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].

```
0. LD R0 #15       //Loads R0 with the shift value (Stored at mem[16])
1. LEA R6 #24      //Loads R6 with PC (= 2) + 24 = 26, Since we cannot directly load because +26 takes up
                   //more than 5 bits
2. NOT R0 R0       //Complements the value in R0
3. ADD R0 R0 #1    //Adds one to R0 to complete the 2's complement conversion
4. LEA R5 #-102    //Loads R5 with ASCII for 'a', This is a negative value for subtraction later
5. 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
6. LDR R3 R1 #19   //Loads the ASCII of the character of the source string, at the index given by R1, into R3.
7. BRZ #7          //Jumps to HALT (at mem[16]) if that character is zero
8. ADD R2 R3 R0    //Adds the shift value(R0) to the current character(R3) and stores it in R2
9. ADD R4 R2 R5    //Adds the negative value of 'a' to the ciphered text, to check if the cipher text has
                   //underflowed
10. BRZP #1        //Skips the next instruction if the result in the prev. instruction is not negative
11. ADD R2 R2 R6   //If it is negative (ie. the encryption has caused an underflow) it adds 26 (R6) to the
                   //ciphered char
12. STR R2 R1 #19  //Replaces the current letter with the final ciphered char result
13. ADD R1 R1 #1   //Increments the index (R1)
14. JSR #-9        //Jumps back to get the next character
15. HALT           //End of Algorithm
16. 000F           //Shift value
17. 0000           //Initial Index Value
18. 0011           //Address of Initial Index Value
19. 0061           //Start of string
20. 0062
21. 0063
22. 0064
23. 0065
```

```
24. 0066
25. 0067
26. 0068
27. 0069
28. 006A
29. 006B
30. 006C
31. 006D
32. 006E
33. 006F
34. 0070
35. 0071
36. 0072
37. 0073
38. 0074
39. 0075
40. 0076
41. 0077
42. 0078
43. 0079
44. 007A
45. 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.