ELE 206 – LAB 6 – FINAL WRITE-UP

**Assembly Code:** This code performs a Ceaser cipher on a given source text, given a shift value. It shifts the characters to backward instead of forwards. The input string is stored from mem[19], and should be null ended. The shift value should be stored at mem[16].

1. LD R0 #15 //Loads R0 with the shift value (Stored at mem[16])
2. LEA R6 #24 //Loads R6 with PC (= 2) + 24 = 26, Since we cannot directly load because +26 takes up

//more than 5 bits

1. NOT R0 R0 //Complements the value in R0
2. ADD R0 R0 #1 //Adds one to R0 to complete the 2's complement conversion
3. LEA R5 #-102 //Loads R5 with ASCII for 'a', This is a negative value for subtraction later
4. LDI R1 #12 //Loads R1 with 0, which is the starting index. R1 is used to store the index when

//iterating through the source string

1. LDR R3 R1 #19 //Loads the ASCII of the character of the source string, at the index given by R1, into R3.
2. BRZ #7 //Jumps to HALT (at mem[16]) if that character is zero
3. ADD R2 R3 R0 //Adds the shift value(R0) to the current character(R3) and stores it in R2
4. ADD R4 R2 R5 //Adds the negative value of 'a' to the ciphered text, to check if the cipher text has

//underflowed

1. BRZP #1 //Skips the next instruction if the result in the prev. instruction is not negative
2. ADD R2 R2 R6 //If it is negative (ie. the encryption has caused an underflow) it adds 26 (R6) to the

//ciphered char

1. STR R2 R1 #19 //Replaces the current letter with the final ciphered char result
2. ADD R1 R1 #1 //Increments the index (R1)
3. JSR #-9 //Jumps back to get the next character
4. HALT //End of Algorithm
5. 000F //Shift value
6. 0000 //Initial Index Value
7. 0011 //Address of Initial Index Value
8. 0061 //Start of string
9. 0062
10. 0063
11. 0064
12. 0065
13. 0066
14. 0067
15. 0068
16. 0069
17. 006A
18. 006B
19. 006C
20. 006D
21. 006E
22. 006F
23. 0070
24. 0071
25. 0072
26. 0073
27. 0074
28. 0075
29. 0076
30. 0077
31. 0078
32. 0079
33. 007A
34. 0000 //Null pointer to end string

**Feedback**

This lab was extremely exciting as we got to literally build a processor!! The process was a lot less intimidating than we thought at first, but it was certainly quite time consuming. It took about 14 hours. A good majority of our time was spent debugging the Verilog and Assembly code. But it was certainly worth it in the end.