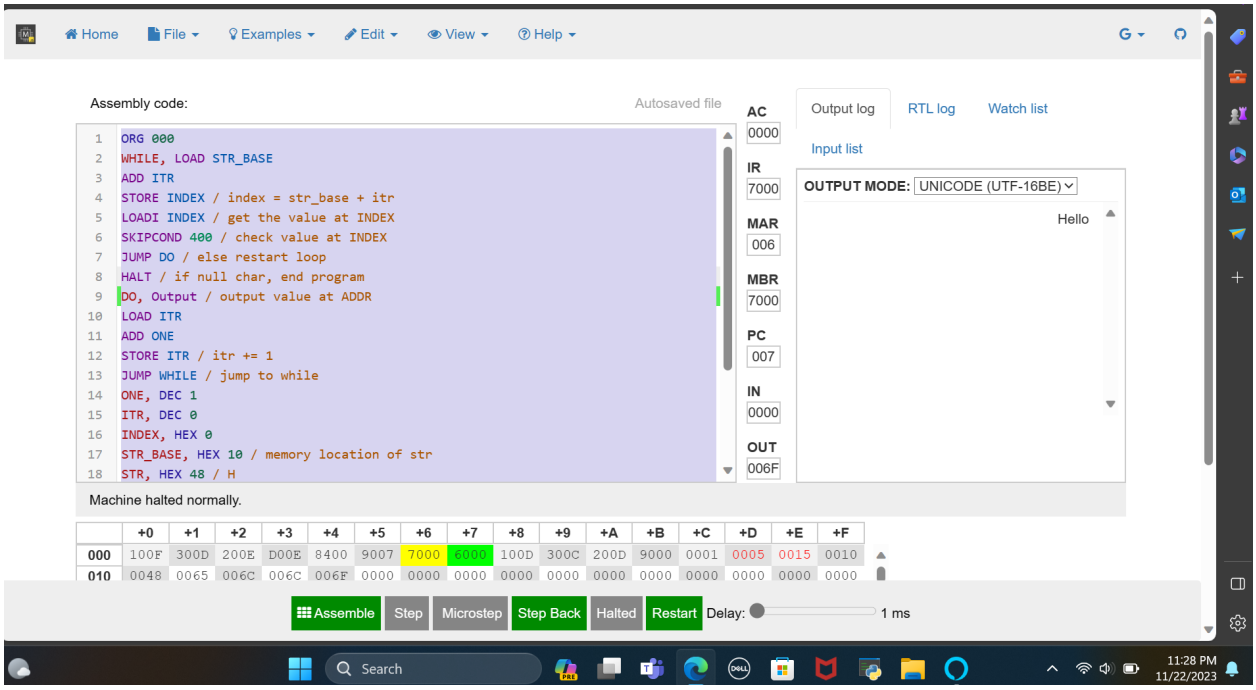
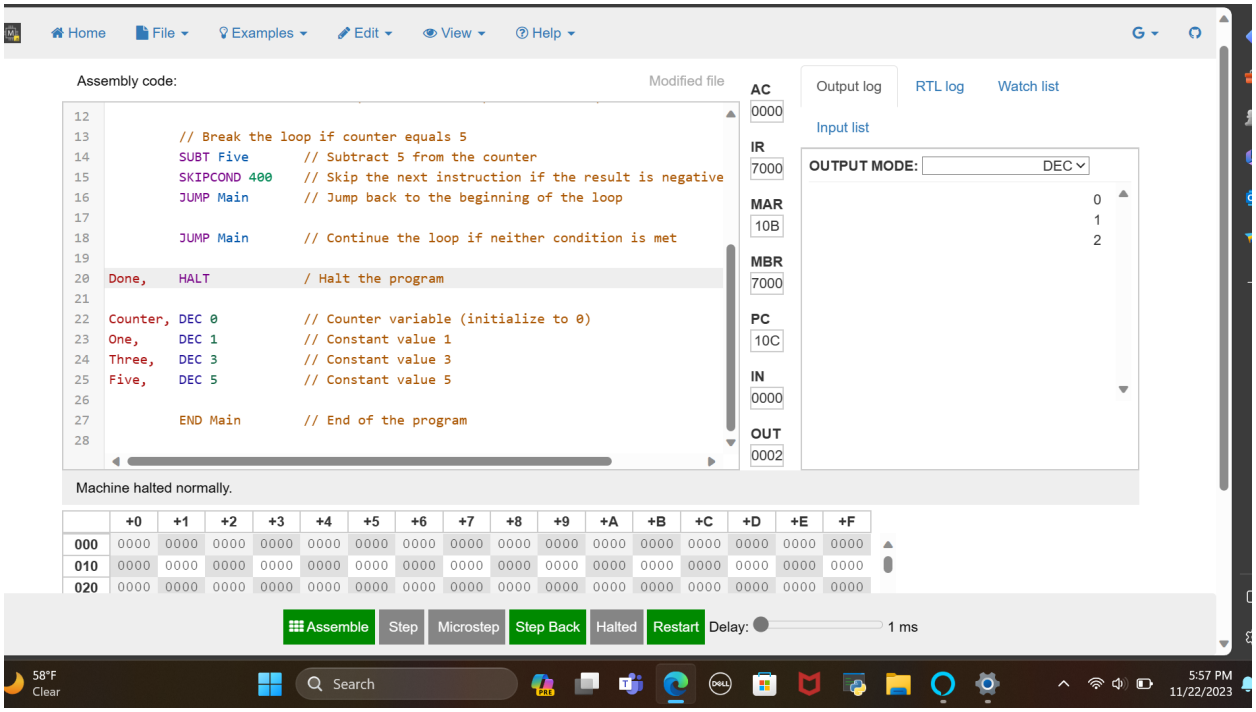
Question no.1

```
ORG 000
WHILE, LOAD STR_BASE
ADD ITR
STORE INDEX / index = str_base + itr
LOADI INDEX / get the value at INDEX
SKIPCOND 400 / check value at INDEX
JUMP DO / else restart loop
HALT / if null char, end program
DO, Output / output value at ADDR
LOAD ITR
ADD ONE
STORE ITR / itr += 1
JUMP WHILE / jump to while
ONE, DEC 1
ITR, DEC 0
INDEX, HEX 0
STR_BASE, HEX 10 / memory location of str
STR, HEX 48 / H
HEX 65 / e
HEX 6C / l
HEX 6C / l
HEX 6F / o
HEX 0 / NULL char
```



#question no 2

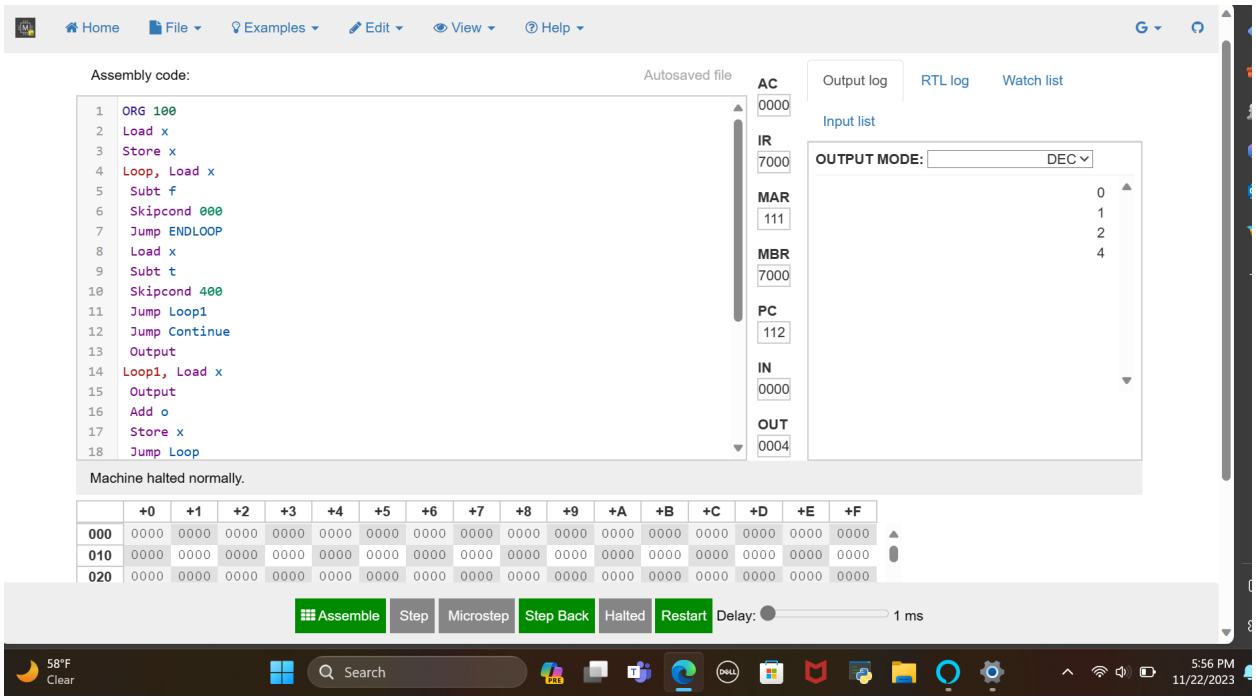
```
   ORG 100      // Start address of the program

Main,    LOAD Counter   // Load the counter
         OUTPUT         / Output the current value of the counter
         ADD One        // Increment the counter
         STORE Counter  // Store the updated counter value

         // Check if counter is equal to 3
         SUBT Three     // Subtract 3 from the counter
         SKIPCOND 000   // Skip the next instruction if the result is zero (counter == 3)
         JUMP Done      // Jump out of the loop if counter equals 3

         // Break the loop if counter equals 5
         SUBT Five      // Subtract 5 from the counter
         SKIPCOND 400   // Skip the next instruction if the result is negative (counter < 5)
         JUMP Main      // Jump back to the beginning of the loop

         JUMP Main      // Continue the loop if neither condition is met

Done,    HALT           / Halt the program

Counter, DEC 0          // Counter variable (initialize to 0)
One,     DEC 1          // Constant value 1
Three,   DEC 3          // Constant value 3
Five,    DEC 5          // Constant value 5

         END Main       // End of the program
```



#question no. 3

```
ORG 100
Load x
Store x
Loop, Load x
 Subt f
 Skipcond 000
 Jump ENDLOOP
 Load x
 Subt t
 Skipcond 400
 Jump Loop1
 Jump Continue
 Output
Loop1, Load x
 Output
 Add o
 Store x
 Jump Loop
ENDLOOP, Halt
Continue, Load x
 Add o
 Store x
 Jump Loop
```

x, Dec 0
t, Dec 3
o, Dec 1
f, DEC 5
END 100



#question no. 4

```
    ORG 100          // Start at address 100
    LOAD ZERO        // Initialize product to 0
    STORE PRODUCT

    LOAD COUNTER     // Load the counter value (3 in this case)
    STORE COUNTER

MULTIPLY_LOOP, LOAD PRODUCT  // Load the current product
    ADD NUMBER           // Add the number (4 in this example) to the product
    STORE PRODUCT        // Store the new product

    LOAD COUNTER         // Load the current counter
    SUBT ONE             // Decrement the counter
    STORE COUNTER        // Store the new counter value

    SKIPCOND 400         // Check if the counter is zero
    JUMP MULTIPLY_LOOP   // If the counter is not zero, repeat the loop
```

```
    LOAD PRODUCT          // Load the final product
    OUTPUT               / Output the final product
    HALT                / End the program

NUMBER,   DEC 4          // The number to be multiplied (4)
COUNTER,  DEC 3          // The number of times to add the number (3)
PRODUCT,  DEC 0          // The product, initially 0
ONE,      DEC 1          // A constant representing the number 1
ZERO,     DEC 0          // A constant representing the number 0
```