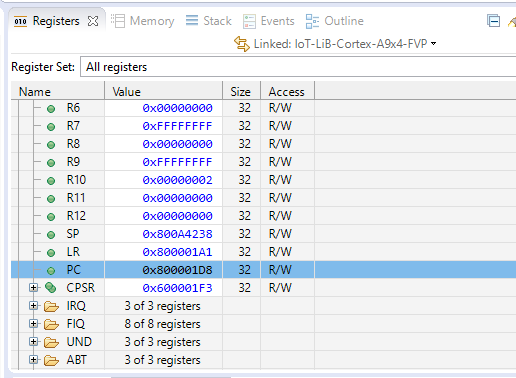
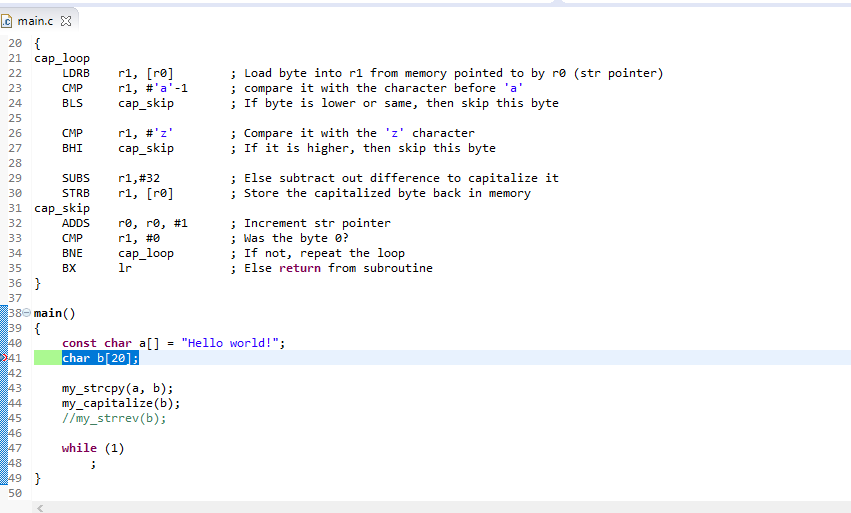
Baron Eiley

EE 277 Embedded SOC Design

1. Run the program until the opening brace in the main function is highlighted. Open the Registers window (Window->Show View->Registers). What are the values of the stack pointer (SP), LR, and the PC? (Insert screenshot of your editor.) **(5 points)**

I tried setting the breakpoint to the first opening brace on the main function the program wouldn’t let me set it there, however, it let me set it on line 41, “char b[20];”

The value of the stack pointer (SP) register at the point is 0x800A4238. The value of the Link register (LR) is 0x800001A1. The value of the program counter (PC) 0x800001D8**.**



\

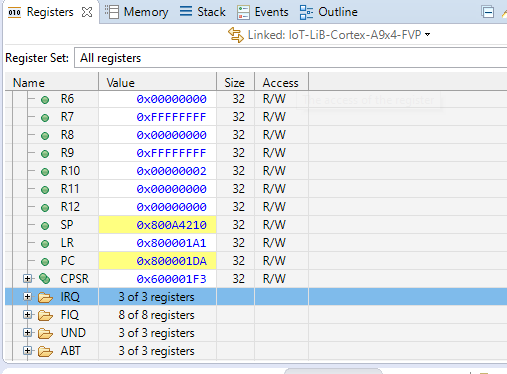
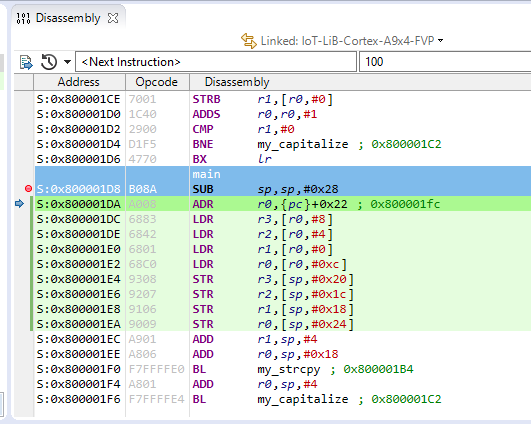
1. Open the Disassembly window (Window->Show View->Disassembly). Which instruction is highlighted, and what is its address? How does this address relate to the value of PC? (Insert screenshot of your disassembly window.) **(5 points)**

A screenshot of a computer

AI-generated content may be incorrect.

The instruction that is highlighted is “SUB sp,sp,#0x28”. The address is S:0x800001D8. the address is related to the program counter as the PC always stores the address of the instruction that is currently being performed. This is evident in that the address in the PC is the same as the address in the disassembly.

1. Switch to “Step by Instruction” mode by clicking  in the Debug Control window. Step one machine instruction using the F5 key while the Disassembly window is selected. Which two registers have changed (they should be highlighted in the Registers window), and how do they relate to the instruction just executed? (Insert the screenshot.) **(5 points)**



The two registers that changed were the the SP and the PC. Knowing that the stack pointer acts as a place holder for the location of the top of the stack, it makes sense, the new value of the SP is 0x800A4210 as the previous execution of the instruction subtracted 0x28 from the previous value of 0x800A4238. Secondly, the PC changed to 0x800001DA as that is the address of the currently place in the routine when we jumped to the next instruction.

1. Look at the instructions in the Disassembly window. Do you see any instructions that are four bytes long? If so, what are the instructions? (Insert screenshot.) **(5 points)**

A screenshot of a computer program

AI-generated content may be incorrect.

Looking at the disassembly window, specifically, the “Opcode” column, we can see that two instructions my\_strcpy and my\_capitilize are 32 bits long. We know this because it is hex value where each hex character is 4 bits long, 8 characters equates to 32 bits.

1. A screenshot of a computer

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

   AI-generated content may be incorrect.Continue execution (using F5) until reaching the BL my\_strcpy instruction. What are the values of the SP, PC, and LR? (Insert screenshot.) **(5 points)**

Once the debug program reached BL my\_strcpy the value of the SP is 0x800A4210, the value of the PC is 0x800001F0, and the value of the LR is 0x800001A1.

1. Watch the Variables window (Window->Show View-> Variables) to analyze the variables “a” and “b” (Insert screenshot). What is the value of “a”? What is the value of “b”? **(5 points)**

The value of “a” is the character string we asserted as “Hello World” Secondly, the value of “b” is variables is metadata as we have not assigned any particular value to that variable in the code “**char** b[20];”

A screenshot of a computer

AI-generated content may be incorrect.

1. A screenshot of a computer

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

   AI-generated content may be incorrect.Which registers hold the arguments to my\_strcpy, and what are their contents? (Insert screenshot.) **(5 points)**

The registers that hold the arguments to the arguments are R0 (source: 0x800A4228) and R1 (destination: 0x800A4214) We can confirm the address within the stack with that loaded into R0 and R1.

1. Use the Expressions window to watch the values in the address held in R0 and R1. Do the values match variables “a” and “b”? (Insert screenshot.) **(5 points)**

A screenshot of a computer

AI-generated content may be incorrect.

Yes, they match.

1. A screenshot of a computer

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

   AI-generated content may be incorrect.Execute the BL instruction. What are the values of the SP, PC, and LR? What has changed and why? Does the PC value agree with what is shown in the Disassembly window? (Insert screenshot.) **(5 points)**

After the execution of the BL, The value of the SP is 0x800A4210, the value of the PC is 0x800001B4, and the value of the LR is 0x800001F5. Yes. The PC, agrees with the address in the disassembly window.

1. A screenshot of a computer

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

   AI-generated content may be incorrect.Single step through the assembly code watching the “Expressions” window to see the string being copied character by character from a to b. Which register holds the character? **(5 points)**

After two iterations, we can see that the letters are being copied correctly from a to b. The register that holds the character is the R2 register.

1. A screenshot of a computer

   AI-generated content may be incorrect.What are the values of the character, the src pointer, the dst pointer, the LR, and the PC when the code reaches the last instruction in the subroutine (BX lr)? (Insert screenshot.) **(5 points)**

A screenshot of a computer

AI-generated content may be incorrect.

The value of the src pointer is 0x8004A225 and the value of the dst pointer 0x8004A221. The value of the LR is 0x800001F5, and the PC is 0x800001C0.

1. A screenshot of a computer

   AI-generated content may be incorrect.Execute the BX lr instruction. Now what is the value of the PC? **(5 points)**

The value of the PC is the 0x800001F4.

1. What is the relationship between the PC value and the previous LR value? Explain. **(5 points)**

After execution, the PC value equals the previous LR value. The LR temporarily held the return address during the subroutine.

1. A screenshot of a computer

   AI-generated content may be incorrect. Now step through the my\_capitalize subroutine and verify it works correctly, converting from “Hello world!” to “HELLO WORLD!”. ((Insert final screenshot.) **(5 points)**

A screenshot of a computer

AI-generated content may be incorrect.

1. Please explain your debugging experience in your own words **(10 points)**

The debugging experience provided real-world hands-on experience with the ARM architecture. The lab provided understanding of critical components within register of R0 – R4, Stack Pointer, Link Register, and the Program Counter. As an embedded engineer, it is important to understand how the software integrates with the hardware, and this is evident how the software arguments in “my\_strcpy” are saved into the into the hardware stack of R0 and R1. Secondly, as the assembly code looped through the debug, we analyzed the increments of the program counter and understood the function of the PC as it relates to current location of the program execution. Thirdly, we analyzed the function of the stack pointer being the register that points to the top of the stack. In the case of assembly functions, we saw the stack pointer pointing to the top of the stack in memory as we looped through the code storing local variables. Finally, as the assembly jumped in and out of the custom functions, the link register showed the location of where the program needed to return to after the completion of the function. Overall, the lab provided critical step by step debugging practice which is critical when pinpointing where in the code a possible bug might be when troubleshooting. Understanding the registers, their functions and deterministic behaviors are paramount when designing embedded systems in the future.