Computer Architecture

Last updated 5/17/19

Architecture

- General Purpose Processor
  - User Programmable
    - Intended to run end user selected programs
  - Application Independent
    - PowerPoint, Chrome, Twitter, Angry birds, …
- Embedded Processor
  - Not User Programmable
    - Programmed by manufacturer
  - Application Driven
    - Non-smart phone, appliances, missiles, automobiles, …
    - Very wide and very deep applications profile

© tj

Architecture

- General Purpose Processor
  - Key Characteristics
    - 32/64 bit operations
    - Support non-real-time/time-sharing operating systems
    - Support complex memory systems
      - Multi-level cache
      - dRAM
      - Virtual memory
    - Support DMA-driven I/O
    - Complex CPU structures
      - Pipelining
      - Superscalar execution
      - Out-of-order execution (OOO)
      - Floating Point HW

© tj

Architecture

- General Purpose Processor
  - Examples
    - ARM 7, 9, Cortex A8, A9,A15
    - Intel Pentiums, Ix, Core ix…
    - AMD Phenom, Athleron, Opteron
    - Apple A4, A5, A6
    - TI OMAPs

© tj

Architecture

- Embedded Processor
  - Key Characteristics
    - 4/8/16/32 bit operations
    - Support real-time operating systems
    - Relatively simple memory systems
    - Memory mapped I/O
    - Simple CPU structures
      - Few registers
      - Limited Instructions
    - Support for multiple I/O schemes
    - Wide range of peripheral support
      - A/D – D/A
      - Sensors
      - Extensive interrupt support

© tj

Architecture

- Embedded Processor
  - Examples
    - Motorola/Freescale 68K, HC11, HCS12
    - ARM Cortex Rx, Mx
    - Atmel AVR

© tj

Architecture

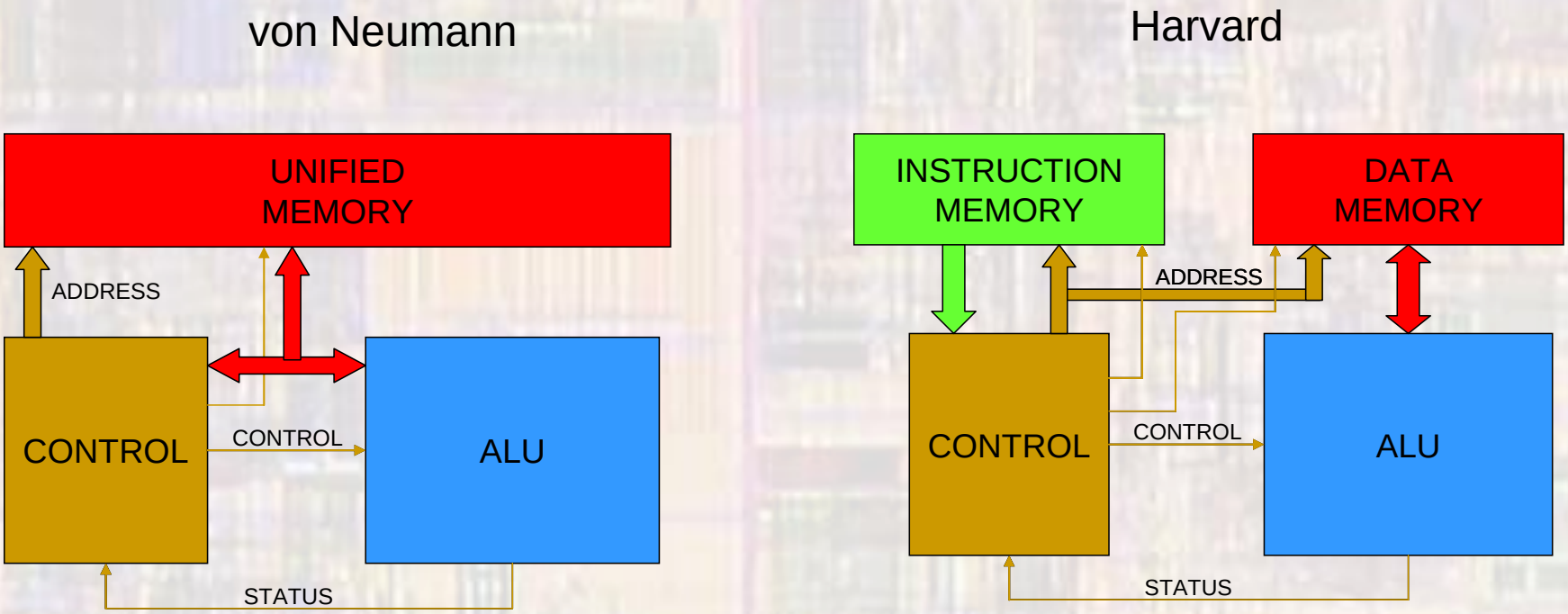
Instruction Sets

- CISC – Complex Instruction Set Computer
  - Name didn't even exist until RISC was defined
  - Used in most processors until about 1980
  - One instruction holds multiple actions
    - Load data from location, add, write data to new location
  - Many times the instructions were designed to emulate high level language constructs
- RISC – Reduced Instruction Set Computer
  - Developed in the '80s
  - Most prevalent architecture today
  - Sometimes called a load/store architecture
  - Instructions are simple
    - Load data from location
    - Add
    - Store data to location
- RISC dominates today
  - Much easier to take advantage of advanced structures like Pipelining, Superscalar, OoO

© tj

Architecture

- Memory Bus Structure



© tj

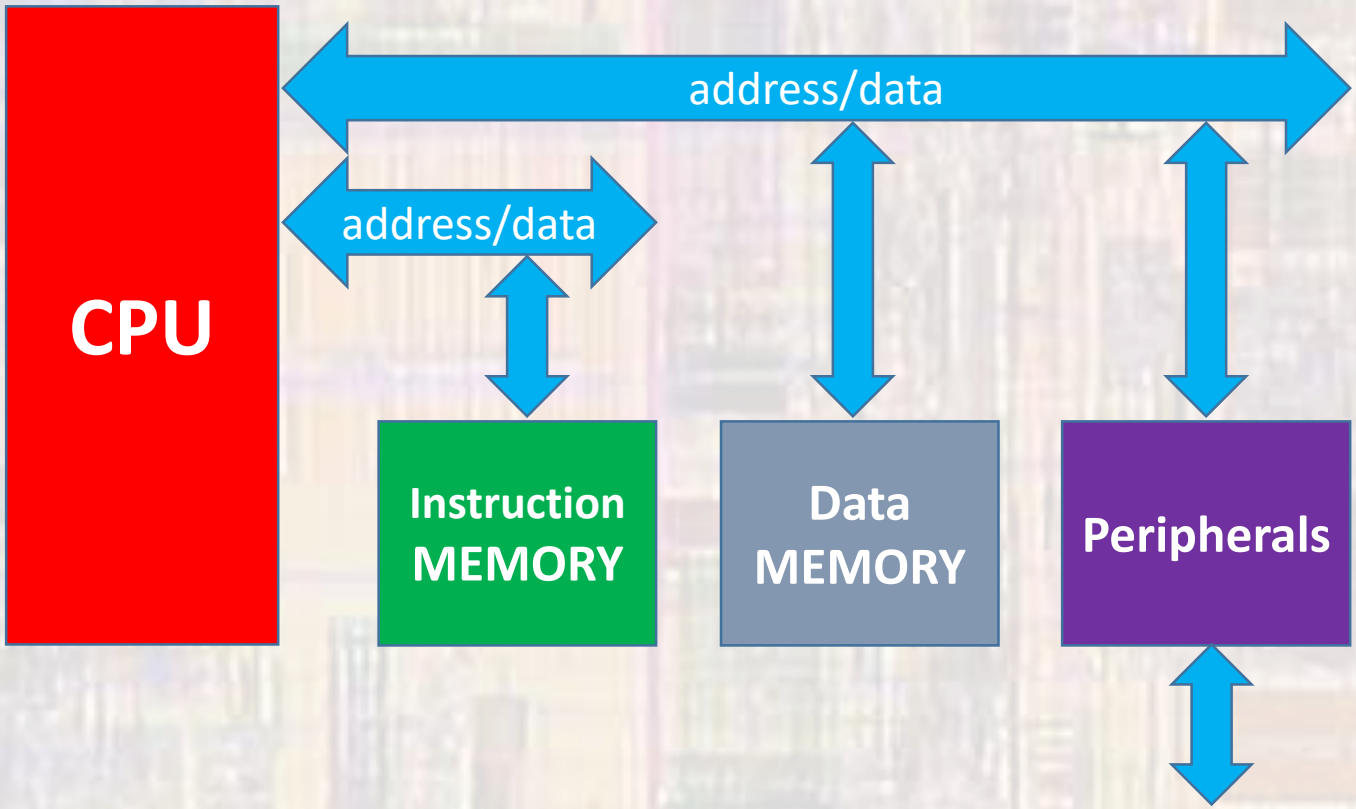
Architecture

- RISC Instruction set
  - 2 basic types of instructions
    - Register based instructions
    - Memory instructions
  - Register Instructions
    - Only require access to the internal registers
      - Arithmetic
      - Logical
      - Control
  - Memory Operations
    - Read or write to memory

© tj

Program Execution

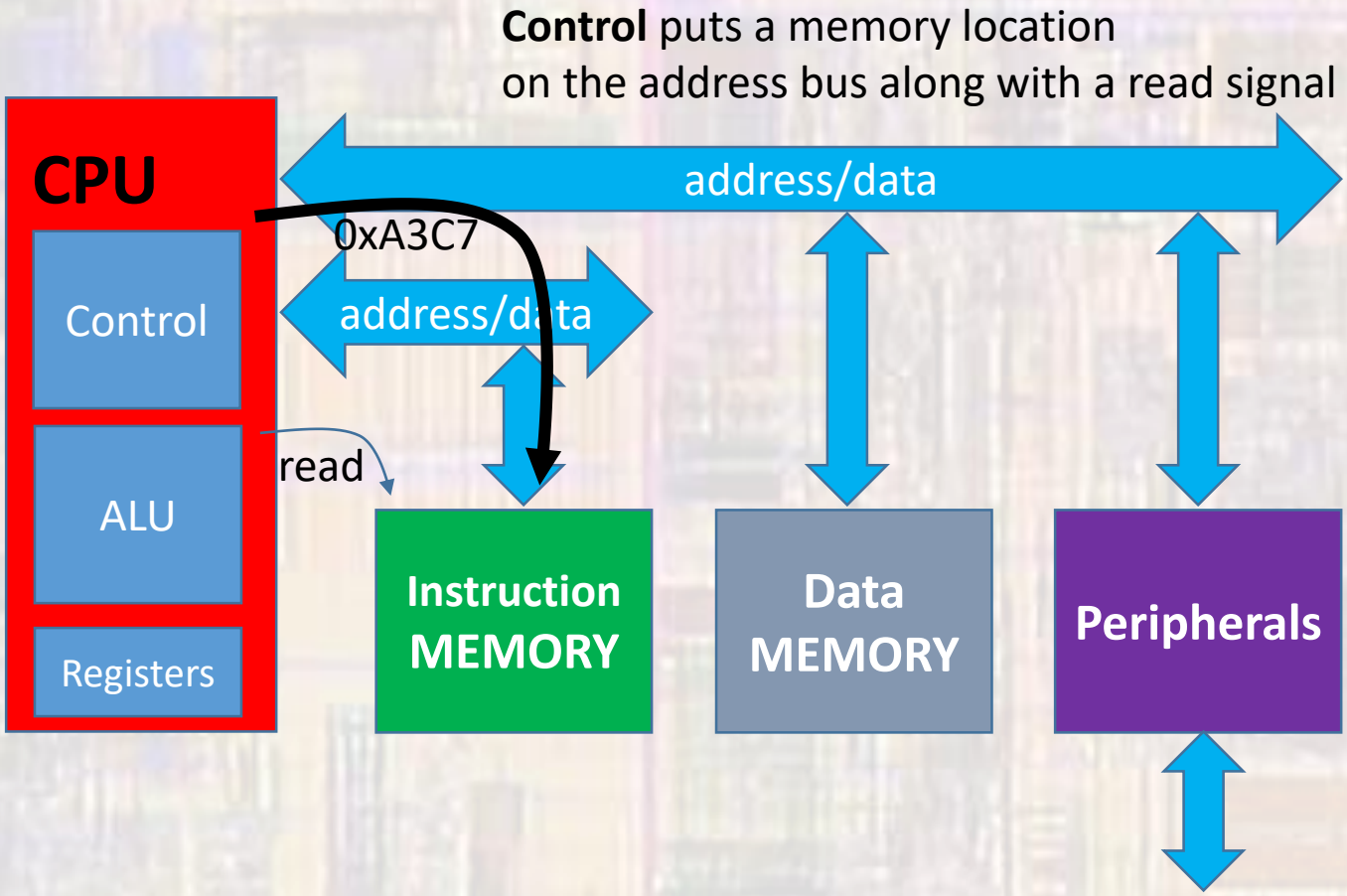
- Simplified Block Diagram



 © tj

Program Execution

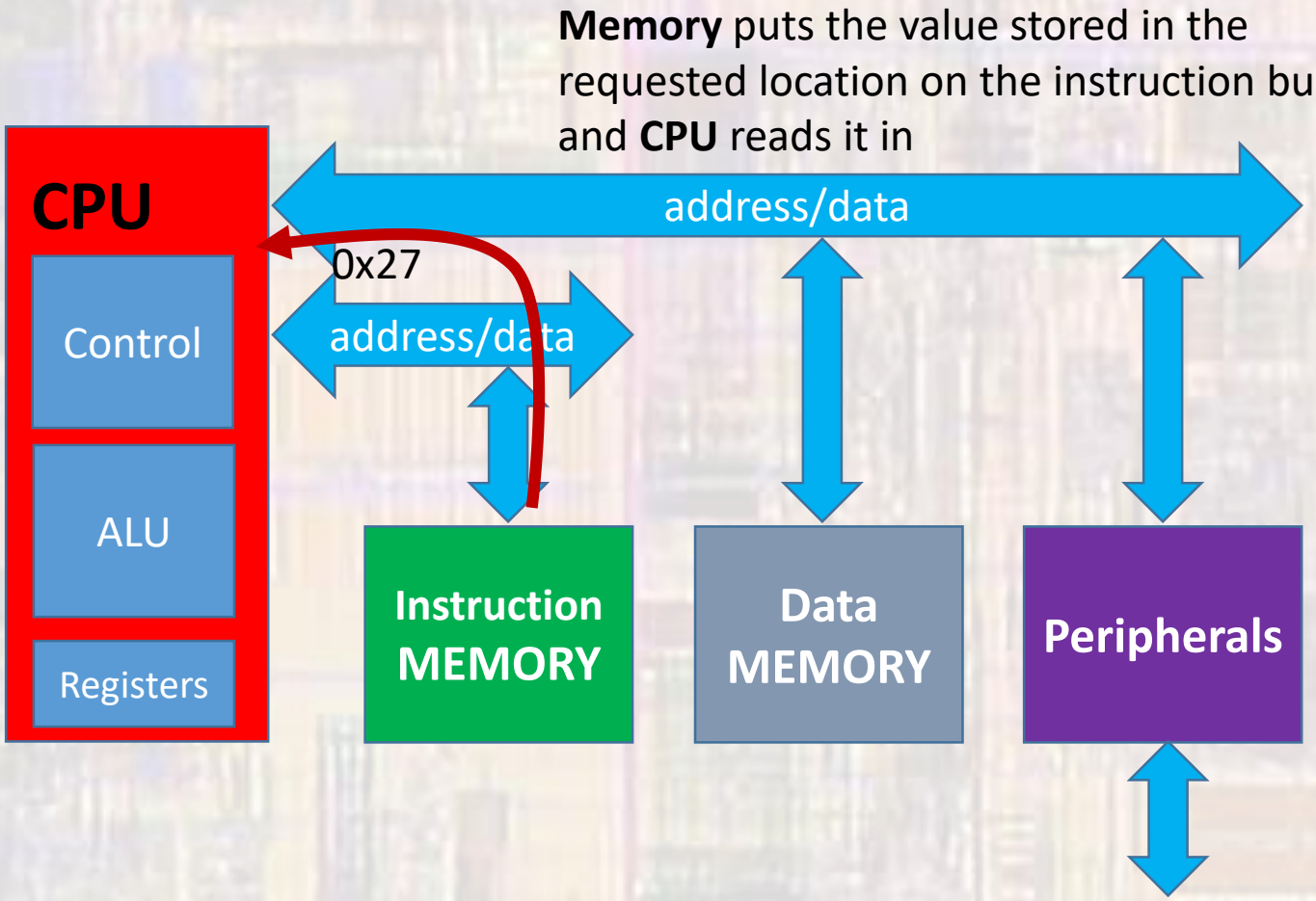
- Request Instruction (fetch)



 © tj

Program Execution

- Request Instruction (fetch)

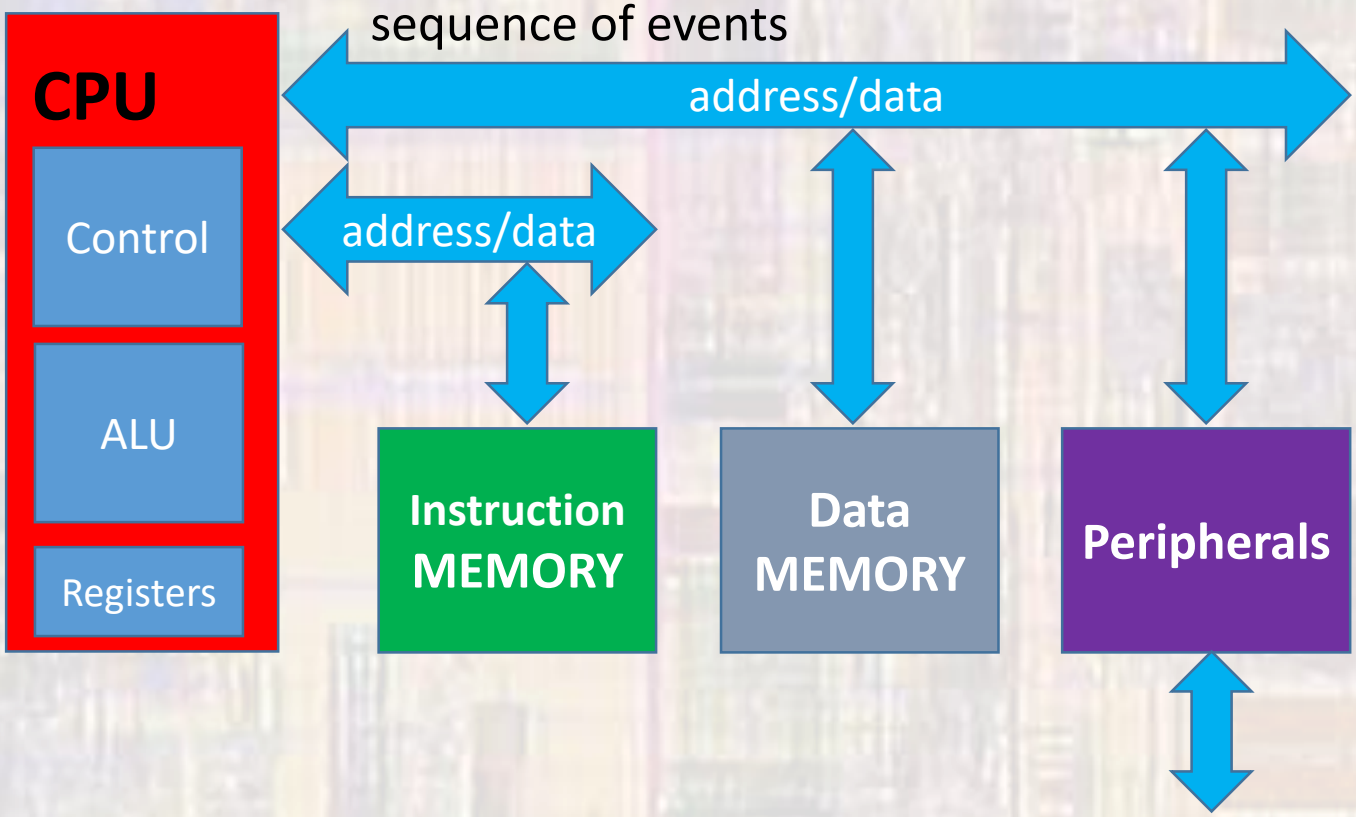


 © tj

Program Execution

- Decode Instruction (decode)

Control decodes the word returned by the memory and prepares to execute a pre-defined sequence of events

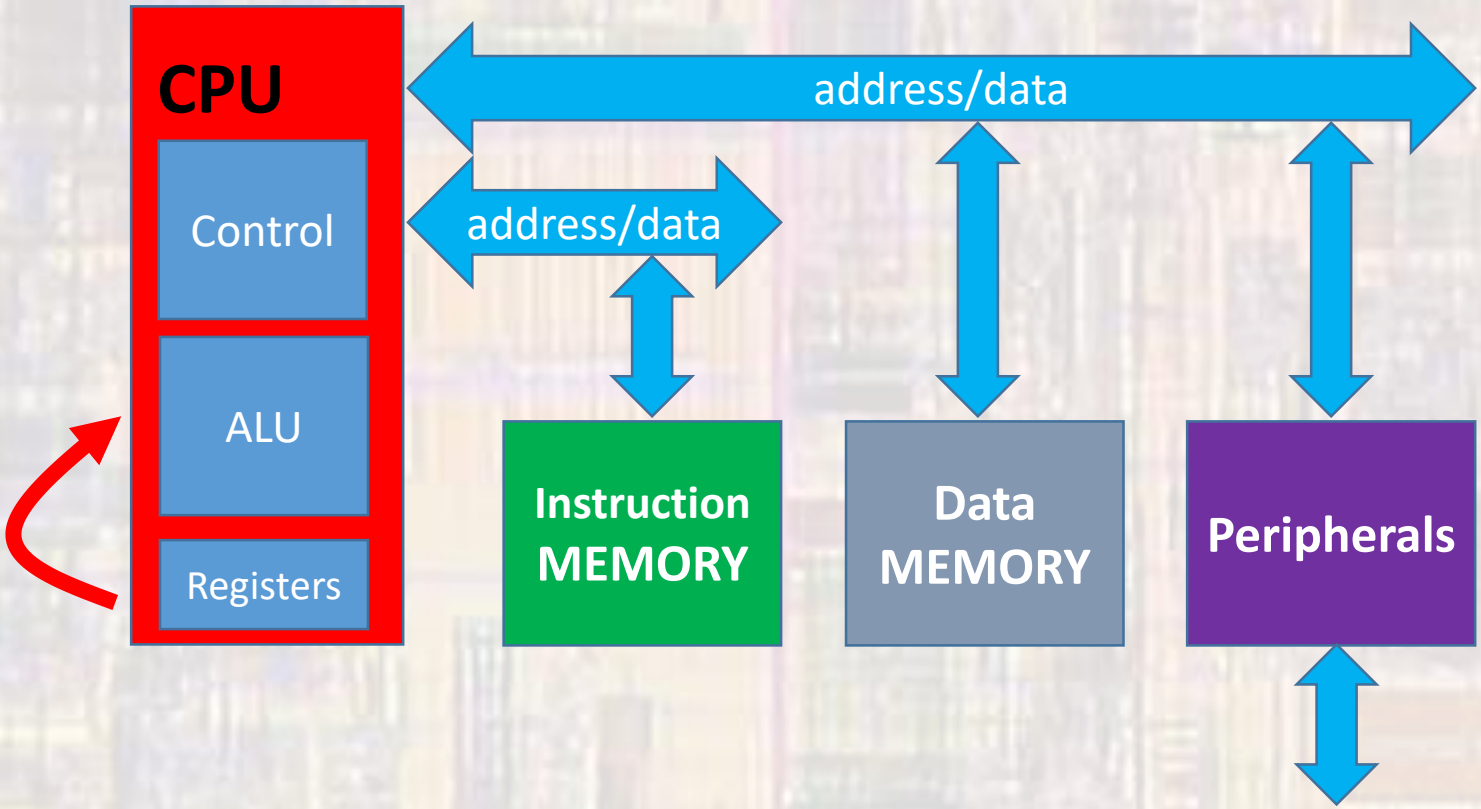


 © tj

Program Execution

- Execute Instruction (execute)

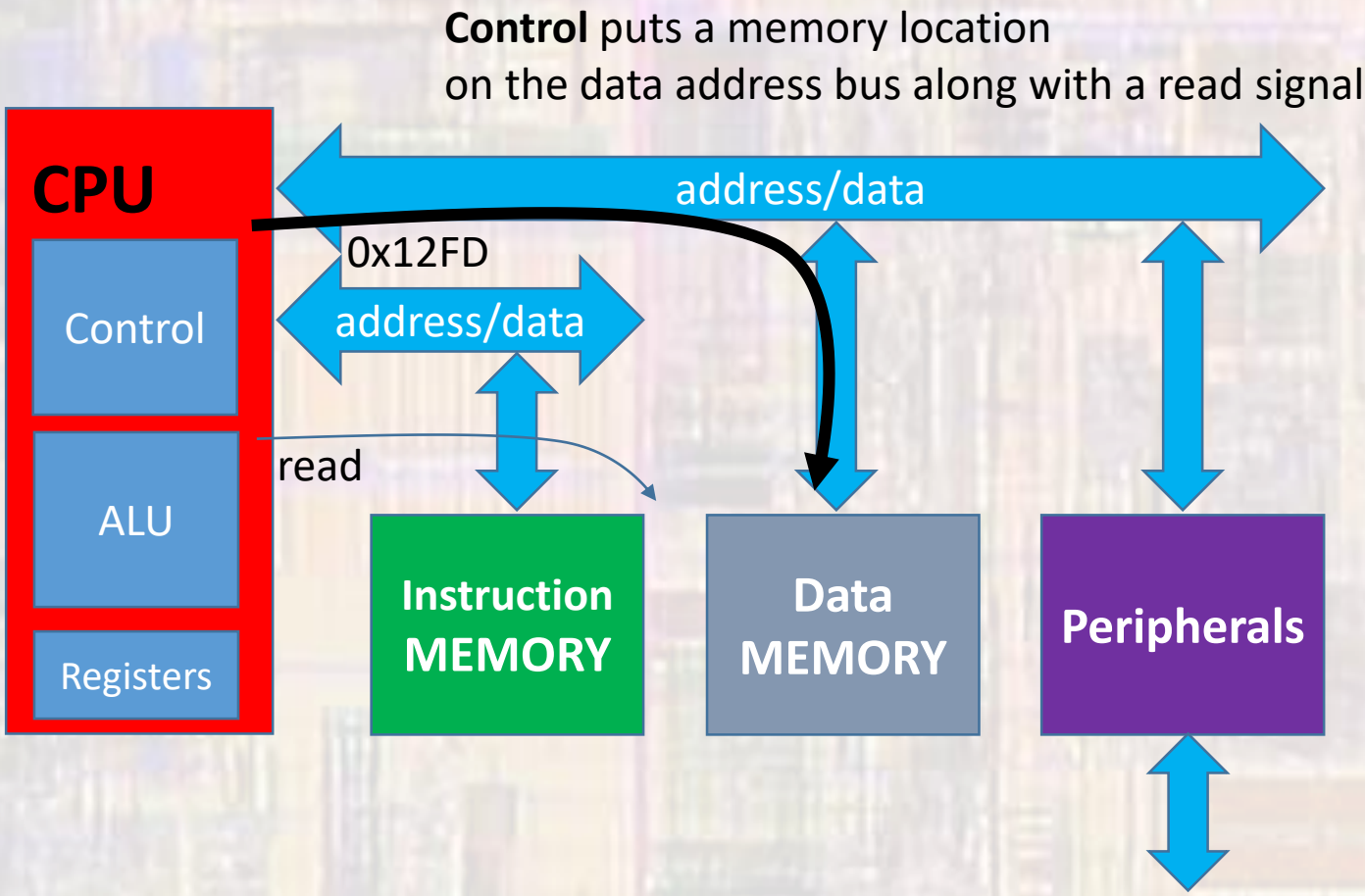
CPU executes the fetched instruction (pre-defined sequence of events) using data from the registers



 © tj

Program Execution

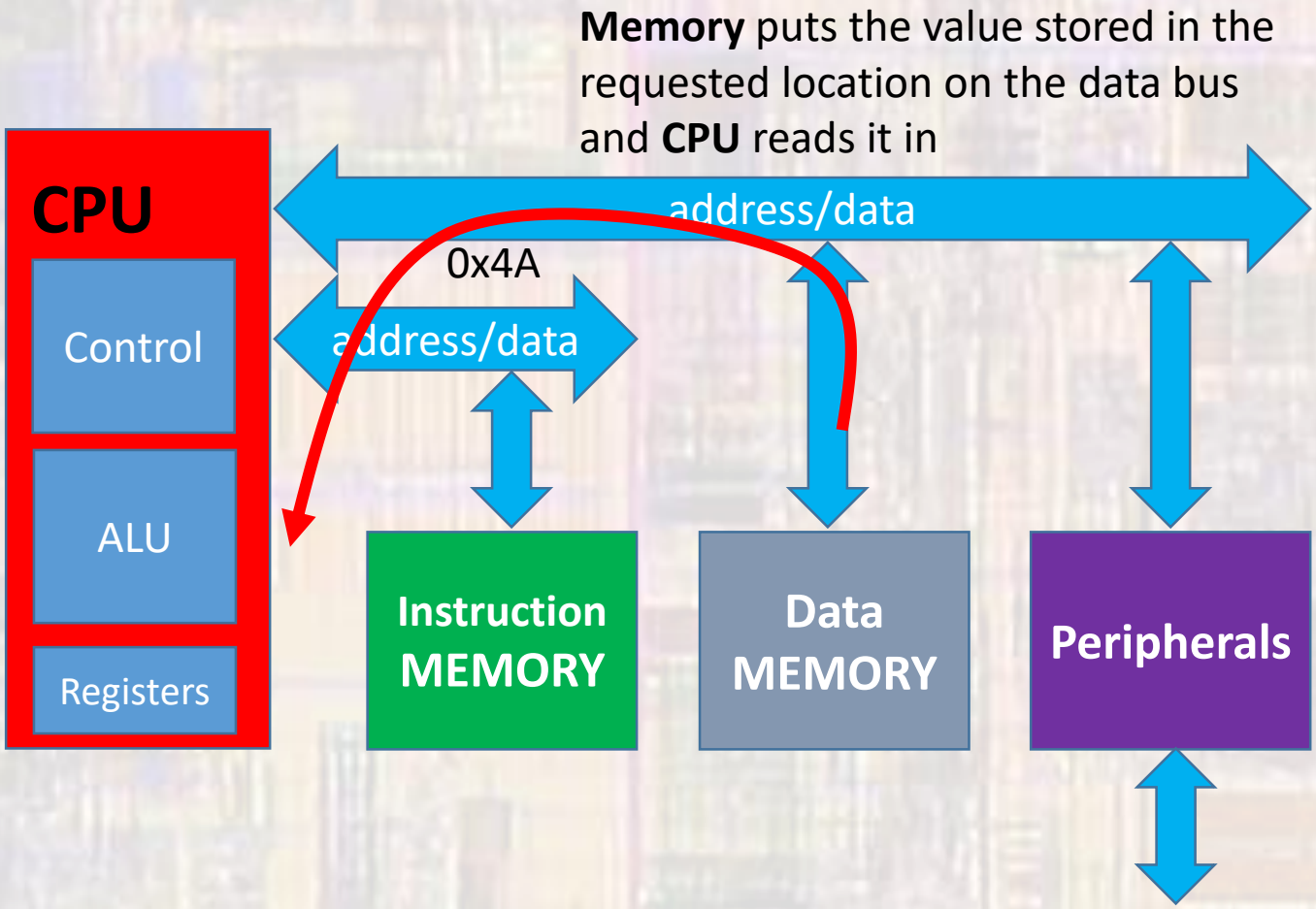
- Read data from memory (mem - Load)



 © tj

Program Execution

- Read data from memory (mem - Load)

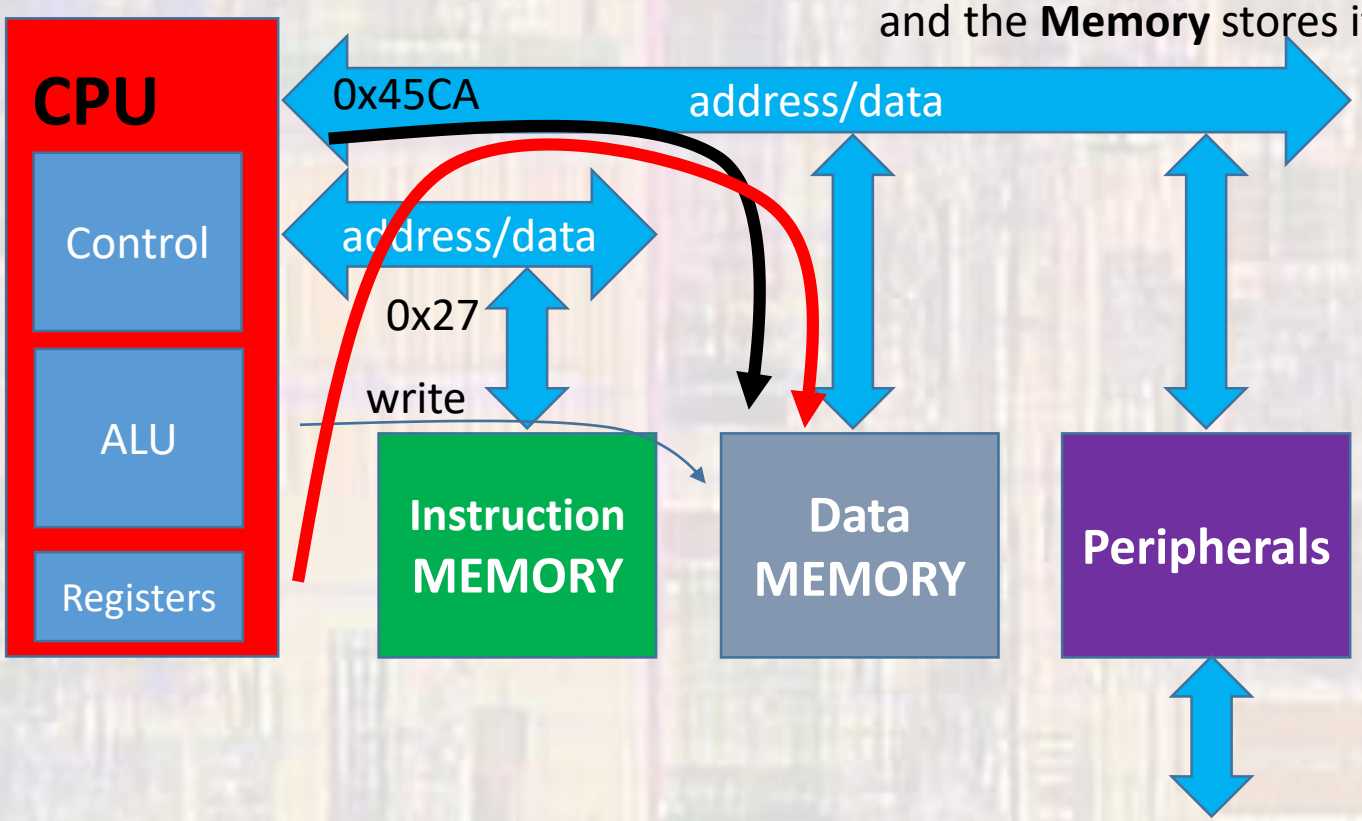


 © tj

Program Execution

- Write data to memory (Store)

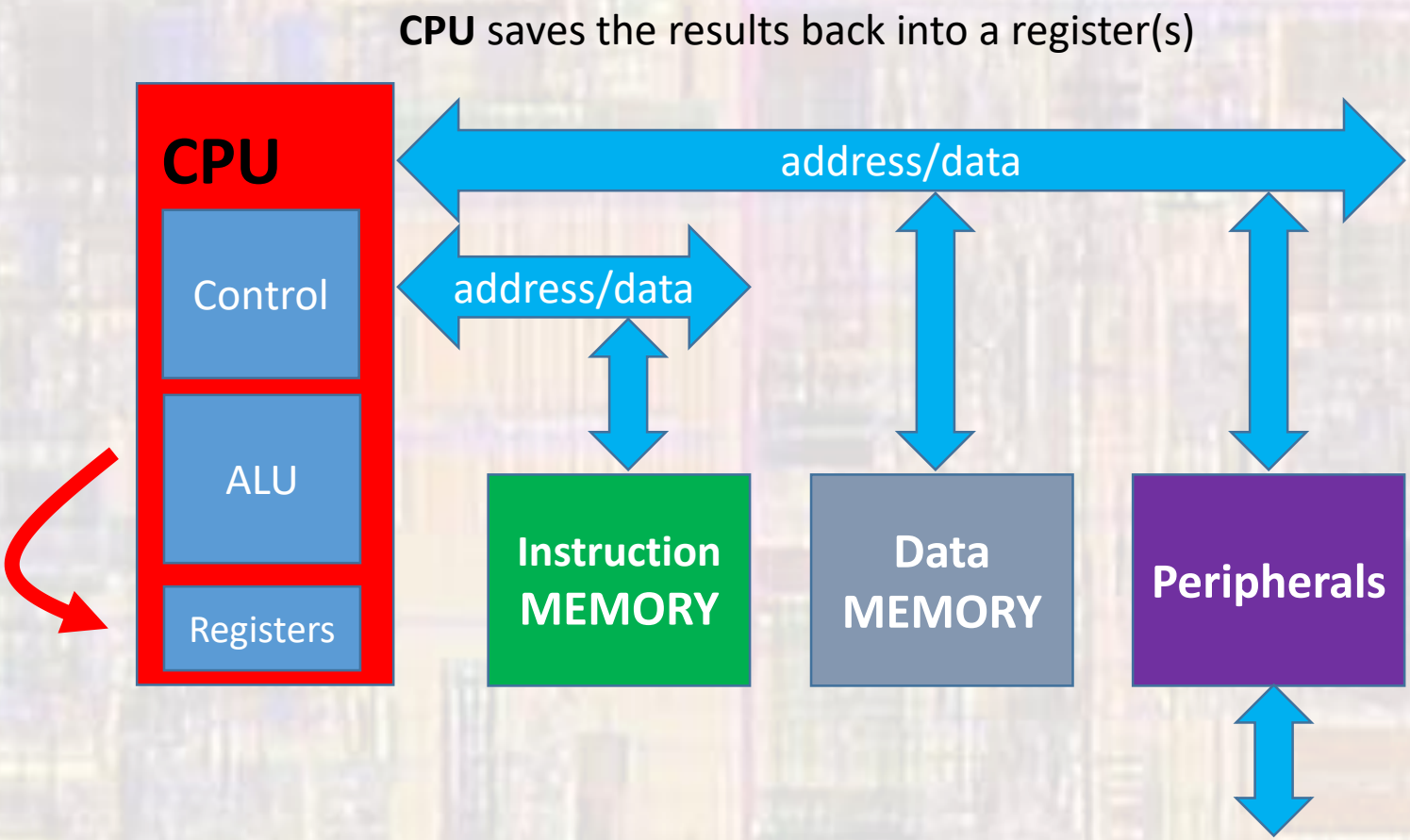
Control puts the location to store the result on the address bus and the value on the data bus and the **Memory** stores it



© tj

Program Execution

- Save the result (writeback)



 © tj

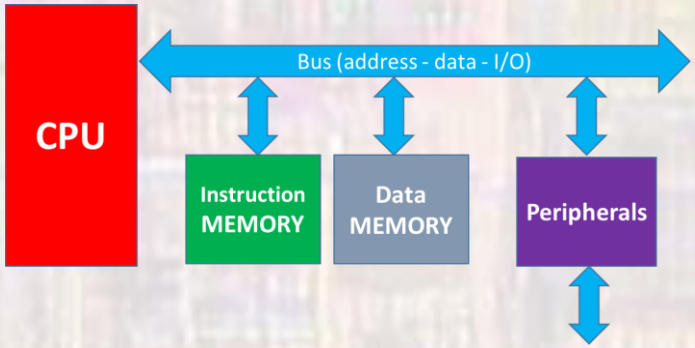
Program Execution

- 5 Stages of Instruction Execution
  - Fetch (IF)
  - Decode / Register Access (ID)
  - Execute (EX)
  - Memory Access (MEM)
  - Write Back (WB)

© tj

Program Execution

- Instruction Sequencing

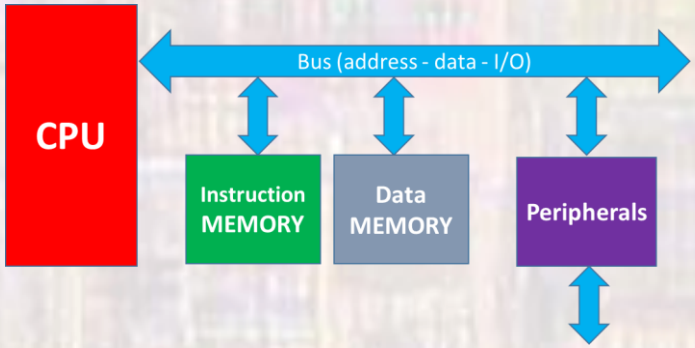


  - Program Counter (PC)
    - Register that holds the NEXT instruction memory location to be fetched
    - Provides the address for the instruction memory read
    - Typically the register is incremented each clock cycle
      - Incremented by the size of an instruction
      - e.g. for a 32 bit instruction word the PC would be incremented by 4

© tj

Program Execution

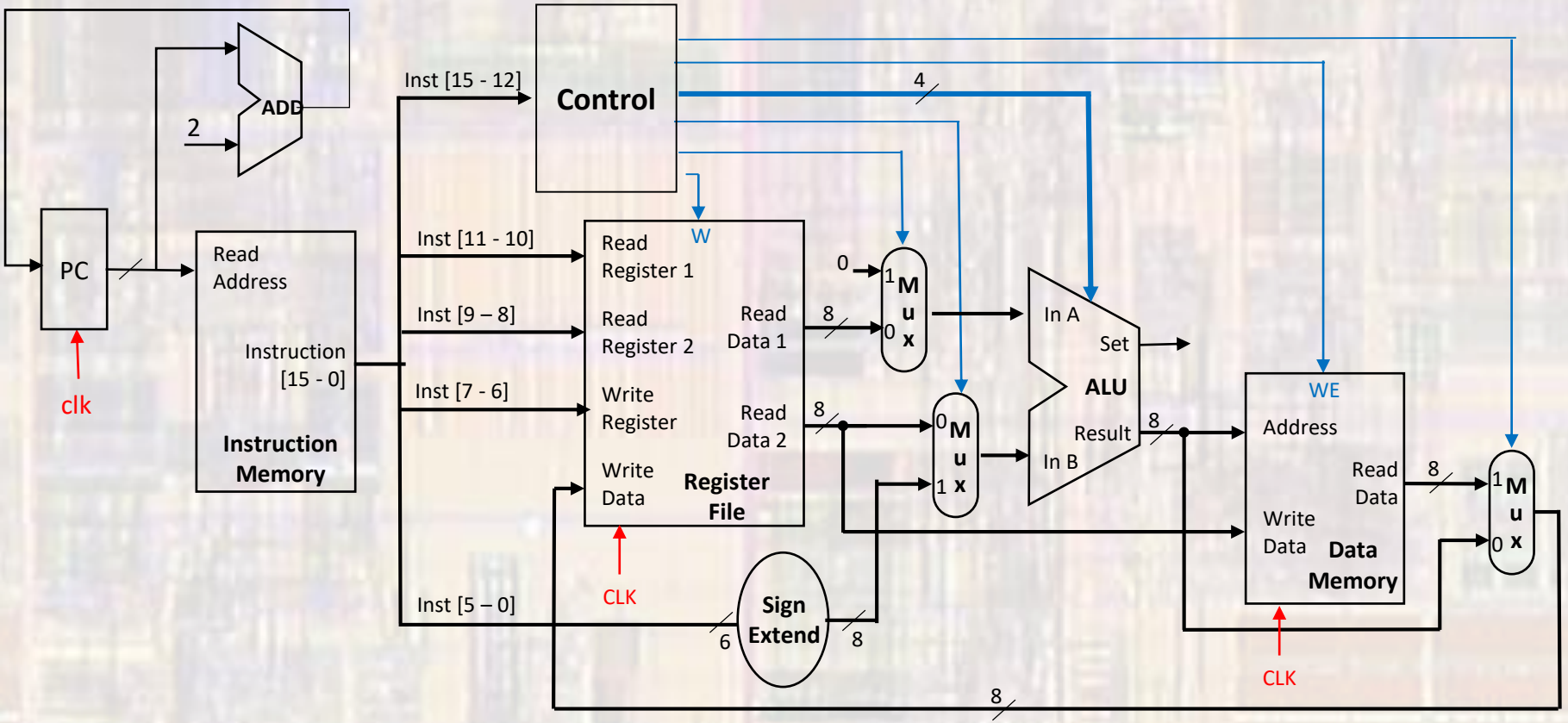
- Instruction Sequencing
  - Program control
    - Linear flow – increment PC normally
    - Function call
      - Store the “planned” next instruction address somewhere
      - Place the first instruction address for the function in the PC
      - Execute linearly in the function until done
      - Restore the “planned” next instruction address



 © tj

Simple Data Path

- Single Cycle Data path



 © tj