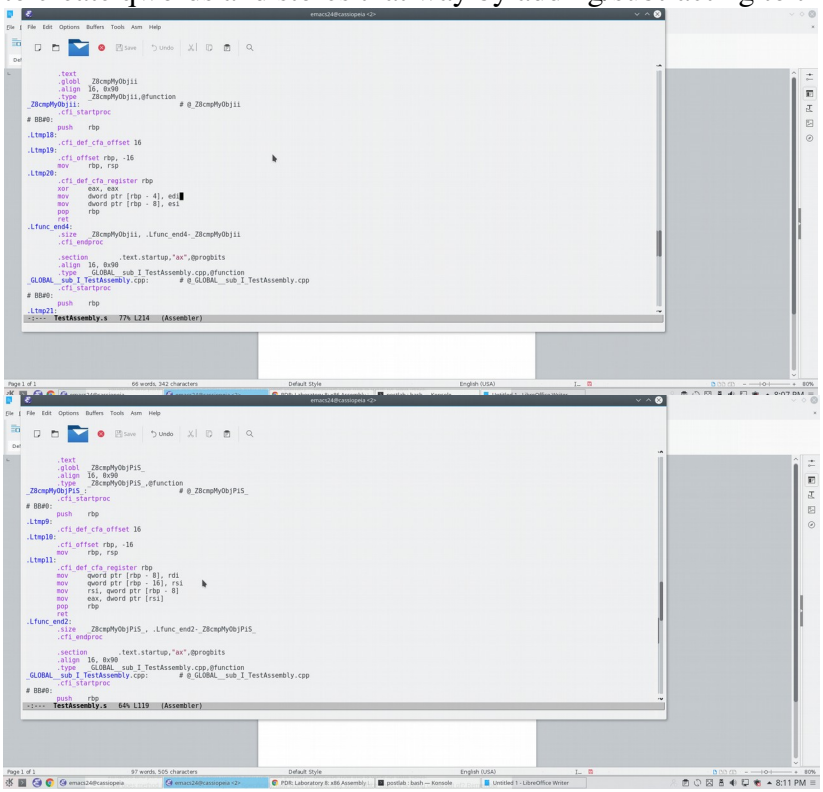
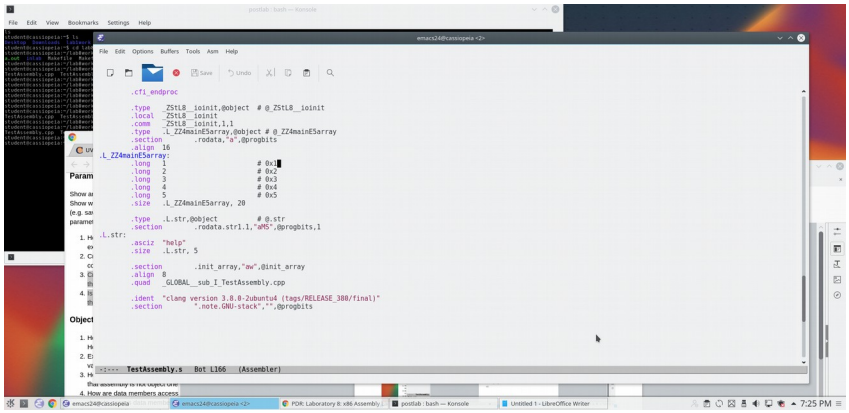
Postlab8.pdf rr5de

Parameter passing

1and 2:::Using the below printscreens as proof. It passed the “di” and “si” and beyond elements when passing by value. It also moved an element in to the stack to save it.For passing by reference, it appears to create qwords and stores that way by adding/subtracting to the current rbi, rsi register.



3:::::Using the below printscreen as proof. It appears to create the array at the end of the code, under a function itself. The callee gets the array data by calling that function plus 4,8,16 to get the index after the 0th. It also stored the array indexes as longs even though the c++ code had the array as an int array.



4. When one of the pointers passed in the above method were changed into a reference, there was no change in assembly code. It can be assumed there is no difference in the code. What is passed in the register is a qword which appears to contain the actual address location on the computer.

Objects

1. Using the source: https://stackoverflow.com/questions/38513625/where-in-which-memory-segment-are-objects-of-a-class-stored-in-c which is a stackoverflow answer (what I saw elsewhere corroborated this). Objects are kept together through the use of heap structures. That is essentially how they are laid out in memory.

2. Using the same source as above. Since objects are stored together in a heap structure, elements of the object are accessed similar to an array. Using the object pointer and an offset to that element in the heap. The struc command can also be used to assign elements.

3. and 4 Using the source: http://www.drdobbs.com/embedded-systems/object-oriented-programming-in-assembly/184408319 From what this source argues, having used the "struc" command to describe an object. It then assigns jmp and get commands in the struc. The call command is then used to get the method needed. There appears to be no error checking here, so assembly will not warn you if you are rnning a method for an object that is not part of the object (or can use the object). This also provides the answer to number 4, as you can use methods to get elements of the object (for outside the member function). Inside the members functions, it appears from the source that you can directly call from the struc lines.

5. I was unable to find the answer to this question. There does not appear to be any privacy to assembly objects.