# RV32IM Pipelined CPU

1st Maya Hussein
*Computer Science & Engineering*
*The American University in Cairo*
Cairo, Egypt
mayamakram@aucegypt.edu

2nd Nada Badawi
*Computer Science & Engineering*
*The American University in Cairo*
Cairo, Egypt
nada_badawi@aucegypt.edu

## I. Introduction

In a pipelined RISC-V RV32I processor with a single port memory used for both instruction and data memory, hazard elimination techniques such as data forwarding and branch prediction can be incorporated to improve performance. Data forwarding can be used to overcome data hazards by forwarding the result of an instruction that has not yet been written back to a register to a subsequent instruction that needs it as an operand. This eliminates the need for pipeline stalling. Branch prediction can be used to overcome control hazards by predicting the outcome of branch instructions and speculatively executing instructions along the predicted path. This can reduce the number of pipeline stalls caused by control hazards. [1]

*Note: Most modules were adopted from this course's lab; however, major changes were made to the following modules.

## II. Discussion

The major adjustments were made to the main components: The data memory, instruction memory, and the ALU control unit. New components and features were introduced to be able to support all forty instructions without any hazards. A forwarding unit was adopted as well as a flushing technique which was implemented using a 2x1 multiplexer that either flushes or executes the current instruction based on the branch output and the ALU Jump control signals. A single byte-addressable memory was adopted as well in the code.

### A. Memory

The instruction memory and data memory no longer exist separately; they are now in one module. To avoid changing out test suite and instructions, we have made use of an offset variable that defines where the data memory is now required to be read from. The instructions are stored at the least memory addresses to get fetched correctly with the program counter starting from 0. Since the RISC-V architecture is Little Endian, the memory component was designed to be of the same byte ordering system, where the least significant bit is stored in the least memory address. The structure of the memory also supports both instruction and data memory where the data memory begin as location 200. The structure is as follows a multiplexer is added before the memory to take input from either the ALU Output or the Program Counter based on the clk. The data is then fetched from the memory and sent to decoder to choose the correct path for the data.

### B. ALU Control Unit

The ALU Control Unit had the signals of the M extension supported to add the multiplication and division calculations.

### C. Forwarding Unit

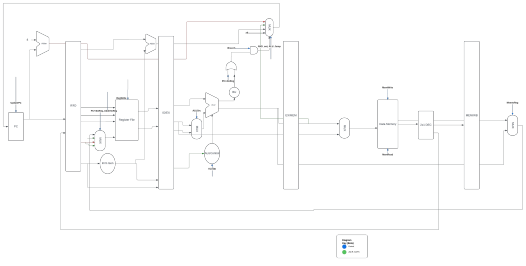
This was a newly introduced module that had the conditions of needing to forward applied. The forwarding would take place when there is a read after write dependency. However, minor amendments were made to remove some unneeded conditions in the unit that relates to the EX/MEM register, that it is because we need to only forward for one cycle and not two after implementing the single memory component. It was also noticed that the single memory removes the need of having a hazard detection unit and that the updated forwarding component and flushing mechanism would be sufficient to have correct results without any nop instructions injected.

### D. Test Programs

The test programs are installed in the instruction memory by reading the file using the readmemh() method. Whenever another test program is needed to be installed the file path should be changed in the instruction memory module. This feature required infinitesimal changes to be made to the instruction memory module only.

## III. Schematic

Pipelined CPU Schematic



## IV. GitHub Repo

The following is a link to the github repo that integrates everything: https://github.com/mayammakram/RISCV32_CPU

## REFERENCES

[1] Andrew Waterman, Yunsup Lee, David Patterson, Krste Asanović, "The RISC-V Instruction Set Manual, Volume I: User-Level ISA Version 2.1", Technical Report UCB/EECS-2016-118, EECS Department, University of California, Berkeley, May 31, 2016.