Mini-MIPS CPU
COMP 273 Project

Due: April 14, 2024, on myCourses

Submission instructions

You can do this project on your own or in a team of 2 students. If you are forming a team of 3, there are additional requirements. All work must be your own and must be submitted to myCourses. Include your name(s) and student number(s) in a comment at the top of your Logisim circuit. **Submit only one file: project.circ**. **All team members must submit the same circuit**. Check your submission by downloading it from myCourses to verify that it was correctly submitted. You will not receive marks for work that is incorrectly submitted.

To help the TAs know who to grade, you must fill out this Google Form with your team members names and email addresses. The TA will use this sheet when grading. Your project will not be graded if it is not registered in this sheet: https://forms.office.com/Pages/ResponsePage.aspx?id=cZYxzedSaEqvqfz4-J8J6ldtrJSsGchLsKruoBPFKv5UNzZVWENaR0VCVDBKU0xWSjVSN0FQR1cwVi4u

If you are working on your own, **you must still fill out** the above Form with your name and email address.

Purpose

- Understand how CPUs function.
- To get used to wiring instructions as micro code.
- To understand how the CU coordinates the different machines that make the CPU.

Helpful

- The lecture recordings on CU.
- Optionally you may use your Mini 5 and Mini 6 solutions in this project, or you can use the solutions provided by the instructor.

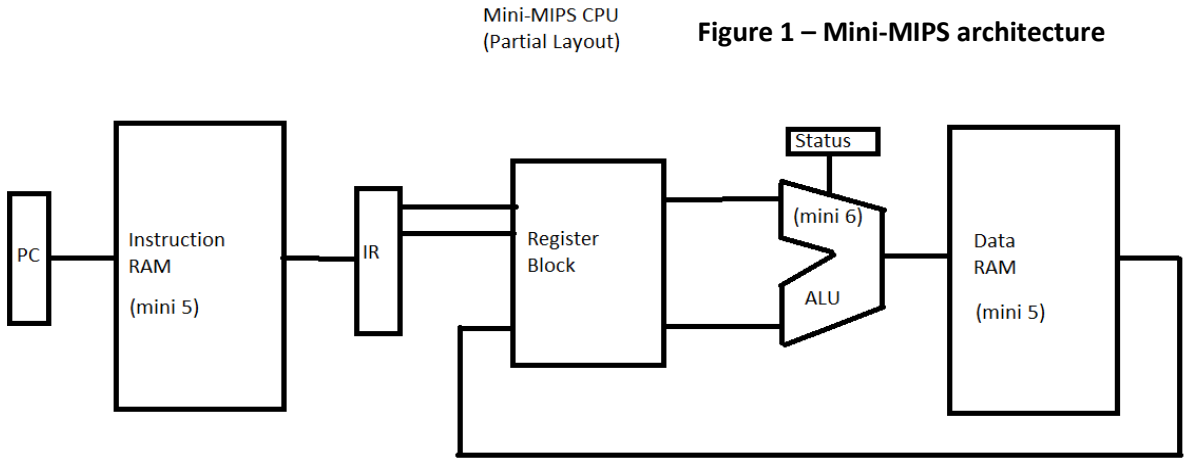

Figure 1 – Mini-MIPS architecture

Overview

Mini-MIPS is a pipeline CPU that executes only one instruction at a time. This means that the architecture of the Mini-MIPS CPU is patterned after the MIPS pipeline CPU, however it only executes one instruction per CPU execution cycle. In the normal MIPS pipeline multiple instructions are in the process of execution at any moment, but not in our Mini-MIPS. We are doing this to simplify the circuit.

The Mini-MIPS CPU diagram, Figure 1, is the basic CPU architecture that everyone's project must adhere to. It represents the basic pipeline architecture. The TA will make sure that your CPU looks and behaves as a pipeline. It is best to make your CPU look like the above diagram. You are permitted to build the entire CPU on the same circuit page, or you can create subcircuit pages to mimic the above diagram's look. If you build your CPU on a single page, it is important to label each part of the circuit and to leave empty spaces to separate the parts. This will both help you while debugging and help the TA understand what you have built. If you chose to use subcircuits you will need to take care of the time delays that Logisim adds to subcircuits. The more subcircuits you nest the more delays are added.

The execution of a single instruction requires multiple tricks. The CU, not shown, controls each section of the pipeline. Pipeline stage 0 constitutes the following elements: address from the PC entering the Instruction RAM (notice RAM and not cache – we are using your mini 5 RAM to make things easier). The output from the instruction RAM overwrites the IR. PC is incremented to the next address (circuit not shown in diagram). This completes Stage 0. Stage 1 uses the Register Block to output the values for the ALU's L and R registers (not shown in the diagram). Bits from the IR will address the specific Register Block registers. Stage 2 causes the ALU to do its thing updating the status register and A-out register (not shown in the diagram) and sending values to intermediate registers (not shown in diagram) for the multiple data paths that are possible at this point in execution (to prepare for: an overwrite of the PC, or overwrite Data RAM at a specific address, or overwrite a specific register in the Register Block). Stage 3 completes what was prepped by Stage 2: either write into Data RAM at a particular address or loads a value from Data RAM into a temporary register (not shown, like Data Register Out) for the Register Block or overwrites the PC, etc. Stage 4 completes the write to Register Block from the temporary register (if needed) and then the CU loops back to Stage 0. The CU operates as an infinite loop until the program asks the computer to shutdown.

Your final submission must run by auto ticking at the default Logisim Evolution 1 Hz rate. When you test your circuit, you can manually click on the clock, but the final submission must auto tick. You can set this up through the menu option Simulate, see Figure 2. Make sure "Auto-propagate" is check marked. Make sure "Auto Tick" is check marked. Make sure "Auto tick frequency" is set to 1 Hz. If you circuit runs at 1 Hz it will run at higher frequencies as well. But the TA will test your circuit with the default speed.

From the Simulate tab you can start and stop the ticking, see Figure 3. The play and pause buttons can be pressed to start and stop the ticking of the clock. When you pause the clock, you can manually tick the clock. When you press play, it will resume ticking where you left of.

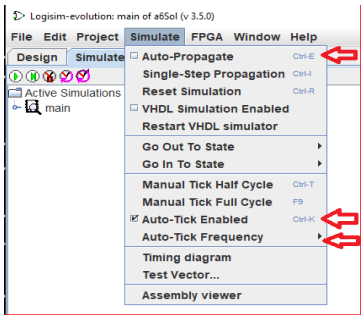

Figure 2

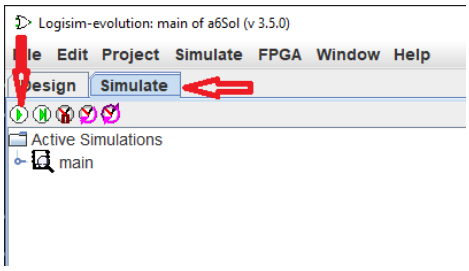

Figure 3

Team of 1, 2 or 3:

- If you are working on your own, then follow the instruction for a team of 2 project.
- Teams of 2 groups will create only the Mimi-MIPS CPU.
- Teams of 3 groups will create the Mini-MIPS CPU and a portion of the system board. The system board will contain a U-Bus and an input number keypad and an output 2-digit digital display.

Implementation

Using Logisim Evolution create the following Mini-MIPS CPU:

- To make this easier: We will not distinguish the system board from the CPU for those required to do the System Board. In other words, there will be only one U-Bus that will surround the CPU and System Board components. It will run on the same clock as the CPU. You can draw the system and CPU components on the same circuit diagram. Please separate and label the parts to help you debug and the TA to distinguish the components.

- To make this easier: You can use your Mini 5 and/or Mini 6 solution in your project. You can also use any solutions provided by the professor from labs or assignments.

- To make things easier: everything is a nibble in size. All data are a nibble (4-bits). The IR must be able to address a maximum of 8 nibbles.

- Your circuit must have the following components:
  - Team of 2
    - A clock
    - The CPU-parts depicted in figure 1 and described in Overview.

- The Register Block only has General Registers R0 and R1.
- The CU controls everything inside the CPU.
- The Status register flags: overflow, signed overflow, zero, and negative.
- Instruction RAM stores 8 nibbles of data.
- Data RAM stores 2 nibbles of data.
- The CU must implement the Fetch and Execute instruction.
- You can add additional components to help you. See the list of legal components.

- Team of 3 additional elements
  - A U-Bus
  - Optional MAR & MBR register. If you like, these registers could be replaced by the D and X registers described below.
  - The CU controls everything outside the CPU.
  - 2-digit digital display plus supporting registers. The 2-digit display will be used as a "slot" output device. It is connected to the U-Bus. This display has a register called D that stores a single 4-bit unsigned integer number. Any value in D is automatically displayed on the 2-digit Display Screen. Data RAM address 0 is connected automatically to register D through the U-Bus. The CU handles the zero-page mapping to the D register. See the IO_FLAG instruction below.
  - A keypad that supports single digit numbers 0 to 9. The user presses a key, and the binary integer value automatically is saved in its helper register X. Register X is connected through the U-Bus to the Data RAM address 1. The CU controls the mapping from the zero-page to the E register. See the IO_FLAG instruction below.

- You are permitted to use the following Logisim pre-built elements:
  - Clock
  - Wire
  - D-Flip-flop
  - AND, OR, NOT, XOR gates
  - Pins and probes
  - Subcircuits (optional)
  - Tunnelling (optional)
  - Multiplexer
  - Decoder
  - Team of 3:
    - The Button component for the keypad (under Input/Output)
    - 7-Segment Display for the Display Screen (under Input/Output)
  - You must build everything else by hand (cannot use other Logisim components)

NOTE 1: The TA must be able to change and see the bits of RAM and registers before and after execution. Provide a way for this to happen (this is especially important for those of you using subcircuits).

NOTE 2: Your final circuit must use designs we covered during class. You cannot use any outside (other sourced) circuit designs.

Instructions

- The Instruction RAM has 8 nibbles. These nibbles will be used to store your assembler instructions. Algorithms must fit within this space and terminate with HALT.

- The Data RAM has 2 nibbles and will store the program's data.

- To run a program, you will need to first input the instruction into the Instruction RAM, the data in the Data RAM, and set the PC pointing to the first instruction in the Instruction RAM. All other registers are assumed to be zero or overwritable. Then the auto-clock is turned on and the programs runs. Make sure to place the clock on your circuit board connected to the circuit.

- Team of 2
    - LOAD REGISTER, RAM_ADDRESS
        - 1st 2 bits : LOAD = 00
        - 3rd bit : REGISTER = 0 for R0, 1 for R1
        - 4th bit : RAM_ADDRESS = 0 for address 0 in RAM, 1 for address 1 in RAM
        - Example: LOAD R1, 0 → 0010
        - Example: LOAD R0, 1 → 0001
    - SAVE REGISTER, RAM_ADDRESS
        - 1st 2 bits : SAVE = 01
        - 3rd bit : REGISTER = as in LOAD
        - 4th bit : RAM_ADDRESS = as in LOAD
        - Example: SAVE R0, 1 → 0101
    - ADD REGISTER1 REGISER2
        - 1st 2 bits : ADD = 10
        - 3rd bit : REGISTER = as in LOAD
        - 4th bit : REGISTER = as in LOAD
        - The solution to the addition is saved in REGISTER1.
        - Example: ADD R1 R0 → R1 = R1 + R0 → 1010
        - Example: ADD R0 R0 → R0 = R0 + R0 → 1000
    - SUB REGISTER1 REGISTER2
        - 1st 2 bits : SUB = 11
        - 3rd bit : REGISTER = as in LOAD

        - 4th bit : REGISTER = as in LOAD
        - The solution to the subtraction is saved in REGISTER1.
        - Example: SUB R1 R0 → R1 = R1 – R0 → 1110
        - Example: SUB R0 R0 → R0 = R0 – R0 → 1100
    - HALT
        - All four bits are 1.
        - Example: HALT → 1111
        - This marks the end of the algorithm. The clock's ticking does not affect the circuit anymore. The PC no longer increments.

- Team of 3:
    - IO_FLAG
        - All four bits are 0.
        - Example: IO_FLAG → 0000
        - By default, the information in the D and X registers are not downloaded into/from the Data RAM. Create a helper register called T that needs to be set to 1 for information to flow from/to D and X. By default, T starts with zero.

- Bonus instruction:
    - JUMP
        - For teams of 2, you can use 0000 since you are not implementing IO_FLAG.
        - For teams of 3, what trick can you use?
        - The PC is updated with whatever is in R0.

- Your circuit must be able to execute any program set in the RAM with the PC pointing to the first instruction of that algorithm. The PC can be set to any starting address.

Execution

Your CPU circuit must be able to do at least the following algorithms:

1. Execute a program that loads two numbers, performs an ALU operation, and then saves the solution.
2. Execute a program that has only the HALT instruction.

Note: The TA will test your CPU by first entering a starting address in the PC and loading a program in your Instruction RAM and data in your Data RAM. Then, they will start the clock and see the result display in the Data RAM (or your Display Screen for the Team of 3 groups).

Note: Your CPU does not need to be optimized, therefore it does not matter the number of micro steps required to execute your instructions or the number of ticks it needs to take.

Marking

- Bonus points for JUMP +3
- Maximum 100 points (group of 2)
  - +5 Reusing A5 circuit (or you can create a new version).
  - +5 Reusing A6 circuit (or you can create a new version).
  - +5 Using one common clock for CPU.
  - +25 The pipeline architecture.
  - +10 CPU Registers R0 and R1 (the Register Block).
  - +50 CU Fetch and execution of instructions (including PC increment)
- Maximum 100 points (group of 3)
  - +5 Reusing A5 circuit (or you can create a new version).
  - +5 Reusing A6 circuit (or you can create a new version).
  - +5 Using one common clock for both CPU and System.
  - +20 The pipeline architecture.
  - +5 U-Bus, MBR, MAR (if needed).
  - +10 CPU Registers R0 and R1 (the Register Block).
  - +40 CU Fetch and execution of instructions (including PC increment).
  - +10 Display Screen and D register.
- No late penalty with max 3 late days.
- -10 to -30 points for not following instructions.
- -20 points for not using the clock.
- Assignment must execute to be graded.

Rubric

Range\Quality	Marginal	Pass	Good
90-100	The entire circuit works but only 2 **minor** issues.	The entire circuit works but only 1 **minor** issues.	The entire circuit runs well with no noticeable issues.
80-89	The entire circuit works but >3 **limited** issues.	The entire circuit works but with 2-3 **limited** issues.	The entire circuit works but with 1 **limited** issue.
70-79	¾ of the components run many **limited** issues.	¾ of the components run with many **limited** issues.	Almost all components run with **limited** issues.
60-69	½ of the components run with many issues.	¾ of the components run with many issues.	¾ of the components run with a few issues.
50-59	Only ½ components run but with major issues.	Only ½ components run with fewer major issues.	Only ½ components run mix of major & issues.
< 50	Most components do not run well, or at all.	¼ of the components run with or without issues.	Less than ½ components run with or without issues.

Marginal = Lower 3 points, Pass = Middle points, Good = Upper 3 points in range. Must run t grade.

Issues: Major=does not work, Issue=works some cases, Limited=works most cases, Minor= technicality.