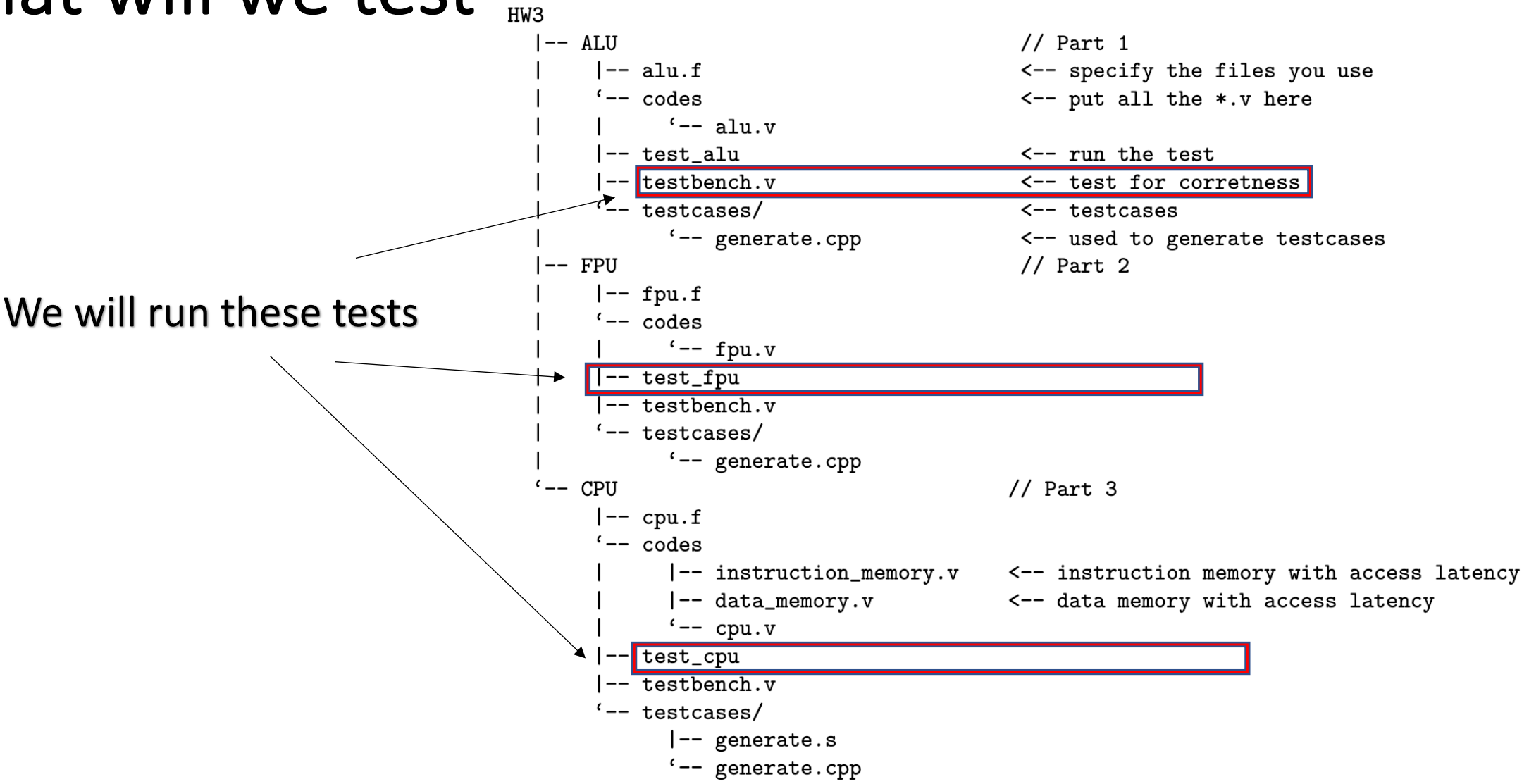
Supplementary

What will we test

We will run these tests



How we score

```
==========================
Test #0: Correct!
Test #1: Correct!
Test #2: Correct!
Test #3: Correct!
ALL PASS!
Finish flip test.
==========================
Test #0: Correct!
Test #1: Correct!
Test #2: Correct!
Test #3: Correct!
ALL PASS!
Finish reverse test.
==========================
Finish all tests.
Score: 30.0/30.0
```

alu

```
ALL PASS!
Finish add test.
======================
Test # 0: Correct!
Test # 1: Correct!
Test # 2: Correct!
Test # 3: Correct!
Test # 4: Correct!
Test # 5: Correct!
Test # 6: Correct!
Test # 7: Correct!
Test # 8: Correct!
Test # 9: Correct!
ALL PASS!
Finish mul test.
======================
Finish all tests.
Score: 20.0/20.0
```

fpu

```
VCD info: dumpfile cpu.vcd opened for output.
Load test
Check memory
Correct!
VCD info: dumpfile cpu.vcd opened for output.
Add sub test
Check memory
Correct!
VCD info: dumpfile cpu.vcd opened for output.
And or xor test
Check memory
Correct!
VCD info: dumpfile cpu.vcd opened for output.
Andi ori xori test
Check memory
Correct!
VCD info: dumpfile cpu.vcd opened for output.
Slli srli test
Check memory
Correct!
VCD info: dumpfile cpu.vcd opened for output.
Bne beq test
Check memory
Correct!
```

cpu

You can test cpu testcase one by one

```
iverilog -D T0 -f cpu.f
vvp ./a.out
iverilog -D T1 -f cpu.f
vvp ./a.out
iverilog -D T2 -f cpu.f
vvp ./a.out
iverilog -D T3 -f cpu.f
vvp ./a.out
iverilog -D T4 -f cpu.f
vvp ./a.out
iverilog -D T5 -f cpu.f
vvp ./a.out
iverilog -D T6 -f cpu.f
vvp ./a.out
iverilog -D T7 -f cpu.f
vvp ./a.out
```

About instruction_memory.v

- We will read the testcase to instruction memory at the beginning
- The starting address of instructions is 0.
- You should send 64-bit address and one cycle valid signals to fetch instructions.
- The 32-bit instruction will be fetched 5 cycles later.
- No pipeline read memory.
- You should send **one cycle** i_valid to read memory

About data_memory.v

- Switch to read mode when MemRead is high
- Switch to write mode when MemWrite is high
- Do not read and write memory simultaneously
- The data will be read 7 cycles later.
- The data will be written 7 cycles later.
- No pipeline read/write memory.
- You should send **one cycle** i_MemRead/i_MemWrite to access memory

Submit format

- Please put your files in the following format. And upload as a zip file

```
Rxxxxxxxx                    <-- zip this folder
    |-- ALU
    |    |-- codes/
    |    |     `-- *.v       <-- files you used
    |    `-- alu.f           <-- list all the files needed
    |-- FPU
    |    |-- codes/
    |    |     `-- *.v       <-- files you used
    |    `-- fpu.f           <-- list all the files needed
    |-- CPU
    |    |-- codes/
    |    |     `-- *.v       <-- files you used
    |    `-- cpu.f           <-- list all the files needed
    `-- report.pdf
```