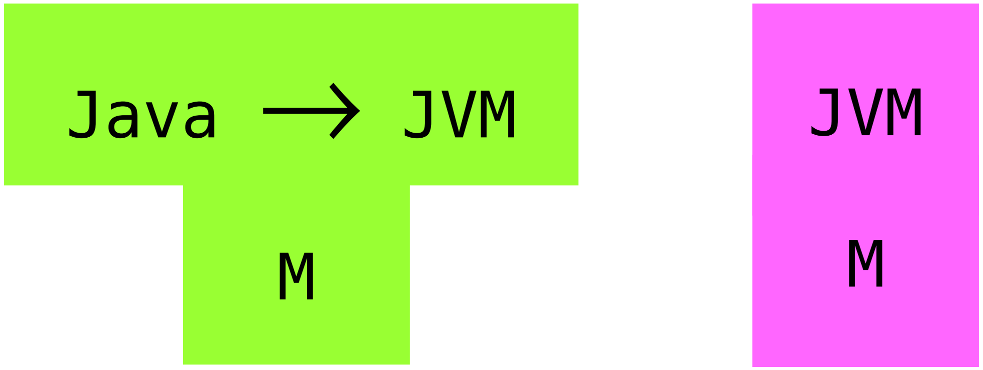




An abstract machine is functionally equivalent to a real machine.

Interpretative compilers

Java SDK components:



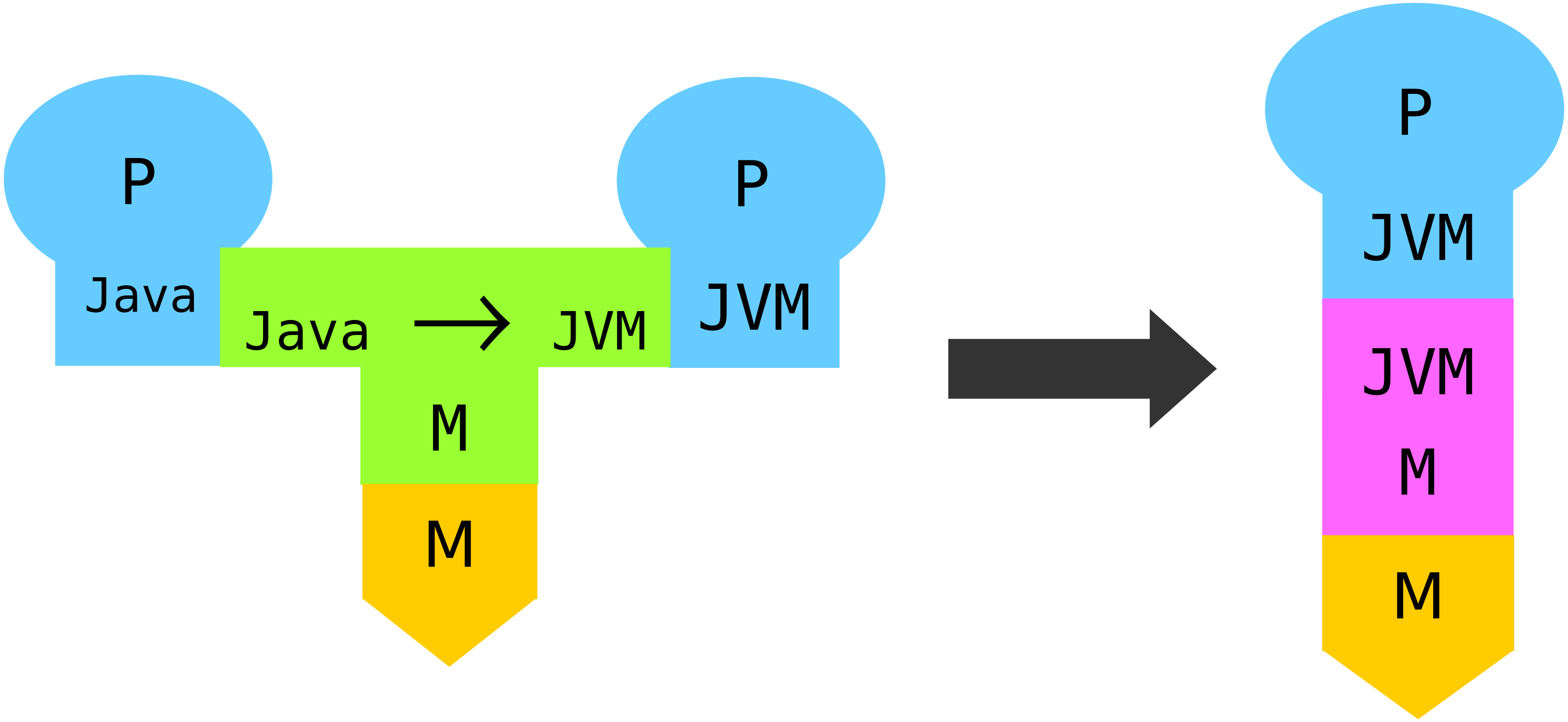
Java → JVM

M

JVM

M

Interpretative compilers (...)



Exercise: Full bootstrap

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