Nathaniel Daniel

Shen

CS 219

23 April 2020

PR2

The first section in the code is called “LC1”, and holds the formatting string for the printf function. The next section is the “LC0” section, which holds the data for the list to be searched. The main function sets up the stack frame. The program then loads a pointer to the list. It also initializes the largest variable. Then, the program initializes the “i” variable and jumps to block “L2”. If it is less than 3, it jumps up to “L4”. Here, the program compares the element at index i in the list and the “largest” variable. If the element in the list is not greater, the program jumps to “L3” and increments the index. Otherwise, it sets “largest” as the variable at index i. The program then executes into the “L3” block and increments the index, and the control flow is reunited. The program then runs into “L2” and jumps back to block “L4”. Eventually, the program will not jump back to “L4” and keep executing when “i” is 3. The program then loads the arguments for the ”printf” call and jumps to that subroutine. The linker adds the code necessary to jump there during the link phase, so no more work is needed. The program then sets up the return for the main function and returns.

One of the main differences between the c code and the assembly is the generated glue code for the functions’ abi. Another big difference is the for loop, where the comparing “i” and the list length occurs after the loop body and not before. The if statement comparing the list element and the largest element is also implemented the opposite of how the programmer

specified. It looks for list elements that are less than or equal to the “largest” variable, then branches; the code conditionally skips assigning the largest var rather than conditionally executing it. The compiler also expands the “LIST_LEN” define and the compiler expands the code in some sections accordingly; the loop exits when “i” is 3, even though the programmer never wrote “3” in the code. The compiler also doesn’t store some of the constant variables in the code as constants, instead putting them in their own sections outside of the main function. The compiler also includes a section of constant sections after the main function,which the main function refers to.