

Computer System Architecture, Spring term 2020
Project Description

Overview

In this project, you are asked to implement a new microcontroller architecture using any language of your choice. You are asked to assemble your own project by selecting **one feature from each of the four main categories**.

To be able to assemble your project, you need to submit your features selection via the following link:
<https://bit.ly/3aHRUW9>

Note: Each feature has a capacity up to 37 team. **FIRST COME, FIRST SERVED**
Deadline for assembling your project is **Sunday; 5th of April 2020, 11.59 pm**

The guidelines for the project submission and testing will be posted later.

1 Microarchitecture

You **MUST** implement one of the following features:

1.1 Von Neumann architecture

- a) It has one memory for data and instructions.
- b) A single set of address/data buses between CPU and memory.
- c) It allows only **ONE** memory fetch at a time.

1.2 Harvard architecture

- a) It has two separate memories for instructions and data.
- b) It has two sets of address/data buses between CPU and memory.
- c) It allows **TWO** simultaneous memory fetches. One fetch for the data and the other for the instructions.

2 Instruction Memory and Data Memory Size

Your implementation of the two memories has to be one of the following memory sizes.

- a) 1024 x 16-bit
- b) 1024 x 32-bit

Requirements

- a) The memory size is reflected on the system registers size.
If you choose choice (a) you will have a register size of 16 bits. If you choose choice (b) you will have a register size of 32 bits

3 Total Number of Registers

Your implementation can have **ONE** of the following total number of registers.

- a) 16 Registers.
- b) 32 Registers.

Requirements

- a) You have to state the names of the registers as well as the purpose of each one.

4 Instruction Format

You **MUST** implement **ONE** of the following instruction sets :

4.1 Instruction Set 1

- a) **Arithmetic Instructions:**
 - 1. Add.
 - 2. Add immediate.
 - 3. Sub.
 - 4. Multiply.
- b) **Logical Instructions:**
 - 1. And.
 - 2. Or immediate.
 - 3. Shift left logical.
 - 4. Shift right logical.
- c) **Data Transfer Instructions:**
 - 1. Load word.
 - 2. Store word.
- d) **Conditional Branch Instructions:**
 - 1. Branch on not equal.
 - 2. Branch on greater than.
- e) **Comparison Instructions:**
 - 1. Set on less than.
- f) **Unconditional Jump Instructions:**
 - 1. Jump.

4.2 Instruction Set 2

- a) **Arithmetic Instructions:**
 - 1. Sub.
 - 2. Add.
 - 3. Add immediate.
 - 4. Multiply.
- b) **Logical Instructions:**
 - 1. Or.
 - 2. And immediate.
 - 3. Shift right logical.
 - 4. Shift left logical.
- c) **Data Transfer Instructions:**
 - 1. Load word.
 - 2. Store word.
- d) **Conditional Branch Instructions:**
 - 1. Branch on equal.
 - 2. Branch on less than.
- e) **Comparison Instructions:**
 - 1. Set on less than immediate.
- f) **Unconditional Jump Instructions:**
 - 1. Jump Register.

Requirements

- a) The instruction set is to be implemented using a new designed format, which **CANNOT** be the **MIPS** instruction format.
- b) You can add other instructions in your implementation, if needed.

5 Data Path

Your Datapath implementation will differ based on your customisation of the first four categories. but the it **MUST** have the following modules:

- a) Instruction Memory.
- b) Data Memory + Cache.
- c) Register File.
- d) Program Counter.
- e) Arithmetic Logic Unit.
- f) Control Unit(s).

Requirements

- a) The changes in the Microarchitecture has to be reflected in the implementation.
- b) The changes in the data memory and registers have to be clear in the implementation and testing.
- c) Each module is implemented independently from the other modules.
- d) The designed Datapath has to be **graphically presented** (Refernce: Lecture 7, Slide 11). You can use any online software to draw the datapath. For example, you can use the following link <https://app.diagrams.net/>.

6 Pipelining Stages

Your pipelining implementation **MUST** have the following stages:

- a) Fetch.
- b) Decode.
- c) Execute.
- d) Memory.
- e) Writeback.

Requirements

- a) You can add more stages, as required by your implementation.
- b) Pipeline registers values has to be reflected at each stage.

7 General Requirements

7.1 Loading Instructions

- a) The whole program has to be loaded before testing.

7.2 Output

- a) At each clock cycle, the following has to displayed:
 - 1. All registers contents.
 - 2. Changes in data memory.
 - 3. All control signals contents.
 - 4. Current executing instruction.For example; in clock cycle 2, what instruction is in each stage of the five stages in the pipelining, what are the operands, their values and the control signals values.

Project Submission Guidelines

- Failing to submit the project at all, results in having **ZERO** grade in the project grade.
- Copying others solution, copying online codes and changing the variables names and attempting to cheat in any other creative way, will only result in having a **ZERO** in the project.
- If you are submitting using the **GUC mail**, you should use **Internet Explorer**, because Google chrome **DOES NOT ATTACH FILES TO THE GUC MAIL**.

Project Testing Guidelines

- A detailed document will be uploaded next week to give you an example for the expected inputs and outputs. As well as the project deadline.