Introduction to Computer Systems
COMP-273

# Assignment #1

Due: January 31, 2015 on myCourses at 23:30

## Question 1: Data Representation (4 points)

A "fraction" is a logical concept that represents a number as two integer values. One of these integer values is on the numerator while the other is on the denominator. The "fraction" is commonly depicted in mathematics as:

$\frac{a}{b}$ where a and b are integers from $-\infty$ to $+\infty$

An example fraction would be 5/7. Another example is 10/20 which would reduce to 1/2. Converting these types of numbers into decimal numbers could lead to an infinite series, which cannot be represented in a computer.

For this question, suggest a binary data representation for the fraction. In class we talked about how to represent a C string in binary. I am asking you to do the same thing for this "fraction" definition. As in all hardware representations you will have to make compromises. There is more than one valid solution to this problem. Please provide a well explained solution.

Your solution MUST have the following parts: (a) a description of your "fraction" data structure in binary (it's physical structure: bit-size, addressing and organization), (b) an example value stored in your binary "fraction", (c) assuming I want to create 10 "fraction" numbers, calculate the amount of memory it will use up, and (d) identify ALL the compromises you had to make and their impact comparing the theoretical "fraction" with your implementation: discuss representation and operations like addition and division.

## Question 2: Number Systems (4 points)

(A) Convert the following numbers and show all your work:

$92A5F_{16}$ to decimal and binary
$100010110101_2$ to decimal and hex
$339_{10}$ to binary and hex

(B) Bob used a pointer to save the C string "I am Mary." at address $110_{16}$ in RAM. Depict the memory as a table with two columns. The first column will be the address in RAM and the second column will be the data stored at that address. Display the memory and it's contents in binary for the C string Bob stored at

$110_{16}$. In other words: write the addresses in binary and the bytes in binary for the given C string as a table having two columns, the left-most column being the addresses.

## Question 3: Hardware Systems at the Logical Level (5 points)

Assuming a classical CPU (having 2 general purpose registers called R1 and R2, an ALU, IP, IR, status register, sequencer, and DR, AR, R/W bit, without a cache) is connected directly only to the system board's 25-bit bus (8-bits for data, 16-bits for address and 1 bit for read/write). Assume a single 8-bit slot connected, on one end, only to the RAM at address $0F_{16}$ and, and on the other end, to some random device. Assume further that the CPU has two instructions called LOAD A,B and SAVE A,B. LOAD copies data from an address in RAM (designated by the B term) to a register (designated by the A term). SAVE copies data from a register (designated by the A term) to an address (designated by the B term). Finally, assume there are 100 bytes starting at address $200_{16}$ of RAM that must be sent to the device connected at our given slot. Do the following:

A) Write a pseudo algorithm using LOAD and SAVE that will move the 100 bytes from RAM to the slot. Use pseudo-C syntax for all your control statements plus the LOAD and SAVE commands. In other words, you can use if-statements and loops with variables, except when you have to use the LOAD and SAVE commands.

B) Describe the path these bytes will take through this imaginary computer from RAM to the slot. Be as precise as you can be. Mention every wire (pathway) and register it must pass through. This is called “tracing”.

## Question 4: Circuits with LOGISIM (7 points)

For the circuit assignments and the mini-project you will be asked to build logic circuits using the LOGISIM simulator. This useful simulator not only helps you draw professional circuit diagrams but it also executes them. You can actually watch them run to completion graphically. This will both help you and the TA validate the correctness of your circuits. All your circuits must be validated in this way for you to receive full marks.

There is a link to the LOGISIM program on the course web page. You can get there through myCourses in the following way: myCourses → Select COMP273 → Select Course Information → Then select the course web site, on the right hand side will be a link to the LOGISIM JAR file. Download that file and then double click it to run. It is programmed in JAVA so it should just run when you double click the JAR file, assuming you have Java installed on your computer. Most computers come with Java already installed. You can also use Google to find it on-line (assuming a more recent version has been posted).

Do the following:

(A) Part 1 (2 points)

1. Download LOGISIM
2. Run LOGISIM's tutorial
    1. Go to the help menu
    2. Select tutorial (it will pop-up a 4 step “Beginner's Tutorial”)
3. Construct the “Beginner's Tutorial” circuit
4. Save this circuit in a file called **BEGINNER**.
5. **Upload** this file to myCourses as your answer.

(B) Part 2: Exploring Memory (5 points)

Using LOGISIM create a functional Read/Write 4-nibble RAM memory system. Your memory system will have an appropriately sized Address Register, a 4-bit nibble Data Register, a 1-bit read/write register, and the actual 4-nibble RAM with it's address space and it's data space. Set input pins on the Data Register, Address Register, Read/Write register so you can manually enter values into those registers. RAM will start off empty.

If the read/write bit is set to 1 then your RAM is in write mode and data in the Data Register will be stored into your RAM at the address specified by the Address Register.

If the read/write bit is set to 0 then your RAM is in read mode and data from the RAM at the address specified by the Address Register will be copied into the Data Register.

You are only permitted to use the following things from LOGISIM: bits, wires, the gates AND/OR/NOT. You are not permitted to use any other tools provided by LOGISIM except for LOGISIM's black-box feature that will help you simplify your circuit diagram (if you want to).

LOGISIM's BIT can be found in the program at: Project → Load Library → Built In Libraries. BIT is called a FLIP FLOP and it can be found in the MEMORY library. You can use any FLIP FLOP model. (Actually, take some time and look around at all the pre-built stuff … but don't get tempted to use any of them, except for the flip-flops).

A black box is a mini circuit diagram. LOGISIM lets you create these. They help make your diagrams simpler to read. When you create a black box LOGISIM will represent it by a picture of a square on the screen with input and output wires/terminals protruding out from the box. Your mini circuit is assumed to be contained within the square. The input/output terminals connect to the input/output wires in your mini circuit. As I said, black boxes are optional and do not effect your mark, but you may find it useful. (The lab with the TA will go over this feature.)

Save the LOGISIM circuit diagram in a file called MINIRAM. This is what you will upload to WEB CT.

**ASSIGNMENT SUBMISSION INSTRUCTIONS**

- Submit your solution to myCourses before the due date
- 5% is removed for each day late up to 2 days at which time the submission box closes
- Please submit your assignment in electronic form. The circuit diagrams must be in LOGISIM while text can be in PDF, RTF or TXT file formats.
- You must work on your own. A grade of zero will be applied to copied or shared solutions.
- The LOGISIM circuits must run (even incorrectly) to be graded. The TA's will not do debugging. If the circuit does not run then the TA will assign a grade of zero for that question.

**GRADING INSTRUCTIONS**

- This assignment is worth a total of 20 points
- The breakdown is stated on each question
- Each question is graded proportionally. In other words, if your question is 50% correct you get 50% of the marks.