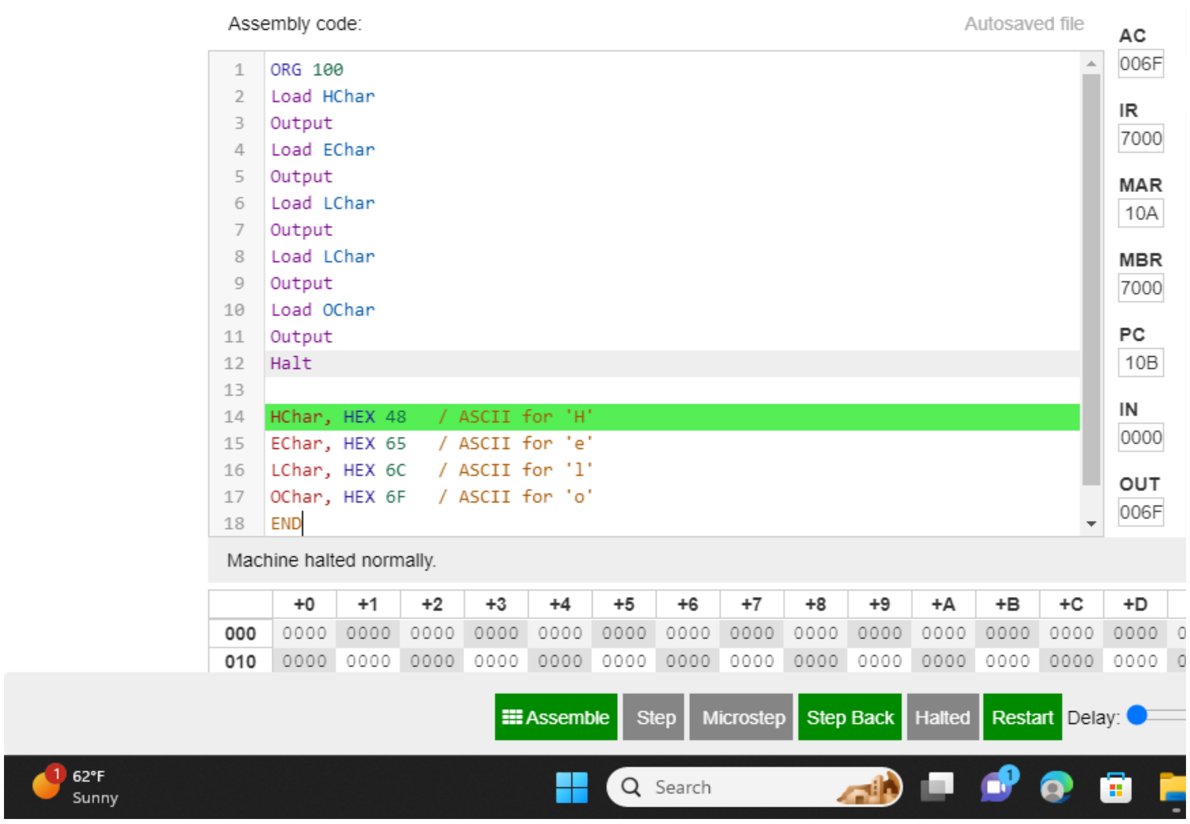
1.
SOURCE CODE:


Assembly code:
Autosaved file

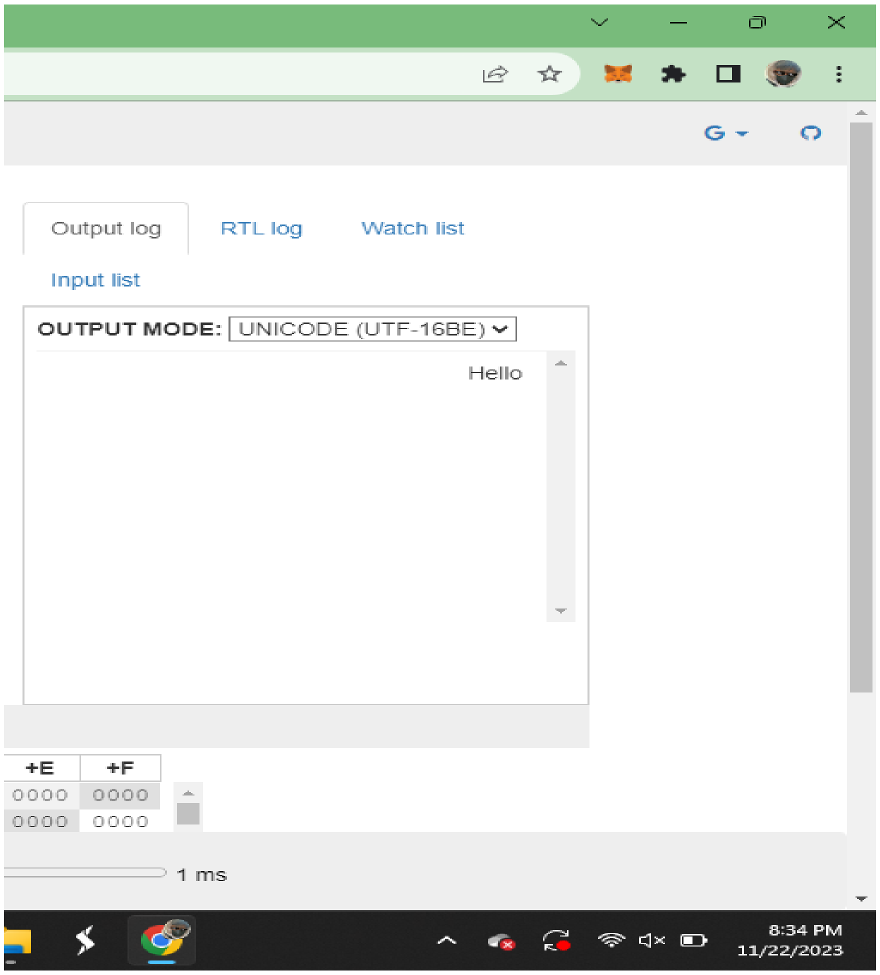
```
ORG 100
Load HChar
Output
Load EChar
Output
Load LChar
Output
Load LChar
Output
Load OChar
Output
Halt

HChar, HEX 48   / ASCII for 'H'
EChar, HEX 65   / ASCII for 'e'
LChar, HEX 6C   / ASCII for 'l'
OChar, HEX 6F   / ASCII for 'o'
END
```

AC 006F
IR 7000
MAR 10A
MBR 7000
PC 10B
IN 0000
OUT 006F

Machine halted normally.

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D
000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

OUTPUT:



OUTPUT MODE: UNICODE (UTF-16BE)

```
Hello
```

```
+E    +F
0000  0000
0000  0000
```

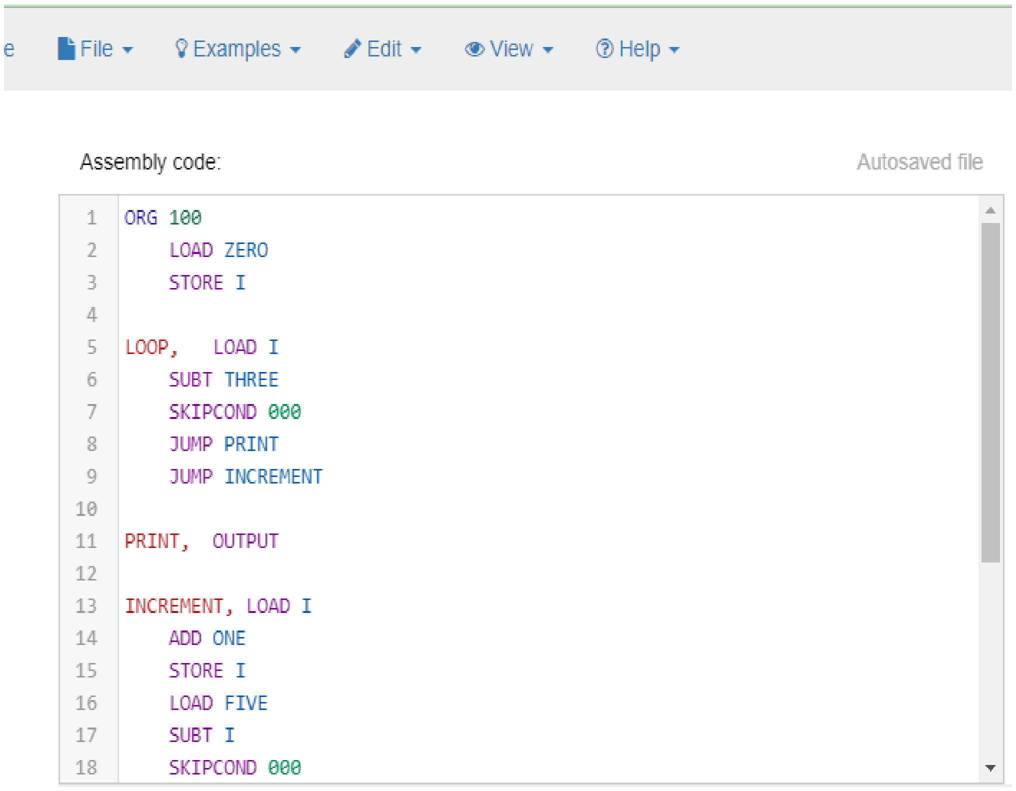
1 ms

8:34 PM
11/22/2023

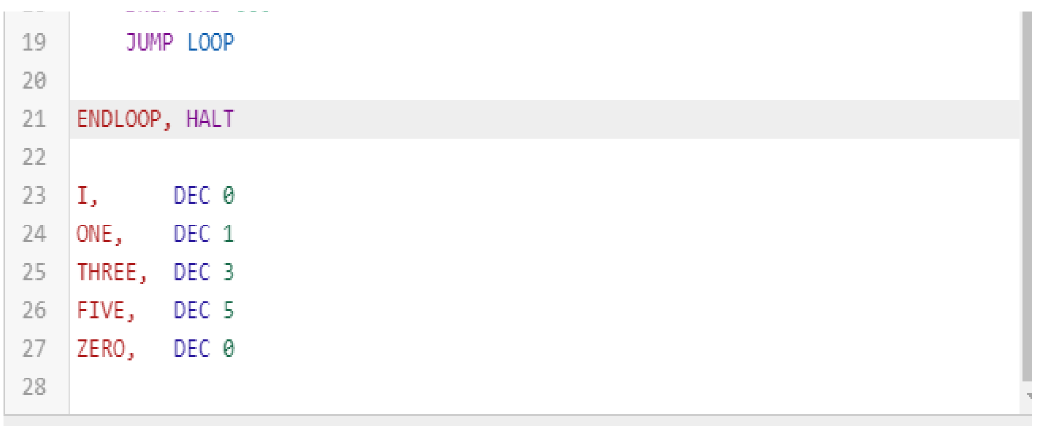
2.
SOURCE CODE:

```
ORG 100
    LOAD ZERO
    STORE I

LOOP,   LOAD I
    SUBT THREE
    SKIPCOND 000
    JUMP PRINT
    JUMP INCREMENT

PRINT,  OUTPUT
```

```
INCREMENT, LOAD I
    ADD ONE
    STORE I
    LOAD FIVE
    SUBT I
    SKIPCOND 000
    JUMP LOOP

ENDLOOP, HALT

I,      DEC 0
ONE,    DEC 1
THREE,  DEC 3
FIVE,   DEC 5
ZERO,   DEC 0
```

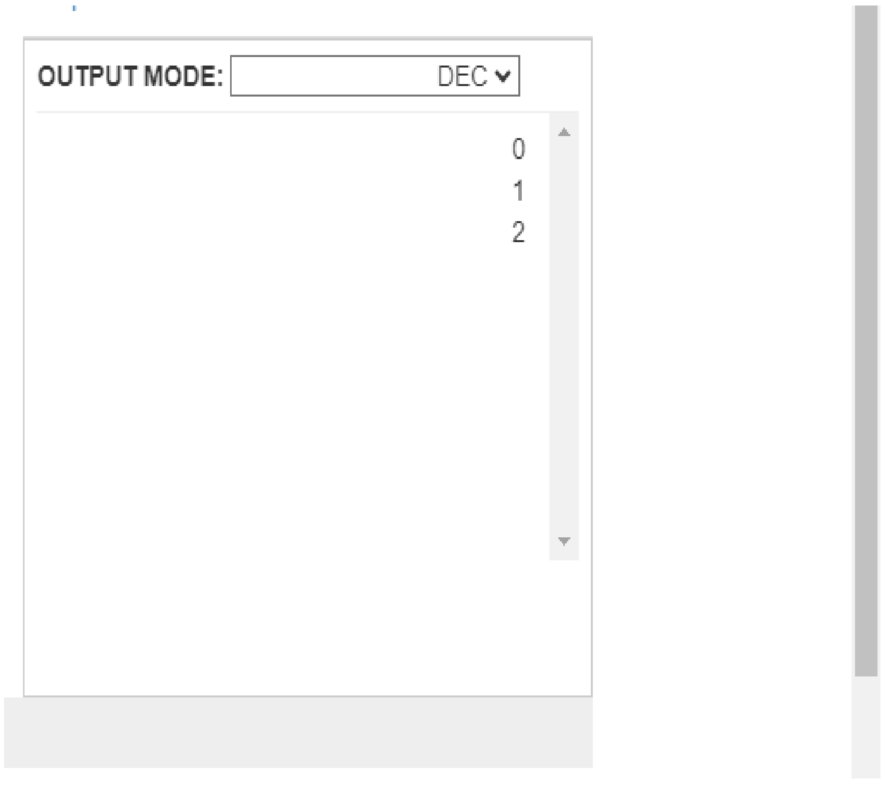


```
ORG 100
    LOAD ZERO
    STORE I

LOOP,    LOAD I
    SUBT THREE
    SKIPCOND 000
    JUMP PRINT
    JUMP INCREMENT

PRINT,  OUTPUT

INCREMENT, LOAD I
    ADD ONE
    STORE I
    LOAD FIVE
    SUBT I
    SKIPCOND 000
```



```
    JUMP LOOP

ENDLOOP, HALT

I,      DEC 0
ONE,    DEC 1
THREE,  DEC 3
FIVE,   DEC 5
ZERO,   DEC 0

```

OUTPUT:



OUTPUT MODE: DEC

```
0
1
2
```

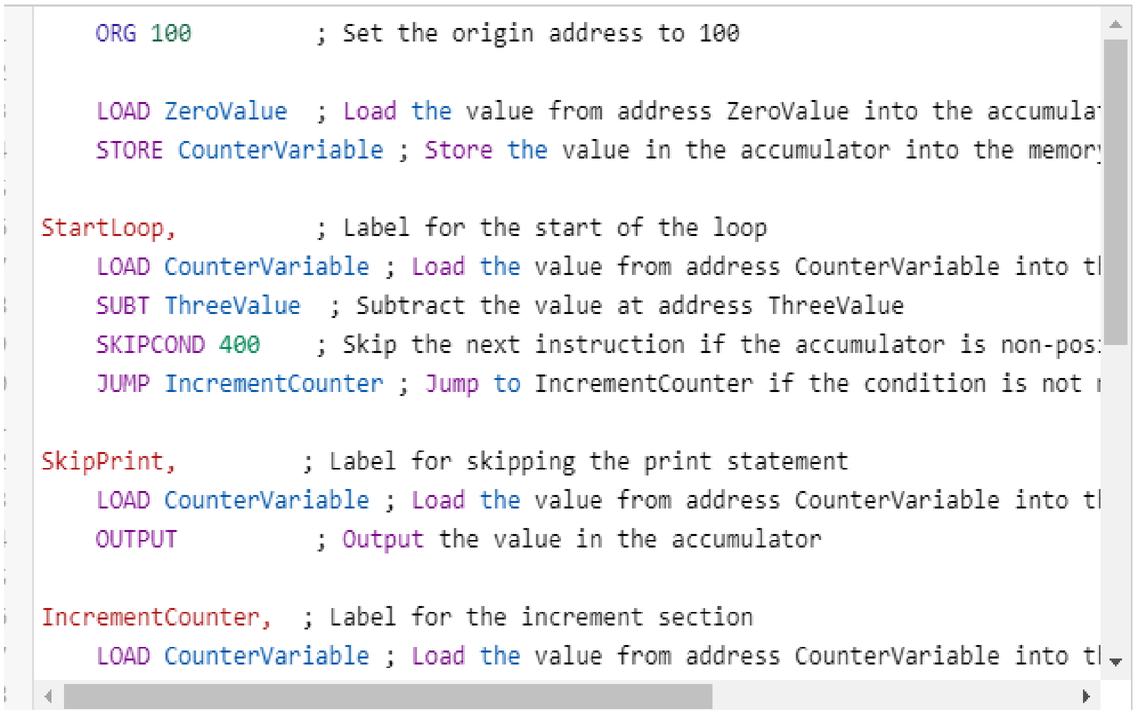
3.

SOURCE CODE:

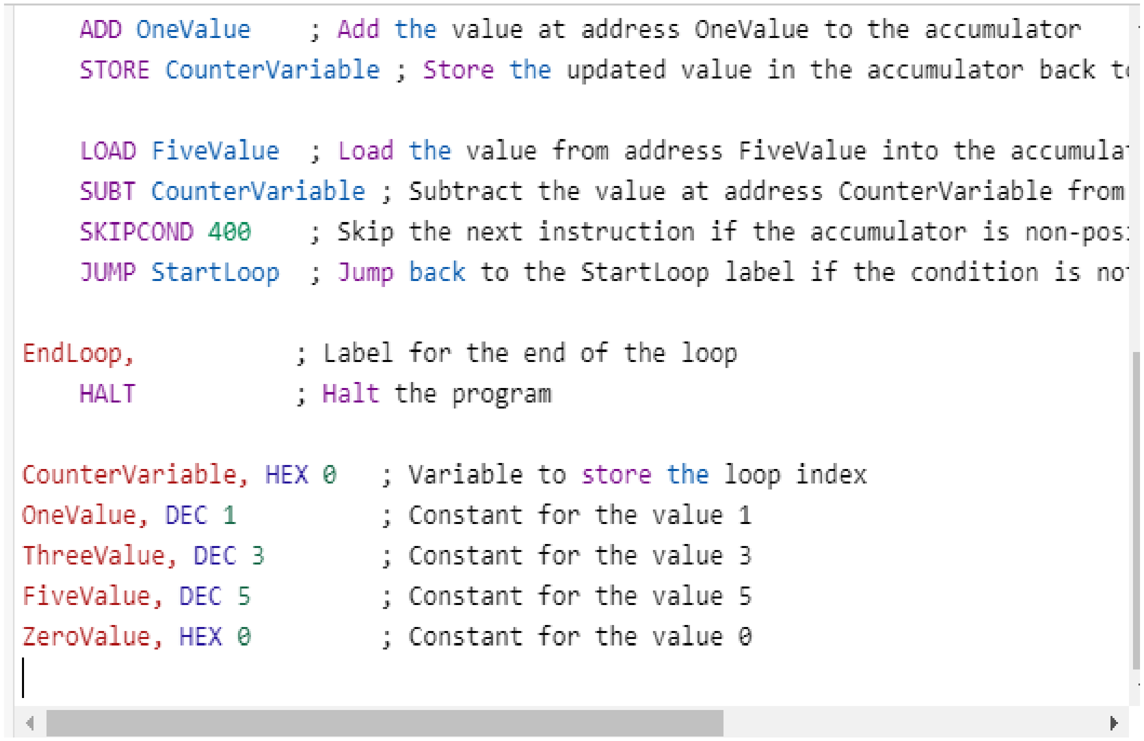
```
    ORG 100         ; Set the origin address to 100

    LOAD ZeroValue  ; Load the value from address ZeroValue into the
accumulator
```

```
    STORE CounterVariable ; Store the value in the accumulator into the memory
location CounterVariable

StartLoop,          ; Label for the start of the loop
    LOAD CounterVariable ; Load the value from address CounterVariable into the
accumulator
    SUBT ThreeValue  ; Subtract the value at address ThreeValue
    SKIPCOND 400    ; Skip the next instruction if the accumulator is non-positive
(i.e., if result is negative or zero)
    JUMP IncrementCounter ; Jump to IncrementCounter if the condition is not
met

SkipPrint,         ; Label for skipping the print statement
    LOAD CounterVariable ; Load the value from address CounterVariable into the
accumulator
    OUTPUT          ; Output the value in the accumulator

IncrementCounter,  ; Label for the increment section
    LOAD CounterVariable ; Load the value from address CounterVariable into the
accumulator
    ADD OneValue    ; Add the value at address OneValue to the accumulator
    STORE CounterVariable ; Store the updated value in the accumulator back to
the memory location CounterVariable

    LOAD FiveValue  ; Load the value from address FiveValue into the
accumulator
    SUBT CounterVariable ; Subtract the value at address CounterVariable from
the accumulator
    SKIPCOND 400    ; Skip the next instruction if the accumulator is non-positive
(i.e., if result is negative or zero)
    JUMP StartLoop  ; Jump back to the StartLoop label if the condition is not met

EndLoop,           ; Label for the end of the loop
    HALT           ; Halt the program

CounterVariable, HEX 0   ; Variable to store the loop index
OneValue, DEC 1          ; Constant for the value 1
```

```
ThreeValue, DEC 3       ; Constant for the value 3
FiveValue, DEC 5       ; Constant for the value 5
ZeroValue, HEX 0       ; Constant for the value 0
```

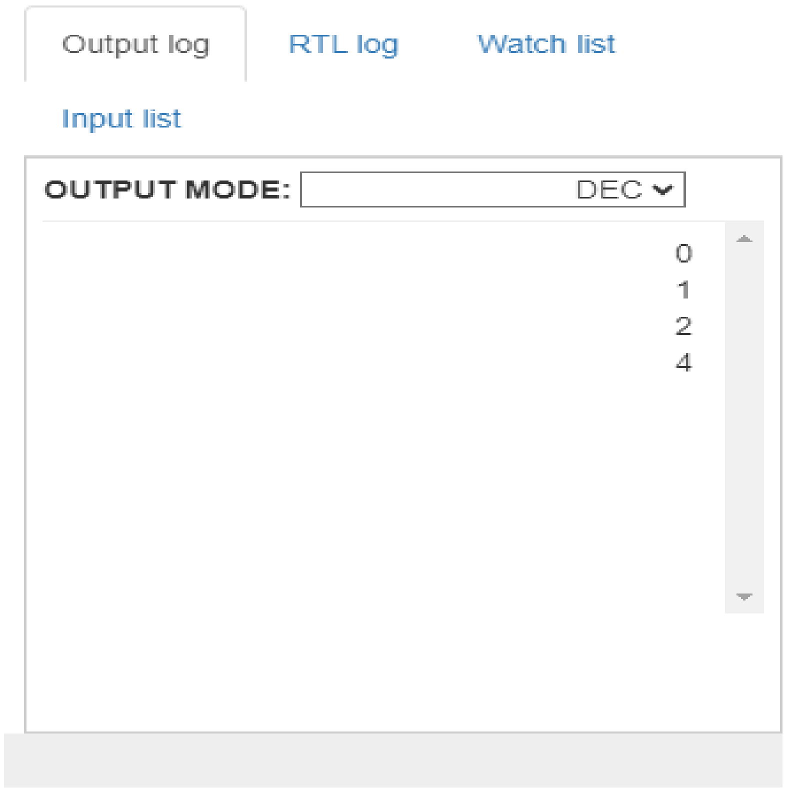


```
    ORG 100         ; Set the origin address to 100

    LOAD ZeroValue  ; Load the value from address ZeroValue into the accumula
    STORE CounterVariable ; Store the value in the accumulator into the memory

StartLoop,          ; Label for the start of the loop
    LOAD CounterVariable ; Load the value from address CounterVariable into tl
    SUBT ThreeValue  ; Subtract the value at address ThreeValue
    SKIPCOND 400    ; Skip the next instruction if the accumulator is non-pos
    JUMP IncrementCounter ; Jump to IncrementCounter if the condition is not

SkipPrint,         ; Label for skipping the print statement
    LOAD CounterVariable ; Load the value from address CounterVariable into tl
    OUTPUT          ; Output the value in the accumulator

IncrementCounter,  ; Label for the increment section
    LOAD CounterVariable ; Load the value from address CounterVariable into tl
```



```
    ADD OneValue    ; Add the value at address OneValue to the accumulator
    STORE CounterVariable ; Store the updated value in the accumulator back t

    LOAD FiveValue  ; Load the value from address FiveValue into the accumula
    SUBT CounterVariable ; Subtract the value at address CounterVariable from
    SKIPCOND 400    ; Skip the next instruction if the accumulator is non-pos
    JUMP StartLoop  ; Jump back to the StartLoop label if the condition is no

EndLoop,           ; Label for the end of the loop
    HALT           ; Halt the program

CounterVariable, HEX 0   ; Variable to store the loop index
OneValue, DEC 1          ; Constant for the value 1
ThreeValue, DEC 3        ; Constant for the value 3
FiveValue, DEC 5         ; Constant for the value 5
ZeroValue, HEX 0         ; Constant for the value 0
```

OUTPUT:



```
OUTPUT MODE: DEC
0
1
2
4
```

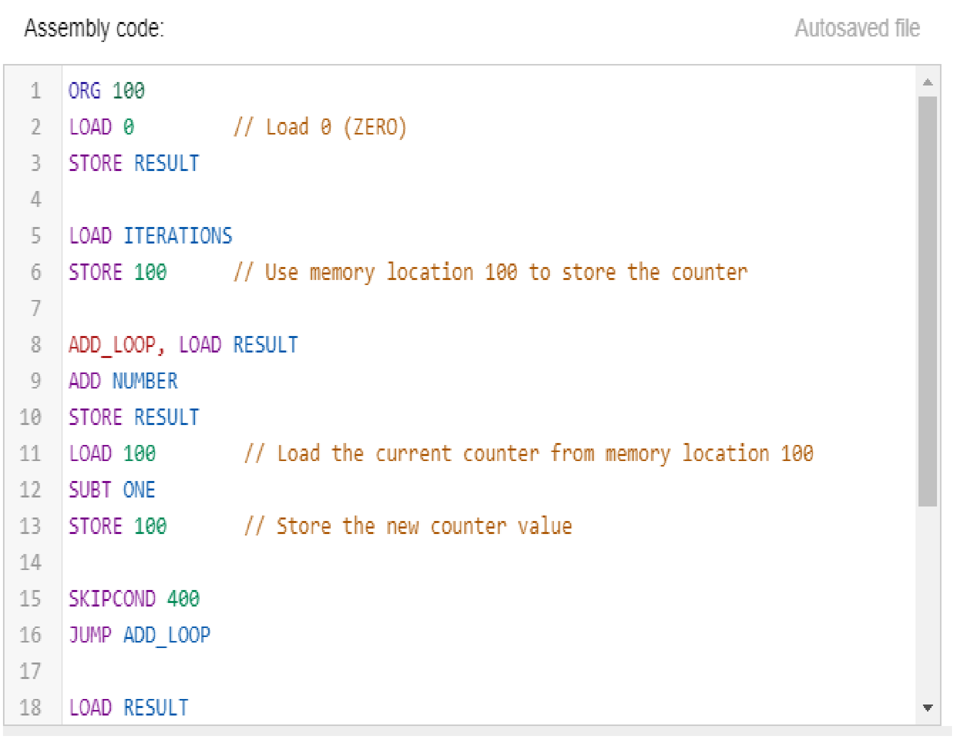
4.
SOURCE CODE:

```
ORG 100
LOAD 0          // Load 0 (ZERO)
STORE RESULT

LOAD ITERATIONS
STORE 100      // Use memory location 100 to store the counter

ADD_LOOP, LOAD RESULT
ADD NUMBER
STORE RESULT
LOAD 100       // Load the current counter from memory location 100
SUBT ONE
STORE 100       // Store the new counter value

SKIPCOND 400
```

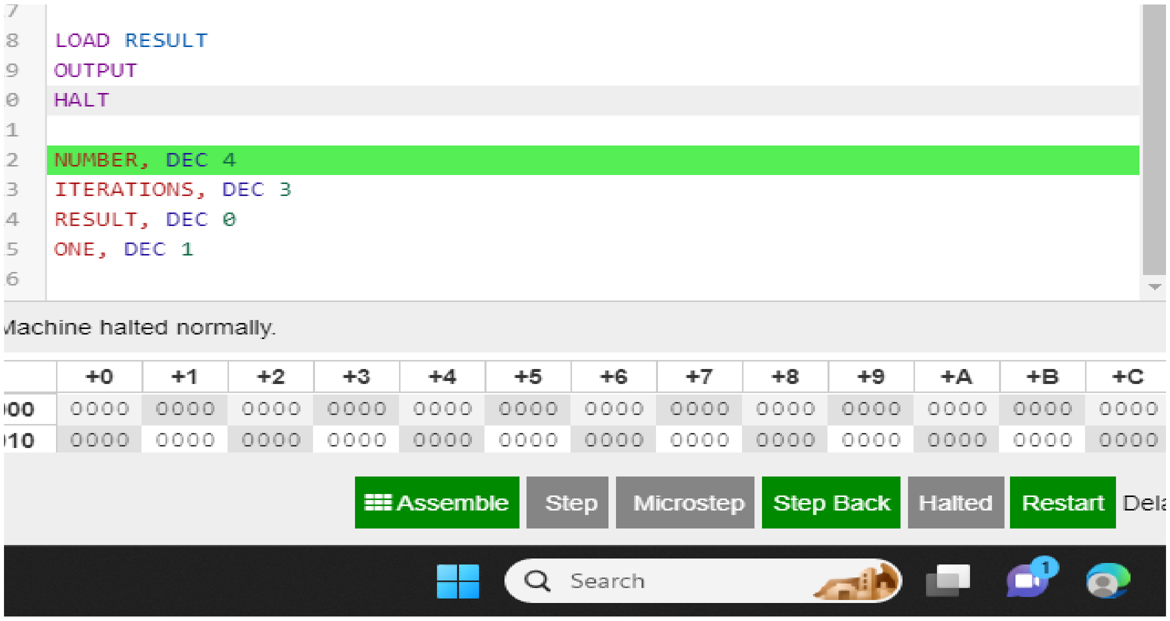
JUMP ADD_LOOP

LOAD RESULT
OUTPUT
HALT

NUMBER, DEC 4
ITERATIONS, DEC 3
RESULT, DEC 0
ONE, DEC 1



```
ORG 100
LOAD 0          // Load 0 (ZERO)
STORE RESULT

LOAD ITERATIONS
STORE 100       // Use memory location 100 to store the counter

ADD_LOOP, LOAD RESULT
ADD NUMBER
STORE RESULT
LOAD 100         // Load the current counter from memory location 100
SUBT ONE
STORE 100        // Store the new counter value

SKIPCOND 400
JUMP ADD_LOOP

LOAD RESULT
```

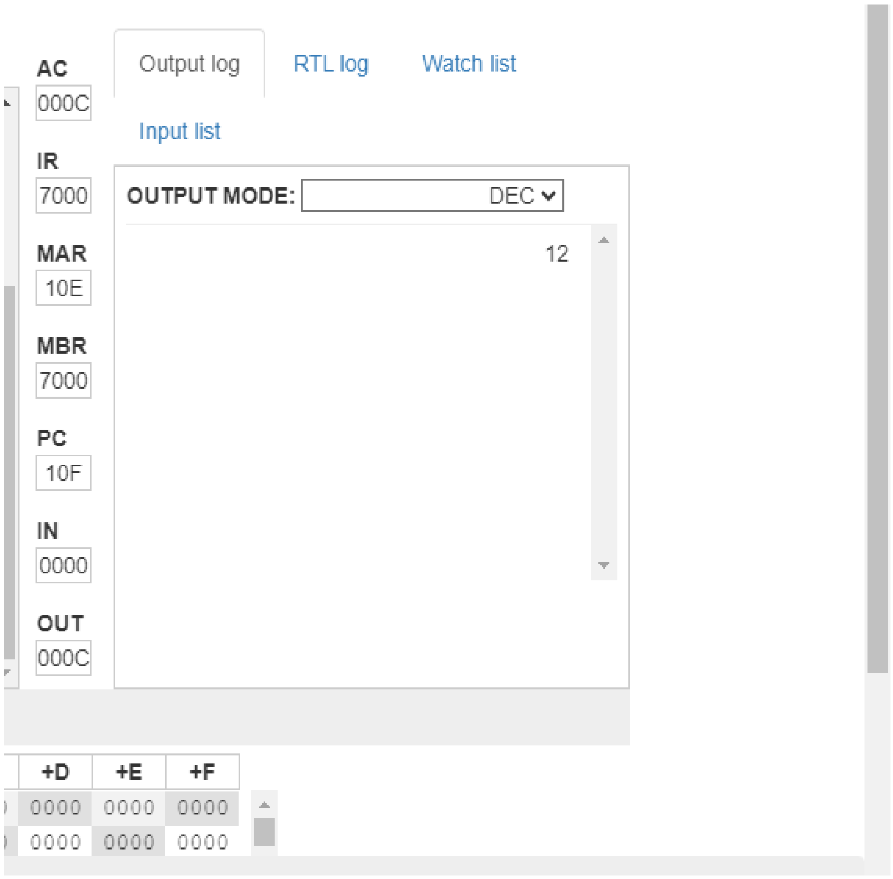


```
LOAD RESULT
OUTPUT
HALT

NUMBER, DEC 4
ITERATIONS, DEC 3
RESULT, DEC 0
ONE, DEC 1
```

lachine halted normally.

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C
00	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
10	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

OUTPUT:



AC 000C
IR 7000
MAR 10E
MBR 7000
PC 10F
IN 0000
OUT 000C

OUTPUT MODE: DEC

```
12
```

+D	+E	+F
0000	0000	0000
0000	0000	0000