Bug Report

Jacob Halsey

All of these bugs can be demonstrated using specific test cases defined in `tests.e`

`error_found` Bugs

1

Addition produces the wrong result when the second operand matches these conditions: For both the least or second least significant bytes of the (4 byte integer): It will break if one of (not both) the 5th or 6th LSBs are set.

2

This bug changes the behaviour of shifting left by 0 (or `din2 mod 32 == 0`) where it returns 0 instead of the un-shifted `din1`

3

This bug causes the `ADD` opcode to always return success (response 2) even when the input should cause an overflow (response 3).

4

This bug causes instructions on port 4 to timeout (i.e. never set a response other than 0).

Extra Bugs

5: Subtraction Response Code

The subtraction (`SUB`) operation always returns an overflow error (response 3) even when the input should not have overflown (the output data value is still correct).

6: Shift Right by One

The shift right (`SHR`) operation fails to shift by one bit (or more specifically when `din2 mod 32 == 1`). Instead the un-shifted operand1 is returned. Shifting by greater than 1 bit works fine.

7: Shift Left by Three or More

The shift left (`SHL`) operation fails when shifting by three or more bits. Where `A` is `din2 mod 32`, and A >= 3. Then the results have an offset of 2^(A-3). E.g. shifting left by 10, the result is 128 larger than expected. (Shifting by 0, 1 or 2 bits still works as expected)

8: Invalid operations

An invalid operation responds with success (1) instead of invalid command (2) as defined in the specification.

9: Priority Logic

When all four ports are sent a series of ADD/SUB (or shift) instructions at the same time, ports 3 and 4 appear to get “stuck”, and do not complete until ports 1 and 2 are no longer in use. This is a violation of the first come first serve priority described in the specification,