CPU

COMP 273 Project

Due: December 5, 2022, on myCourses

Submission instructions

You can do this project on your own or in a team of 2 students. 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 documents / Logisim circuits. **Submit only one file: project.circ**. 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 Sheet with your team members 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://docs.google.com/spreadsheets/d/1_0TDM7jCe9rw6UT3RTZ2_71B5iAbLX7QPUtPpZ5sqaU/edit?usp=sharing

If you are working on your own, **you must still fill out** the above Sheet with your name.

Purpose

- Learning about the basic circuitry in the CPU
- To get used to wiring the execution of macro code as micro instructions
- To understand how the CU coordinates the different machines that make the CPU

Helpful

- You can use A5 and A6 circuits in this project.
- The lecture recordings on CU.
- You will need to merge the A5 and A6 circuit into this project. You may use your own solutions, or you can use the solutions provided by the instructor.

Implementation

Using Logisim Evolution create the following basic classical CPU:

- To make this easier: We will not distinguish the system board from the CPU. In other words, there will be only one U-Bus that will surround the CPU and System Board components. Everything will use this single bus. In addition, a single clock will control everything.
- 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.
- To make things easier: everything is a nibble in size. All data and addresses are a nibble (4-bits). The IR is a nibble.
- Your circuit must have the following components:
    - A clock
    - A U-Bus
    - CPU-part: MAR, MBR, IR, PC, ALU (R, L, Status, A), General Registers (R0, R1), and CU
    - The CU controls everything inside the CPU and outside the CPU
    - The Status register flags: overflow, signed overflow, zero, and negative.
    - System-part: RAM (AR, Mode, DR) and a 2-digit Display Screen (this is new)
    - RAM stores 8 nibbles of data.
    - 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. RAM address 0 is connected automatically to register D through the U-Bus.
    - The CU must implement the Fetch instruction, Execute instruction, and zero-page mapping to the D register.
    - You can add additional things (like registers or D-flipflops, etc.) to help you.
- You are permitted to use the following Logisim pre-built elements:
    - Clock
    - Wire
    - D-Flip-flop
    - AND, OR, NOT, XOR gates
    - Pins (in/out)
    - 7-Segment Display (under Input/Output) for the Display Screen
    - Black boxes (optional)
    - Tunnelling (optional)
    - 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.

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

Steps

- Begin by deciding the location of the U-Buse and the A5 and A6 circuits. I suggest you use the "adding a circuit" project option to import your A5 and A6 circuits, but this is optional.
- Decide where the CPU registers will be placed and where the CU will be located.
- Decide if you will need extra registers.
- RAM is 8 nibbles. These nibbles will be used to store assembler instructions and data.
  Our assembler instructions are nibble sized:
    - 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 REGISTER
        - 1st 2 bits : ADD = 10
        - 3rd bit : REGISTER = as in LOAD
        - 4th bit : is zero (unused)
        - The solution to the addition is saved in R0 automatically.
        - Example: ADD R1 → R0 = R0 + R1 → 1010
        - Example: ADD R0 → R0 = R0 + R0 → 1000
    - SUB REGISTER
        - 1st 2 bits : SUB = 11
        - 3rd bit : REGISTER = as in LOAD
        - 4th bit : is zero (unused)
        - The solution to the subtraction is saved in R0 automatically
        - Example: SUB R1 → R0 = R0 – R1 → 1110
        - Example: SUB 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.
- Your circuit must be able to execute any program set in the RAM with the PC pointing to the first instruction of that algorithm. You may set the PC's starting address anywhere you like.

Execution

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

1. Execute a program that loads two numbers, performs and 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 RAM. Then, they will start the clock and see the result display on your Display Screen.

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

Marking

- Maximum 100 points
  - +5 Reusing A5 circuit
  - +5 Reusing A6 circuit
  - +5 Using one common clock for both CPU and System
  - +5 U-Bus
  - +10 Managing MBR, MAR, and PC with RAM
  - +10 CPU Registers R0 and R1
  - +10 PC Increment
  - +40 CU and execution of instructions
  - +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