Assignment:5

CSL3020: Computer Architecture
AY 2024-25, Semester – V
Due on: 6-10-2024

Total:150 Marks

General Instructions:

1. Clearly mention the assumptions you have made, if any.
2. Clearly report any resources you have used while attempting the assignment.
3. Any submission received in another format or after the deadline will be Penalized
4. Make sure to add references to the resources that you have used while attempting the assignment.
5. Plagiarism of any kind will not be tolerated and will result in zero marks.
6. Select the correct and working program for Testing.

Submission Guidelines:

You can form a group of at most three members, but only the leader is permitted to submit the assignment.

Clearly list the details of your group members in the report, along with their specific contributions to the assignment.

Submit a single report outlining the architecture, methods, results, and observations. Ensure the report is concise and thoroughly professional.

No need to add your code in the Report.

Preparing a report is mandatory; failing it will lead to non-evaluation of the assignment.

1. Name your files as yourRollNo._YourGroupMembersRollNo.pdf and yourRollNo._YourGroupMembersRollNo.c/cpp/py
   your test programs.

2. Adhere to the instructions given, failing them may result in a penalty.

NOTE: ONLY GROUP LEADER IS SUPPOSED TO SUBMIT THE ASSIGNMENT.

MIPS Processor Simulation

Objective:

In this assignment, you will develop a simulation of a MIPS processor that performs the following:

1. **Compile MIPS Assembly Code into Machine code (binary instructions).**
2. **Simulate the execution of the Machine code using a simulated MIPS datapath, including generating control signals and simulating the ALU and the rest of the operations.**

By the end of the assignment, your simulation should be able to:

- **Translate MIPS assembly into machine code.**
- **Simulate the execution of basic MIPS instructions including arithmetic, memory access, and branching.**
- **Generate control signals and simulate the data flow within an MIPS-like architecture.**

Note: You are free to use any language (c, cpp, python, etc.)

Task - 1: [50]

Implement a MIPS compiler that reads MIPS assembly instructions and converts them into binary machine instructions. The compiler should be able to handle the .data and .text sections having memory allocation and around 10 MIPS instructions, including:

- **R-type instructions (`add`, `sub`, `and`, `or`, `slt`)**
- **I-type instructions (`lw`, `beq`, `addi`)**
- **J-type instructions (`j`)**

Note: You can also add more instructions on your own.

Instructions:

1. **Read Assembly Input: Your program should take a MIPS assembly program as input (you can use a text file with MIPS code).**
2. **Parse the Instructions: For each line of MIPS assembly, identify the type of instruction (R-type, I-type, J-type), the operation, and the registers or immediate values involved.**
3. **Convert to Binary: For each instruction, convert it into the corresponding binary format, adhering to the standard MIPS instruction format (opcode, funct, registers, etc.).**
4. **Output the Binary: Output the binary code corresponding to the MIPS instructions.**

Example:

An Instruction add $t1, $t2, $t3 should be compiled to somewhere like

000000 01010 01011 01001 00000 100000

Task- 2: MIPS Execution [50]

Once the binary instructions are generated, simulate the execution of the MIPS instructions using a simulated processor. Your executioner should:

- **Simulate the MIPS datapath.**
- **Generate and apply control signals.**
- **Simulate ALU operations, memory access, and branch instructions.**

Instructions:

You should follow the standard MIPS instruction Pipeline. i.e

- **Instruction Fetch**
- **Instruction Decode**
- **Execution**
- **Memory Write back**

Your execution should be able to perform

- **ALU operations (e.g., addition, subtraction).**
- **Instruction fetch, decode, etc.**
- **Generate Control signals that will drive the execution.**
- **Simulate memory access (for lw and sw). (handle offsets calculation as well)**
- **Handle branching (beq) by updating the program counter.**
- **Simulate main memory and registers (32 registers for MIPS).**
- **Ensure proper reads and writes to registers and memory.**
- **Simulate the program counter (PC) that keeps track of the current instruction to be executed.**

Note: Your code should be modular and well-formatted.

Task - 3: Testing and Reporting [30] + [20]

You are required to test your simulator with a total of 5 MIPS programs. Test the simulator for the two programs given here. You can write your own 3 programs which tests your simulator.
(You should write fairly complex programs which can challenge your simulator)
You will be required to mention the results of all the programs in the report and analyze them.
Note: For printing the final register output you may use the default printing methods from the respective languages.

Deliverables:

- **Compiler simulator file**
- **Mips_processor file**
- **Your 3 test codes.**
- **A detailed report which contains all the points mentioned in.**

Report should contain the following points:

- **Describe your understanding with the tasks.**
- **Provide a brief explanation of the Architecture you implemented. focusing on its objective and key operations.**
- **Highlight any challenges faced and how you overcame them.**
- **Explain the simulator's working in detail.**
- **Provide the outputs of the test codes with detailed explanations.**
- **Reflect on what you learned from the assignment**
- **Explain the individual contribution.**

Evaluation Criteria

- **Accurate execution of the test codes.**
- **Thorough analysis of the simulator and its architecture**
- **Clarity: Clear and well-organized report presentation.**
- **Insightfulness: Depth of understanding and interpretation of execution cycle.**