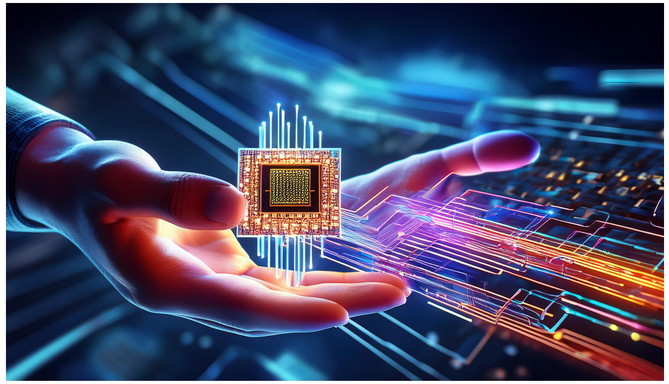
25.10.24 17:51

CSCI 250
Introduction to Computer Organisation
Lecture 3: CPU Architecture V



Jetic Gū

2024 Fall Semester (S3)

Overview

- Architecture: von Neumann
- Textbook: -
- Core Ideas:
    1. Case Study: a CPU without Pipeline, MIPS

A Case Study: MIPS

Why MIPS?

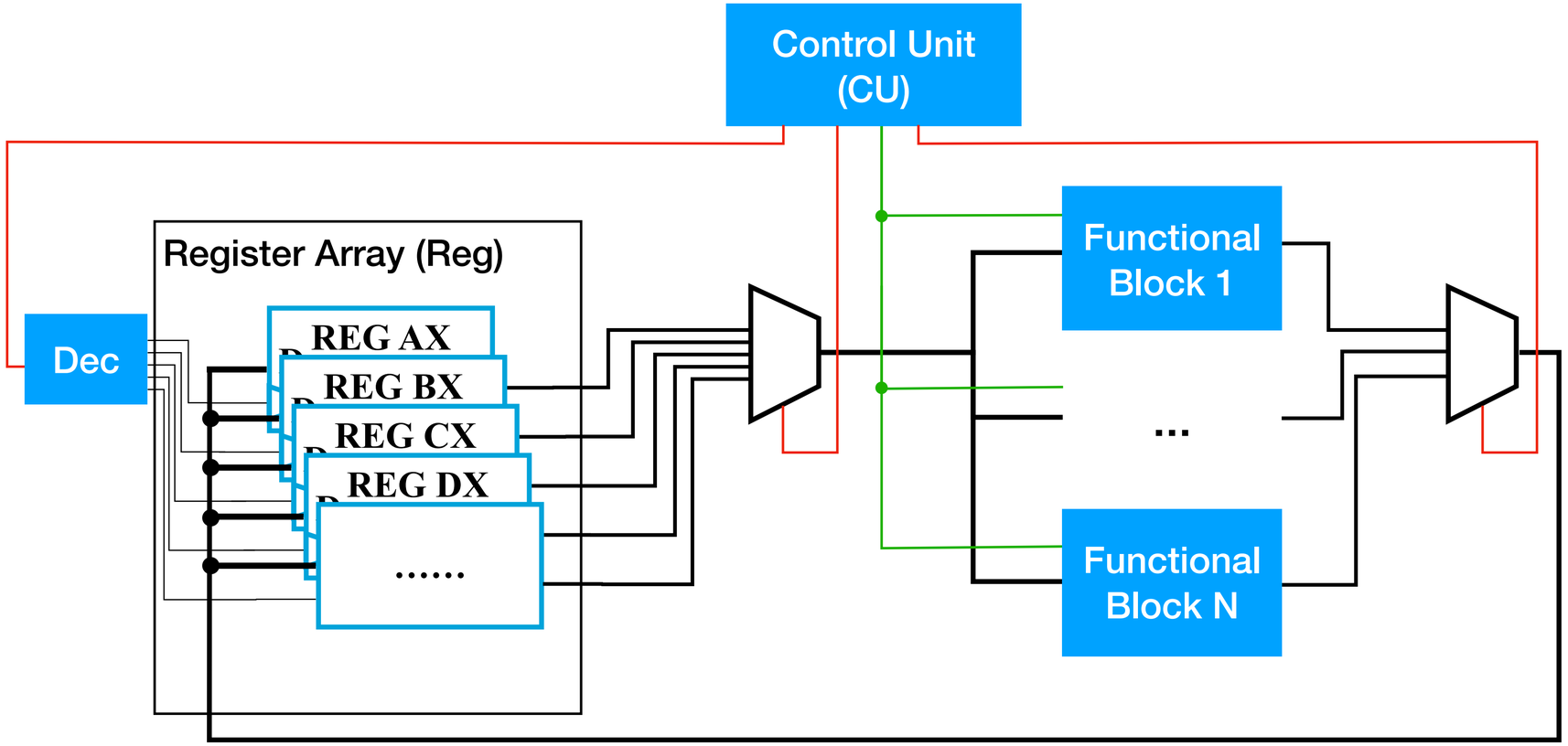
- A common MIPS CPU example is one with a separate instruction memory and main data memory
  - This makes it easier to implement the CPU without involving pipeline
  - It still works as a fully functional CPU, just not as sophisticated as our modern CPUs
  - After we learned how this CPU works, we can then understand better how a pipelined modern CPU like ARM works

Concept

Components

- A Programme Counter that is separate from the register array
  This is a bit weird for ARM, but for MIPS this is OK
- Instruction Memory can use normal SRAM module, but read only. Address goes in, instruction comes out
- ALU and adder, as well as other necessary components

Datapath from CSCI150

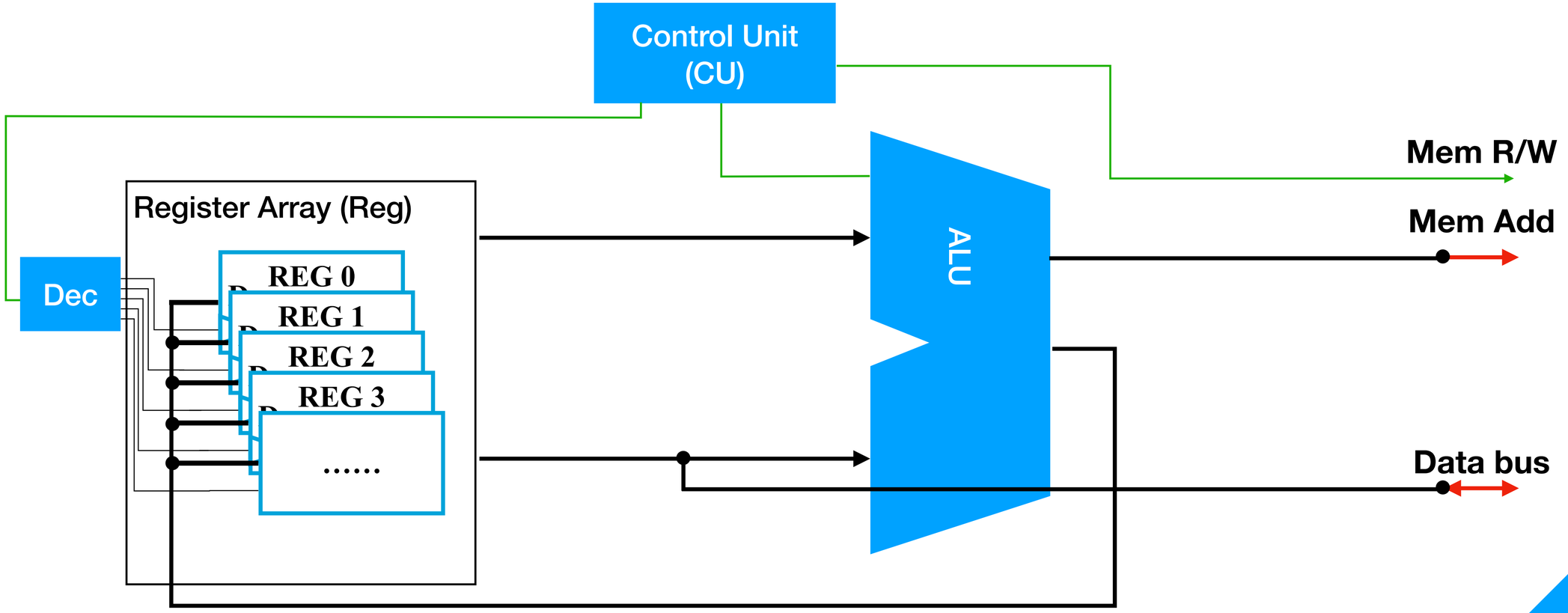


- There's no memory, we didn't discuss the control unit. Well, now we will

Review

P1
MIPS

Step 1: Adding Memory (Main Data Bus)

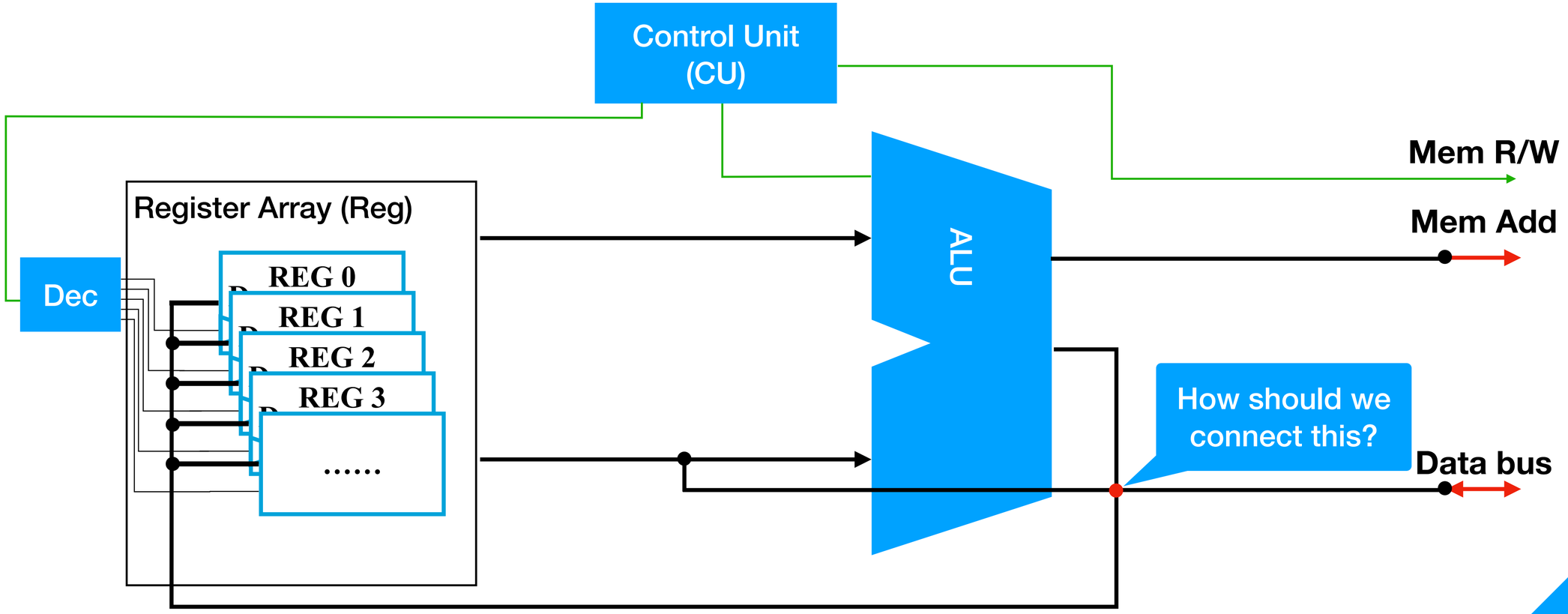


- ALU should be able to output to the main data bus: Address, Data

Technical

P1
MIPS

Step 1: Adding Memory (Main Data Bus)



- Register Array should be able to accept data from the main data bus

Technical

P1
MIPS

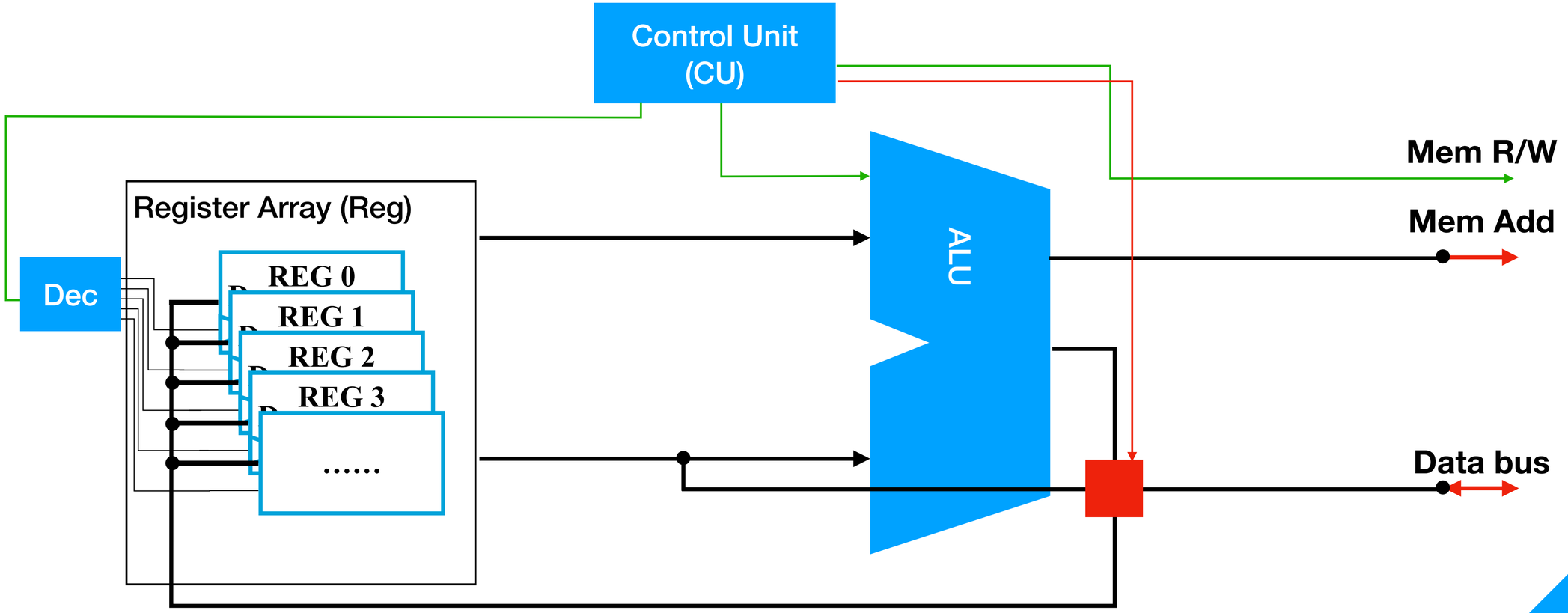
Step 1: Adding Memory (Main Data Bus)

- Bidirectional Data Bus
    - You can activate data going from CPU to memory
    - You can activate data going from memory to CPU
    - What if you want it just OFF? (Open circuit?)

Technical

P1
MIPS

Step 1: Adding Memory (Main Data Bus)

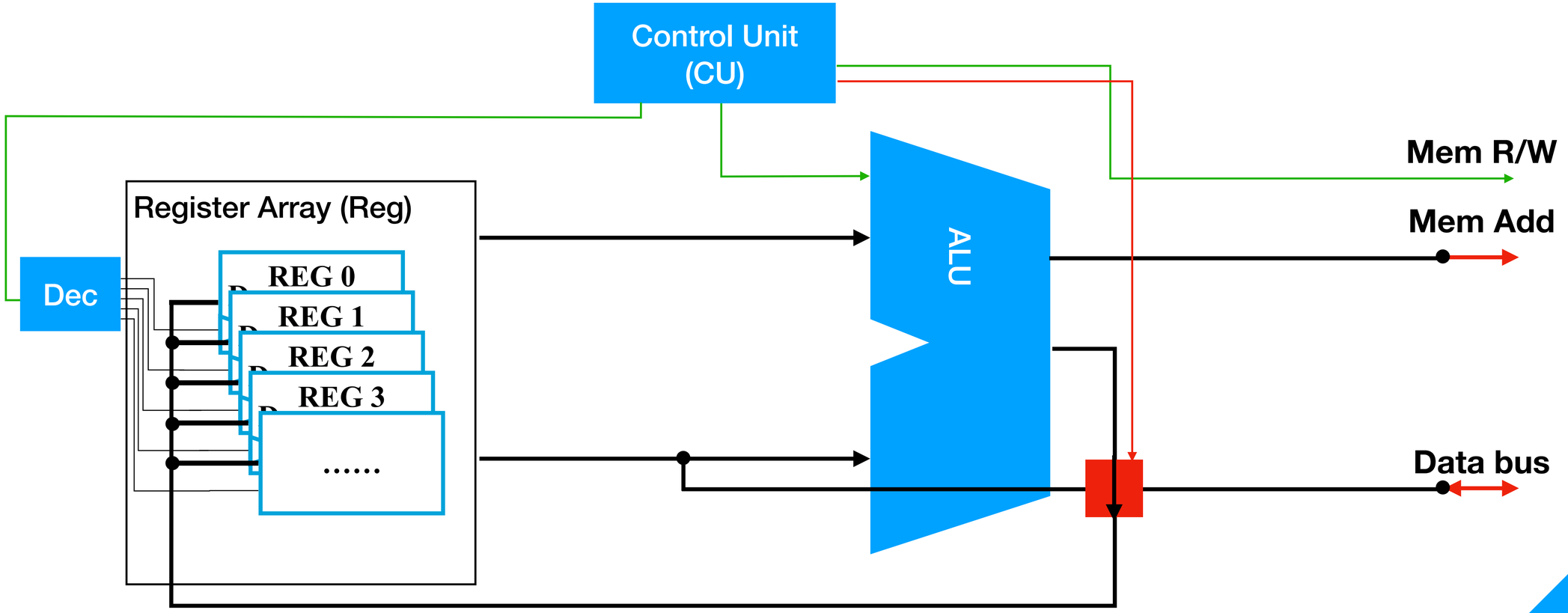


- Register Array should be able to accept data from the main data bus need to ensure your data is going in the right direction

Technical

P1
MIPS

Step 1: Adding Memory (Main Data Bus)

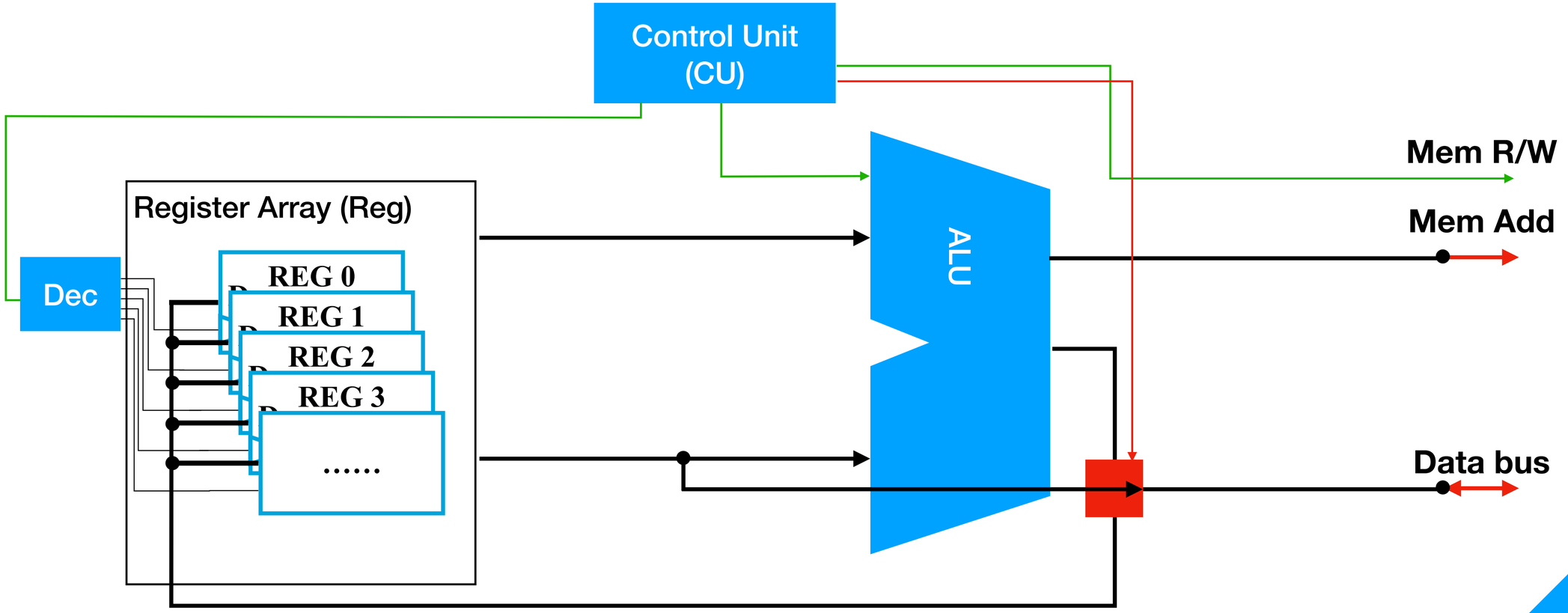


1. ALU -> RegArray

Technical

P1
MIPS

Step 1: Adding Memory (Main Data Bus)

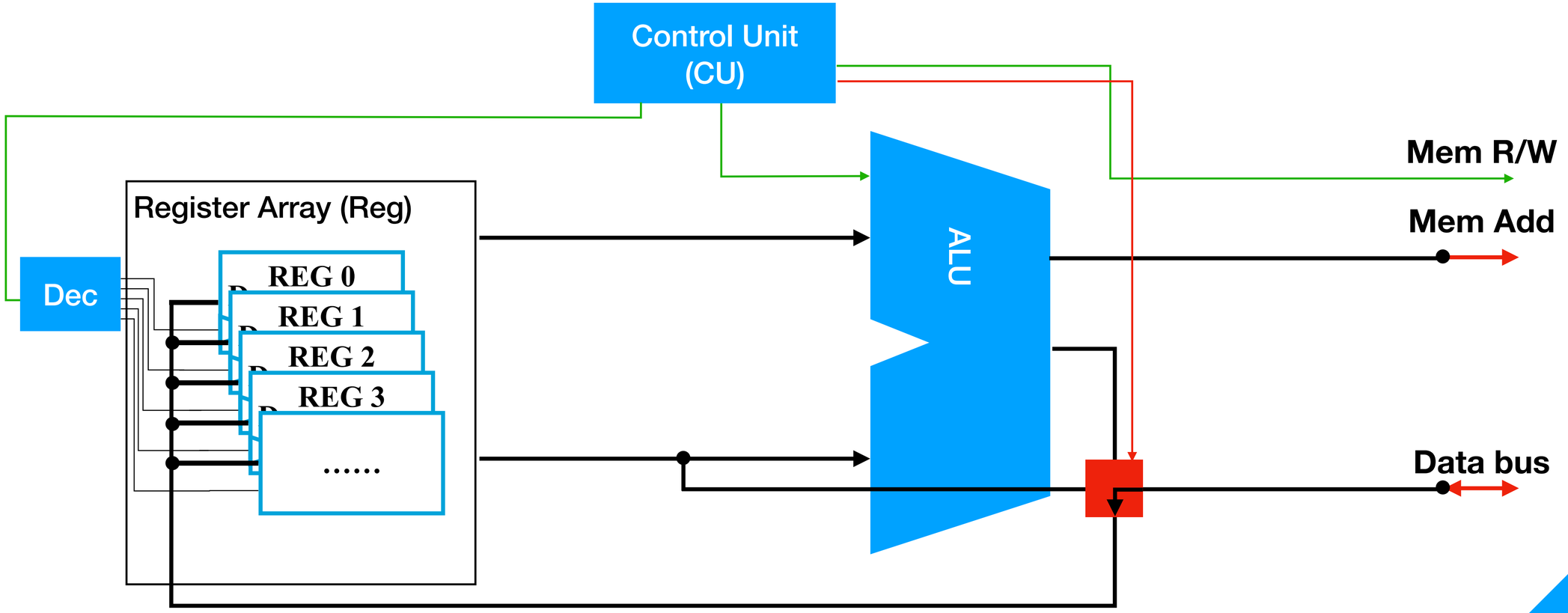


2. Reg2 -> Data Bus

Technical

P1
MIPS

Step 1: Adding Memory (Main Data Bus)

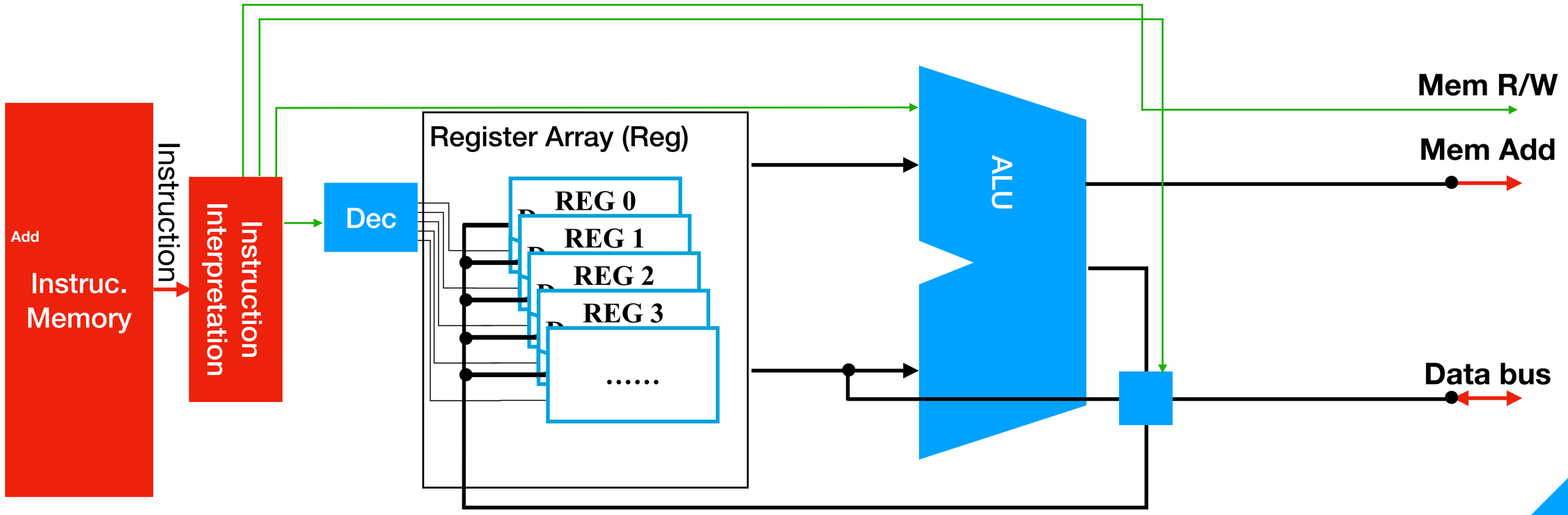


3. Data Bus -> RegArray

Technical

P1
MIPS

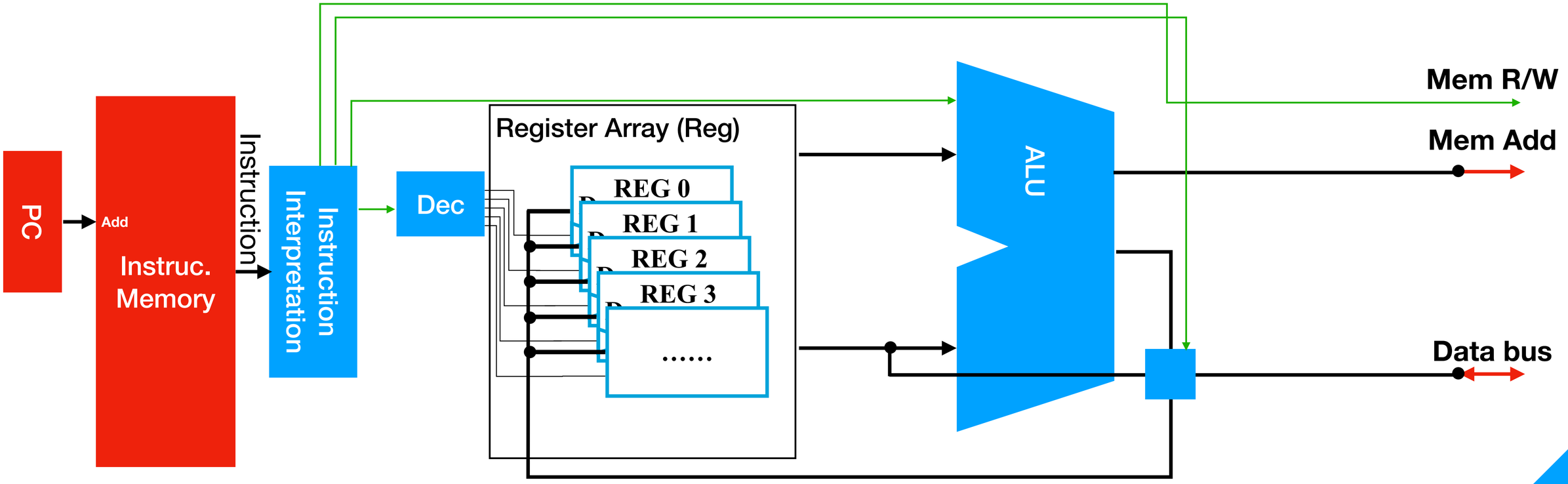
Step 2: Adding Instruction Memory



- Your control unit here is broken into several pieces

P1
MIPS

Step 2: Adding Instruction Memory

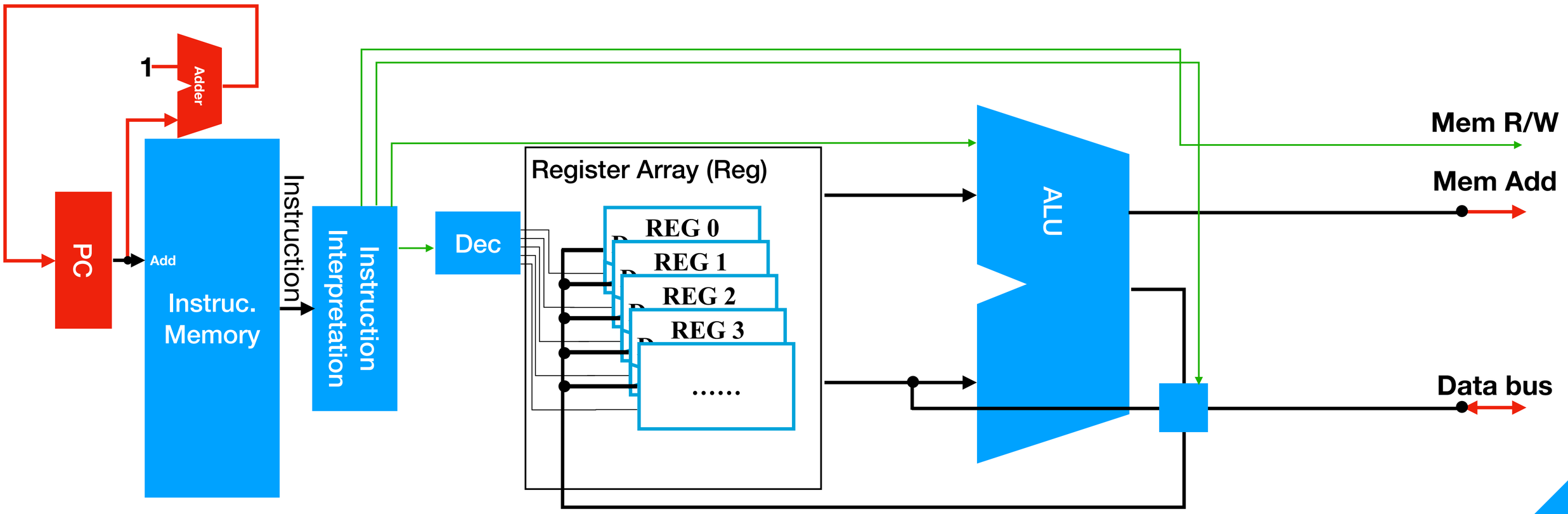


- Programme Counter provides the address of the instruction to retrieve

Technical

P1
MIPS

Step 2: Adding Instruction Memory



- Programme Counter increases its value for every CLK, moving to the next instruction to execute

Technical

P1
MIPS

Branching
MIPS: J type instructions

- As we discussed before, you should be able to change the address of the next instruction

- This can happen when you perform a function call, or when you use a for loop, or even an if condition

- **Branching**
  Changes PC value to an immediate, or from a register, or original PC with **Offset**

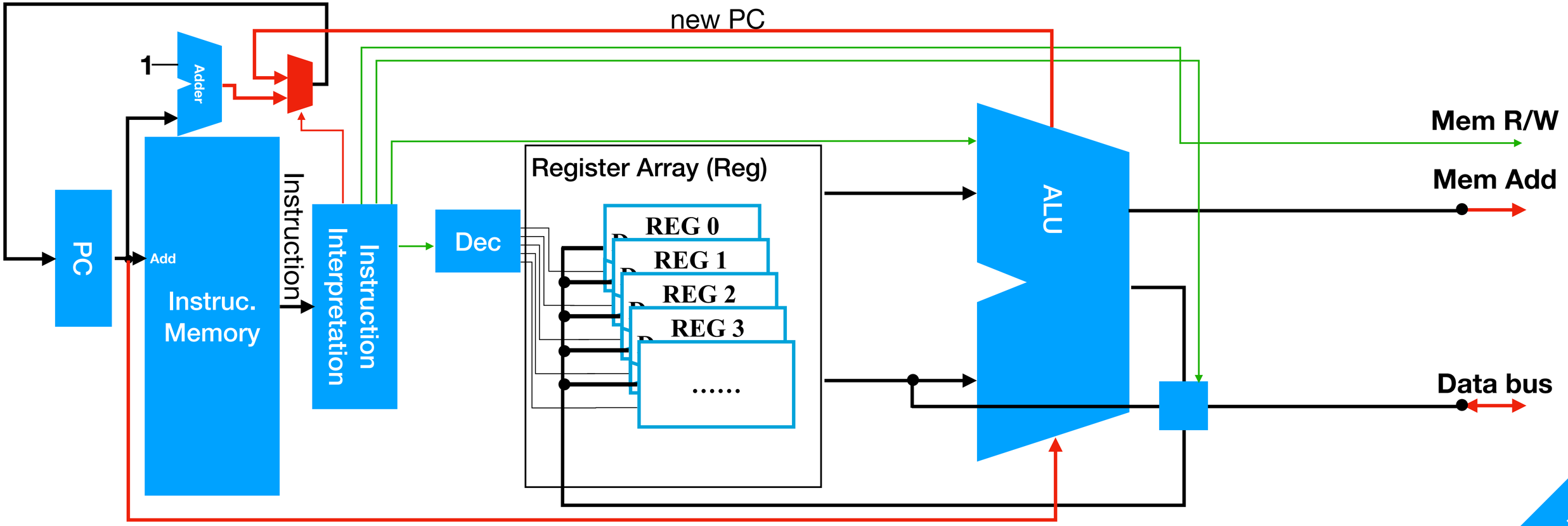
- **Conditional Branching** is triggered when certain conditions are met

    - For MIPS, an example is **beq**:
      compares two registers value, if they are the same, **PC <= some other value**

Concept

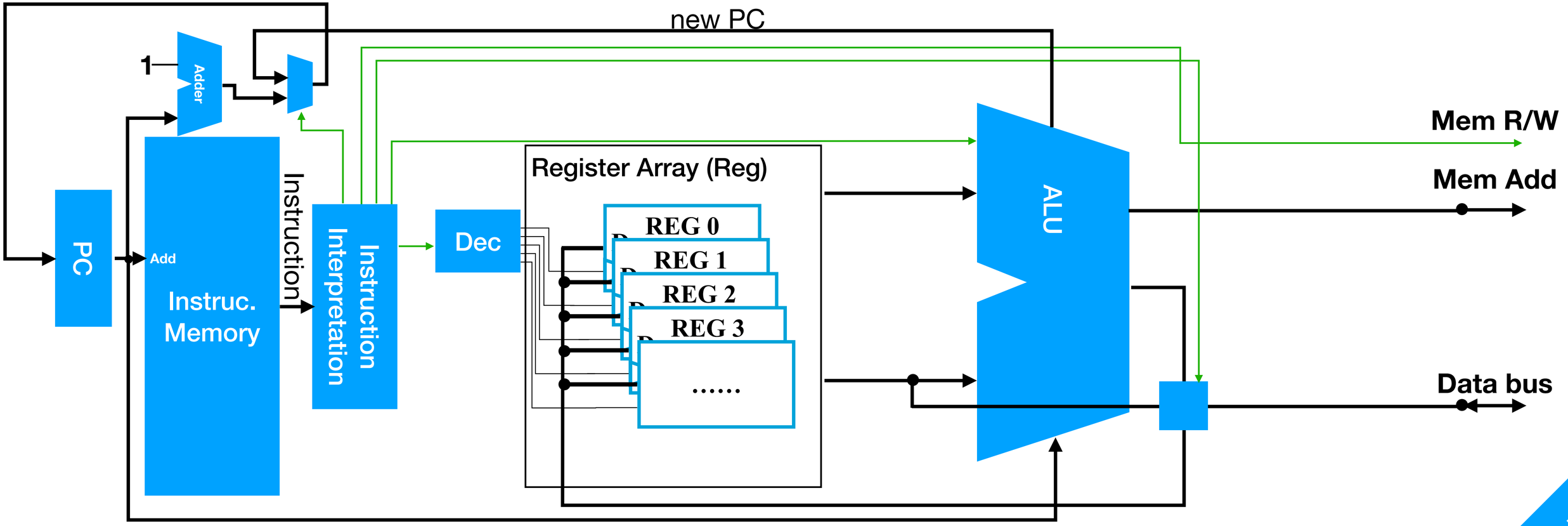
P1
MIPS

Step 3: Adding Branching



- Programme Counter increases its value for every CLK, moving to the next instruction to execute

MIPS Example CPU



- Programme Counter increases its value for every CLK, moving to the next instruction to execute

What is still missing?

- MIPS retrieval of instructions is instantaneous, and is executed within the same CLK cycle
    - This is not realistic in x64 nor ARM, where a single instruction coming from memory/cache will always suffer from delay
- All MIPS instructions finish within one CLK cycle
    - This is also not realistic in x64 nor ARM
- This MIPS CPU has separate instruction memory and main memory
    - In reality, ARM/x64 maintains a small list of instructions to execute, but make no mistake, there's only **ONE** memory
- This MIPS doesn't have Interrupt
    - This is unlikely in most modern CPUs, not just ARM and x64. There should always be interrupt

Concept

P1
MIPS

What is still missing?

- How easy is it to merge the instruction memory and main memory into a single one? Is it possible within the limits of what we've discussed? What would be needed?

Future