COMP 273

Classical CPU Project

Project Due: April 12 to 16, 2013 at 23:55 on My Courses & Demo

Assemble a team of 2 or 3 students, 3 are preferable, who will work together to construct the solution to this project. Only your team leader will submit the project to the Project assignment box on My Courses. Make sure to include a text file identifying your team member's by name and ID number, so that everyone can get the grade. Include your own name and ID number in this list. The what-to-hand-in section details how this submission should be constructed.

The reason there are 3 days for the submission is because you will demo the program to the TA or Professor during one of those three days (April 12, 15 or 16). You will have to make an appointment, so the earlier you do this the better. You can even start now! (do this by email)

PROJECT (15 points + 3 bonus points)

This project is segmented into two parts: the basic question and the bonus question. All teams must do the basic question. The bonus question is available if you found the basic question too easy. If you are a group of 4 then you must do the bonus questions but you will not receive any bonus points. If you want to also receive bonus points then you will have to propose something to the prof for approval.

You must construct your project using LOGISIM.

BASIC QUESTION

Build an operational Classical CPU circuit diagram using LOGISIM. Your CPU is contained within a "system board". The project is described below divided into the System Board section and the CPU section. Your system board's clock speed and your CPU clock speed can be anything you like, even the same speed.

You are permitted to build your CPU from the following items: not-gate, and-gate, or-gate, xor-gate, flip-flop, half-adder, full-adder, clock, wire, wire bundles, input pins, output pins, Logisim's single digit display, multiplexers, decoders, encoders, PLA, or PAL. You can build your own black-boxes. Logisim behaves funny if you nest your black-boxes deeper than 3.

The System Board will be composed of the following items:

- Full Read/Write 16-byte RAM with AR, DR and Mode registers (can process one byte at a time)
- A 13-bit system bus. 8-bits for the byte, 4-bits for the address and 1-bit for the mode.
- A 2-digit digital display with it's companion 8-bit output data register.
- An 8-bit input buffer connected to a number-keypad (simulated by Logisim's input pins where each pin is one of the numeric digits from 0 to 9 from a keypad). This keypad will be pressed before your CPU starts running. While running it will read what is in the buffer.

The Classical CPU will be composed of the following items:

- An 8-bit CPU bus
- MAR, MBR, and MMode registers which will interface with the RAM
- A PC with an increment adder (each instruction is 1-byte long)
- A IR with a sequencer (control unit), which you are free to format as you see fit
- Two general purpose registers
- An ALU with status (overflow?, zero?, negative?), 8-bit Left, Right and A-out registers.
- Two private buses, one to the keypad and the other to the display.
- Data can be moved to any of the GP registers, PC, RAM, keypad or display.
- Any other "classical" pathways are up to you

Your CPU's execution cycle:

Step 1: IR ← RAM[PC] & PC++ (note RAM[PC] implicitly uses MAR, MBR, AR, etc.)
Step 2: Get instruction's arguments (from RAM, GP Register, or Keypad data register, if any)
Step 3: Execute the instruction (eg. ALU operation, JUMP, or MOVE)
Step 4: Save the result (if any, into RAM, GP Registers, or display buffer register)

In order to make the drawings easier you may use black boxes in the following cases:

- Only after you have designed a full sample circuit in detail
- You can reuse black-boxed components in other areas of your diagram.
- You can reuse components from your assignments, specifically:
  - Register
  - Full-RAM
  - Adder

Your programming language:

- You must supply your own binary definition for these instructions (ie. The op-codes, parameters, etc.). Note that Rxy indicates either of the two general-purpose registers can exist at that spot.
- ADD Rxy, Rxy (addition of two registers with the result is stored in the first register)
- STR Rxy, Address (store the value in Rxy into the byte at address in RAM)
- LD Rxy, Address (load value into Rxy from the address in RAM)
- BZ Rx, Ry Address (branch if Rxy is zero to Address – use an XOR circuit if you do not plan to implement SUB)
- PRT Rxy (output contents of Rxy to your digital display)
- INP Rxy (store data to Rxy from the keypad buffer)
- STOP (mysteriously, the program stops running)
- Bonus Instructions!! (see bonus section below for more information)
  - SUB Rxy, Rxy (subtraction of two registers with the result stored in the first register)
  - COMP Rxy (complement of Rxy, saved back to the same register)
  - MULT Rxy, Rxy (multiply two registers and save result in the first register)

Notes:

1. You do not need to concern yourself with how the first instruction's address got into the PC. You do not need to concern yourself with how information arrived in the RAM in the first place. Just assume that the PC is initialized to zero and the RAM has a program with data already positioned at address zero .
2. Please provide the following program written in your language that can fit in your memory and be executed by your CPU. We will execute that program when we test your design.

The program you must write in your programming language, running on your CPU:

- Write a program that multiplies two numbers without using the MULT instruction. One number comes from the RAM and the other comes from the input buffer. The solution is displayed on your digital display.

Other than the items listed above, you are free to design your classical CPU in any way you like.

2-digit Digital Display

Your 2-digit digital display should be patterned along the slides presented in class. Specifically this would be Lecture 6/7 the slides covering the topic Binary Code Decimal Example. You will build a simple digital calculator-like 2-digit display. LOGISIM has an additional black-box that you are permitted to use. It is part of the built-in library extensions called Input/Output. Within that extension is a component called 7-Segment Display. You will need two of these. The CPU instruction PRT will convert the integer number stored in a register and display that to this digital display.

8-bit Input Buffer

Your circuit will also contain an input device. We will imagine that it is connected to a 10-digit keypad, but for our purposes it will simply be 10 LOGISIM input-pin connectors sitting to one side of your circuit diagram. The user can "press" one "button" and a binary digit will be stored in the buffer. Your CPU can read these bits with the INP instruction. The instruction does not wait for the user.

Please provide a name for your CPU (something cool would be nice).

BONUS QUESTION

Listed below are three possible upgrades to your CPU. Each upgrade is worth 1 bonus point. You do not need to answer this in any particular order. Select 1, 2 or all three features to implement. There are no part marks for the bonus questions. You either get 0 or 1 point.

- Implement a 2's complement feature for your integer numbers in the ALU by providing for these two additional CPU programming instructions:
  - COMP Rx → Rx is converted to 2's complement, stored back into Rx (uses ALU)
  - SUB Rx, Ry → Rx = Rx – Ry, using 2's complement addition (use your own implementation)
  - Note: by default your memory and registers will now store signed integer numbers (in unsigned or 2's complement formats)

- ON/OFF Switch
  - Provide circuitry that combines the clock with an ON/OFF switch for the computer. When the switch is on and the clock ticks the PC is set to zero and the computer starts executing the program in memory. How the program got there, we don't know. But you must initialize the PC to zero. You MUST use a LOGISIM black-box for this option as well. The Input/Output library has a Button box. The user must press this button to turn on/off your computer.
  - Implement an additional computer instruction: HALT. This command turns off the computer.

- Multiplication
  - Upgrade the ALU so that it can multiply! There is an easy way and a hard way to do this…
  - Provide an additional instruction: MULT Rx, Ry. The result it stored in Rx.

WHAT TO HAND IN

Everything should be handed in electronically on My Courses.

Students must work in teams of 2 or 3 people. Only the team leader hands in the project to My Courses. In addition to his or her solution, please also submit a text file identifying each team member by name and student ID number. Each person in the team will receive the same grade. This text file will help us during grading.

Hand in all the LOGISIM files (including the black boxes) with a README.TXT file if the TA needs special instructions for running your circuit. Only those portions of your design that actually run will be graded.

Make sure to make an appointment with either the TA or the Professor to demo your project. You will have about 5 to 10 minutes to show that your project works (or which parts work).

Also submit a simple document that provides the following information: A black box summary of your entire CPU. This should fit on a single printed page. It would be a high-level view of the entire CPU. This includes an additional page of text describing how it functions. Your document should give some information about each LOGISIM file you uploaded, what it is and how it should be used. The very last page of your document should be your assembler-like program and a description of your assembler language.

Submit your document as a WORD, EXCEL, PDF or JPG file. Submit your design as LOGISIM file(s). We will execute these files to confirm that your design runs. Copied files will lose all grades. Make sure to submit this in the box called PROJECT. There are no late days.

Name your CPU. Provide this information in the document.

Project Document

Your project document should have the following elements:

1. A cover page
   1.1. Project title/CPU name
   1.2. Name of each team member with ID number
2. Section 1: Overview of CPU
   2.1. High level diagram of the entire CPU in one picture (one page)
   2.2. Overview of its functionality and flow (one page)
3. Section 2: LOGISIM File Breakdown
   3.1. Using a top-down approach, describe each part of the CPU (one or two pages)
4. Section 3: The Assembly Language (two to four pages)
   4.1. Define your instructions in assembly
   4.2. Define your instruction bit formatting
   4.3. Describe how your instructions execute over the CPU
   4.4. Provide the requested program
5. Section 4: Appendix (optional)
   5.1. This place is for miscellaneous other things important for TA to have

HOW IT WILL BE GRADED

The mini-project is worth 15 points plus 3 bonus points.

- The basic project is worth 15 points
  - Graded proportionally
  - Design 10 points
    - CPU – 7 points
      - RAM – 2 points
      - Registers – 1 point
      - ALU – 1 point
      - PC + IR + CU – 3 points (includes proper execution cycle)
    - DISPLAY – 1 points
    - INPUT – 1 point
    - Document – 1 point (but can loose points in other places if design is hard to read)
  - Running program 5 points
    - Branching – 1 point
    - Instruction bit format – 1 point
    - Addressing – 1 point
    - Program runs – 2 points
- Bonus points (3 in total) for mini-project
  - 1 point for each bonus implemented correctly, 0 points otherwise (not graded proportionally, graded binary = a full yes or a zero)