Asst3: Architecture - in Spaaaace

You are a the Chief Computronist on the first intersolar survey vessel. While studying a planet likely suitable for colonization your ship encountered an alien ship. During attempts at contact there was some misunderstanding or error. The alients opened fire and quickly retreated. They left you in a decaying orbit with heavy damage to astrogation and navigation. You and the Chief Engineer have found that, while extensive, much of the damage seems superficial. The main control and calculation circuits are intact, but the interface to them is a pile of slag. You need to rebuilt the interface from spare parts before your orbit decays and you become unintentional colonists - at high speed. Luckily you had some of the datasheets for the hardware printed out, although they suffered some plasma burns.

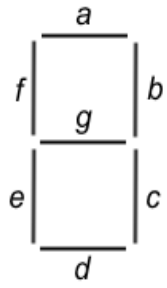
N.B. You will need to generate wire diagrams of some sophistication and size below. It is *STRONGLY* recommended that you use a drawing tool to generate them. There are a number of free drawing tools like DIA or GiMP that run on many operating systems. It is also recommended that you color-code the path your inputs take through your diagrams. For instance, red for 'A', blue for 'B' and so on.

Part 0: Re-establish Astogration Interface

You need to read output from the computer so you can run some function tests to be sure the programming and data is intact as the hardware seems to be. So your first task to re-establish a computer interface. Hexadecimal notation is used to interface with the navigational computer and to display all its data, although the computer itself uses Binary Coded Decimal (positive magnitude only, little endian) inputs and outputs (I/O). The parts that coded/decoded information from hexadecimal into BCD and vice versa were destroyed. You will need to design an I/O demultiplexer.

Your demultiplexer will need to select one of 16 different circuits. Each circuit needs to take a BCD input in 4 bits and translate it to a pattern of 7 LED segments to represent one of the 16 hexadecimal values: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, b, C, d, E, F. For instance, the bit pattern 0001 should turn on the two rightmost segments of the LCD to represent a "1" and the bit pattern 1010 should turn on all but the bottomost LCD segment to represent an "A".

Refer to the truth table in the attached datasheet for the output demultiplexer and draw a wire diagram that supports the arrangement and design of its pins. If you look at the "Connection Diagram" you'll see outputs labelled a through g. These correspond to each of the 7 LED segments. By convention, the 7 LED segments are named, starting at the topmost vertical segment and proceeding clockwise: a, b, c, d, e, f. The middle segment is named g.



Draw the wire diagram for your demultiplexer.
Be sure it implements the functionality detailed in the data sheet.

Part 1: Repair the Astrogation Input Module

You got the input and output LCDs up and running, but once you did you found some of the command keys for the navigation cluster are not responding. On investigation, you found a shard of deck plating wedged in the input module. You need to rebuild a few basic computational commands, but you need to use the output going to the LCDs to do it since you don't have the time to do the delicate work of tapping the traces on the board. The commands to increment, decrement add, and subtract coordinates by values are missing. Without these functions, the navigator can not key in course corrections as you fly. Each value consists of 8 hexadecimal values.

While you need to take the 7-segment LCD signals in as input, and must output them the same way, you do not need to compute on them. As you recall, addition and subtraction in twos complement is the same operation, so if you first designed a multiplexer to convert the 7-segment LCD signals in to a binary value and then made a circuit to encode *that* in to a twos complement number, all you'd need to do is feed it in to a full adder and decode the result.

Repair sequence:
a. You will first need to design a multiplexer to convert each 7-segment LCD pattern to a BCD. That shouldn't be hard since you designed the demultiplexer to do the opposite transformation in Part 0. Once you design this part, presume it has the same shape and interface as the microcontroller in the attached data sheet. If you want to refer to this part, draw it as a similar rectangle with the part number: "WTF211AST3"

Draw out the wire diagram for your multiplexer.

b. Design an encoder to take the 8 BCD numbers your multiplexer above will output and convert them to one twos complement value. Remember that each BCD value is a different power of 16. For instance the input: E00000AB would give you the BCD values:

1110 0000 0000 0000 0000 0000 1010 1011

.. which would be outputted by your array of 8 multiplexers. You need to take those and multiply them by 16 the correct number of times to reflect the column of the hexadecimal value they came from. So, "1011" needs to be multiplied by 16^0, "1010" by 16^1, and "1110" by 16^7. Recall though - once you have values in binary, multiplying by a power of 2 is pretty easy. You can presume a 'negative' input line/pin will be on if the hexadecimal input is negative.

Draw out the wire diagram for your encoder.

c. You do not have to draw a circuit to implement twos complement addition as it is very well documented, but you should look one up. Find a microcontroller that does twos complement addition of 32 bit integers and attach/submit its data sheet and the URL where you found it to your assignment.

Include the data sheet of a twos comlpement adder in 32 bits.
Be sure to document your source.

Extra Credit:
Reprogram the Course Plotting Coprocessor

You've restored primary helm control. It isn't perfect, but at least now you'll be actually orbiting rather than crashing very slowly. Helm control and Astogration *can* work, but they are having a tough time of it. Helm controls are input as Spherical coordinates, but Astrogation is done in Cartesian coordinates. Unfortunately the Course Plotting Coprocessor took a power surge in the attack and was partially erased. The coordinates can be calculated by hand, but it is one thing to know the math, it is another to compute it while hundreds of lives are on the line when you haven't done any of the calculations in years. Besides, helmsmen and naivgators should leave the computation to the Chief Computronist.
You can find details on the coordinate systems at:
https://math.oregonstate.edu/home/programs/undergrad/CalculusQuestStudyGuides/vcalc/coord/coord.html

The main problem is square root. You are getting your data coded as two twos complement binary values that each are long enough to hold 8 hexadecimal digits (i.e. 64 bits). The first 8

hexadecimal digits is for the integer part, and the next 8 are for the fractional part. Be sure that any limits or calculations or error boundaries presume the full length representation. For instance, if your method requires an error bound, it should be below $1/16^8$.

Repair sequence: (each counted separately, although must be completed in order)
a. Consult your 'technical documents' on digital square root (i.e. look it up online) and keep track of your sources.

> *Find an algorithm for a discrete approximation of square root and attach it.*
> *Document your source.*

b. Detail the algorithm or technique you will use. Write it out in C.

> *Implement the attached algorithm in C.*

c. In order to reprogram the Coprocessor you will need to program it directly in assembly language. The unit uses the Y86-64 language. You can find it specified in your textbook.

> *Transcribe the C code to Y86-64 and attach it.*