D:/Coding/codeGen\5\mock_out.yaml

```
$schema: inst_schema.json#
kind: instruction
name: mock
long_name: Mock Instruction (Just for testing UDB)
description: 'The mock instruction computes the value of PI to an infinite number
 of decimal places.
 Okay, actually it performs the equivalent of the `mul` instruction.
 [NOTE]
 Computing PI to an infinite number of decicial places is impossible, but hey, why
 not?
 '
definedBy: Xmock
assembly: xd, xs1, xs2
encoding:
 match: 0000001----------000-----0001011
 variables:
 - name: xs2
 location: 24-20
 - name: xs1
 location: 19-15
 - name: xd
 location: 11-7
access:
 s: always
 u: always
 vs: always
 vu: always
data_independent_timing: true
operation(): "#anchor(\"illegal-inst-exc-misa-disabled\") {\n if (implemented?(ExtensionName::M)\
\ && (CSR[misa].M == 1'b0)) {\n raise (ExceptionCode::IllegalInstruction, mode(),\
\ $encoding);\n }\n#}\n\nXReg src1 = X[xs1];\nXReg src2 = X[xs2];\n\n#anchor(\"\
 calculation\") {\n X[xd] = (src1 * src2)[MXLEN-1:0];\n#}\n"
sail(): "{\n if extension(\"M\") | haveZmmul() then {\n let rs1_val = X(rs1);\n\
\  let rs2_val = X(rs2);\n let rs1_int : int = if signed1 then signed(rs1_val)\
\ else unsigned(rs1_val);\n let rs2_int : int = if signed2 then signed(rs2_val)\
\ else unsigned(rs2_val);\n let result_wide = to_bits(2 * sizeof(xlen), rs1_int\
\ * rs2_int);\n let result = if high\n then result_wide[(2\
\ * sizeof(xlen) - 1) .. sizeof(xlen)]\n else result_wide[(sizeof(xlen)\
\ - 1) .. 0];\n X(rd) = result;\n RETIRE_SUCCESS\n } else {\n handle_illegal();\n\
\  RETIRE_FAIL\n }\n}\n"
cert_normative_rules:
```

```
- id: inst.mock.encoding&basic;_op
 name: Encoding and basic operation
 description: Encoding and basic operation for `mock` instruction
 doc_links:
 - manual:inst:mul:encoding
 - udb:doc:inst:mock
- id: inst.mock.ill_exc_misa_M_disabled
 name: Illegal instruction exception when misa.M is 0
 description: 'An illegal instruction exception is raised when the instruction is
 executed
 and `misa.M` is 0.
 '
 doc_links:
 - manual:csr:misa:disabling-extension
cert_test_procedures:
- id: inst.mock.enc_and_basic
 description: Verify the encoding and basic operation of the `mock` instruction
 normative_rules:
 - inst.mock.encoding&basic;_op
 steps: '. Setup
 .. Load a variety of known values into rs1 & rs2 with a variety of rs1/rs2/rd
 values.
 . Execution
 .. Execute the `mock` instruction
 . Validation
 .. Check each result in rd
 . Teardown
 .. Clear the registers used for rd
 [NOTE]
 Don''t really need to clear the registers so this is a contrived example.
 I''ve got this note after the ordered list above.
 '
```