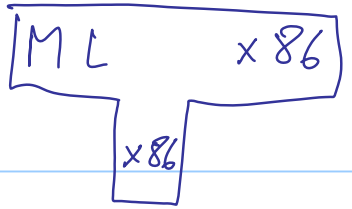
531 2013-01-24

Note Title 2013-01-24

HW2: Exercises 2.2, 2.3, 2.4, 2.5, 2.6, 2.8, and 2.9 from Watt + Brown, due on January 31.

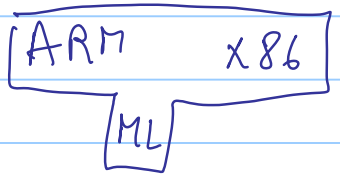
Bootstrapping (from ch. 13 of Mogensen's book)

Suppose we want

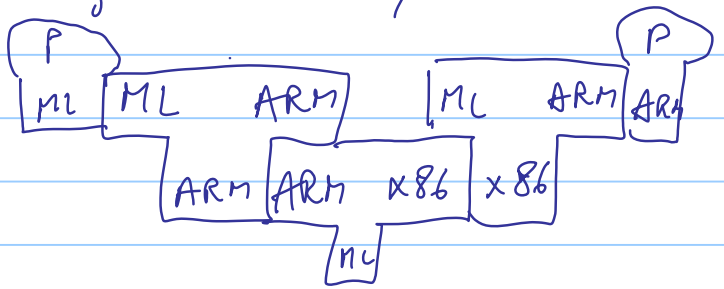


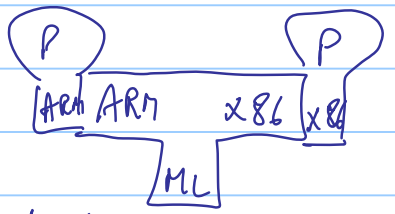
Suppose we have

Option 1: do binary translation, i.e. build



We can then translate the compiler we have using the "binary translator"

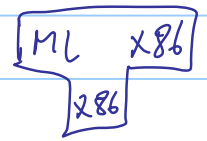


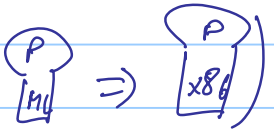


But: requires an extra pass;

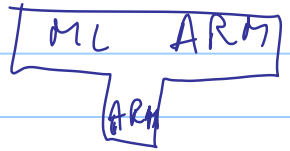
and the ARM->x86 needs to run on the x86 machine

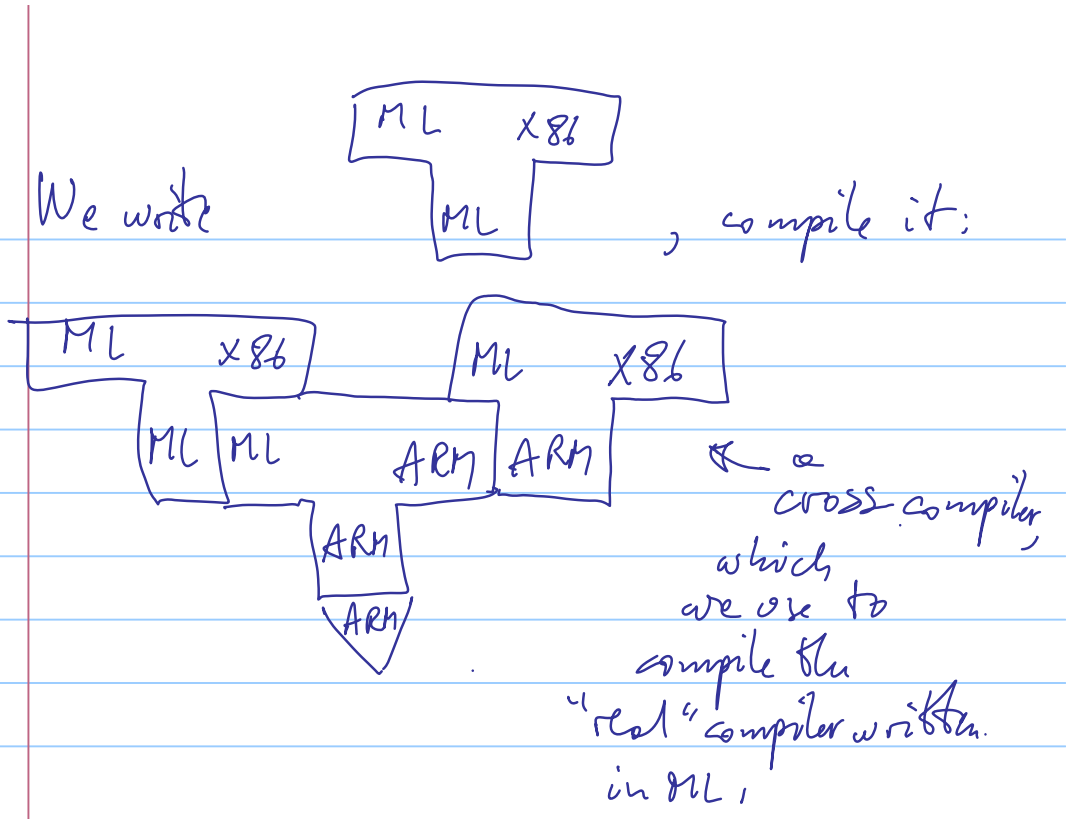
So, instead of binary translation, it usually better to solve this problem by half-bootstrapping

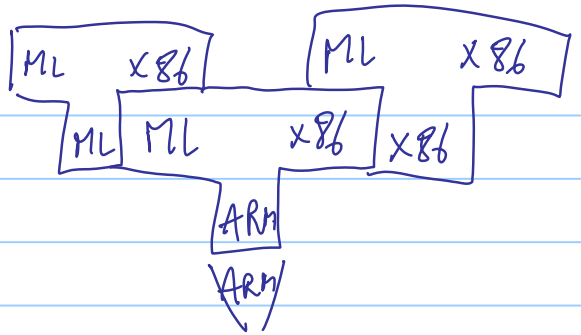
We want 

(

)

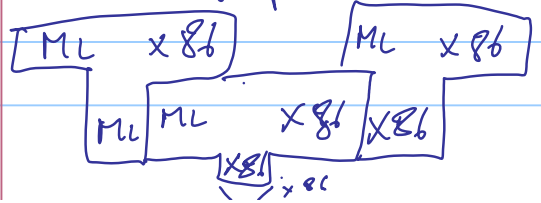
We have: 


We write
ML
x86
ML
, compile it:
ML
x86
ML
ML
ML
x86
ML
ML
ARM
ARM
ARM
ARM
a cross-compiler, which we use to compile the "real" compiler written in ML.



done!

You can then compile again, using the native compiler for testing purposes



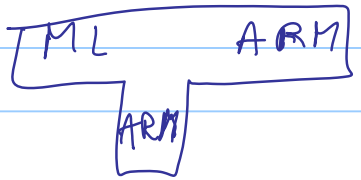
if we do not get the same object code, we have an error.

Now: full-bootstrap. In half-bootstrap,

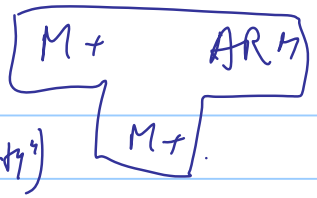
we had compiler for the desired language but running on a different machine.

We want a compiler for the new language M+:

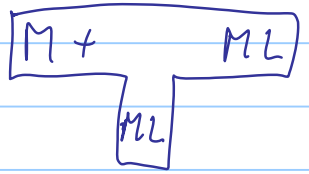
Have



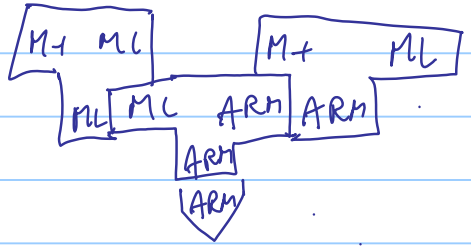
We write the "real" compiler



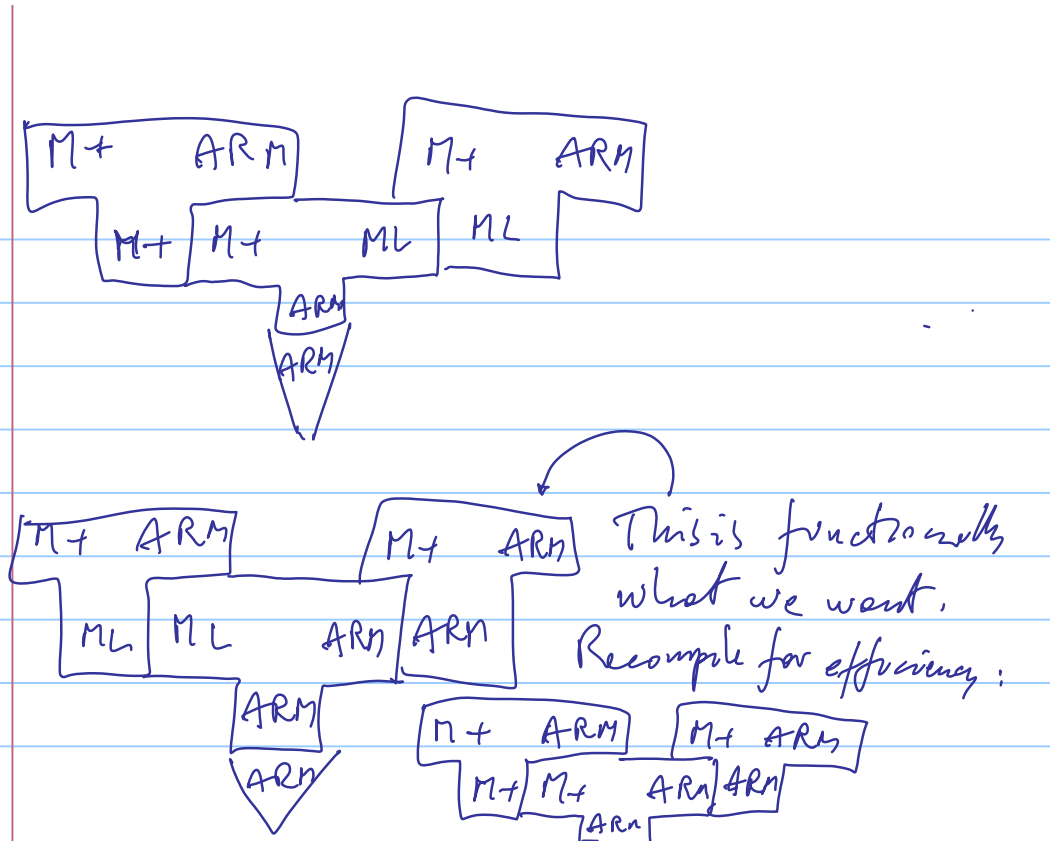
We write a QAD ("Quick and Dirty") translator



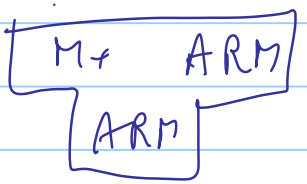
. We compile the QAD compiler:

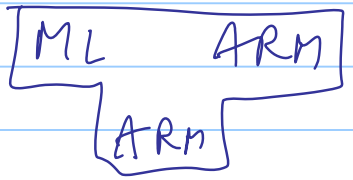


; we use this to compile the "real" compiler


M+ ARM
M+ ARM
M+ M+ ML ML
ARM
ARM
M+ ARM
M+ ARM
ML ML ARM ARM
ARM
ARM
This is functionally what we want. Recompile for efficiency:
M+ ARM
M+ ARM
M+ M+ ARM ARM
ARM

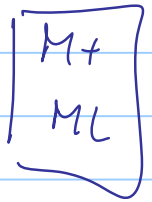
Alternatively, one could do full bootstrapping using an interpreter

Want: 

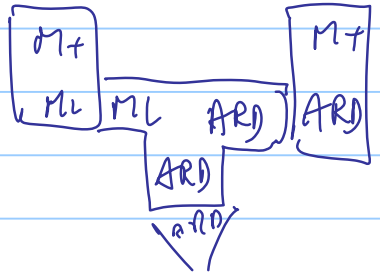
Have: 

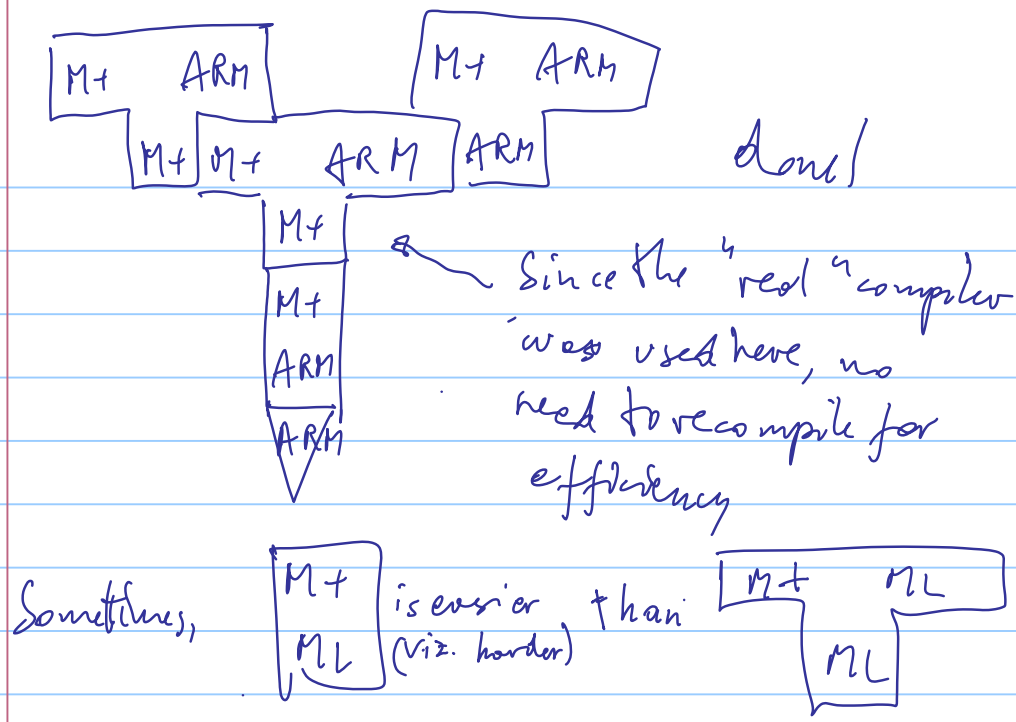
Write the "real" compiler

Write a QAD interpreter



Compile it:




M+ ARM
M+ ARM
M+ M+ ARM ARM
M+
M+
ARM
ARM
done!
Since the "real" compiler was used here, no need to recompile for efficiency
Sometimes,
M+
ML
is easier than
(viz. harder)
M+ ML
ML