COMP 273

Classical CPU Project

Project Due: April 10 to 14, 2015 at 23:30 on myCourses & Demo
20 points + 3 bonus points

Assemble a team of 2 or 3 students, 3 are preferable, who will work together to construct the solution to this project. Everyone in your team 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 same grade. Include your own name and ID number in this list. The what-to-hand-in section details how this submission should be submitted.

The reason there are 3 days for the submission is because you will demo the project to the TA or the Professor during one of those days (April 10, 13 or 14). You will have to make an appointment, so the earlier you do this the better. You can even start now! (do this by email)

You will create the following for this project:

A two bus executable classical computer using Logisim. This computer will have a system bus and a CPU bus.

The system bus has the following machines connected to it: a numeric key pad (digits 0 to 9), a 3 digit display, a 16 byte RAM, the MAR and MBR of the CPU. RAM has the AR, DR, and Mode.

The CPU bus has the following machines connected to it: two general purpose registers, the ALU, the MAR, and the MBR. The MAR is, at the same time, connected to the PC. The PC is connected to an adder. The MBR is, at the same time, connected to the IR. The IR is connected to the CU.

The ALU is made of the following machines: the L and R input registers, the A and STATUS output registers. The STATUS register flags the following conditions: overflow, negative result, and zero result. The L, R and A registers are connected to the CPU bus. The STATUS register is connected to the ALU and various CPU gates, as needed. Only the L register is connected to a 2's complement machine.

The PC increments by 1 at each new instruction. The PC is connected to the MAR.

The CU is constructed as you see fit. It must control the operation of the entire computer, both in CPU and on System Board. This will make things simpler for you. The CU must support the following instructions:

LOAD:	LD	Register, Address	; Loads to either general-purpose registers from RAM
STORE:	STR	Register, Address	; Saves from either general-purpose register to RAM
ADD:	ADD	Register1, Register2	; Register1 = Register1 + Register2
SUBTRACT:	SUB	Register1, Register2	; Register1 = Register1 - Register2
Branch equal:	BEQ	R1, R2, Address	; if R1 == R2 then PC = Address
Branch not equal:	BNQ	R1, R2, Address	; if R1 != R2 then PC = Address
PRINT:	PRT	Register	; Contents of Register displayed on display
INPUT:	INP	Register	; Register receives a value from key pad
STOP:	STOP		; Program stops executing

Register, Register1, Register2, R1, R2 means: Any general-purpose register, but not the same register in both parameters.

Optional Bonus Questions: [3 points - one point for completing each bonus question]

- Create a multiplication circuit and insert that into your classical CPU. With a new instruction called MULT Register, Constant, where Register = Register * Constant.
- An ON/OFF switch that performs a STOP and then sets all registers and RAM to zero.
- A ROM that permanently stores the program we ask you to write for this project. This ROM will download the entire program into RAM at the “flick” of an input pin.

ADDITIONAL INSTRUCTIONS

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 bundle, input pin, output pin, Logisim's single digit display, multiplexer, decoder, and encoder. You can build your own black-boxes. Logisim behaves funny if you nest your black-boxes deeper than 3.

Your buses can only transport one byte at a time.

Your instructions and data must be only 1 byte long.

The 3-digit digital display has a companion 8-bit output data register.

The numeric-keypad has an 8-bit input buffer connected to the number-keys (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. There is no wait.

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

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 code and 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.

3-digit Digital Display

Your 3-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 3-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 three 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).

WHAT TO HAND IN

Everything should be handed in electronically on myCourses.

Students must work in teams of 2 or 3 people. All the members of the team hand in the same project to myCourses. In addition to your 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).

Label your Logisim circuit diagram or provide a single page describing the structure of your circuit, so that we can know how to read your circuit.

Name your CPU. Provide this information in the circuit diagram or in the single page document.

Provide and additional page or two describing the bit pattern of your assembler language.

Lastly, on myCourses Discussion board, it a Project Registration section. Post a single message there with your CPU name and your team member's names.

HOW IT WILL BE GRADED

The mini-project is worth 20 points plus 3 bonus points.

- The basic project is worth 20 points
  - Graded proportionally
  - Circuit design 10 points
  - Running program 10 points
- Bonus points (3 in total)
  - 1 point for each bonus implemented correctly, 0 points otherwise
    (not graded proportionally, graded binary = a full yes or a zero)