A close-up of a sign

AI-generated content may be incorrect.

Baron Eiley

EE 277 Embedded SoC Design

A black text on a white background

AI-generated content may be incorrect.

Note to professor:

A screenshot of a computer

AI-generated content may be incorrect.I was unable to attain the correct license from the arm sales-person, however, I found a workaround to complete the lab using STM32CubeIDE software and my NucleoSTM-F429Z1 microcontroller board (https://os.mbed.com/platforms/ST-Nucleo-F429ZI/). The STM32CubeIDE software has the same base infrastructure as the ARM DS IDE. See figure below. It supports debugging of the stack register and supports the compilation of inline assembly using custom startup files (.s). I will explain more in the video submission on how I was able to run lab two using this software and this hardware.

A screenshot of a computer

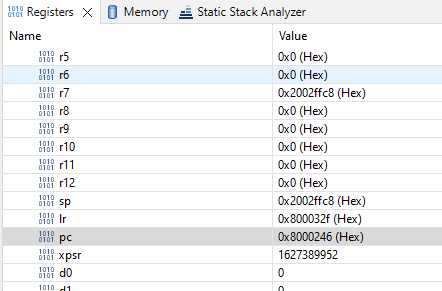
AI-generated content may be incorrect.This is my implementation of the string reverse function in inline assembly.

The reverse string function operates by using two pointers to scan forward to find the null terminator and the steps back to point to the last char. The program continuously swaps the character they point to from one end to the other. Once they meet in the middle, the string has been reversed and the stores the value into the register and the branch returns.

A white text with black text

AI-generated content may be incorrect.

The value of the stack point (SP) is 0x2002ffc8, the value of the link register (LR) is 0x800032f and the value of the PC is 0x8000246.



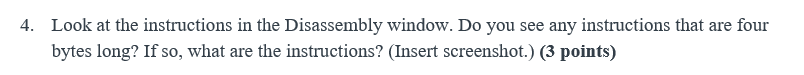
A white background with black text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.The instruction that is highlighted is “08000246: ldr r3, [pc, #48] ; (0x8000278 <main+56>)” This makes sense as this is the first part of the assembly instruction that will load “Hello World” into const char a[]: The address at this point is 0x8000246, which is the same value as the program counter as the PC is responsible for keeping track of where the current execution of the code.

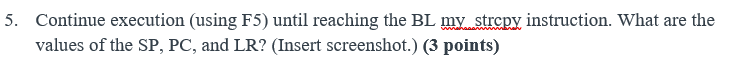
Since I am using a different software for this lab, I have a similar button that lets me step by instruction. When I press the F5 key, the two registers that changed were r0, and r1, in addition to the PC. They relate to the instruction being executed because they are loading in the arguments from the function to the desired registers. (More registers are highlighted because the software compiler I am using compiles the c code in my main.c different than the ARM DE IDE). Nonetheless, this shows the arguments from the function being loaded in.



A yellow text on a white background

AI-generated content may be incorrect.

According to my disassembly window, there is one instruction that is 4 bytes long as it relates to my\_strcpy function: 0800025c: bl 0x80001ec <my\_strcpy>

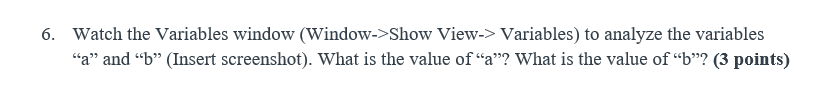
A screenshot of a computer program

AI-generated content may be incorrect.

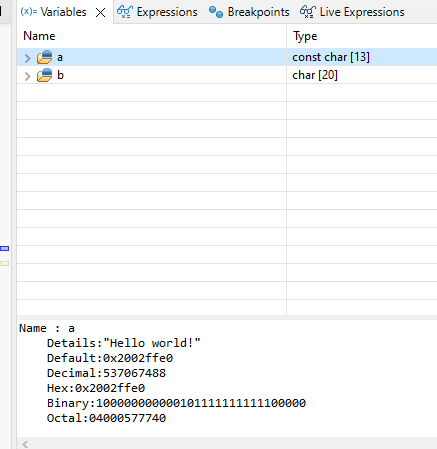
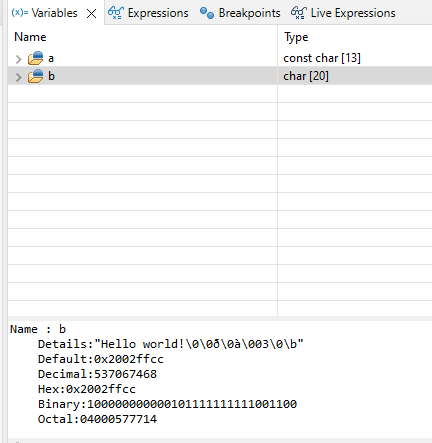
Once we loop through the end of my\_strcpy, the values of the SP is 0x2002ffc8, the values of the PC is 0x80001f8, and the value of the LR register is 0x8000261. The SP did not change during these execution and this is because the function did not use the command “PUSH” or “POP”. The SP point always points to the top of the stack and if these commands aren’t used with the compiler within this STM software, therefore the SP won’t change.

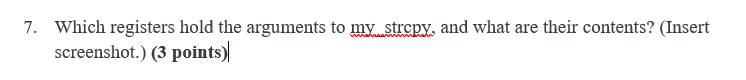
A screenshot of a computer

AI-generated content may be incorrect.



At this point in the execution, the values of variable a is "Hello world!" and the value of variable b is "Hello world!\0\0ð\0à\003\0\b". The value of b makes sense as the as it copied the value of a and the rest is just the uninitialized garbage since the char b is set to a element array of 20.

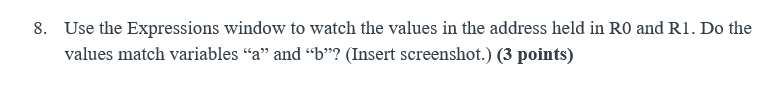




According to my software version, the registers that hold the arguments of my\_strcpy is r0 and r1.

A screenshot of a computer

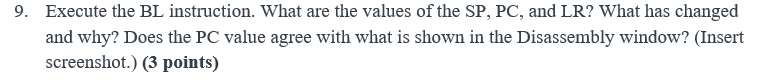
AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.Yes, the values match.

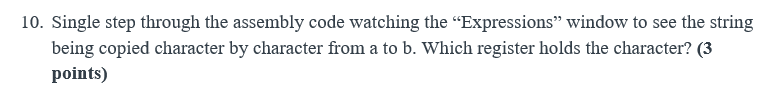
After the execution of the BL instruction , the values of the SP is 0x2002ffc8, the PC is 0x80001fa, and the value of the LR is 0x8000269. The program counter changed because the we looped through to the next instruction. The link register is pointing to the location we need to jump back to after the function finishes. In this scenario, the program counter matches that of the disassembly window. Note: You must click on the value to see the real value of the register. This software uses a different representation.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.



In step 5 of this lab, we branched through the my\_strcpy function already. So now we are in the my\_capitalize function. I will loop through this function and analyze the Expressions window from there. At this point, the register that holds the character is R1.

A screenshot of a data

AI-generated content may be incorrect.



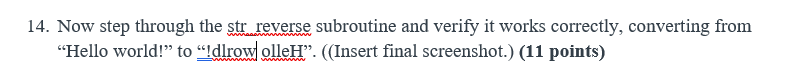
After execution of BX lr, the value of the of the PC is 0x8000268.

A screenshot of a computer

AI-generated content may be incorrect.



As we learned from the previous lab, we learned that the relationship between the PC and the LR is that the LR holds the place where the program needs to jump back to in the main subroutine and once it does, the value of the PC will be the same as the LR because that is now where the current execution is.



I ran the code with the my\_capitalize function still active but this question assumes that we run the copied lower case char b through my str\_reverse. I will include both screenshots and recompile the code to do #14

A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

