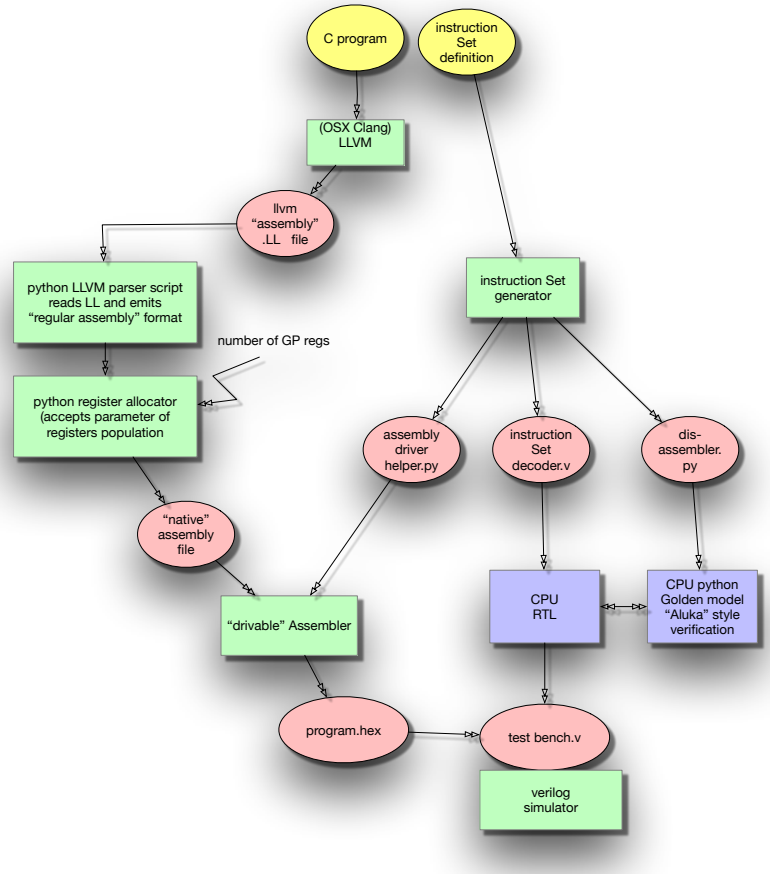
RISX (hacking C compiler)

OBJECTIVE: “C” COMPILER GENERATING ASSEMBLY OF MODEST CPU WITH FLUID (READ: NOT TOTALLY FINALIZED) INSTRUCTION SET.

My ASIC projects often include controller. Flowchart FSM comes to mind. Alas, usually there are too many unknowns and complications during (and after) the development. Going with simple CPU makes sense. I did several of those, and always programmed them with assembler. RISC5 and commercials are too bulky for this task, and older generation (8085 and 6809) are too weak. I want **squeezable design with easy options to adapt driven by “C” compiler.**

Here is a flow i devised. Your are welcome to laugh at it. Or not. It suits C code up to complexity somewhat beyond control.

First step - use LLVM (**Apple CLANG**) to turn control C code to **IR** (intermediate representation). IR is complex. Next is to dive into backend of LLVM. Official path is way over my head. Instead, i am using **python** to parse IR and generate “simple” format. Followed by a script to assign variables to registers of my target CPU and ending in accommodating it to target assembly language.



THE CPU.
The CPU can have as few registers as to fit the area, and registers can be narrower than usual.

Control software in C, can live with single data type (integer) and modest resources. Single data type also makes addressing load/store simple.

Instruction set of my CPU can be divided into two:
1. Instructions that serve directly LLVM produced code.
2. Specials that cater to control needs. In C represented by calls to predefined functions.

My instruction set generator supports the flow and can fit the opcode to 16bits.

Python flow allows to add optimizations of software and cpu itself.

This flow works today with some limitations. RTL CPU can be optimized and upgraded, but has all the bells and whistles now. Verification with Python keeps the RTL honest.

Typical integration scenario for this kind of controller ———>
If You feel this approach has merit and is beyond ridiculous, talk to me. It will probably end up in open source, when it is more mature.