**CE-321L/CS-330L:**

**Computer Architecture**

**Implementing Single & Pipeline Processor**

**with hazard detection in risc-v**

**Date: 10–April-2025**

**Submitted by:**

Muhammad Wasiq Shaikh

Syed Muhammad Kazim Raza

Haris Ahmed Khan

**Introduction**

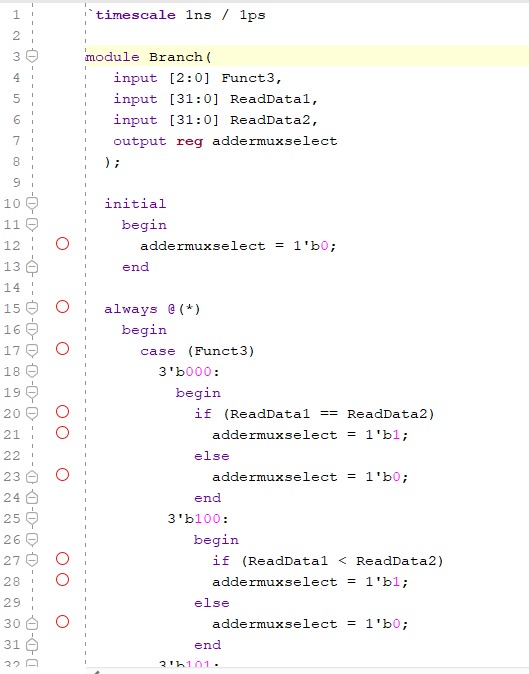
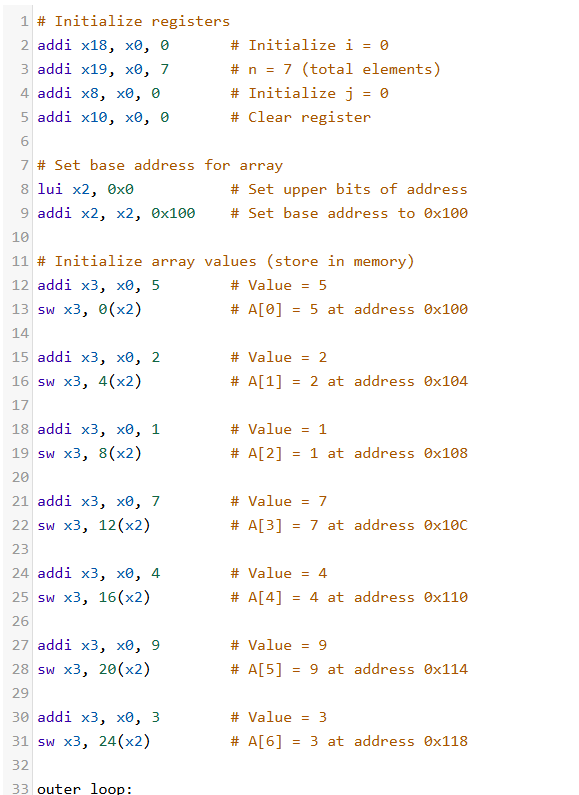
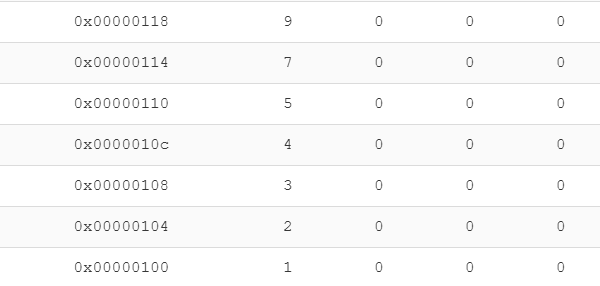
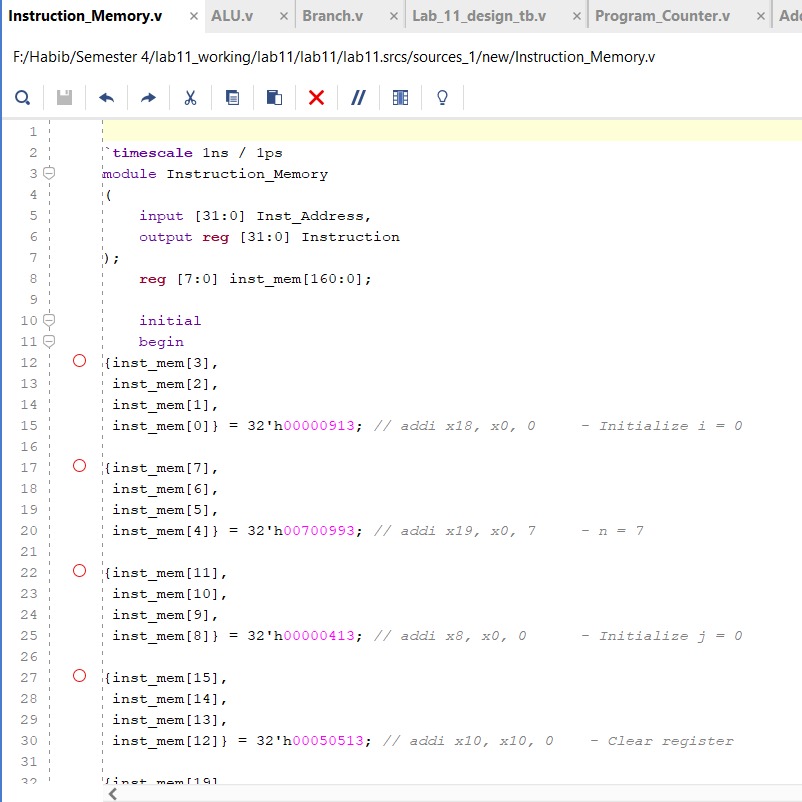
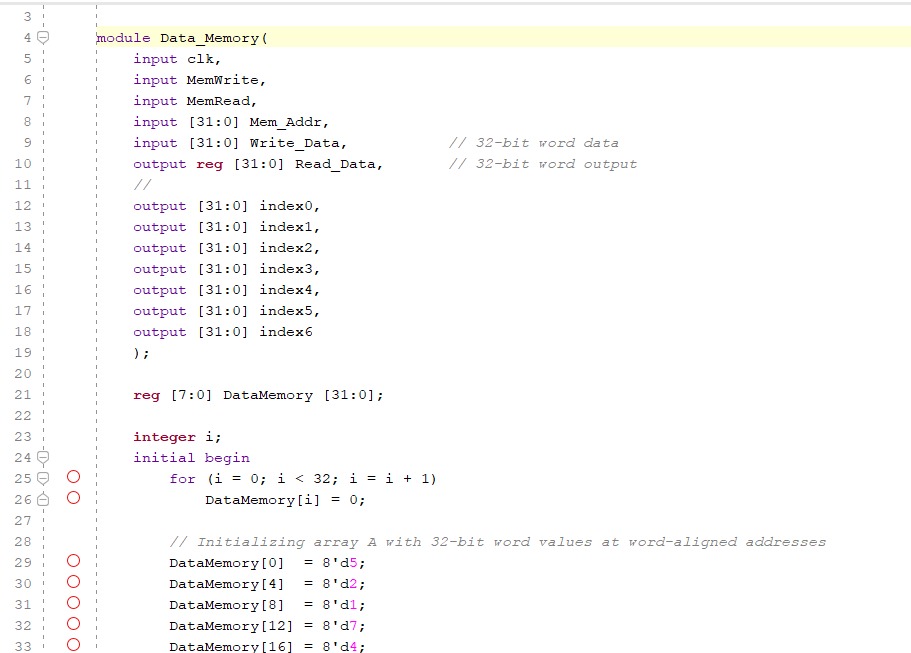
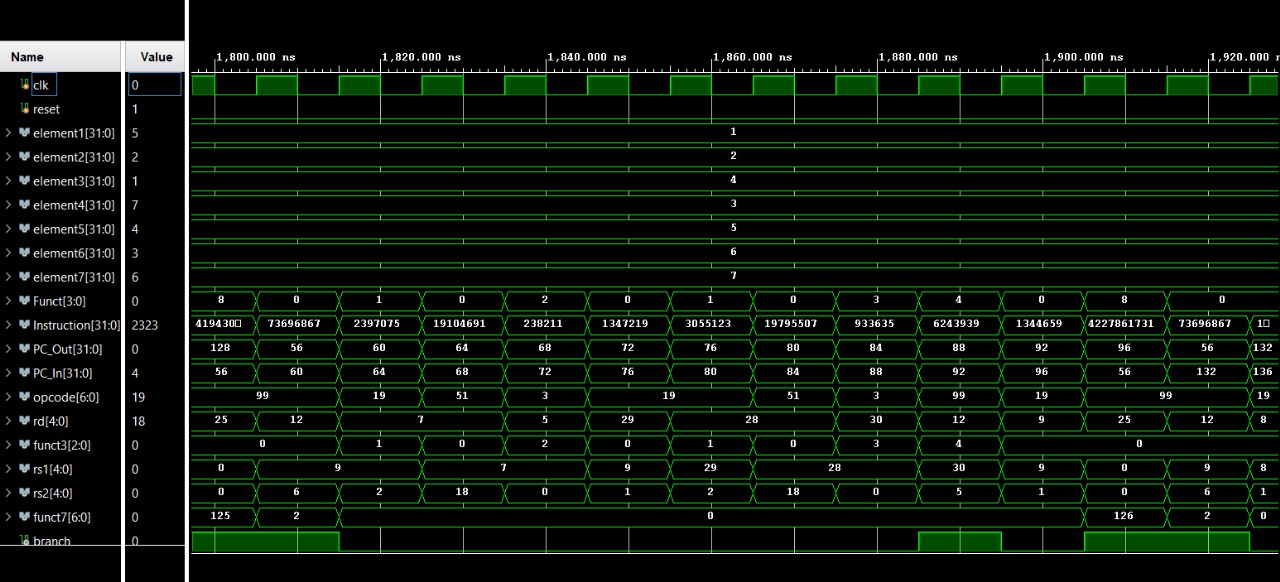
**Problem Statement**: Design a 5-stage RISC V pipelined processor that can run the bubble sorting algorithm on it and deals with hazards using forwarding, flushing and hazard detection and compare its performance with a single cycle processor.

**Objectives:**

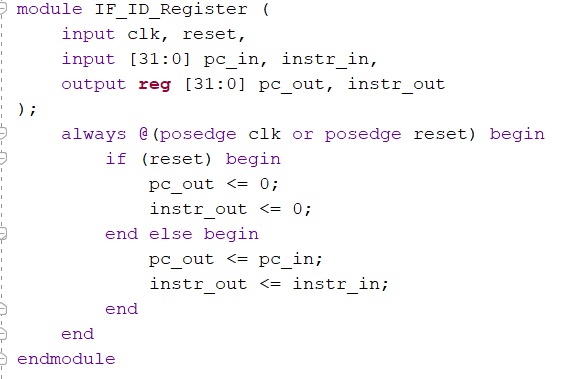
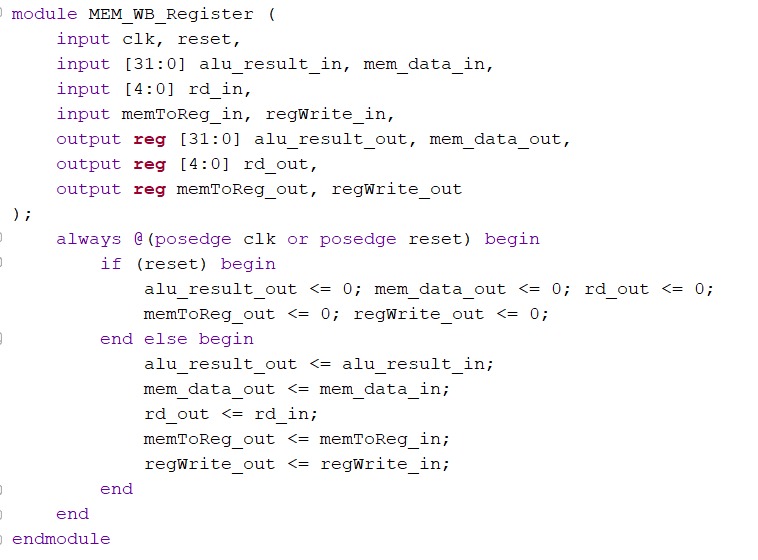
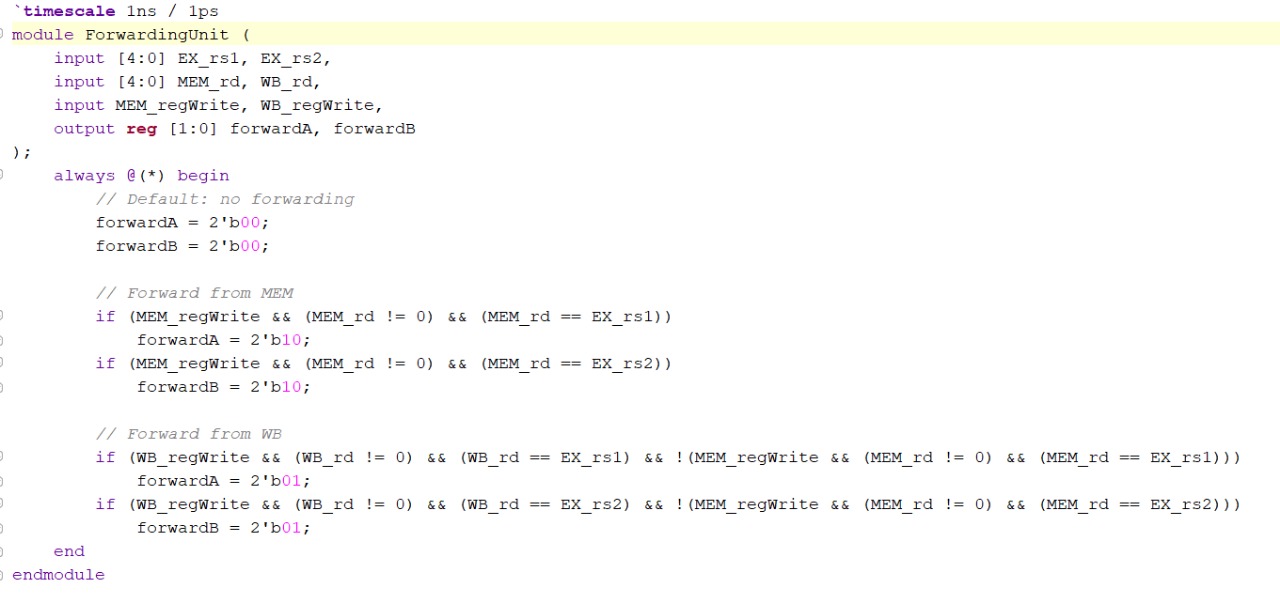
* Create a baseline single cycle processor that can run the bubble sort algorithm.
* Convert this baseline into 5 stage pipeline processor with forwarding to deal with data hazards.
* Add control unit and hazard detection unit on top of pre-existing code to deal with any hazards such as memory or control hazard.
* Calculate CPI and execution time for both the complete pipelined version and the single cycle baseline, and compare the performances

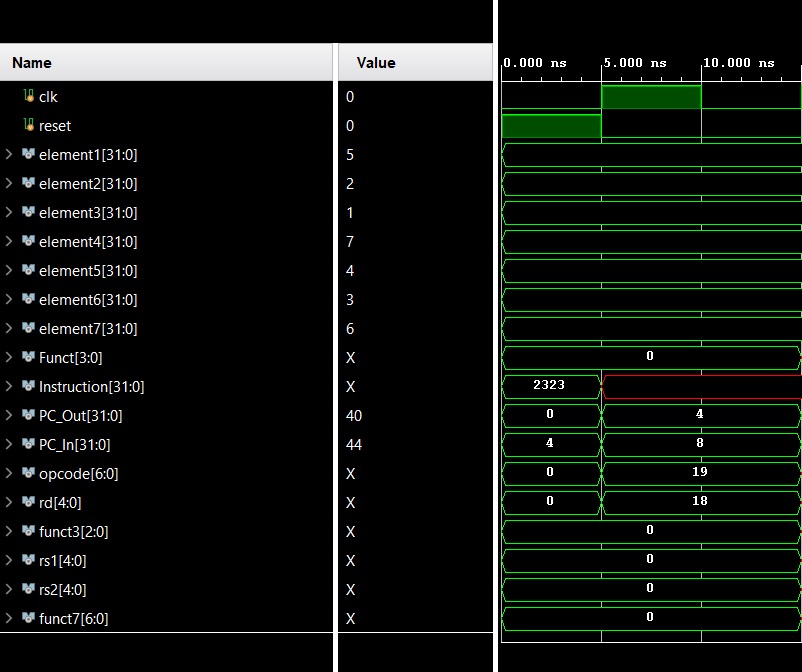
**Methodology**

**Task 1:**

* We started off with a single cycle basic processor from lab 011, then we added a branch module given below:
*   
    
  We used the branch module in order to implement bubble sort as the prexisting code was insufficient to execute it.
* We started off with writing an assembly code and testing it on venus for bubble sort as given:  
    
    
    
    
    
  We also modified instruction memory with machine code of our RISC V assembly code and modified data memory in order to run the code for single cycle as given below along with the waveform:  
    
    
    
    
  

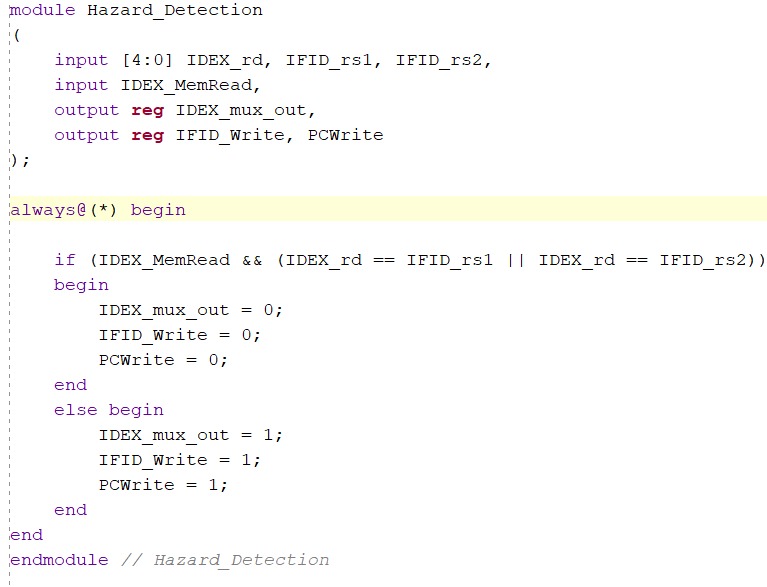
**Task 2:**

* To implement Pipelining, instead of using wires between modules, which would update values in real time instead of at clock cycles hence causing problems, we use intermediate registers that store values from wires and then forward these values according to clock cycle change. Some snippets of registers are given:  
    
  
* 
* We also implemented a forwarding unit whereby according to conflicts between rd and either rs1, rs2 or both, we update the values of forward A or forward B or both and by this process, we reduce data hazards, we can either forward from MEM stage or from WB stage back to Execute stage
* 

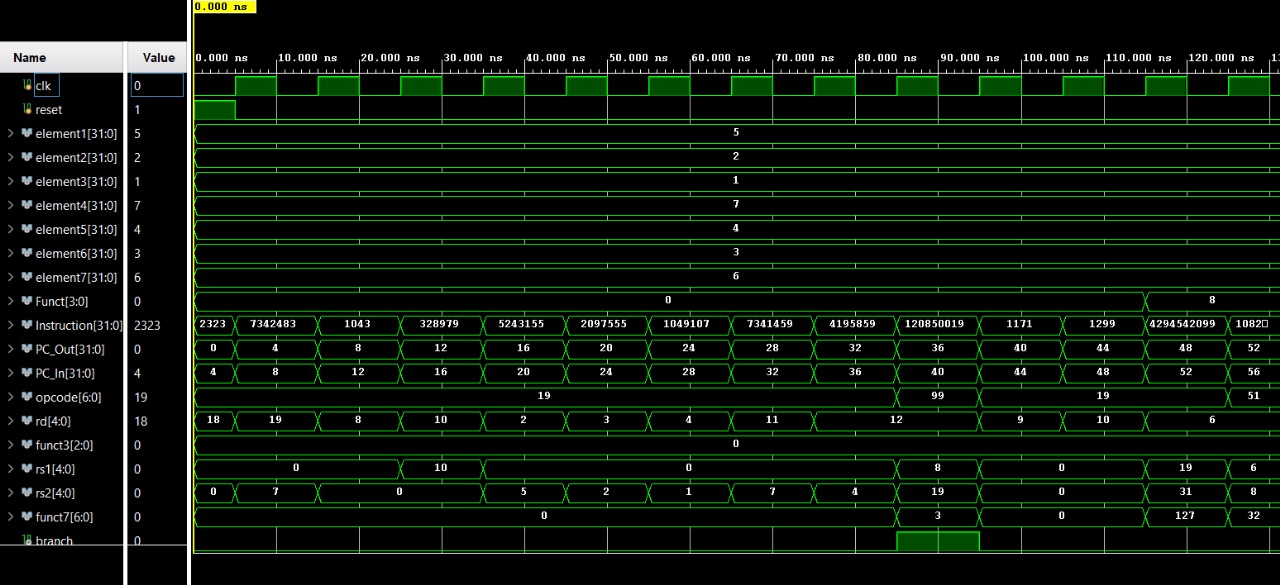


Above is a waveform showing the execution of one instruction with forwarding and pipelining, there are still issues at this stage as we are yet to resolve control and memory hazards.

**Task 3:**

* To resolve memory hazard, we implemented hazard detection unit, which installs hardware nops so that memory hazards are resolved as shown below:  
    
  
* **Task 4:**  
    
    
  **CPI (Cycles Per Instruction) = ~1 (but takes multiple cycles per instruction, e.g., 5 stages = 5 cycles per instruction).**
* Throughput = ~1 instruction / 5 cycles (assuming 5-stage pipeline).
* CPI ≈ 1 (ideally, 1 instruction completes per cycle).
* Throughput = ~1 instruction / cycle (5x speedup over non-pipelined).
* Speedup ≈ Number of Pipeline Stages (theoretical max).
* Speedup≈1.35 ×1.2≈4.6x

**Results**

  
We observed that pipelined processor is significantly faster almost 4.6 times fast as compared to single cycle processor.

**Challenges**

* It was difficult to extend from sorting 3 elements to 7 elements due to bugs, we were able to resolve this by rectifying our loop in code,
* It was difficult to make the base address of array to be from 0x100 we had to resort to using 0 as base address.

**Task Division**

|  |  |
| --- | --- |
| **Team Member** | **Tasks Contribution** |
| **Syed Muhammad Kazim Raza** | **Task 1: Single Cycle Pipeline + Report** |
| **Muhammad Wasiq Shaikh** | **Task 2: Pipeline Processor with Forwarding + Report** |
| **Haris Ahmad Khan** | **Task 3: Added Hazard detection and Control Unit + Report** |

**Conclusions**

* Our project still has some bugs to fix but overall, we learned that pipelined processor with hazard detection and forwarding is almost 4.6 times faster and has better throughput than single cycle processor.

**References** (If any)

Patterson & Hennessy, Computer Organization and Design (Chapter 4).

**Appendices**

Either add your code in appendix or provide link to ***github*** of your project.

GitHub Link:  
  
<https://github.com/Zealox12/CA_ProcessorProject.git>

**Note:** The report must be well formatted, and must include title page and page numbers. Figures, results and simulations must be clear and readable.

Refer to the guidelines for what content to be added in each task, check the evaluation rubric.