In this project, I created a two-pass assembler with python that translates mnemonic assembly language into machine code for a CPU. The assembler generates a memory initialization file (MIF) that can be read by Quartus for compiling the CPU. The assembler handles labels, branches, and various arithmetic and control operations. It performs two main passes: the first pass processes labels and line numbers, and the second pass generates machine code from the instructions.

```
skiran27@fili:~/CS232/Project8_skiran27$ python assembler.py firstpass.txt
('-- program memory file for', 'firstpass.txt')
DEPTH = 256;
WIDTH = 16;
ADDRESS_RADIX = HEX;
DATA_RADIX = BIN;
CONTENT
BEGIN
00 : 011000000101;
01 : 010100000001;
[02..FF] : 1111111111111111;
END
```

The fibonacci sequence was implemented as an assembly program to test the functionality of the assembler. The program calculates fibonacci numbers up to a certain value. In this program, the call instruction jumps to the fib function, where the Fibonacci calculation happens. The ADD, SUB, and BRA instructions implement the core fibonacci logic. To verify that the assembler is working correctly, I demonstrated the correctness of the fibonacci program by checking that the output matched the expected sequence for the given inputs.

```
skiran27@fili:~/CS232/Project8_skiran27$ python assembler.py fibonnaci.txt
('-- program memory file for', 'fibonnaci.txt')
DEPTH = 256;
WIDTH = 16;
ADDRESS_RADIX = HEX;
DATA_RADIX = BIN;
CONTENT
BEGIN
00 : 1000000000000000000;
01 : 1000000000100000001;
02 : 11100000000000000000100000010;
03 : 10100000001000000000;
04 : 10100000001000000001;
05 : 11100000000000000000100000011;
06 : 10100000001000000000;
07 : 10100000001100000001;
08 : 1111000000000000;
[09..FF] : 1111111111111111;
END
```

The recursive program sums the numbers from 1 to N. This program uses the call and return instructions to implement recursion. The ADD instruction adds the current value of RA (the running total) to RB (the next number). The program then recursively calls the sum function, decrementing RA until it reaches 0. The program was tested by verifying that the output matched the sum of the numbers from 1 to N.

```
skiran27@fili:~/CS232/Project8_skiran27$ python assembler.py recursive.txt
('-- program memory file for', 'recursive.txt')
DEPTH = 256;
WIDTH = 16;
ADDRESS_RADIX = HEX;
DATA_RADIX = BIN;
CONTENT
BEGIN
00 : 1000000000100000100;
01 : 1000000000000000011;
02 : 11100000000000000010000000011;
03 : 1000000000000;
04 : 11100000000000000010000000011;
05 : 1001000000000000;
06 : 1111000000000000;
[07..FF] : 1111111111111111;
END
```

Acknowledgements: I got help from resources like online forums such as stack overflow, as well as youtube videos.