CSE 341: Computer Organization Report (Spring 2022)

Name (Last, First): McGeough, Brecken

UB 8-digit number: 50301442

Project A Report: MIPS Assembly Programming

1. Describe your implementation in detail for the Minimum-Maximum problem. (less than 200 words)

My implementation starts out with 2 variables that will keep track of the current minimum and current maximum value. They are initialized to -1. The implementation then iterates through the input text by 3, loading the byte of the current iteration index in the input and loading the byte of the current iteration index plus 1. Then, it gets the ascii value of the current character and the next character, subtracting 48 from them to get their decimal value, then combining them to create a 2-digit number. It combines the two ints into the 2-digit int by multiplying the first int by 10 and adding the second int to that (Ex. 13 = (1 \* 10) + 3). If the number is less than the current minimum, update minimum to be the number. If the number is greater than the current maximum, update maximum to be the number. If current minimum and maximum are -1, then update both to be number. Then, return the final minimum and maximum numbers.

1. Describe your implementation in detail for the Sorting problem. (less than 200 words)

My implementation first loops through the input text by 3 in order to load the byte and next byte of the current iteration index in the input, in order to construct the 2-digit number at index i of the input (same process as minimum-maximum except here it is done to construct an array of 2-digit number). Then, it adds the constructed 2-digit numbers to an array that I allocated of size 10 (words). Then, my implementation iterates over the length of the array, and for every iteration, it iterates over the length of the array again, and checks if the 2-digit number at the current index of the iteration is greater than the 2-digit number at the current index plus 1. If it is, it swaps the numbers in the array. After doing this for every number in the array, and doing that n times (n=size of array), it returns the updated, bubble sorted array.

1. Describe your implementation in detail for the Bonus problem (If attempted). (less than 200 words)

I reused a lot of the same code from the min\_max and bubble sort problems. I first checked t, and if its 1, I run the min\_max code modified for n 2-digit integers, then call jal display (look at implementation detail for min\_max). If t is 2, then I run the bubble sort code modified for n 2-digit integers, where the array allocated is of size 800 (words) for sufficiently large inputs (look at implantation detail for bubble sort). Then I call jal display\_array.

1. Highlight major challenges that you faced while working on Part A of the project (if any). (less than 200 words)

Some major challenges that I faced working on part A of the project were keeping track of what values are supposed to be in certain registers in order to not overwrite important registers, and making sure I’m correctly iterating through arrays by the number of bytes. Also, extrapolating out my algorithms to work for n 2-digit integers was a challenge.

1. Please mention any comments or suggestions for Part A of the project (If any). (less than 200 words)

Very fun and engaging project.