Michael Masenheimer
CSC 252 - ASIM 1 report

Section 1

1) The first variable is at $t0 which loads the address of sort. It's at 0x00400010

2) First line of LA instruction: lui $1, 0x00001001
Second line of LA instruction: ori $8, $1, 0x0000003c

3) First line of LA instruction in hex: 0x3c011001
Second line of LA instruction in hex: 0x3428003c

Section 2

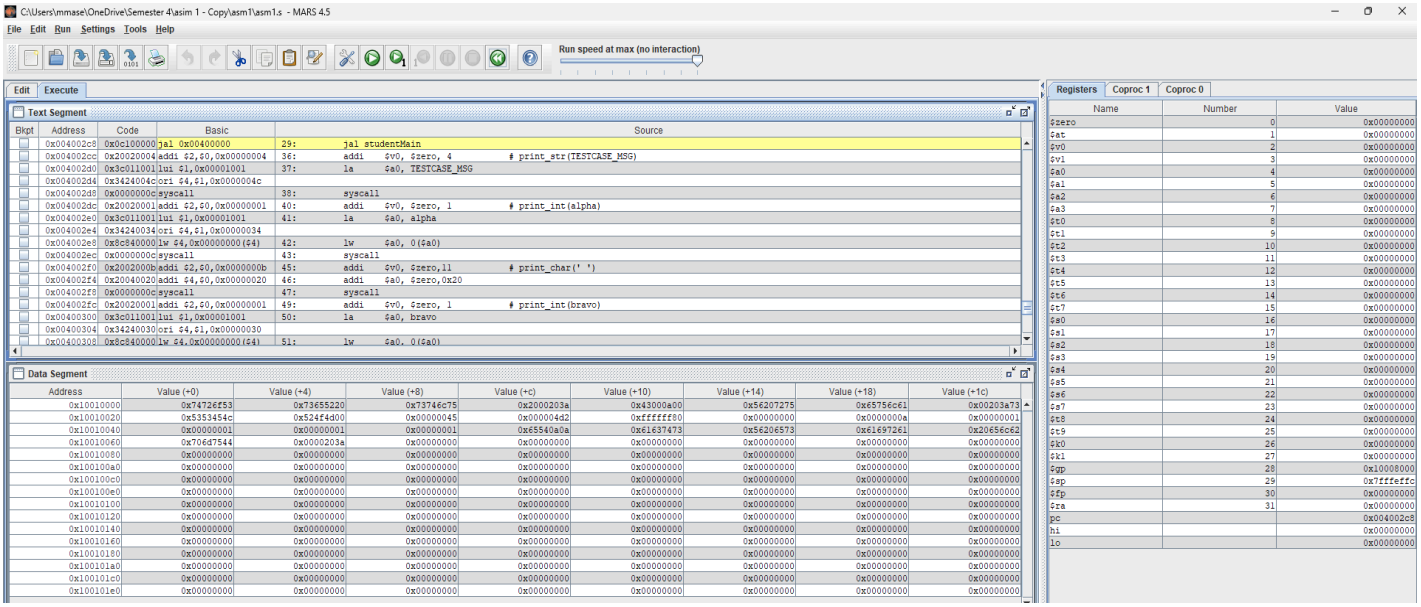
1) The first string I loaded was called **sorted** with "Sort Results: ".
2) Its first 4 characters translate to 53 6F 72 74 0A.
3) I looked at the register that said $at which said 0x10010000
4) Ended up at 0x74726f53
5) T r o s and s e R and s t l u

Reflection

This exercise made me step back and think about how loops really work, where conceptually it is just a bunch of code jumping over other code until it meets a condition. I learned how to use jump statements for repetition, this will be useful because I have a feeling knowing how to branch and jump is going to be a core part of writing assembly code. A few questions I have regard the interaction of .s files. I'm still a little confused on how to run code on a different file from a main file (although that might be covered later). Another question is about the difference between while and do-while loops: how they look in assembly. I know we haven't learned about do-whiles but it sounds interesting to me.

How did I find the information in the first part of this report?

After putting my main code into a directory with only one test case, I assembled the code and looked at the MARS "execute" and register section.



This is the whole screen where I was able to locate addresses and understand how string loading and la works.

These buttons are the bulk of how I ran my code. The left one actually assembled it, the middle runs the current program, and the last one increments the code running by one line (or one instruction). I was able to use these to find the addresses and hex values for all of 1.1-1.2.

I noticed that for the LA instructions, the address of the label is embedded into the actual machine code for values in lui and ori. The 32 bit address is made of two 16 bit constants.