

Module: R5: RV-fpga

Guide for testing RISC-V Instructions

Guide for Testing Assembly

VeeR EH1

➤ Test Plan Guide for Testing RISC-V Assembly Instructions

This guide will walk you through the steps required to test different RISC-V assembly instructions using the VeeR EH1 core. Follow the steps carefully to ensure proper setup and execution.

Step 1: Download the Required Files

1. Download the “**testplan_VeeR_EH1.tar.xz**” files provided.
2. Ensure the download is complete and the files are not corrupted.

Step 2: Extract the Files

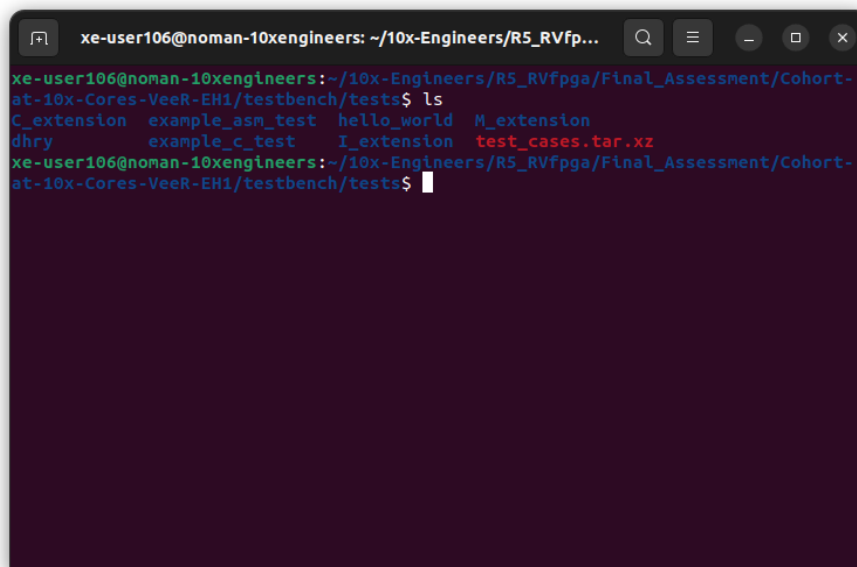
1. Locate the downloaded “**testplan_VeeR_EH1.tar.xz**” file in your system.
2. Extract the contents of the tar file to the appropriate path. The path where you should extract the files will look something like this:

`path-to-VeeR-core/testbench/tests/`

In your case, it should be:

`$RV_ROOT/testbench/tests/`

3. Verify that the files are correctly extracted to the specified directory. A screenshot is attached for reference.

A terminal window with a dark purple background. The title bar shows the user 'xe-user106@noman-10xengineers' and the current directory '~/10x-Engineers/R5_RVfpga...'. The terminal text shows the user navigating to the directory '~/10x-Engineers/R5_RVfpga/Final_Assessment/Cohort-at-10x-Cores-VeeR-EH1/testbench/tests' and running the 'ls' command. The output of 'ls' lists several files: 'C_extension', 'example_asm_test', 'hello_world', 'M_extension', 'dhry', 'example_c_test', 'I_extension', and 'test_cases.tar.xz'.

```
xe-user106@noman-10xengineers: ~/10x-Engineers/R5_RVfpga/Final_Assessment/Cohort-at-10x-Cores-VeeR-EH1/testbench/tests$ ls
C_extension  example_asm_test  hello_world  M_extension
dhry         example_c_test    I_extension  test_cases.tar.xz
xe-user106@noman-10xengineers:~/10x-Engineers/R5_RVfpga/Final_Assessment/Cohort-at-10x-Cores-VeeR-EH1/testbench/test$
```

Step 3: Create a Temporary Directory for Binaries and Hex Files

1. Choose a location on your system where you want to generate and store the binary and hex files.
2. Create a new directory called “**temp**” in that location.

```
mkdir /path-to-your-temp-directory/temp
```

3. Navigate into the “temp” directory.

```
cd /path-to-your-temp-directory/temp
```

4. Open a terminal in the “temp” directory to prepare for the next steps.

Step 4: Run the Make Command

1. In the terminal, enter the following command to test a specific RISC-V instruction:

```
make -f $RV_ROOT/tools/Makefile verilator TEST=test_xor  
TEST_DIR=$RV_ROOT/testbench/tests/C_extension/test_xor
```

2. Modify the command as follows:

1. **TEST**: Replace “**test_xor**” with the specific instruction you want to test.
2. **TEST_DIR**: Set the path to the directory containing the test files for the specific extension of the RISC-V base architecture you are working with. Ensure this path is accurate.
3. For example, you want to test **mul instruction** from M_extension, the command should be formatted like this:

```
make -f $RV_ROOT/tools/Makefile verilator TEST=test_mul  
TEST_DIR=$RV_ROOT/testbench/tests/M_extension/test_mul
```

```

VerilatorTB: End of sim
xe-user106@noman-10xengineers:~/10x-Engineers/R5_RVfpga/Final_Assessment/Outputs/example-test$ make -
f $RV_ROOT/tools/Makefile verilator TEST=test_mul TEST_DIR=$RV_ROOT/testbench/tests/M_extension/test_
mul
riscv64-unknown-elf-cpp -Isnapshots/default /home/xe-user106/10x-Engineers/R5_RVfpga/Final_Assessmen
t/Cohort-at-10x-Cores-VeeR-EH1/testbench/tests/M_extension/test_mul/test_mul.s > test_mul.cpp.s
riscv64-unknown-elf-as -mabi=ilp32 -march=rv32imc_zicsr test_mul.cpp.s -o test_mul.o
Building test_mul
riscv64-unknown-elf-gcc -mabi=ilp32 -march=rv32imc_zicsr -Wl,-Map=test_mul.map -lgcc -T/home/xe-user1
06/10x-Engineers/R5_RVfpga/Final_Assessment/Cohort-at-10x-Cores-VeeR-EH1/testbench/link.ld -o test_mu
l.exe test_mul.o -nostartfiles
riscv64-unknown-elf-objcopy -O verilog test_mul.exe program.hex
riscv64-unknown-elf-objdump -S test_mul.exe > test_mul.dis
Completed building test_mul
./obj_dir/Vtb_top

VerilatorTB: Start of sim

-----
TEST FAILED
-----

Finished : minstret = 922, mcycle = 2891
See "exec.log" for execution trace with register updates..

- /home/xe-user106/10x-Engineers/R5_RVfpga/Final_Assessment/Cohort-at-10x-Cores-VeeR-EH1/testbench/tb
_top.sv:344: Verilog $finish

VerilatorTB: End of sim
xe-user106@noman-10xengineers:~/10x-Engineers/R5_RVfpga/Final_Assessment/Outputs/example-test$

```

Please note that the TEST_DIR has been changed to /M_extension/test_mul to test the mul instruction.

Step 5: Instruction Legend

Refer to the legend below for the appropriate instruction names to use in the “TEST” field:

RV32I		RV32M		
add	sb	mul	mulhu	rem
sub	sh	mulh	div	remu
xor	sw	mulhsu	divu	
or	beq			
and	bne			
sll	blt			
srl	bge			
sra	bltu			
slt	bgeu			
sltu	jal			
addi	jalr			
xori	lui			
ori	auipc			
lh				
lhu				
lw				

RV32C		
add	nop	bnez
mv	swsp	srli
jalr	addi4spn	srai
jr	lw	andi
lwsp	sw	
li	and	
lui	or	
addi	xor	
addi16sp	sub	
slli	beqz	

Make sure to use the correct instruction name as per the extension you are testing.

Final Notes:

- Ensure your environment variables such as “**RV_ROOT**” are set correctly.
- The “**verilator**” target will simulate the test and generate the relevant binaries and hex files in your “**temp**” directory.

Follow these steps closely, and you'll be able to test various RISC-V assembly instructions effectively.