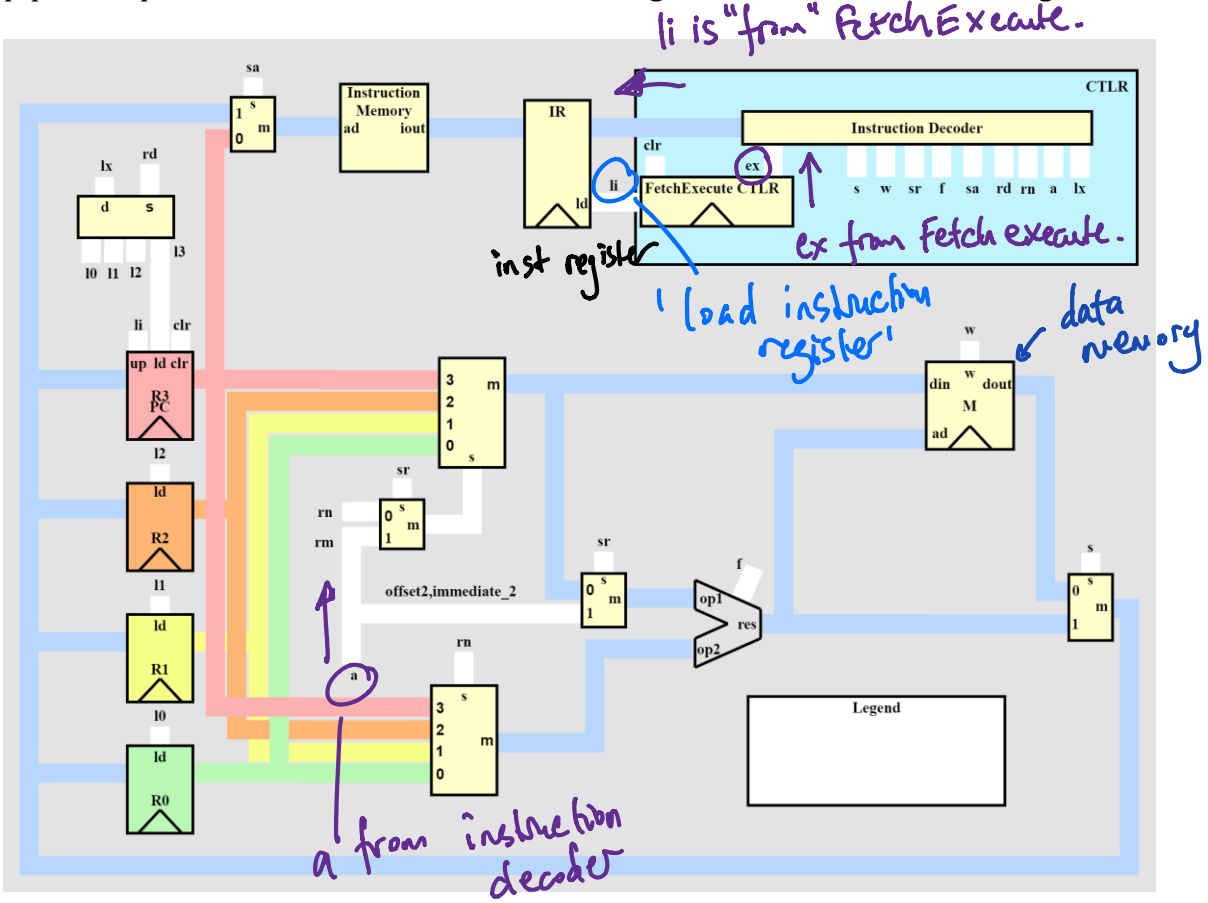
The ARM0 processor – Part 2

by Craig Scratchley.

Make a pipelined processor – 2-stage Harvard architecture

When we added a separate instruction memory to our ARM0 processor under development, we arrived at a Harvard architecture for the processor. That instruction memory might need to be relatively large, and it might be put off-chip from the processor. Such has certainly been done in different periods of technology. An off-chip memory could be relatively slow to access by the processor.

If the instruction memory is slow, it might be interesting to load one instruction at the same time as another instruction is being decoded and then executed. A pipelined processor can do that. Here is a 2-stage Harvard architecture design.



My apologies, this is currently a draft version of the figure with multiplexors and demultiplexors shown with rectangles and not with trapezoids as should be the case. I will try to update the document with an updated figure.

Start with FetchExecuteCTLR executes when ex == '1'

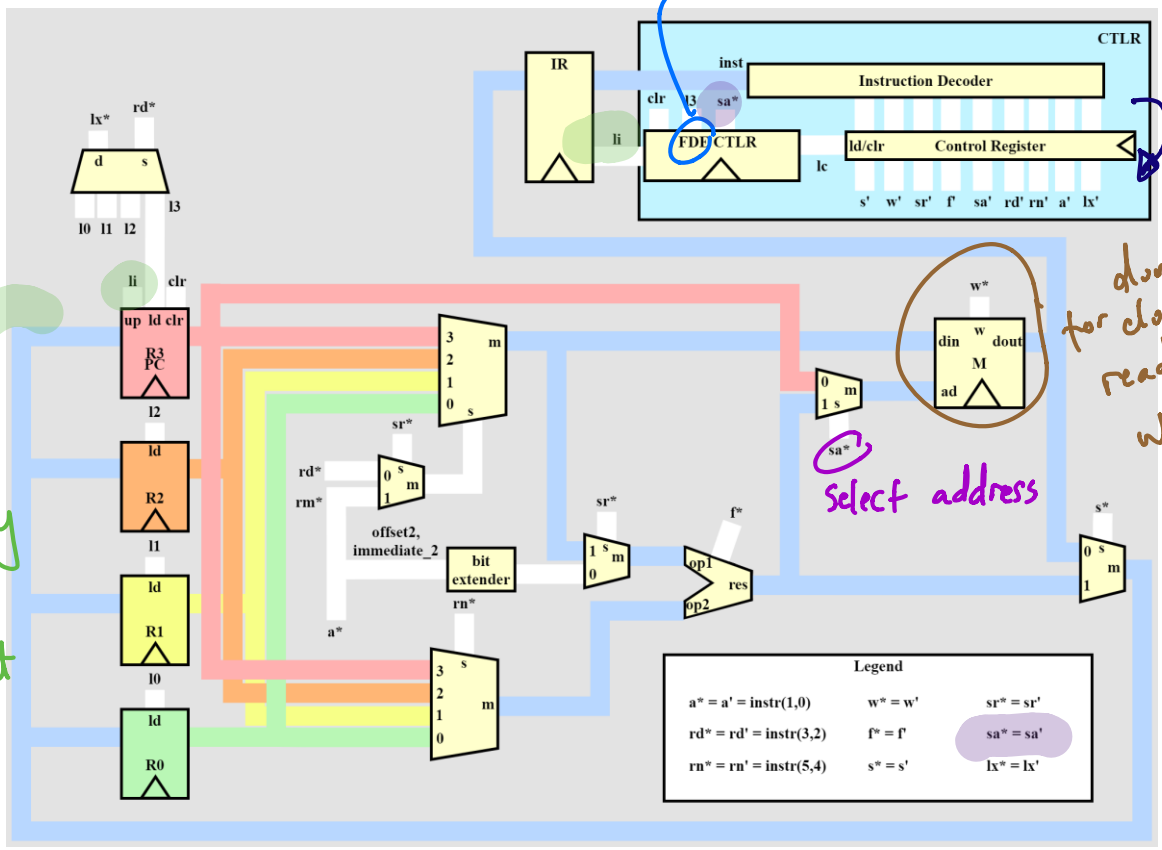
The most important addition here is IR, the instruction register at the top centre of the figure. It is a place to temporarily hold an instruction within the processor.

In the CTLR component we now also need a sequential controller, called the FetchExecute controller, which will be especially useful when the processor is powered on or cleared. For this case, the main thing that this new controller does is stop any execution from making changes when an "undesired" instruction appears in the IR, such as upon powering on of the processor or immediately after a reset. This controller has just 2 simple states. Reset state, and running state. Signal *ex* will be set to 1 only when in the running state. Importantly, the *Instruction Decoder* will only allow writing to memory component *M* when *ex* is set to 1.

Von Neumann architecture and a 3-state pipeline

Can't read from or write to M and load next instruction at the same time

At the moment, I am now going to jump to our final architecture. We are switching from Harvard to von Neumann architecture. That means that we will no longer have a separate instruction memory, but instead will have both instructions and data in a common main memory. Also, we are adding a 3rd stage to the pipeline. This follows the design of early versions of the real ARM processor, when a 3-state pipeline got cemented into the ARM Instruction Set Architecture (ISA), as we will find out in the labs today.



When loading instructor, we increment PC. "li"

one address holds instruction & data.

from PC

from ALU

Harvard

Von Neumann.

- flexibility.
  put data in whatever address space
- we can't read data and write to memory instruction at the same time.

Note that the Instruction Memory is gone, and we now have a new multiplexor, the *sa* multiplexor, to decide between addresses coming from the PC (for fetching instructions) or the ALU (for Load or Store operations).

The "FetchExecute CTLR" from the 2-state architecture has now been updated to handle decoding separately from fetching and executing. A block diagram and a finite state machine for the controller are shown in the below figure.

The lc signal controls loading or clearing of a "Control Register", which holds decoded control signals. Note that the presense of a Control Register causes a 1-cycle delay between the output of the Instruction Decoder and the controlling of various other components. So w', for example, is delayed by one cycle compared with the w signal coming out of the Instruction Decoder.