# Pipelined CPU Implementation

## -Without IF/ID forwarding

b01902023 章莉 b01902066 楊耀源 b01902079 謝競容

---

## Implementation

### ALU.V

This module reads in the 32 bit inputs as operands and a 3 bit control input and do the arithmetic operations due to the 3 bit input.

| | AND | OR | ADD II ADDI | SUB | MUL |
|---|---|---|---|---|---|
| **Control bits** | 0 | 1 | 2 | 3 | 4 |

### CONTROL.V

This module assign all other control signal due to the corresponding operation code.

| | R Type | I Type | lw | sw | beq | Jump |
|---|---|---|---|---|---|---|
| **ALUSrc** | 1 | 0 | 0 | 0 | x | x |
| **ALUOp** | 2 | 0 | 0 | 0 | xx | xx |
| **RegDst** | 0 | 1 | 1 | x | x | x |
| **EX** | 12 | 1 | 1 | 000x | xxxx | xxxx |
| **MemWrite** | 0 | 0 | 0 | 1 | 0 | 0 |
| **MemRead** | 0 | 0 | 1 | 0 | 0 | 0 |
| **MEM** | 0 | 0 | 1 | 2 | 0 | 0 |
| **RegWrite** | 1 | 1 | 1 | 0 | 0 | 0 |
| **MemtoReg** | 0 | 0 | 1 | 0 | 0 | 0 |
| **WB** | 2 | 2 | 3 | 0 | 0 | 0 |
| **JumpCtrl** | 0 | 0 | 0 | 0 | 0 | 1 |
| **BrenchCtrl** | 0 | 0 | 0 | 0 | 1 | 0 |

### DATA_MEMORY.V

This module contains an initial 32 memory registers of size 1 byte. The data are stored in little endian order.

**FORWARD_UNIT.V**

This module controls the data forwarding mechanism, which only occurred while data hazard appears. Because data in EX stage will write over MEM stage, while forwarding is needed, and data in both stages are needed to be forwarded, we will forward the EX stage first.

**HAZARD_DETECTION_UNIT.V**

Sometimes while forwarding can't help the CPU will have to stall for a cycle, and this unit is implemented for this functionality. It checks whether it'll have to stall while lw instruction following addi, Rtype and branch.

---

# Problems Encountered and Solutions

**1. = VS. <=**

We've used <= in most of our code for implementation, in order to run faster. However, this causes a serious problem in Control.v where at first we assign value to internal registers with =, and then concat them together with <=. At last we've found out that changing all <= into = will work. We guess that <= and = can't use together which will cause unknown errors.

**2. CPU.V**

Because CPU.v is the first module implemented, without careful thinking many wires are connected wrongly, and because that we take the code of last homework of a prototype, some value that should be changed isn't modified. We've spent many time debugging on this single file.

**3. NAMING OF MODULES**

Without thinking thoroughly through the whole procedure of the CPU, it's hard to give a nice easy-understood name to each input and outputs. We've been changing the name frequently after knowing what is really going on inside the modules while implementing and debugging.

**4. DIFFICULT TO DEBUG**

Because that pipelined CPU contains four instructions inside a clock, we will have to spend up many working memory in our brain in order to trace all the functions working well. However, it's also through debugging which makes us getting more clear of what each module in pipelined CPU is doing.

**5. GROUP CORPORATION**

We uses github to share our code together, sometimes we forget to pull and implement the same thing twice. Also, the most difficult thing is that everyone implement their components, sometime it is hard to debug without going through other people's code and we certainly forget to do this well which cause a lot of codes. Maybe next time we

should setup detailed specification for interfaces between modules before everyone starts to implement their module.

---

## Team Work

**章莉 B01902023**

Connect modules together and setup interfaces for each modules.
Implement Hazard_Detection_Unit, Signed_Extend , ALU, Forward_Unit.
Write the report.

**謝競容 B01902079**

Implemented modules: Mux32, AND, Flush_MUX, Equal, ID_EX,ForwardMUX, MUX5
Encode Fibonacci code.

**楊燿源 B01902066**

Implemented modules: IF_ID, Shift_Left2_Concat, Control, Shift_Left2, Registers, ALU_Control, Data_Memory, MEM_WB, EX_MEM

---

## GitHub Repo

https://github.com/lunaivory/ComputerArchitecture/tree/master/project1