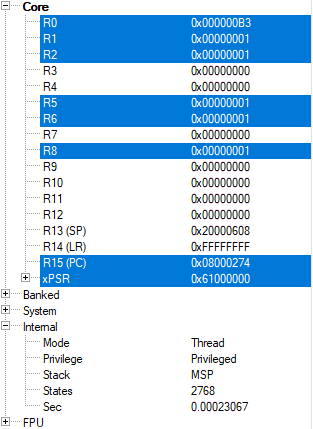
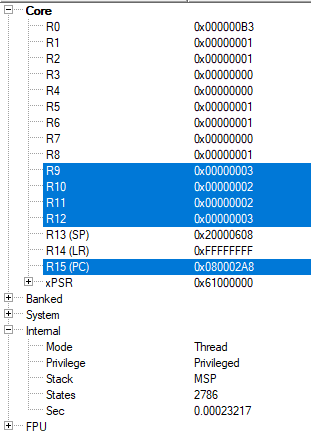
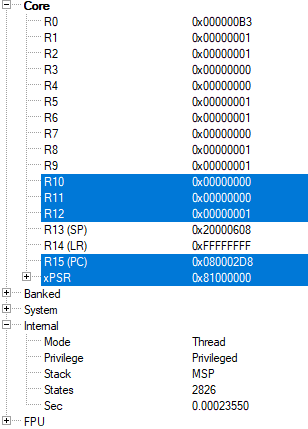
Q3

In this question, firstly we give a constant 8-bit data in memory and 13-bit memory section with full of zeros. We give the 8-bit data to r0 with LDR operation. We shifted the r0 to access the bits of the locations and give them to registers such as r1=first bit,r2=second bit etc. For doing that, we used MOVS operation with barrel shifter, and we iterated the wanted bit into carry. After that we used ADC with the cleaned registers. After that we cleared the register for the wanted p0, p1, p2, p4, p8 bits. Then we perform the given rules and find the values of the parity bits before the mode operation. For doing the mode operation, we created a loop and checked the given value, if it is less than two, we give the mode operation’s result to the p1, p2, p4, p8 parity bits. After that, we added all the bits and performed mode operation for the p0 parity bit. Then we perform Logical Shift Left operation for the moving proper places to our bits. And after, we added the results of the Logical Shift Left operation and stored in r0. Finally, we LDR the 13-bit data location to r1 and we stored the final result r0 in there with STR operation.

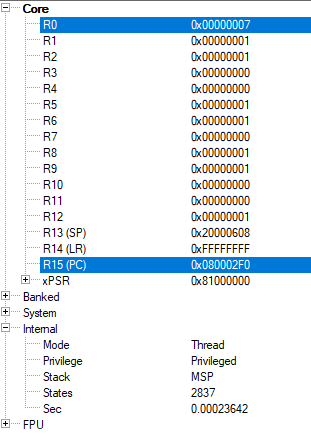
We used the value given in assignment pdf 0xB3.r8 assigned to the eighth bit, r7 assigned to seventh bit etc. Before finding the parity bits our memory looked like this.



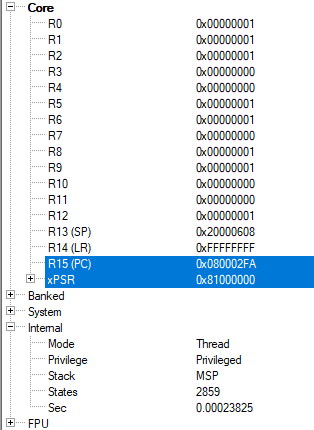
Before the mode operation, our parity bits (r9=p1, r10=p2, r11=p4, r12=p8) looked like this.



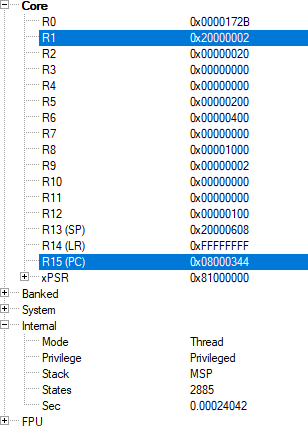
After the mode operation our parity bits looked like this. Still, we did not calculate the p0 bit. Now we calculate the p0 parity bit.



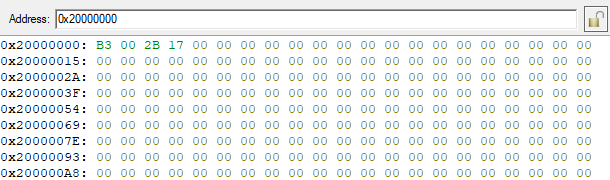
After calculating the parity bit p0, we found it 7.we stored it in the r0 because we used all other registers to hold the 12 bit data bit by bit and we are not using the address of the 8-bit data anymore.



After all the calculations, we found the parity bits and held them in the registers. After this step of the project, we added properly located bits to the r0 which holds the p0 bit. We did not perform any shift operation in r0, and because of this we added all other values to the r0.



In this photo, we showed that our result is in the r0 and the allocated memory address for the output of the program is in r1. After that we stored the r0 into that address.



In this photo, our memory is shown after our program stopped. As we see, our compiler added 2 bytes of padding to the memory because of the performance. And we see that our result is written as 2B-17. It is because the lower memory address holds the lower bits of the data. Here is the program we wrote.

