VM-specific v1.3.0 opcodes simulation

NOTES:

- changed META - it can be used for MSIZE simulation
- setting ergs per pubdata is done by separate opcode now (not part of `near_call`)
- incrementing TX counter is done by separate opcode now (not part of `far_call`)

Our VM has some opcodes that are not expressible in Solidity, but we can simulate them on compiler level by abusing “CALL” instruction. We use 2nd parameter of “CALL” (address) as a marker, and remaining 6 parameters as input parameters (we use “address”-like field since it’s kind of shorter type, if assembly block cares about types in Solidity). Unfortunately “CALL” returns only 1 stack parameter, but it looks sufficient for our purposes.

Please note, that some of the methods don’t modify state, so STATICCALL instead of CALL should be used for them. The type of the needed method is indicated in the rightmost column.

Call types are not validated and do not affect the simulation behavior, unless specified otherwise, like in `raw_far_call` and `system_call` simulations, where the call type is passed through.

For some simulations below we assume that there exist a hidden global pseudo-variable called `ACTIVE_PTR` for manipulations, since one can not easily load pointer value into Solidity’s variable.

Simulated opcode	CALL param 0 (gas)	CALL param 1 (address)	CALL param 2 (value)	CALL param 3 (input offset)	CALL param 4 (input length)	CALL param (output offs
`to_l1(is_first, in0, in1`	if_first (bool)	`0xFFFF`	in0 (u256)	in1 (u256)	`0xFFFF` to prevent optimizing out by Yul	0
`code_source`	0	`0xFFFE`	-	0	`0xFFFF` to prevent optimizing out by Yul	0
`precompile(in0, ergs_to_burn, out0)`	in0 (u256)	`0xFFFD`	-	ergs_to_burn (u32)	`0xFFFF` to prevent optimizing out by Yul	0
`meta`	0	`0xFFFC`	-	0	`0xFFFF` to prevent optimizing out by Yul	0
`mimic_call(to, abi_data, implicit r5 = who to mimic)`	who_to_call	`0xFFFB`	0	abi_data	who_to_mimic	0
`system_mimic_call(to, abi_data,`		`0xFFFA`				

`implicit r3, r4, r5 = who to mimic)`	who_to_call		0	abi_data	who_to_mimic	value_to_pu
`mimic_call_byref(to, ACTIVE_PTR, implicit r5 = who to mimic)`	who_to_call	`0xFFF9`	0	0	who_to_mimic	0
`system_mimic_call_byref(to, ACTIVE_PTR, implicit r3, r4, r5 = who to mimic)`	who_to_call	`0xFFF8`	0	0	who_to_mimic	value_to_pu
`raw_far_call`	who_to_call	`0xFFF7`	0	0	abi_data (CAN be with “to system = true”)	output_offse
`raw_far_call_byref`	who_to_call	`0xFFF6`	0	0	`0xFFFF` to prevent optimizing out by Yul	output_offse
`system_call`	who_to_call	`0xFFF5`	value_to_put_into_r3 (only for `call` with 7 arguments)	value_to_put_into_r4	abi_data (MUST have “to system” set)	value_to_pu
`system_call_byref`	who_to_call	`0xFFF4`	value_to_put_into_r3 (only for `call` with 7 arguments)	value_to_put_into_r4	`0xFFFF` to prevent optimizing out by Yul	value_to_pu
`set_context_u128`	0	`0xFFF3`	value	0	`0xFFFF` to prevent optimizing out by Yul	0
`set_pubdata_price`	in0	`0xFFF2`	0	0	`0xFFFF` to prevent optimizing out by Yul	0
`increment_tx_counter`	0	`0xFFF1`	0	0	`0xFFFF` to prevent optimizing out by Yul	0

`ptr_calldata`	0	`0xFFF0`	-	0	`0xFFFF` to prevent optimizing out by Yul	0
`call_flags`	0	`0xFFEF`	-	0	`0xFFFF` to prevent optimizing out by Yul	0
`ptr_return_data`	0	`0xFFEE`	-	0	`0xFFFF` to prevent optimizing out by Yul	0
`event_initialize`	in1	`0xFFED`	-	in2	`0xFFFF` to prevent optimizing out by Yul	0
`event_write`	in1	`0xFFEC`	-	in2	`0xFFFF` to prevent optimizing out by Yul	0
`load_calldata_into_active_ptr`	0	`0xFFEB`	-	0	`0xFFFF` to prevent optimizing out by Yul	0
`load_returndata_into_active_ptr`	0	`0xFFEA`	-	0	`0xFFFF` to prevent optimizing out by Yul	0
`ptr_add_into_active`	in1	`0xFFE9`	-	0	`0xFFFF` to prevent optimizing out by Yul	0
`ptr_shrink_into_active`	in1	`0xFFE8`	-	0	`0xFFFF` to prevent optimizing out by Yul	0
`ptr_pack_into_active`	in1	`0xFFE7`	-	0	`0xFFFF` to prevent optimizing out by Yul	0
`multiplication_high`	in1	`0xFFE6`	-	in2	`0xFFFF` to prevent optimizing out by Yul	0
`extra_abi_data`	0	`0xFFE5`	-	0	`0xFFFF` to prevent optimizing out by Yul	0

Requirements for calling system contracts

By default, all system contracts up to the address `0xFFFF` require that the call was done via system call (i.e. `call_flags&2 != 0` .

Exceptions:

- BOOTLOADER_FORMAL address as the users need to be able to send money there.

Meaning of ABI params:

- MSG_VALUE_SIMULATOR: `extra_abi_data_1 = value || whether_the_call_is_system`, where || denotes the concatenation, value should occupy first 128 bits, while `whether_the_call_is_system` is a 1-bit flag that denotes whether the call should be a system call. `extra_abi_data_2` is the address of the callee.
- No meaning for the rest