# Snow64 Operand Manager Details

- General Notes
  - This module determines the operands that are used in the EX stage.
  - Note that this module is the one that actually performs operand forwarding.
  - Also, it can produce a stall in some cases.
- Ports of `Snow64OperandManager`
  - This module is clocked.
  - Inputs:
    - An enable signal. This module will not do anything unless this enable signal is high.
      - This enable signal should only be high during the first clock cycle that an instruction is in the EX stage.
      - Perhaps an easy way to do this is to just check if the program counter value in the EX stage has changed? This might not work for a single-instruction infinite loop....
    - Whether or not to perform type casting of register B and whether or not to perform type casting of register C.
      - These are two separate signals.
      - Not all instructions actually need the type casting of both registers. Some instructions only need register B to be casted, and some instructions only need register C to be casted. Still others potentially need both B and C to be casted, and others do not need either register B or C to be casted (mostly those that depend only on register A, such as the control flow instructions).
    - If casting registers B and C, an ove
    - The LAR file's read ports (including all metadata, such as the data type and scalar data offset)
    - The EX stage's past output (and whether or not it's valid)
  - Outputs:
    - The vector values of data LARs a, b, and c.
    - True scalar values of data LARs a, b, and c.
    - Whether or not the EX stage should stall on the current cycle.
- Functionality
  - Operand forwarding.
    - "Pure" scalar result to scalar instruction operand forwarding:
      - For "pure" scalar result to scalar instruction operand forwarding, the `base_addr`, `data_offset`, and the data type of the scalar are used to determine how to perform operand forwarding.
      - "Pure" scalar result to scalar instruction operand forwarding only happens if these conditions are met:
        - Same `base_addr`
        - Same `data_offset`

        - One of the following must be the case:
            - Both the data to forwarded and the register the forwarding will happen for are integers, and they have the same `int_type_size`
            - Both the data to be forwarded and the register the forwarding will happen for are of the BFloat16 type.
    - If these conditions are met, then operand forwarding will look very similiar to the operand forwarding of typical scalar architectures.
- All other forwarding:
    - For every
- What happens regardless of the type of forwarding:
    - Inject previously computed scalar results into captured vectors of data.
    - This is done to speed up any operand forwarding besides the ideal case of "pure" scalar result to scalar instruction operand forwarding.