The LD and ST instructions access main memory.

Note that its the same main memory that holds the instructions, even though for drafting convenience we show it has two separate boxes in our datapath diagram.

In the form we show it here, main memory has three ports: two read ports for fetching instructions and reading load data, and one write port used by the ST instruction to write data into main memory.

The address calculation is exactly the same computation as performed by the ADDC instruction: the contents of the RA register are added to the sign-extended 16-bit literal from the low-order 16 bits of the instruction.

So we'll simply reuse the existing datapath hardware to compute the address.

For the LD instruction the output of the ALU is routed to main memory as the address of the location we wish to access.

After the memory's propagation delay, the contents of the addressed location is returned by the memory and we need to route that back to the register file to be written into the RC register.

The memory has two control signals: MOE (memory output enable), which we set to 1 when we want to read a value from the memory.

And MWE (memory write enable) which is set to 1 when we want main memory to store the value on its write data (WD) port into the addressed memory location.

We need to add another MUX to select which value to write back to the register file: the output of the ALU or the data returning from main memory.

We've used a 3-to-1 MUX and we'll see the use for the other MUX input when we get to the implementation of branches and jumps.

The two-bit WDSEL signal is used to select the source of the write-back value.

Let's follow the flow of data when executing the LD instruction.

The ALU operands are chosen just as they are for the ADDC instruction and the ALU is requested to perform an ADD operation.

The ALU result is connected to the address port of main memory, who's control signals are set for a read operation.

The WDSEL control signals are set to 2 to route the returning data to the register file.

Execution of the ST instruction is very similar to the execution of the LD instruction, with one extra complication.

The value to be written to memory comes from the RC register, but the RC instruction field is not connected a register file read address.

Happily, the RB register address isn't being used by the ST instruction since the second ALU operand comes from the literal field.

So we'll use a MUX to enable the RC field to be selected as the address for the register file's second read port.

When the RA2SEL control signal is 0, the RB field is selected as the address.

When RA2SEL is 1, the RC field is selected.

The output from the second read data port is connected to the write data port of main memory.

The ST instruction is the only instruction that does not write a result into the register file.

So the WERF control signal will be 0 when executing ST.

Here's the flow of data when executing ST.

The operands are selected as for LD and the ALU performs the address computation with the result sent to main memory as the address.

Meanwhile the RC field is selected as the address for the second register file read port and the value from the RC register becomes the write data for main memory.

By setting the MWR control signal to 1, the main memory will write the WD data into the selected memory location at the end of the cycle.

The WERF control signal is set to zero since we won't be writing a value into the register file.

And, since we're not writing to the register file, we don't care about the value for the WDSEL signal.

Of course, the logic will need to supply some value for WDSEL.

The "don't care" annotation is telling the logic designer that she's welcome to supply whatever value is most convenient.

This is particularly useful when using Karnaugh maps to optimize the control logic, where the value can be chosen as either 0 or 1, whichever results in the best minimization of the logic equations.