Unpipelined Design:

With the project design review phase, you have already designed your datapath and control path for a single-cycle unpipelined implementation for WISC-SP13 ISA. So the obvious next step is to write the code for the single-cycle unpipelined implementation. In this document, you will find details regarding the expectations, testing, and grading of your written implementation. Note that your implementation may be different from your design review and that is ok. Changes in the design are expected when you implement it.

As usual, all the Verilog coding rules, naming conventions, etc apply to this project also. If you have forgotten them please visit the hw1.pdf (or Canvas-> Files-> Verilog Tools, Scripts & Guides).

Expectations:

Memory:

In your single-cycle unpipelined design you will use 2 separate memories. The first is for instruction fetch and the second is for data memory. Your memories will be single-cycle perfect memories. You will use the same memory2c.v that you used in hw3 but now instead of instantiating it only once for instruction memory (as you did earlier), you should instantiate it twice, once for instruction memory and the other for the data memory.

Modularize the design:

We recommend that you make a modular design. The suggested approach is to design the 5 stages of MIPS as their own sub-high-level modules. Your proc.v should be as simple as just calling all the 5 stages with wires to pass the data. All the logic is handled inside the 5 stages modules.

Other things to note:

Your implementation should run the complete WISC-SP13 ISA except for the bonus part. The instructions for extra credits (More details in “Special Instruction” of WiscSP13 Microarchitecture and ISA specification pdf) will only be tested at the end of the semester in the final demo and you are not required to implement them for this phase of the project.

For testing and verifying your implementation you will be using wsrun.pl command with -list flag (similar to -prog option). More details later in the document (and also in this pdf).

Testing:

Names and Testbench:

Similar to hw3_3 you will be provided with a testbench, memory module, dff, etc. Your top module will be named proc.v. You will **need to change** some **variable names** in **proc_hier_bench.v** (they will be clearly demarcated). Please make sure to change the testbench else the code will not run.

Feel free to add as many modules as you like and also reuse the older modules. Homeworks are designed in such a way that you can reuse many of the older modules easily.

wsrun.pl -list option:

To test your unpipelined design you will be using wsrun.pl with -list option. The following shows an example of how you can use the -list option with wsrun.pl:

```
wsrun.pl -list rand_mem.list proc_hier_bench *.v
```

This will run all the programs from rand_mem.list eg:

```
/u/s/w/swamit/courses/cs552/fall2021/handouts/testprograms/public/rand_mem/t_1_mem.asm
/u/s/w/swamit/courses/cs552/fall2021/handouts/testprograms/public/rand_mem/t_2_mem.asm
/u/s/w/swamit/courses/cs552/fall2021/handouts/testprograms/public/rand_mem/t_3_mem.asm
/u/s/w/swamit/courses/cs552/fall2021/handouts/testprograms/public/rand_mem/t_4_mem.asm
/u/s/w/swamit/courses/cs552/fall2021/handouts/testprograms/public/rand_mem/t_5_mem.asm
/u/s/w/swamit/courses/cs552/fall2021/handouts/testprograms/public/rand_mem/t_6_mem.asm
/u/s/w/swamit/courses/cs552/fall2021/handouts/testprograms/public/rand_mem/t_7_mem.asm
/u/s/w/swamit/courses/cs552/fall2021/handouts/testprograms/public/rand_mem/t_8_mem.asm
```

After running all the programs in it will write the output in summary.log as follows:

```
/u/s/w/swamit/courses/cs552/fall2021/handouts/testprograms/public/rand_mem/t_1_mem.asm
SUCCESS CPI:1.0 CYCLES:2258 ICOUNT:2255 IHITRATE: 0 DHITRATE: 0
/u/s/w/swamit/courses/cs552/fall2021/handouts/testprograms/public/rand_mem/t_2_mem.asm
SUCCESS CPI:1.0 CYCLES:2338 ICOUNT:2335 IHITRATE: 0 DHITRATE: 0
/u/s/w/swamit/courses/cs552/fall2021/handouts/testprograms/public/rand_mem/t_3_mem.asm
SUCCESS CPI:1.0 CYCLES:2174 ICOUNT:2171 IHITRATE: 0 DHITRATE: 0
/u/s/w/swamit/courses/cs552/fall2021/handouts/testprograms/public/rand_mem/t_4_mem.asm
SUCCESS CPI:1.0 CYCLES:2028 ICOUNT:2025 IHITRATE: 0 DHITRATE: 0
/u/s/w/swamit/courses/cs552/fall2021/handouts/testprograms/public/rand_mem/t_5_mem.asm
SUCCESS CPI:1.0 CYCLES:2234 ICOUNT:2231 IHITRATE: 0 DHITRATE: 0
/u/s/w/swamit/courses/cs552/fall2021/handouts/testprograms/public/rand_mem/t_6_mem.asm
SUCCESS CPI:1.0 CYCLES:2176 ICOUNT:2173 IHITRATE: 0 DHITRATE: 0
/u/s/w/swamit/courses/cs552/fall2021/handouts/testprograms/public/rand_mem/t_7_mem.asm
SUCCESS CPI:1.0 CYCLES:2288 ICOUNT:2285 IHITRATE: 0 DHITRATE: 0
/u/s/w/swamit/courses/cs552/fall2021/handouts/testprograms/public/rand_mem/t_8_mem.asm
SUCCESS CPI:1.0 CYCLES:2156 ICOUNT:2153 IHITRATE: 0 DHITRATE: 0
```

The -list option will take each asm file from list file and run it with -prog option as discussed in the hw3.

Tests for Unpipelined Implementation:

To test the working of your code we will have many tests. Below is the points distribution of each test as well as what to submit for each test. Note that after running each test you will get summary.log generated, which you will rename to appropriate names as written below:

1. **Simple tests: (10 points)**

```
wsrun.pl -list simple.list proc_hier_bench *.v
```

Rename the generated `summary.log` to `simple.summary.log`

Note: If you fail more than 5 simple tests. You will directly lose 75% points for the single-cycle unpipelined project. Make sure you do rigorous testing and ensure that you are not failing the tests.

2. **Complex tests: (18 points)**

```
wsrun.pl -list complex.list proc_hier_bench *.v
```

Rename the generated `summary.log` to `complex.summary.log`

3. **rand_simple tests: (18 points)**

```
wsrun.pl -list rand_simple.list proc_hier_bench *.v
```

Rename the generated `summary.log` to `rand_simple.summary.log`

4. **rand_complex tests: (18 points)**

```
wsrun.pl -list rand_complex.list proc_hier_bench *.v
```

Rename the generated `summary.log` to `rand.summary.log`

5. **rand_ctrl tests: (18 points)**

```
wsrun.pl -list rand_ctrl.list proc_hier_bench *.v
```

Rename the generated `summary.log` to `rand_ctrl.summary.log`

6. **rand_mem tests: (18 points)**

```
wsrun.pl -list rand_mem.list proc_hier_bench *.v
```

Rename the generated `summary.log` to `rand_mem.summary.log`

Additional Notes:

- After running the command check the generated summary.log file. It will show you whether your tests have been successful or not.
- After running the command and successfully passing the test please rename the summary.log file (manually) as requested. You need to submit the renamed file in your final submission.
- Note that if you fail more than 5 tests in the simple test you will automatically lose 75% points. So make sure you test things rigorously.
- Make sure you update the variable names in testbench before you proceed with verifying the code. The variables which you need to change are clearly demarcated in proc_hier_bench.v.
- Make sure you run the vcheck as well as name-convention-check script before submitting.

What to submit

You need to submit a single tar file named `project-unpipelined.tar`.You will submit it directly on the canvas. It will contain the following files

```
- All *.v files needed to run and verify the implementation
- Updated testbench (proc_hier_bench.v)
- All *.vcheck.out files
- simple.summary.log
- complex.summary.log
- rand_simple.summary.log
- rand_complex.summary.log
- rand_ctrl.summary.log
- rand_mem.summary.log
```