Having established the basic bit-sliced design, we add functionality to it to implement more instructions.

32-Bit ALU
Compose 32 bit-slices

Addition

AND

OR

Negate

Overflow Conditions

Separate add and subtract cases

\[
\begin{array}{c|c|c|c}
\text{Operation} & \text{Op A} & \text{Op B} & \text{Overflow} \\
\hline
\text{Add} & A+B & \geq 0 & <0 \\
\text{Sub} & A-B & <0 & \geq 0 \\
\end{array}
\]
Less-than-test

There are three cases:
- Bit 0
- Bit i
- Bit 31

Less Than, MSB

- Capture adder output for set bit

AND-OR-Add-Sub-SLT ALU

The Set line feedback to become the input to Less for bit 0. All other bits have Less set to zero.

Abstracting ...

Using the ALU

The datapath uses the ALU structure several times, though not always in its full generality.

Zero-detecting ALU

If all output bits are zero, the OR is false, so its negation is true.

Testing for alternative representations?

Using the ALU