Effat University

FALL 2021

COLLEGE OF ENGINEERING – DEPARTMENT OF COMPUTER SCIENCE

Marie project

Students Information:

Student Name:	Reem Alsharabi	Layan Jaman	Joud Kaki
Student ID:	S20106353	S20106590	S20106234
Date of submission:		13/12/2021	

The goal of the project is to implement a complete Marie program. It is also a chance to practice Marie programming before your final lab exam.

The steps are the following:

1- Choose your problem (any problem you would like to solve on Marie like sorting an array for example)
The problem should be complicated enough to use all of the Marie instructions categories (Arithmetic, Data transfer, I/O, Branch, Subroutine, Indirect Addressing)
Take some time to learn about the two categories “Subroutine” and “Indirect Addressing”
2- Write your program in C++ (do not forget to comment your program)
3- Write your program on Marie (do not forget to comment your program)
4- Test it and verify that is working well using many test cases.
5- Write your report
6- Prepare your presentation

Your report should:

1- State the problem
2- Include your C++ program
3- Include your Marie program
4- All the explanations of the programs
5- The tests
6- Comparison between the C++ program (written in high-level language) and the Marie program (written in low-level language)
7- Conclusion

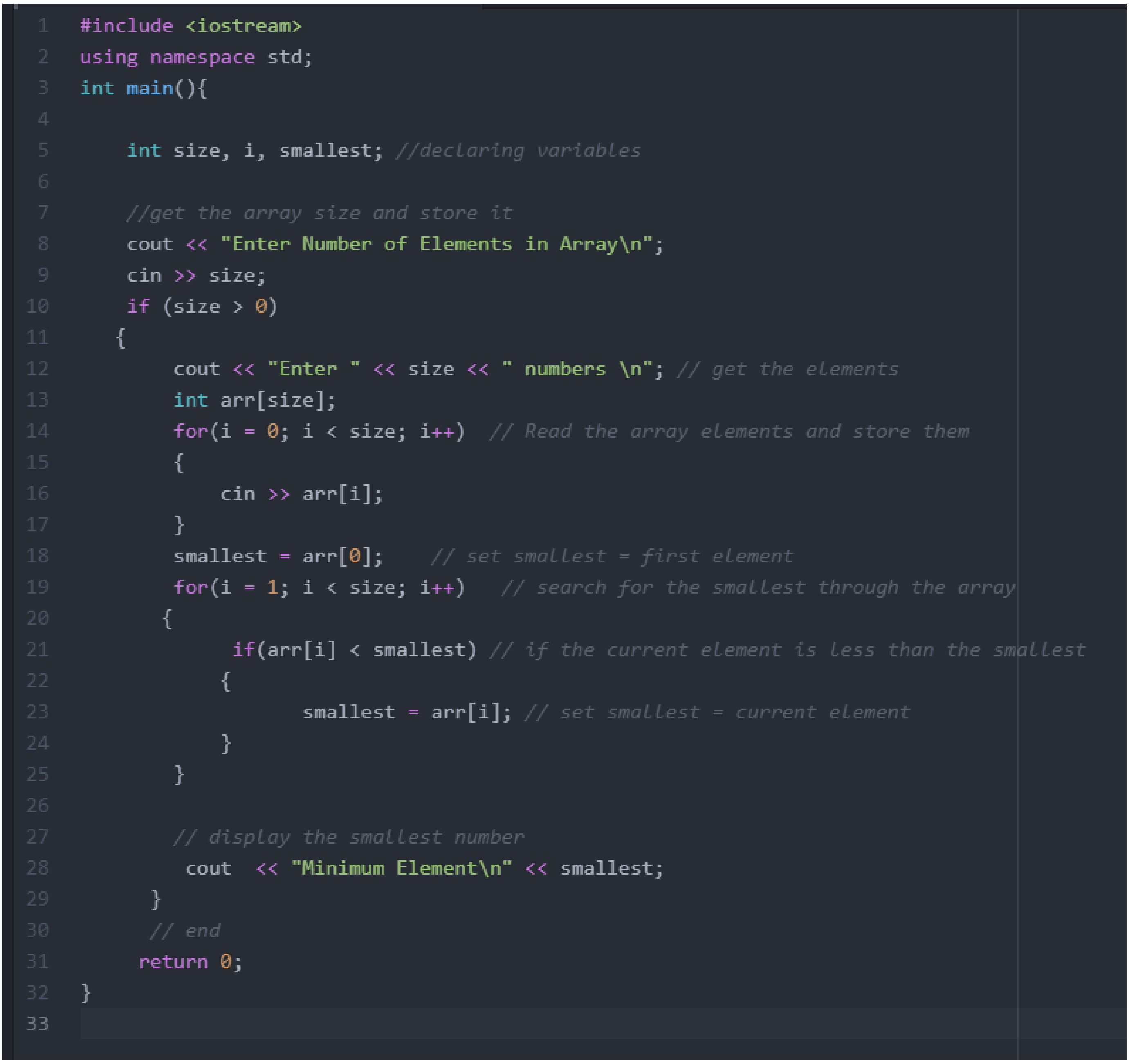
Take care of the report clarity and presentation.
The coding rubrics are available in your syllabus.
Your presentation duration is 10 minutes + 10 minutes of questions about the code.
The rubrics of the presentation will be available soon on BB.

The Problem:

Our program requires the user to input an array of elements of his/her choice of numbers, while our C++ and Marie assembly code provides a solution to this problem by outputting the smallest element within the list. Our program aims to find the smallest element from the user’s input.

C++ Code:

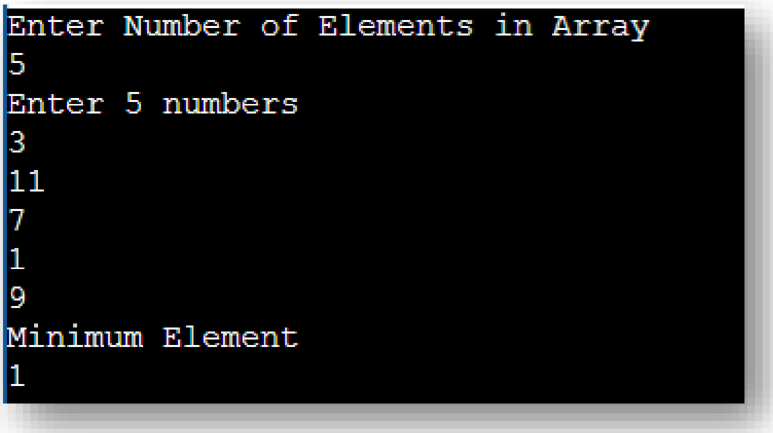
```
#include <iostream>
using namespace std;
int main(){

    int size, i, smallest; //declaring variables

    //get the array size and store it
    cout << "Enter Number of Elements in Array\n";
    cin >> size;
    if (size > 0)
   {
        cout << "Enter " << size << " numbers \n"; // get the elements
        int arr[size];
        for(i = 0; i < size; i++)  // Read the array elements and store them
        {
            cin >> arr[i];
        }
        smallest = arr[0];    // set smallest = first element
        for(i = 1; i < size; i++)   // search for the smallest through the array
       {
             if(arr[i] < smallest) // if the current element is less than the smallest
            {
                   smallest = arr[i]; // set smallest = current element
            }
        }

        // display the smallest number
         cout  << "Minimum Element\n" << smallest;
      }
      // end
     return 0;
}
```



```cpp
#include <iostream>
using namespace std;
int main(){

    int size, i, smallest; //declaring variables

    //get the array size and store it
    cout << "Enter Number of Elements in Array\n";
    cin >> size;
    if (size > 0)
   {
        cout << "Enter " << size << " numbers \n"; // get the elements
        int arr[size];
        for(i = 0; i < size; i++)  // Read the array elements and store them
        {
            cin >> arr[i];
        }
        smallest = arr[0];    // set smallest = first element
        for(i = 1; i < size; i++)   // search for the smallest through the array
       {
             if(arr[i] < smallest) // if the current element is less than the smallest
            {
                   smallest = arr[i]; // set smallest = current element
            }
        }

        // display the smallest number
         cout  << "Minimum Element\n" << smallest;
      }
      // end
     return 0;
}

```

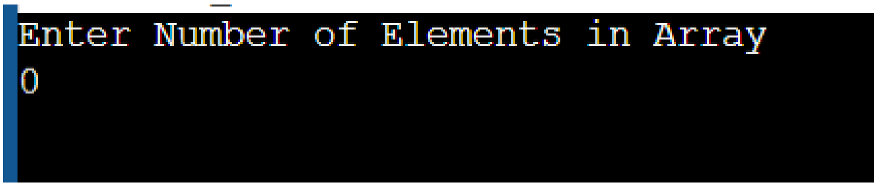
The Execution:

Case 1: size > 0



```
Enter Number of Elements in Array
5
Enter 5 numbers
3
11
7
1
9
Minimum Element
1
```

Case 2: size <= 0



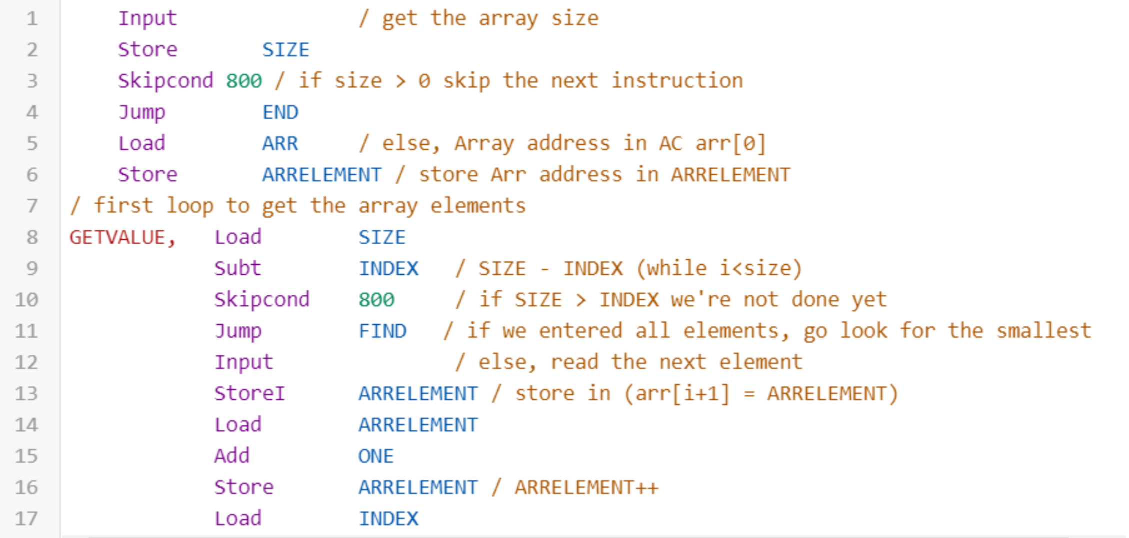
```
Enter Number of Elements in Array
0
```

In the C++ code:

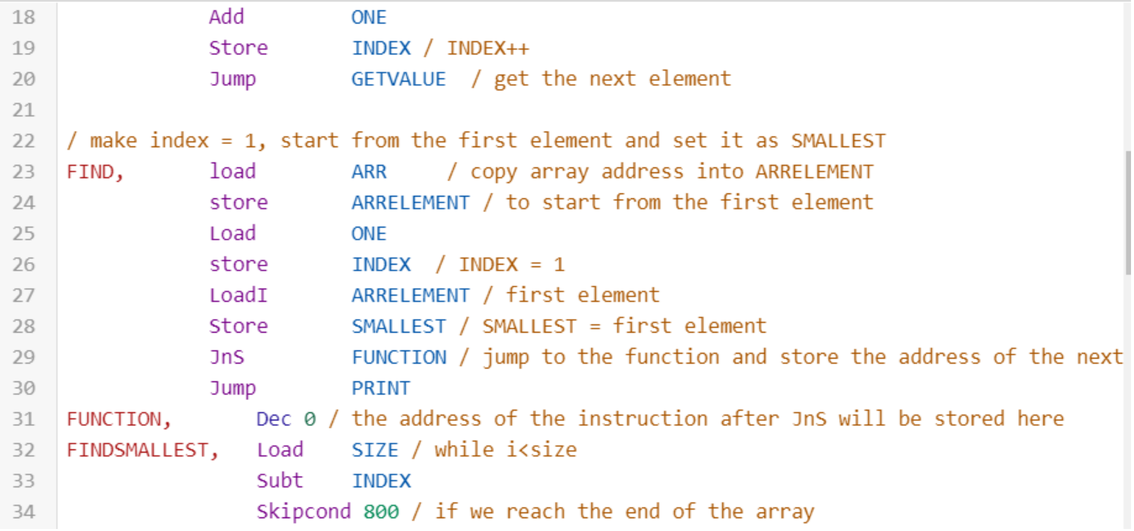
- We started the code by declaring three integers "size", "i", and "smallest" .
- Then, the code will ask the user to input the number of elements in the array and store it in the size variable (cin >> size)
- If size < = 0, the program will end.
- Else, the code will repeat a set of instructions to get the values of the elements from the user and store them in the array. This will repeat until we reach the end of the array.
- The smallest number will be set as the first element in the array
- The program will go to the next loop to find the actual smallest number, it will compare each element with the smallest. If arr[i] < smallest, it will set smallest = arr[i]. Else, the smallest won't change. Then, it will check the next element. This will be repeated until we check all the elements in the array. (while the counter is less than array size, and the counter will be increased each time)
- Finally, the code will then output the smallest element within the list.

Marie Program:

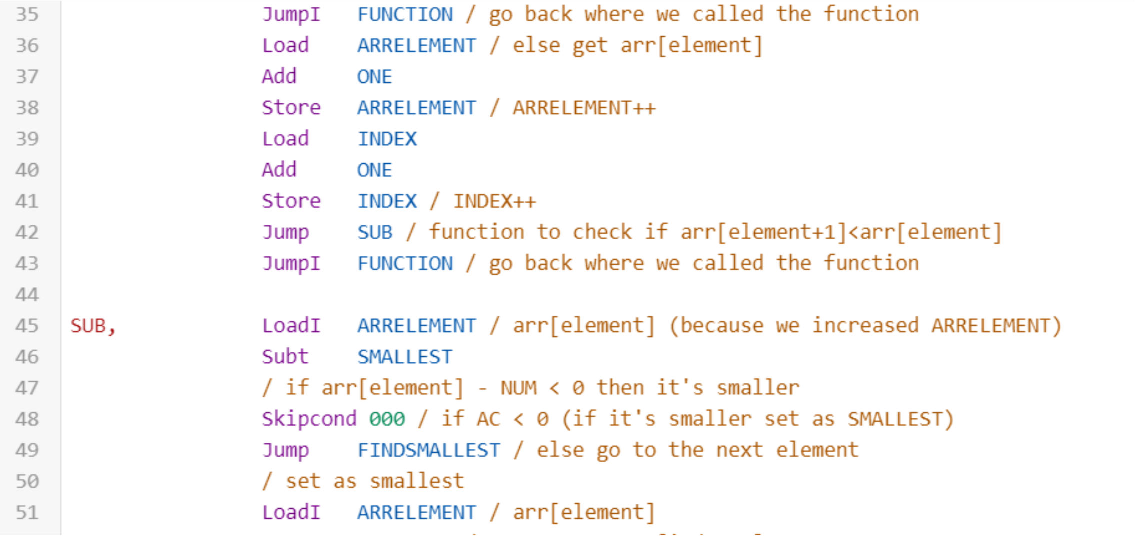
```
        Input           / get the array size
        Store     SIZE
        Skipcond 800 / if size > 0 skip the next instruction
        Jump           END
        Load            ARR     / else, Array address in AC arr[0]
        Store           ARRELEMENT / store Arr address in ARRELEMENT
/ first loop to get the array elements
GETVALUE,          Load       SIZE
                   Subt       INDEX      / SIZE - INDEX (while i<size)
                   Skipcond    800                / if SIZE > INDEX we're not done yet
                   Jump       FIND   / if we entered all elements, go look for the smallest
                   Input              / else, read the next element
                   StoreI     ARRELEMENT / store in (arr[i+1] = ARRELEMENT)
                   Load       ARRELEMENT
                   Add        ONE
                   Store      ARRELEMENT / ARRELEMENT++
                   Load       INDEX
                   Add        ONE
                   Store      INDEX / INDEX++
                   Jump       GETVALUE  / get the next element

/ make index = 1, start from the first element and set it as SMALLEST
FIND,              load            ARR     / copy array address into ARRELEMENT
                   store           ARRELEMENT / to start from the first element
                   Load            ONE
                   store           INDEX  / INDEX = 1
                   LoadI           ARRELEMENT / first element
                   Store           SMALLEST / SMALLEST = first element
                   JnS             FUNCTION / jump and store the address of the next instruction
                    Jump           PRINT
FUNCTION,          Dec     0       / the address of the instruction after JnS will be stored here
FINDSMALLEST,      Load            SIZE / while i<size
                   Subt            INDEX
                   Skipcond 800 / if we reach the end of the array
                   JumpI           FUNCTION / go back where we called the function
                   Load            ARRELEMENT / else get arr[element]
                   Add             ONE
                   Store           ARRELEMENT / ARRELEMENT++
                   Load            INDEX
                   Add             ONE
                   Store           INDEX  / INDEX++
                    Jump         SUB / function to check if arr[element+1]<arr[element]
                    JumpI        FUNCTION / go back where we called the function

SUB,         LoadI        ARRELEMENT / arr[element] (because we increased ARRELEMENT)
             Subt         SMALLEST / if arr[element] - NUM < 0 then it's smaller
             Skipcond 000      / if AC < 0 (if it's smaller set as SMALLEST)
             Jump         FINDSMALLEST / else go to the next element
             / set as smallest
             LoadI        ARRELEMENT / arr[element]
             Store        SMALLEST / SMALLEST = arr[index+1]
             Jump         FINDSMALLEST / repeat

/ display the smallest number on the screen
PRINT,       Load         SMALLEST
             Output
             Jump         END / end the program

SIZE,        Dec          0     / user chosen array size
INDEX,       Dec          0     / current array index
SMALLEST,  Dec            0     / value of smallest element
ONE,         Dec          1     / for ++
ARRELEMENT,    Dec        80    / address of current array index (current element) arr[0]
ARR,         Dec     80    / address for start of the array storage arr[0]
END,         Halt
```



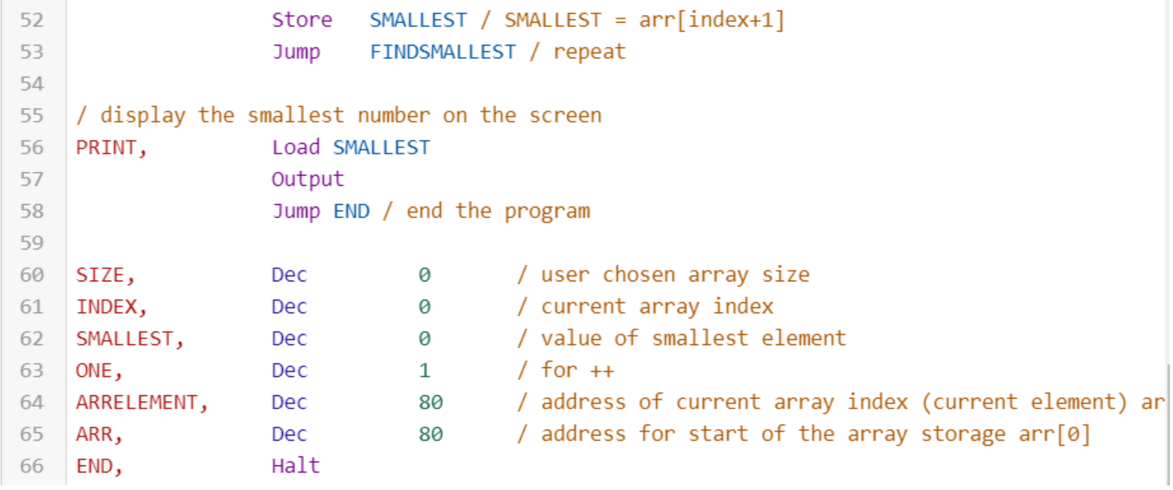
```
    Input              / get the array size
    Store       SIZE
    Skipcond 800 / if size > 0 skip the next instruction
    Jump        END
    Load        ARR      / else, Array address in AC arr[0]
    Store       ARRELEMENT / store Arr address in ARRELEMENT
/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX (while i<size)
            Skipcond    800     / if SIZE > INDEX we're not done yet
            Jump        FIND   / if we entered all elements, go look for the smallest
            Input               / else, read the next element
            StoreI      ARRELEMENT / store in (arr[i+1] = ARRELEMENT)
            Load        ARRELEMENT
            Add         ONE
            Store       ARRELEMENT / ARRELEMENT++
            Load        INDEX
```



```
            Add         ONE
            Store       INDEX / INDEX++
            Jump        GETVALUE  / get the next element

/ make index = 1, start from the first element and set it as SMALLEST
FIND,       load        ARR      / copy array address into ARRELEMENT
            store       ARRELEMENT / to start from the first element
            Load        ONE
            store       INDEX  / INDEX = 1
            LoadI       ARRELEMENT / first element
            Store       SMALLEST / SMALLEST = first element
            JnS         FUNCTION / jump to the function and store the address of the next
            Jump        PRINT
FUNCTION,       Dec 0 / the address of the instruction after JnS will be stored here
FINDSMALLEST,   Load     SIZE / while i<size
                Subt     INDEX
                Skipcond 800 / if we reach the end of the array
```



```
                JumpI    FUNCTION / go back where we called the function
                Load     ARRELEMENT / else get arr[element]
                Add      ONE
                Store    ARRELEMENT / ARRELEMENT++
                Load     INDEX
                Add      ONE
                Store    INDEX / INDEX++
                Jump     SUB / function to check if arr[element+1]<arr[element]
                JumpI    FUNCTION / go back where we called the function

SUB,            LoadI    ARRELEMENT / arr[element] (because we increased ARRELEMENT)
                Subt     SMALLEST
                / if arr[element] - NUM < 0 then it's smaller
                Skipcond 000 / if AC < 0 (if it's smaller set as SMALLEST)
                Jump     FINDSMALLEST / else go to the next element
                / set as smallest
                LoadI    ARRELEMENT / arr[element]
```



```
                Store   SMALLEST / SMALLEST = arr[index+1]
                Jump    FINDSMALLEST / repeat

/ display the smallest number on the screen
PRINT,          Load SMALLEST
                Output
                Jump END / end the program

SIZE,           Dec         0       / user chosen array size
INDEX,          Dec         0       / current array index
SMALLEST,       Dec         0       / value of smallest element
ONE,            Dec         1       / for ++
ARRELEMENT,     Dec         80      / address of current array index (current element) ar
ARR,            Dec         80      / address for start of the array storage arr[0]
END,            Halt
```

In the Marie code:

- Reading the input and storing it in the variable we declared (SIZE).
- If the size < 0, the program will stop. Else, the elements will be stored in the address and each time we will go to the next address. So, we will increase the variable ARRELEMENT (which has the array address) by adding one and storing it in the same variable (ARRELEMENT), then we will get the element as an input and store it in the address in ARRELEMENT.
- Then, the index will be increased by adding one to it and storing it in the same variable (INDEX) each time we get an input. This will be repeated as long as the index is less than size, which we tested be (SIZE - INDEX) and then Skipcond 800.

- Then, we start searching for the smallest value, we started from the first element in the array by loading the array address to the AC (variable ARR) and storing it in the variable ARRELEMENT, then we loaded the number 1 to the AC (variable ONE) and stored it in INDEX to start from index one.
- then we loaded the data in the ARRELEMENT address which represents the first element in the array, and stored it in the variable SMALLEST, we tagged this set of instructions as FIND.

- Then, to find the smallest, we jump to FUNCTION and store the address of the next instruction (which is print)
- After FUNCTION, we will start the loop from FINDSMALLEST, here we will repeat the following instructions as long as INDEX<SIZE. The index will be increased by 1, it will stop (jump back to the stored address in FUNCTION) if INDEX<SIZE (we tested that condition by size – index and then Skipcond 800). Else, it will continue, also, we added one to ARRELEMENT to go to the next element. This set of instructions was tagged as FINDSMALLEST.
- So, while we did not go through the whole array, we jumped to SUBT tag, which tested if the element is less than the SMALLEST by loading the element in the address ARRELEMENT and subtracting the SMMALLEST, if the result was less than zero, it means the element is less than SMALLEST, this was tested using Skipcond 000. If the element was smaller, it will be loaded in the AC and stored in the variable SMALLEST, then it will jump back to FINDSMALLEST to check the next element. (if the subtraction result was not less than 0, it will jump to FINDSMALLEST without setting the element to SMALLEST)
- After checking all elements, we will jump back to the stored address in FUNCTION which will take us to the instruction (Jump PRINT)
- In PRINT, we will load the SMALLEST, output it, and Jump END which will halt the program

Cases:

Case1: Sorted array of positive numbers (ascending)

Case2: Sorted array of positive numbers (descending)

Case3: Sorted array of negative numbers (ascending)

Case4: Sorted array of negative numbers (descending)

Case5: Unsorted array of negative numbers

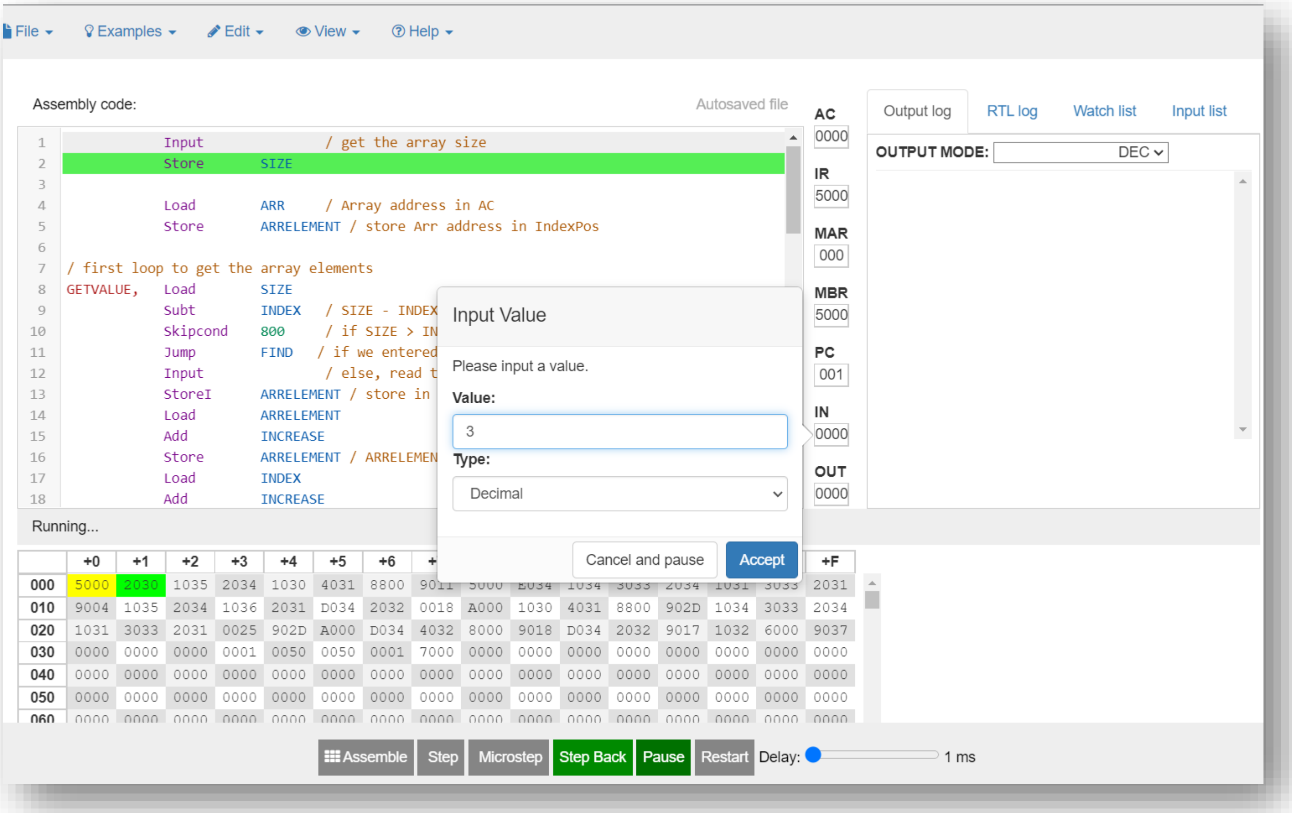
Case6: Unsorted array of positive numbers

Case7: array of the same number

Case8: array of integers

Case9: array of size 0 or less

In all cases: array size = 3 (except for the last case)



Assembly code:

Autosaved file

```
            Input                / get the array size
            Store       SIZE

            Load        ARR      / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX    / SIZE - INDEX
            Skipcond    800      / if SIZE > IN
            Jump        FIND   / if we entered
            Input                / else, read t
            StoreI      ARRELEMENT / store in
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMEN
            Load        INDEX
            Add         INCREASE
```

Input Value

Please input a value.

Value: 3

Type: Decimal

AC 0000

IR 5000

MAR 000

MBR 5000

PC 001

IN 0000

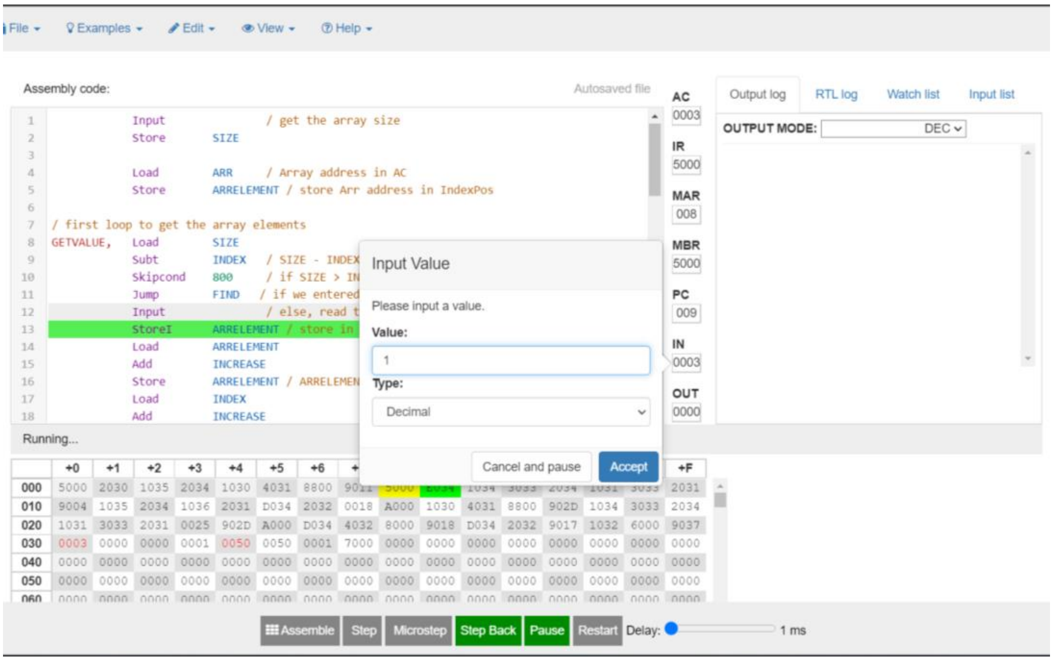
OUT 0000

OUTPUT MODE: DEC

Running...

	+0	+1	+2	+3	+4	+5	+6	+							+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	A000	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0000	0000	0000	0001	0050	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Case 1: Sorted array of positive numbers (ascending) [1, 2, 3]



Assembly code:

```
            Input               / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > IN
            Jump        FIND   / if we entered
            Input               / else, read t
            StoreI      ARRELEMENT / store in
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMEN
            Load        INDEX
            Add         INCREASE
```

Running...

Input Value

Please input a value.

Value: 1

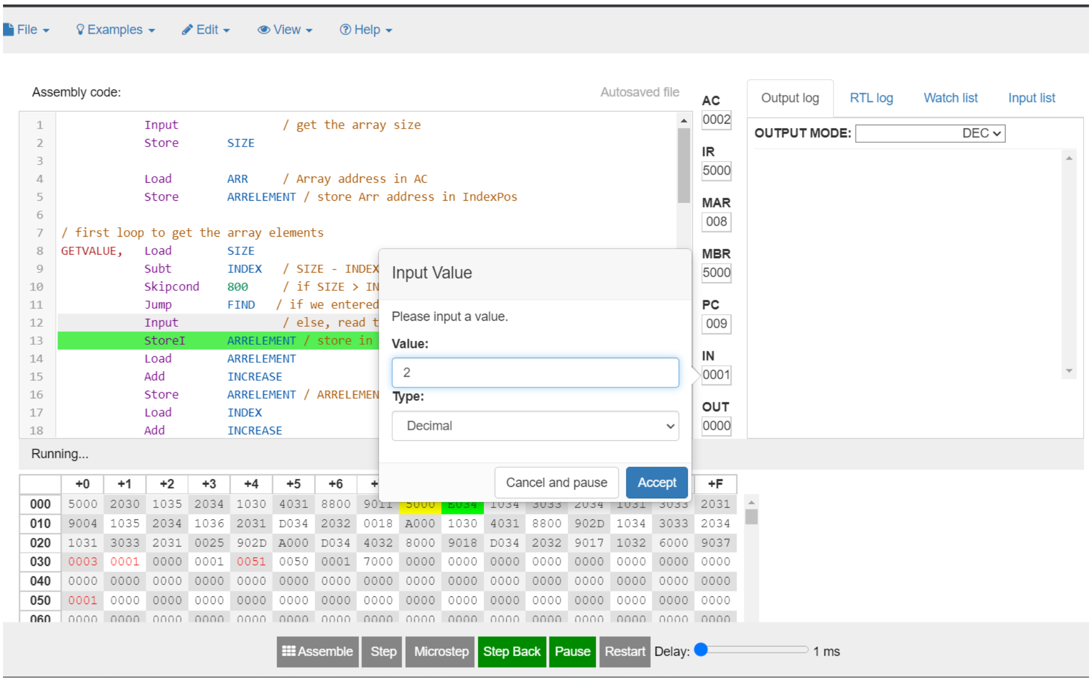
Type: Decimal

Cancel and pause | Accept

AC 0003
IR 5000
MAR 008
MBR 5000
PC 009
IN 0003
OUT 0000

OUTPUT MODE: DEC

	+0	+1	+2	+3	+4	+5	+6	+							+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	A000	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0000	0000	0001	0050	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000



Assembly code:

```
            Input               / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > IN
            Jump        FIND   / if we entered
            Input               / else, read t
            StoreI      ARRELEMENT / store in
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMEN
            Load        INDEX
            Add         INCREASE
```

Running...

Input Value

Please input a value.

Value: 2

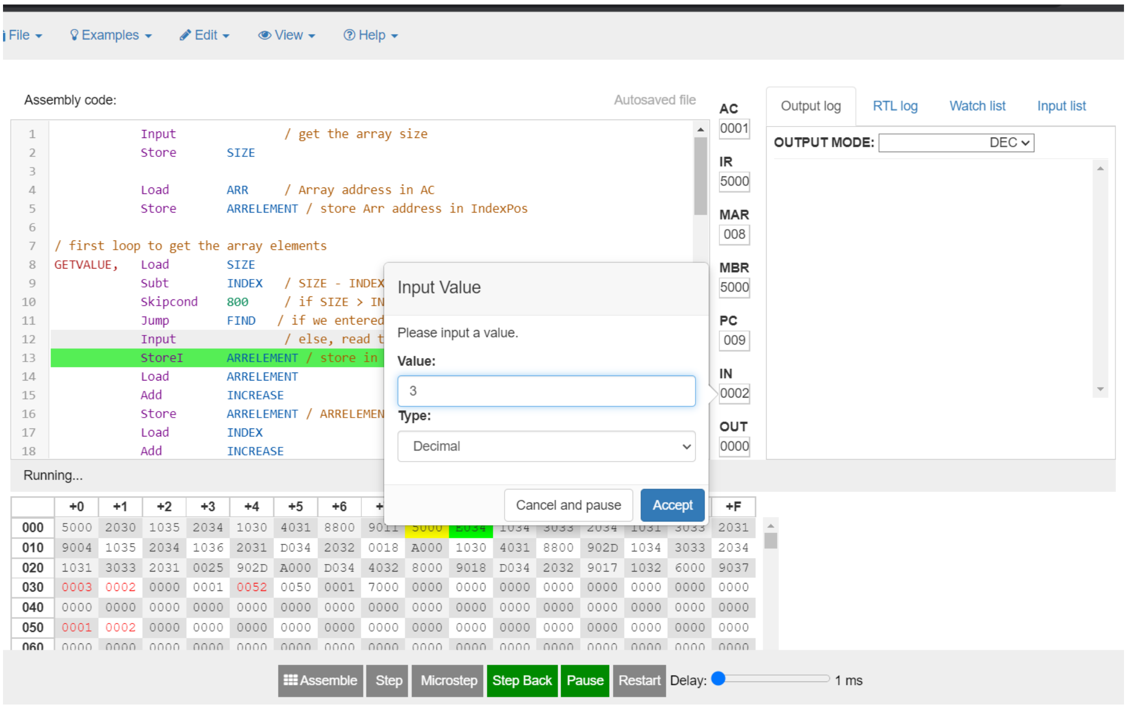
Type: Decimal

Cancel and pause | Accept

AC 0002
IR 5000
MAR 008
MBR 5000
PC 009
IN 0001
OUT 0000

OUTPUT MODE: DEC

	+0	+1	+2	+3	+4	+5	+6	+							+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	A000	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0001	0000	0001	0051	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0001	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000



Assembly code: Autosaved file

```
        Input           / get the array size
        Store   SIZE

        Load    ARR     / Array address in AC
        Store   ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load    SIZE
        Subt    INDEX   / SIZE - INDEX
        Skipcond    800 / if SIZE > IN
        Jump    FIND    / if we entered
        Input           / else, read t
        StoreI  ARRELEMENT / store in
        Load    ARRELEMENT
        Add     INCREASE
        Store   ARRELEMENT / ARRELEMEN
        Load    INDEX
        Add     INCREASE
```

Running...

Input Value

Please input a value.

Value: 3

Type: Decimal

Cancel and pause | Accept

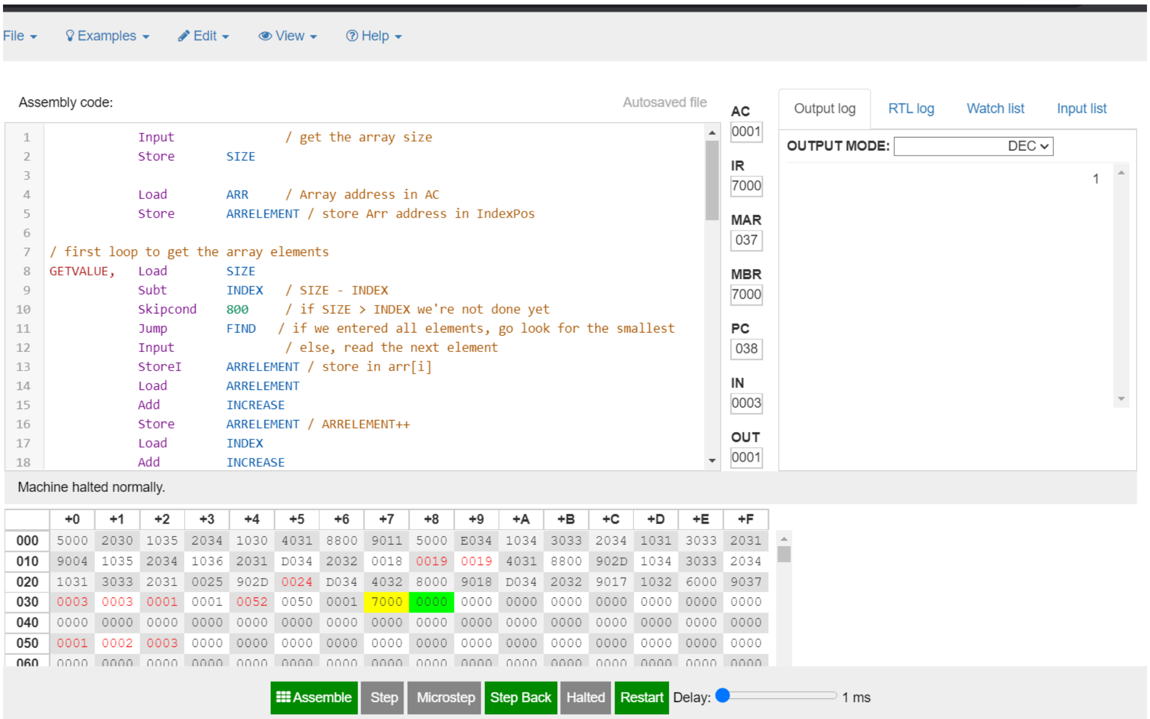
AC 0001 | IR 5000 | MAR 008 | MBR 5000 | PC 009 | IN 0002 | OUT 0000

Output log | RTL log | Watch list | Input list

OUTPUT MODE: DEC

	+0	+1	+2	+3	+4	+5	+6	+								+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	A000	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0002	0000	0001	0052	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0001	0002	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Output:



Assembly code: Autosaved file

```
        Input           / get the array size
        Store   SIZE

        Load    ARR     / Array address in AC
        Store   ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load    SIZE
        Subt    INDEX   / SIZE - INDEX
        Skipcond    800 / if SIZE > INDEX we're not done yet
        Jump    FIND    / if we entered all elements, go look for the smallest
        Input           / else, read the next element
        StoreI  ARRELEMENT / store in arr[i]
        Load    ARRELEMENT
        Add     INCREASE
        Store   ARRELEMENT / ARRELEMENT++
        Load    INDEX
        Add     INCREASE
```

Machine halted normally.

AC 0001 | IR 7000 | MAR 037 | MBR 7000 | PC 038 | IN 0003 | OUT 0001

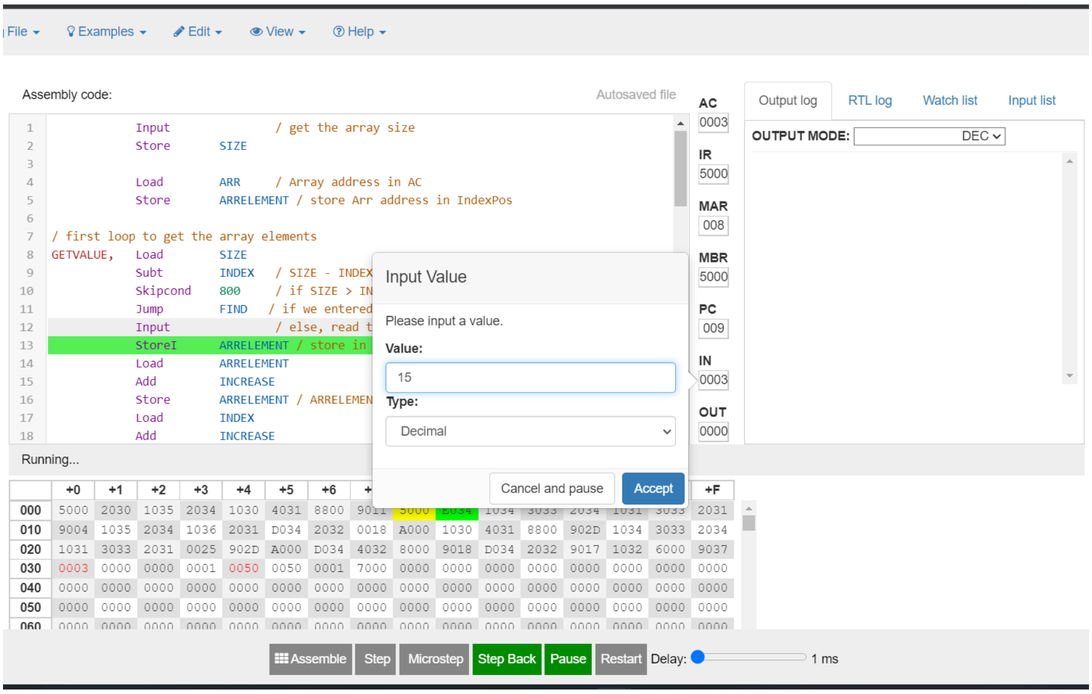
Output log | RTL log | Watch list | Input list

OUTPUT MODE: DEC

```
1
```

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	0019	0019	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	0024	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0003	0001	0001	0052	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0001	0002	0003	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Case 2: Sorted array of positive numbers (descending) [15, 10, 5]



Assembly code:

Autosaved file

```
            Input                / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > IN
            Jump        FIND   / if we entered
            Input                / else, read t
            StoreI      ARRELEMENT / store in
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMEN
            Load        INDEX
            Add         INCREASE
```

Running...

AC 0003
IR 5000
MAR 008
MBR 5000
PC 009
IN 0003
OUT 0000

OUTPUT MODE: DEC

Input Value

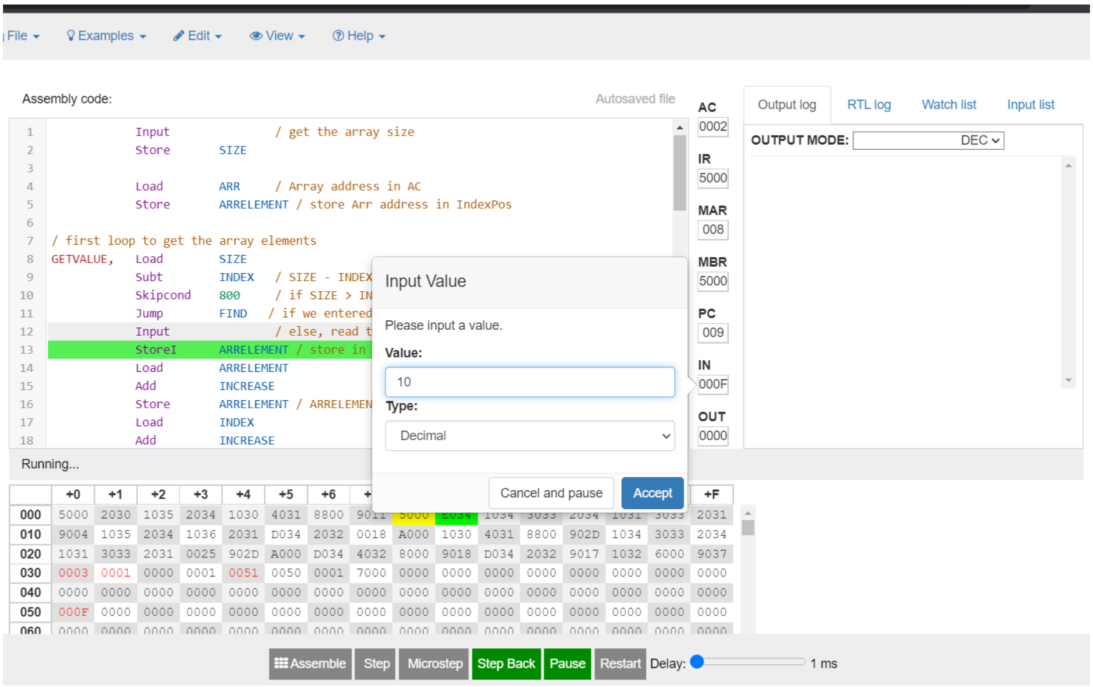
Please input a value.

Value: 15

Type: Decimal

Cancel and pause | Accept

	+0	+1	+2	+3	+4	+5	+6	+								+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	A000	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0000	0000	0001	0050	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000



Assembly code:

Autosaved file

```
            Input                / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > IN
            Jump        FIND   / if we entered
            Input                / else, read t
            StoreI      ARRELEMENT / store in
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMEN
            Load        INDEX
            Add         INCREASE
```

Running...

AC 0002
IR 5000
MAR 008
MBR 5000
PC 009
IN 000F
OUT 0000

OUTPUT MODE: DEC

Input Value

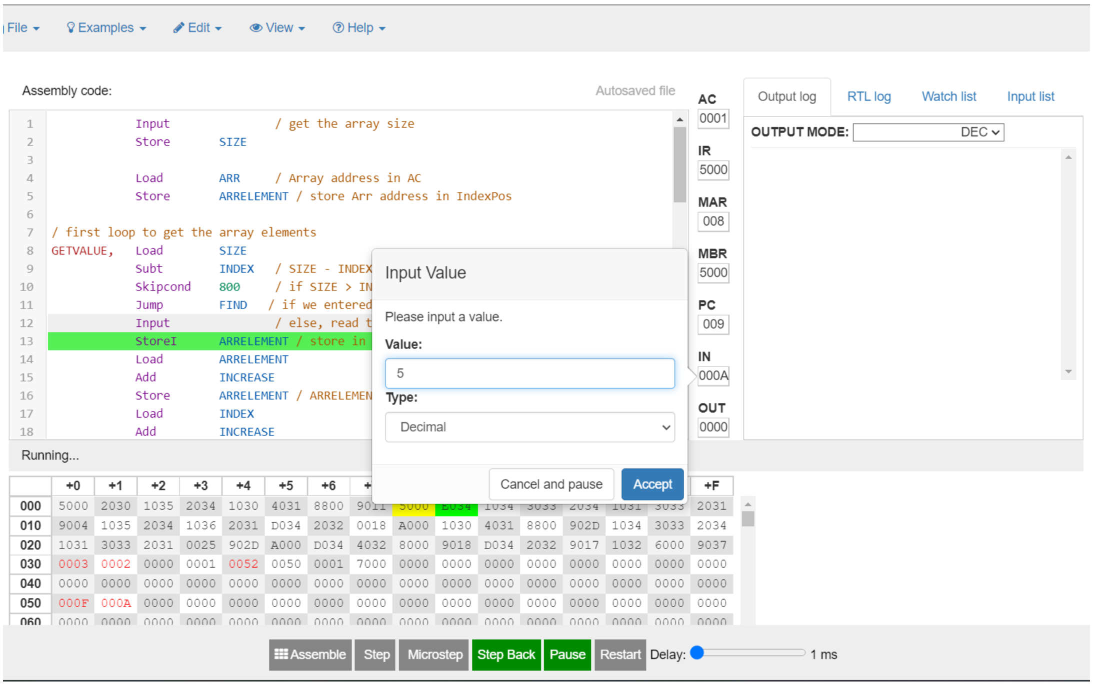
Please input a value.

Value: 10

Type: Decimal

Cancel and pause | Accept

	+0	+1	+2	+3	+4	+5	+6	+								+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	A000	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0001	0000	0001	0051	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	000F	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000



Assembly code:

```
            Input               / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > IN
            Jump        FIND   / if we entered
            Input               / else, read t
            StoreI      ARRELEMENT / store in
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMEN
            Load        INDEX
            Add         INCREASE
```

Input Value

Please input a value.

Value: 5

Type: Decimal

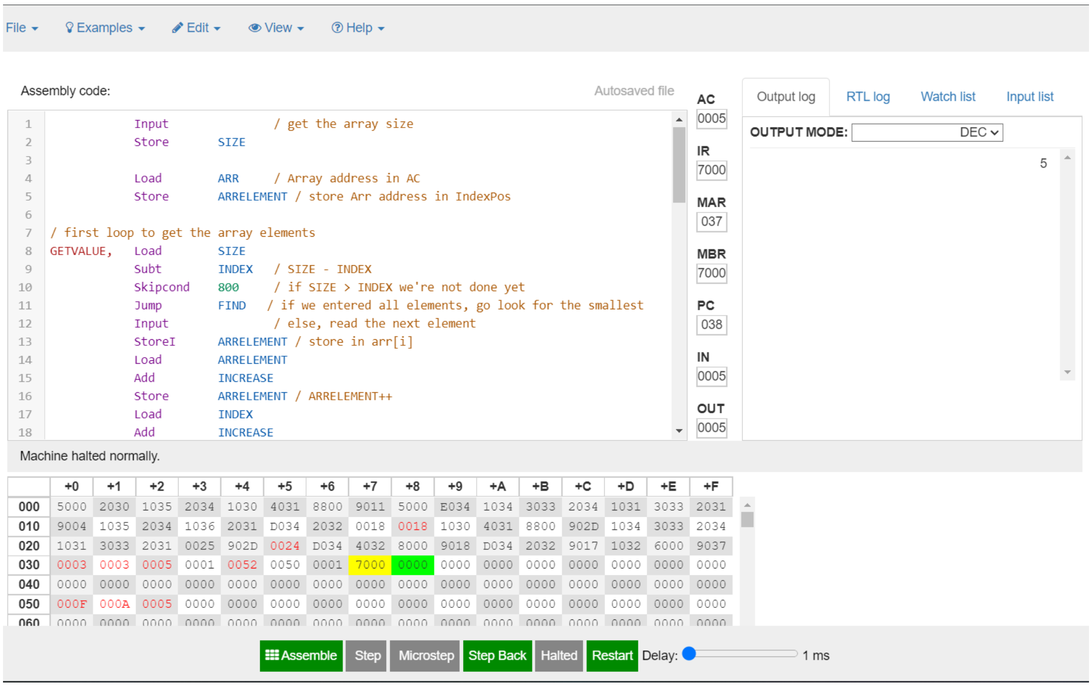
AC 0001
IR 5000
MAR 008
MBR 5000
PC 009
IN 000A
OUT 0000

OUTPUT MODE: DEC

Running...

	+0	+1	+2	+3	+4	+5	+6	+								+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	A000	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0002	0000	0001	0052	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	000F	000A	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Output:



Assembly code:

```
            Input               / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > INDEX we're not done yet
            Jump        FIND   / if we entered all elements, go look for the smallest
            Input               / else, read the next element
            StoreI      ARRELEMENT / store in arr[i]
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMENT++
            Load        INDEX
            Add         INCREASE
```

AC 0005
IR 7000
MAR 037
MBR 7000
PC 038
IN 0005
OUT 0005

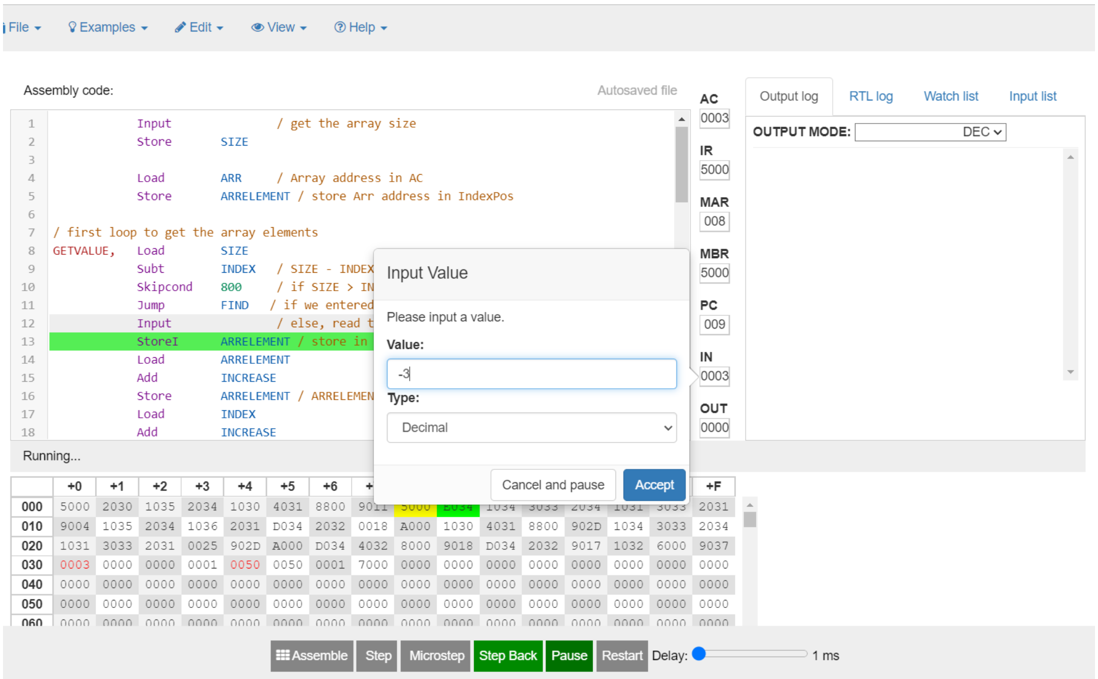
OUTPUT MODE: DEC

```
5
```

Machine halted normally.

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	0018	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	0024	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0003	0005	0001	0052	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	000F	000A	0005	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Case 3: Sorted array of negative numbers (ascending) [-3, -2, -1]



Assembly code: Autosaved file

```
            Input               / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > IN
            Jump        FIND   / if we entered
            Input               / else, read t
            StoreI      ARRELEMENT / store in
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMEN
            Load        INDEX
            Add         INCREASE
```

Input Value

Please input a value.

Value: -3

Type: Decimal

Cancel and pause | Accept

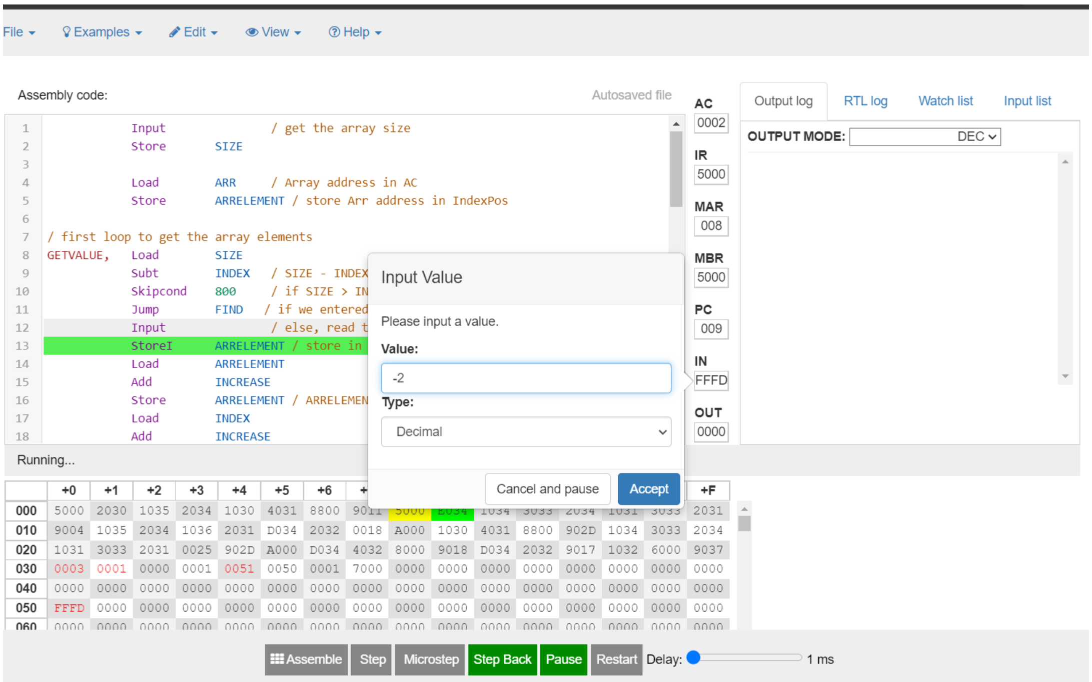
AC 0003
IR 5000
MAR 008
MBR 5000
PC 009
IN 0003
OUT 0000

OUTPUT MODE: DEC

Running...

	+0	+1	+2	+3	+4	+5	+6	+								+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	A000	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0000	0000	0001	0050	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Delay: 1 ms



Assembly code: Autosaved file

```
            Input               / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > IN
            Jump        FIND   / if we entered
            Input               / else, read t
            StoreI      ARRELEMENT / store in
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMEN
            Load        INDEX
            Add         INCREASE
```

Input Value

Please input a value.

Value: -2

Type: Decimal

Cancel and pause | Accept

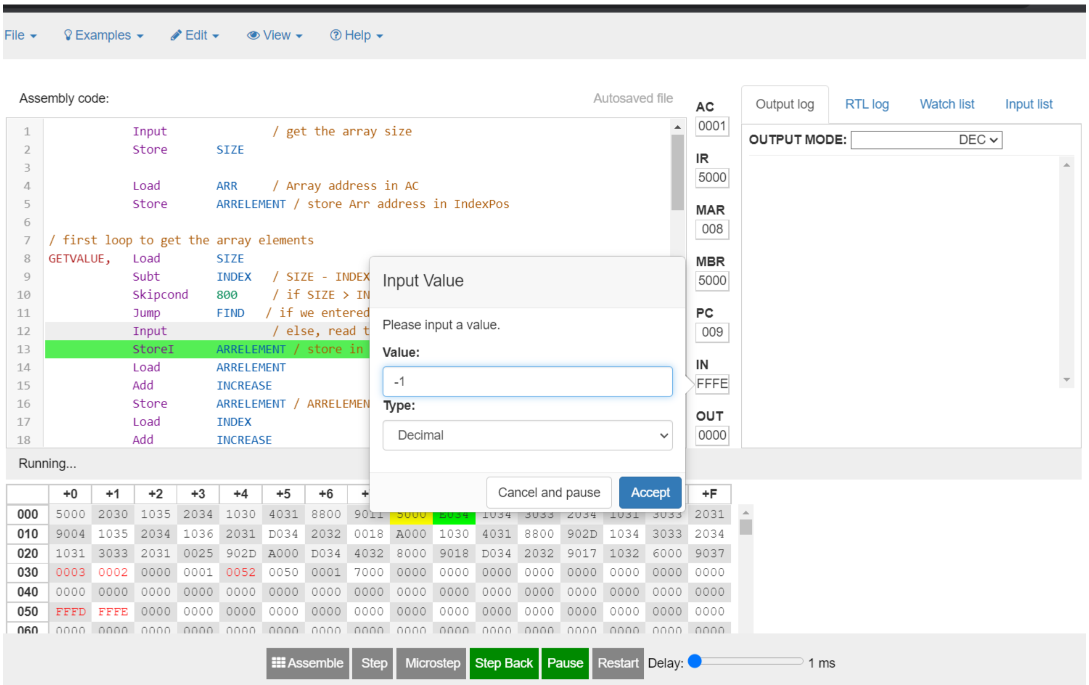
AC 0002
IR 5000
MAR 008
MBR 5000
PC 009
IN FFFD
OUT 0000

OUTPUT MODE: DEC

Running...

	+0	+1	+2	+3	+4	+5	+6	+								+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	A000	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0001	0000	0001	0051	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	FFFD	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Delay: 1 ms



Assembly code:

```
            Input                   / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > IN
            Jump        FIND   / if we entered
            Input                   / else, read t
            StoreI      ARRELEMENT / store in
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMEN
            Load        INDEX
            Add         INCREASE
```

Input Value

Please input a value.

Value: -1

Type: Decimal

Cancel and pause | Accept

AC 0001

IR 5000

MAR 008

MBR 5000

PC 009

IN FFFE

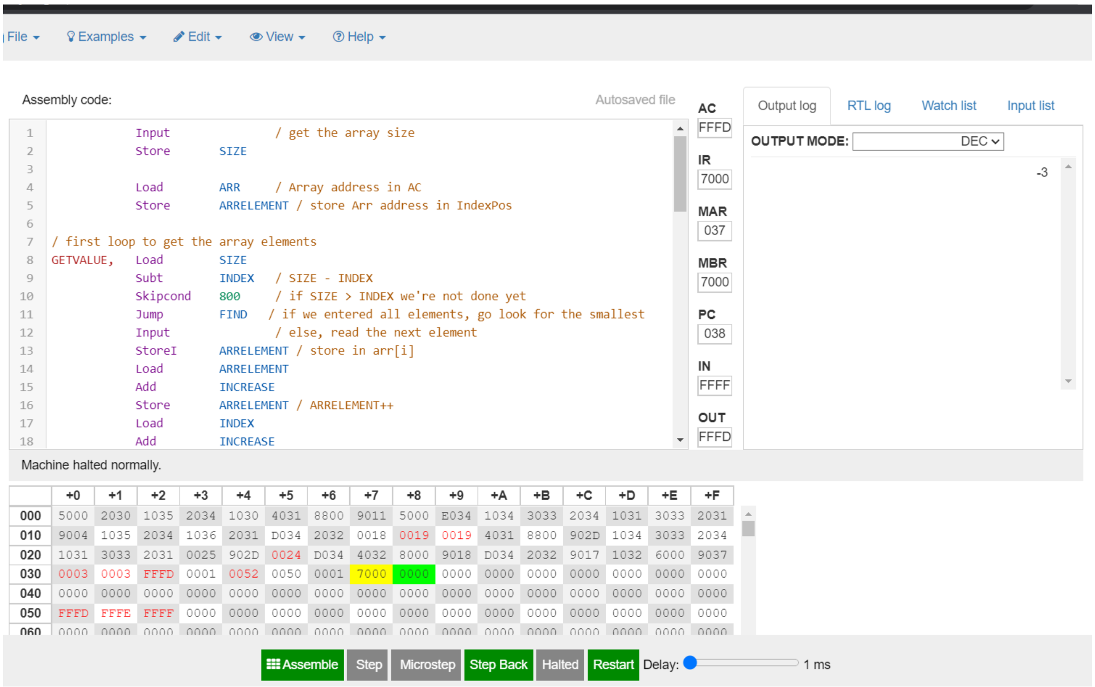
OUT 0000

OUTPUT MODE: DEC

Running...

	+0	+1	+2	+3	+4	+5	+6	+								+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	A000	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0002	0000	0001	0052	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	FFFD	FFFE	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Output:



Assembly code:

```
            Input                   / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > INDEX we're not done yet
            Jump        FIND   / if we entered all elements, go look for the smallest
            Input                   / else, read the next element
            StoreI      ARRELEMENT / store in arr[i]
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMENT++
            Load        INDEX
            Add         INCREASE
```

AC FFFD

IR 7000

MAR 037

MBR 7000

PC 038

IN FFFF

OUT FFFD

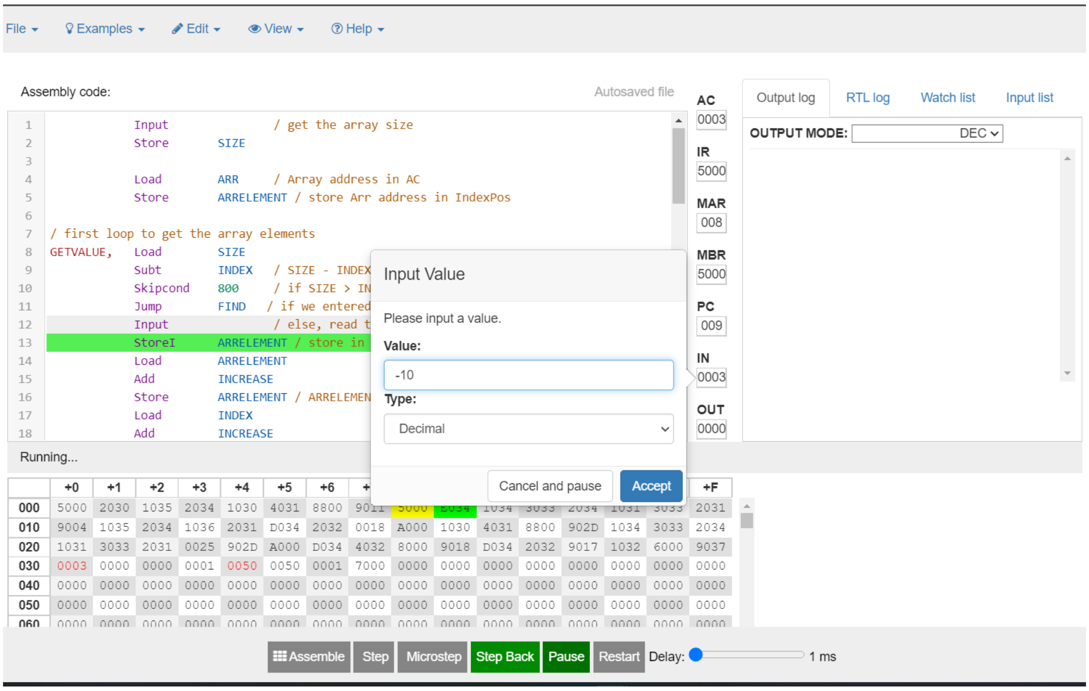
OUTPUT MODE: DEC

```
-3
```

Machine halted normally.

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	0019	0019	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	0024	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0003	FFFD	0001	0052	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	FFFD	FFFE	FFFF	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Case 4: Sorted array of negative numbers (descending) [-10, -9, -8]



```
            Input               / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > IN
            Jump        FIND   / if we entered
            Input               / else, read t
            StoreI      ARRELEMENT / store in
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMEN
            Load        INDEX
            Add         INCREASE
```

Running...

Input Value

Please input a value.

Value: -10

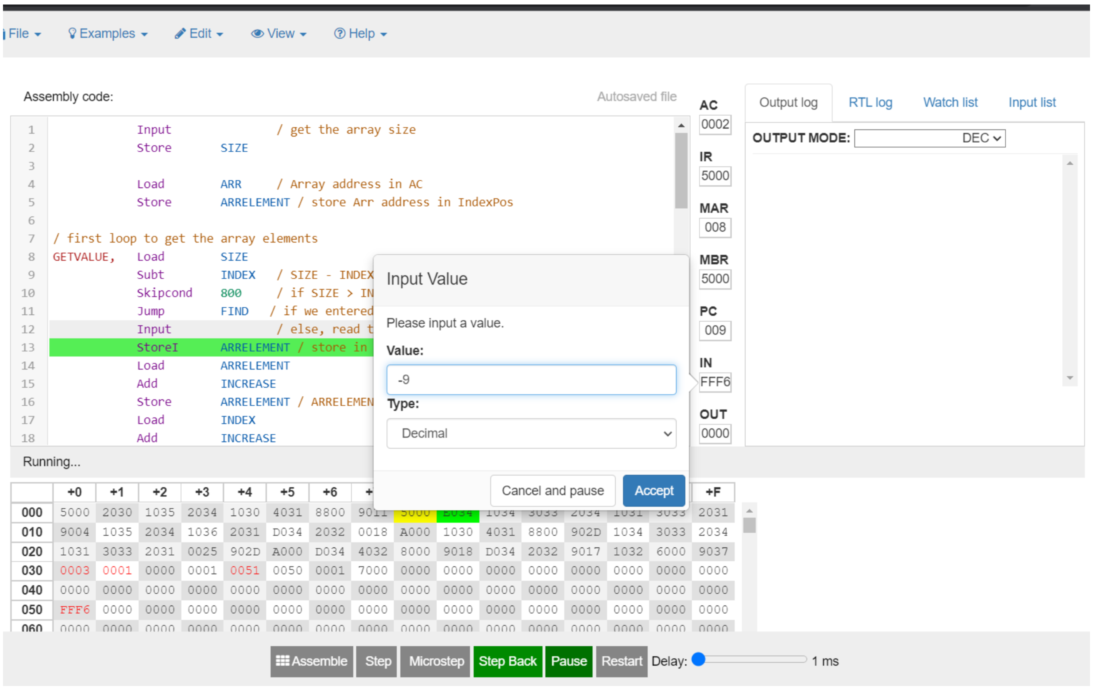
Type: Decimal

Cancel and pause | Accept

AC 0003
IR 5000
MAR 008
MBR 5000
PC 009
IN 0003
OUT 0000

OUTPUT MODE: DEC

```
      +0   +1   +2   +3   +4   +5   +6   +                                  +F
000  5000 2030 1035 2034 1030 4031 8800 9011 5000 E034 1034 3033 2034 1031 3033 2031
010  9004 1035 2034 1036 2031 D034 2032 0018 A000 1030 4031 8800 902D 1034 3033 2034
020  1031 3033 2031 0025 902D A000 D034 4032 8000 9018 D034 2032 9017 1032 6000 9037
030  0003 0000 0000 0001 0050 0050 0001 7000 0000 0000 0000 0000 0000 0000 0000 0000
040  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
050  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
060  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
```



```
            Input               / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > IN
            Jump        FIND   / if we entered
            Input               / else, read t
            StoreI      ARRELEMENT / store in
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMEN
            Load        INDEX
            Add         INCREASE
```

Running...

Input Value

Please input a value.

Value: -9

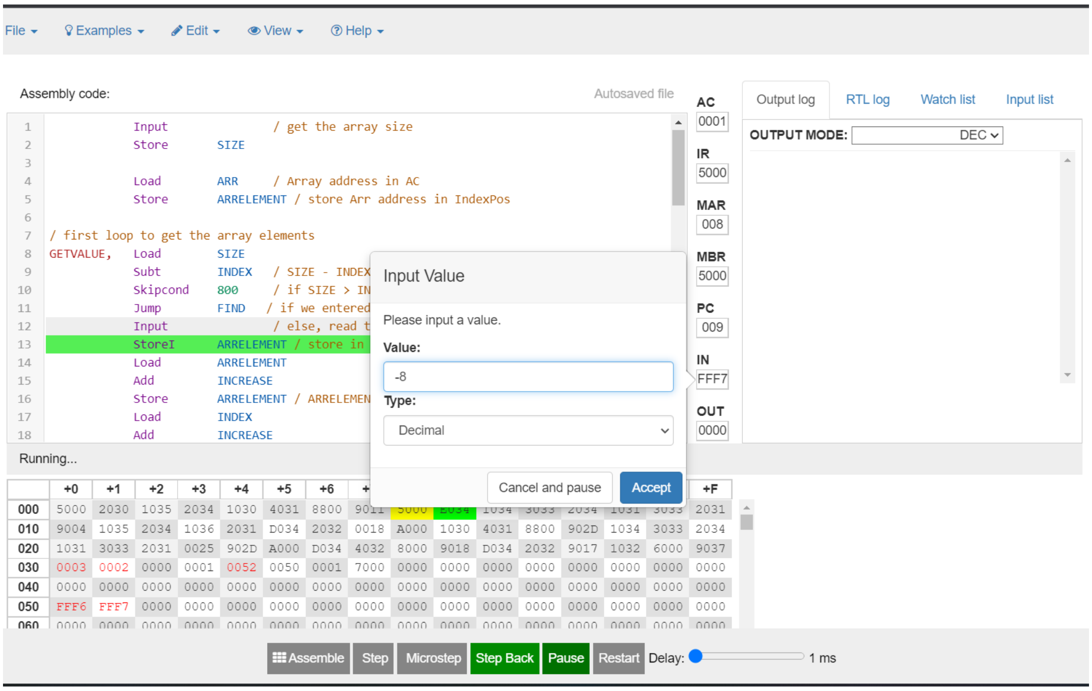
Type: Decimal

Cancel and pause | Accept

AC 0002
IR 5000
MAR 008
MBR 5000
PC 009
IN FFF6
OUT 0000

OUTPUT MODE: DEC

```
      +0   +1   +2   +3   +4   +5   +6   +                                  +F
000  5000 2030 1035 2034 1030 4031 8800 9011 5000 E034 1034 3033 2034 1031 3033 2031
010  9004 1035 2034 1036 2031 D034 2032 0018 A000 1030 4031 8800 902D 1034 3033 2034
020  1031 3033 2031 0025 902D A000 D034 4032 8000 9018 D034 2032 9017 1032 6000 9037
030  0003 0001 0000 0001 0051 0050 0001 7000 0000 0000 0000 0000 0000 0000 0000 0000
040  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
050  FFF6 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
060  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
```



Assembly code:

Autosaved file

```
            Input           / get the array size
            Store      SIZE

            Load       ARR     / Array address in AC
            Store      ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load       SIZE
            Subt       INDEX   / SIZE - INDEX
            Skipcond   800     / if SIZE > IN
            Jump       FIND   / if we entered
            Input              / else, read t
            StoreI     ARRELEMENT / store in
            Load       ARRELEMENT
            Add        INCREASE
            Store      ARRELEMENT / ARRELEMEN
            Load       INDEX
            Add        INCREASE
```

Running...

Input Value

Please input a value.

Value: -8

Type: Decimal

AC 0001

IR 5000

MAR 008

MBR 5000

PC 009

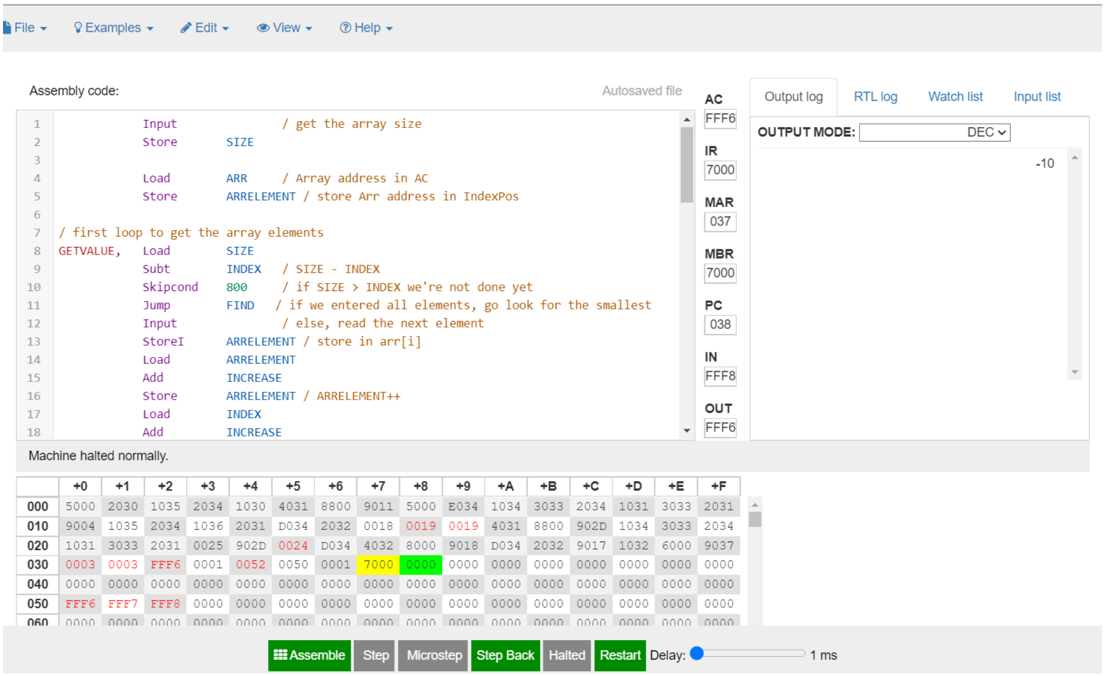
IN FFF7

OUT 0000

OUTPUT MODE: DEC

	+0	+1	+2	+3	+4	+5	+6	+								+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	A000	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0002	0000	0001	0052	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	FFF6	FFF7	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Output:



Assembly code:

Autosaved file

```
            Input           / get the array size
            Store      SIZE

            Load       ARR     / Array address in AC
            Store      ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load       SIZE
            Subt       INDEX   / SIZE - INDEX
            Skipcond   800     / if SIZE > INDEX we're not done yet
            Jump       FIND   / if we entered all elements, go look for the smallest
            Input              / else, read the next element
            StoreI     ARRELEMENT / store in arr[i]
            Load       ARRELEMENT
            Add        INCREASE
            Store      ARRELEMENT / ARRELEMENT++
            Load       INDEX
            Add        INCREASE
```

Machine halted normally.

AC FFF6

IR 7000

MAR 037

MBR 7000

PC 038

IN FFF8

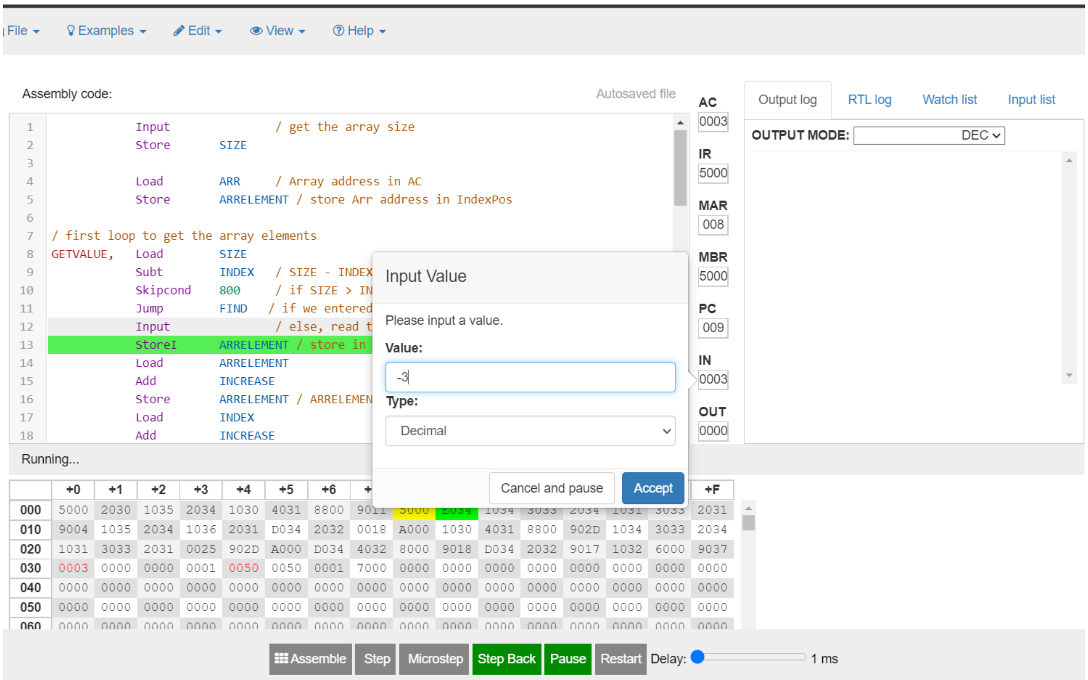
OUT FFF6

OUTPUT MODE: DEC

```
-10
```

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	0019	0019	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	0024	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0003	FFF6	0001	0052	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	FFF6	FFF7	FFF8	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Case 5: Unsorted array of negative numbers [-3, -9, -1]



Assembly code:

```
            Input               / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > IN
            Jump        FIND   / if we entered
            Input               / else, read t
            StoreI      ARRELEMENT / store in
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMEN
            Load        INDEX
            Add         INCREASE
```

Input Value

Please input a value.

Value: -3

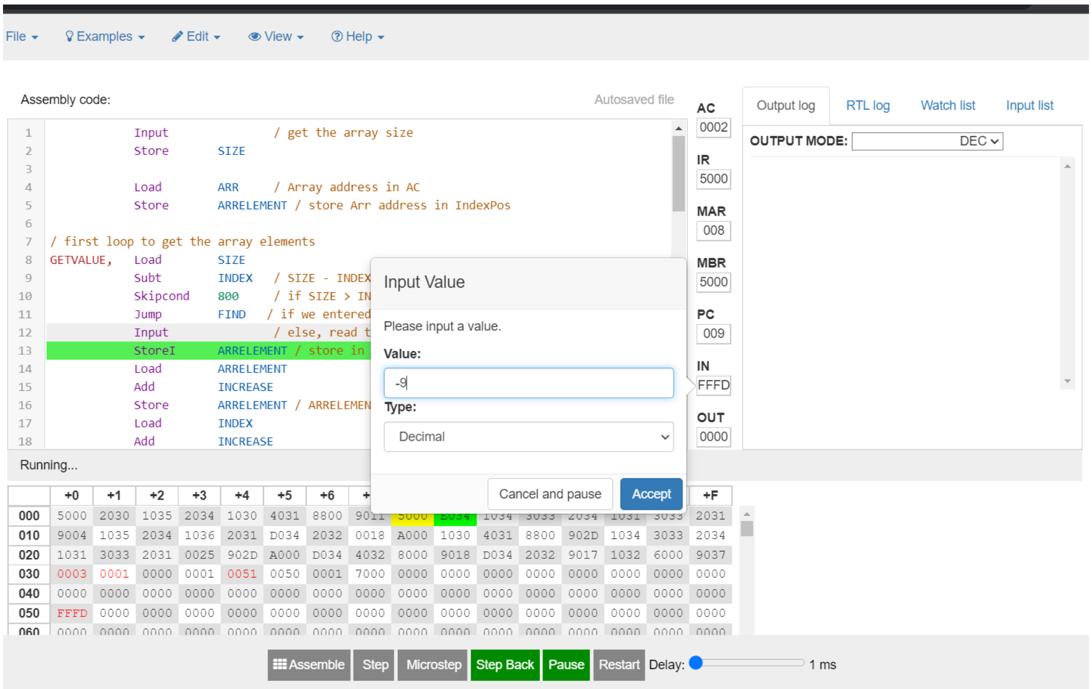
Type: Decimal

AC 0003 | IR 5000 | MAR 008 | MBR 5000 | PC 009 | IN 0003 | OUT 0000

OUTPUT MODE: DEC

Running...

	+0	+1	+2	+3	+4	+5	+6	+								+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	A000	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0000	0000	0001	0050	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000



Assembly code:

```
            Input               / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > IN
            Jump        FIND   / if we entered
            Input               / else, read t
            StoreI      ARRELEMENT / store in
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMEN
            Load        INDEX
            Add         INCREASE
```

Input Value

Please input a value.

Value: -9

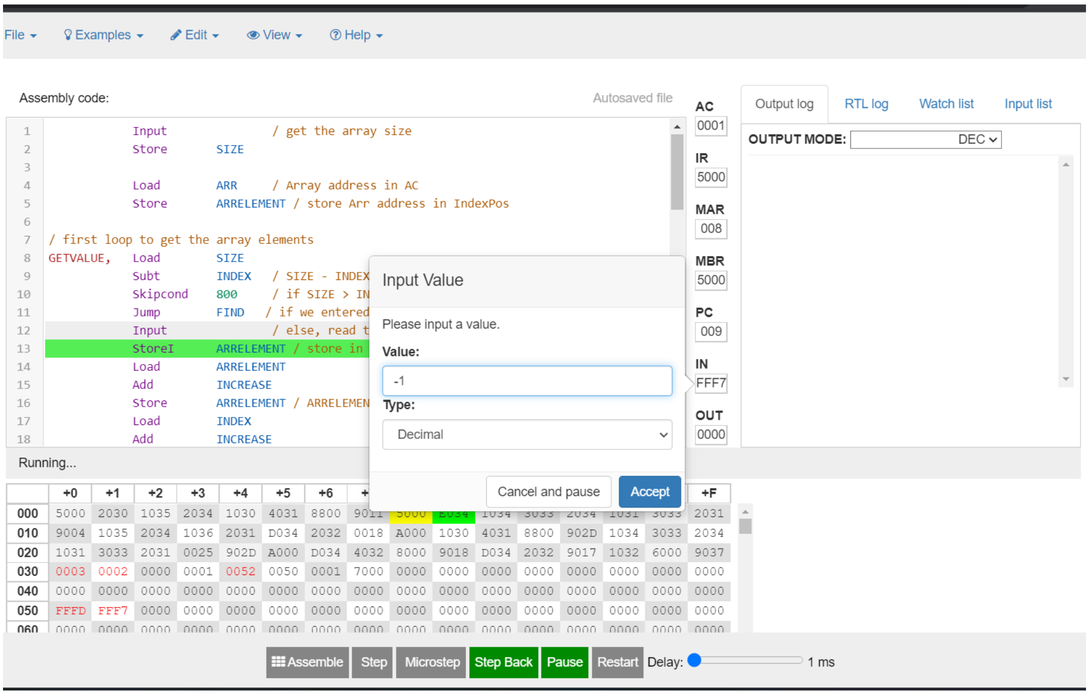
Type: Decimal

AC 0002 | IR 5000 | MAR 008 | MBR 5000 | PC 009 | IN FFFD | OUT 0000

OUTPUT MODE: DEC

Running...

	+0	+1	+2	+3	+4	+5	+6	+								+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	A000	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0001	0000	0001	0051	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	FFFD	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000



Assembly code:

```
            Input                   / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > IN
            Jump        FIND   / if we entered
            Input                   / else, read t
            StoreI      ARRELEMENT / store in
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMEN
            Load        INDEX
            Add         INCREASE
```

AC 0001
IR 5000
MAR 008
MBR 5000
PC 009
IN FFF7
OUT 0000

OUTPUT MODE: DEC

Input Value

Please input a value.

Value: -1

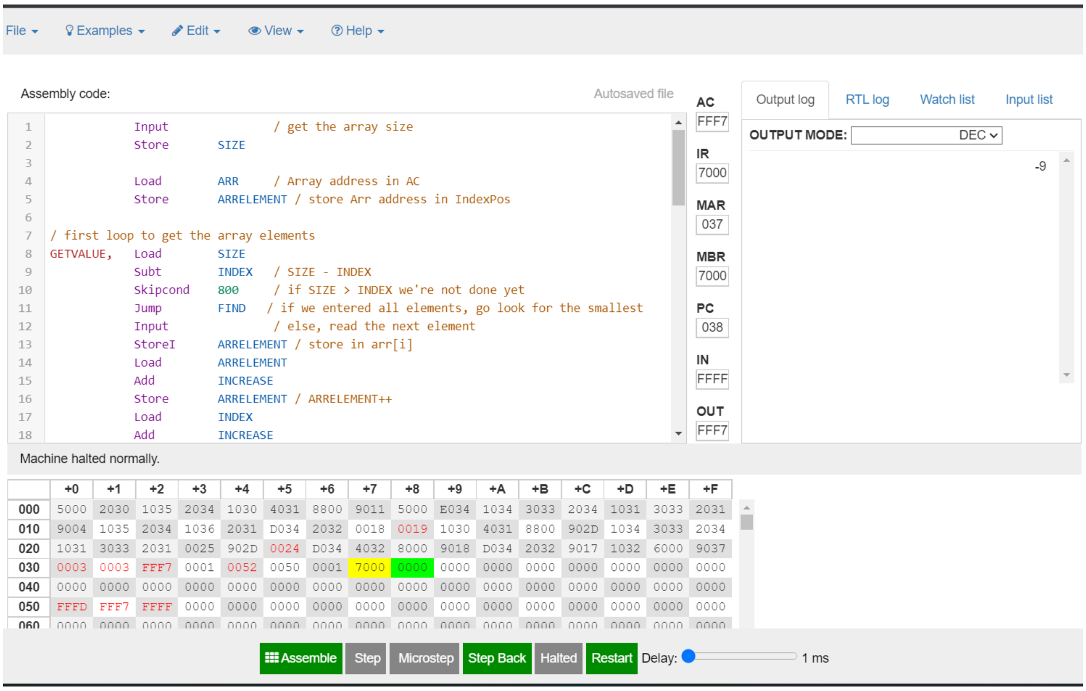
Type: Decimal

Cancel and pause | Accept

Running...

	+0	+1	+2	+3	+4	+5	+6	+								+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	A000	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0002	0000	0001	0052	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	FFFD	FFF7	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Output:



Assembly code:

```
            Input                   / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > INDEX we're not done yet
            Jump        FIND   / if we entered all elements, go look for the smallest
            Input                   / else, read the next element
            StoreI      ARRELEMENT / store in arr[i]
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMENT++
            Load        INDEX
            Add         INCREASE
```

AC FFF7
IR 7000
MAR 037
MBR 7000
PC 038
IN FFFF
OUT FFF7

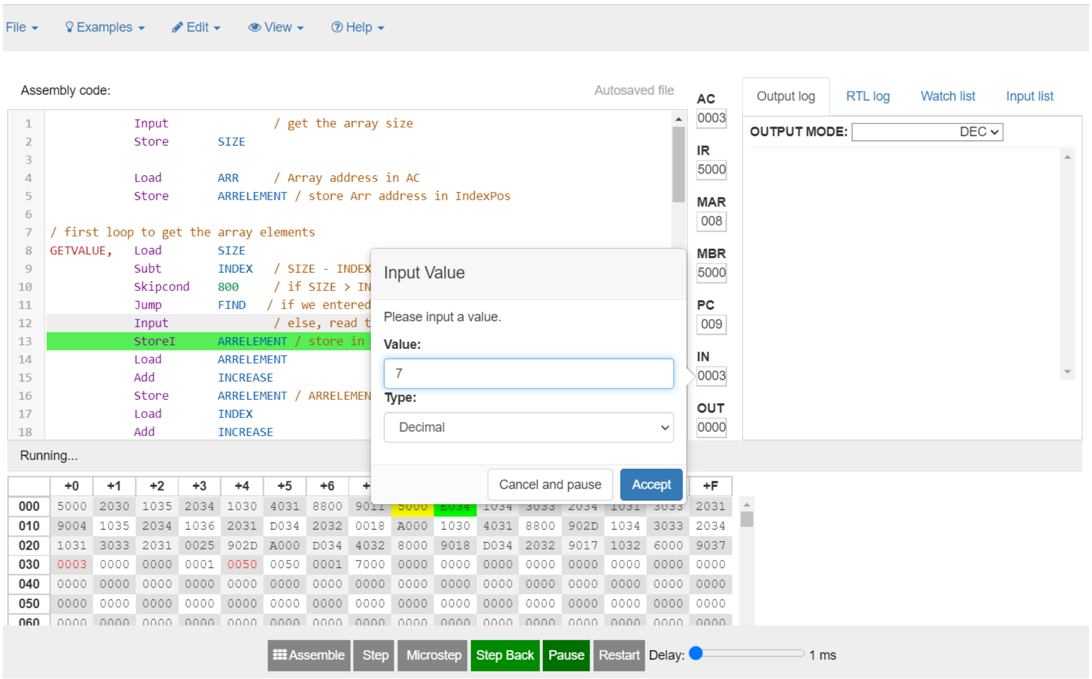
OUTPUT MODE: DEC

```
-9
```

Machine halted normally.

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	0019	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	0024	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0003	FFF7	0001	0052	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	FFFD	FFF7	FFFF	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Case 6: Unsorted array of positive numbers [7, 1, 9]



Assembly code:

Autosaved file

```
          Input                 / get the array size
          Store       SIZE

          Load        ARR       / Array address in AC
          Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
          Subt        INDEX     / SIZE - INDEX
          Skipcond    800       / if SIZE > IN
          Jump        FIND     / if we entered
          Input                 / else, read t
          StoreI      ARRELEMENT / store in
          Load        ARRELEMENT
          Add         INCREASE
          Store       ARRELEMENT / ARRELEMEN
          Load        INDEX
          Add         INCREASE
```

Running...

Input Value

Please input a value.

Value: 7

Type: Decimal

Cancel and pause | Accept

AC 0003

IR 5000

MAR 008

MBR 5000

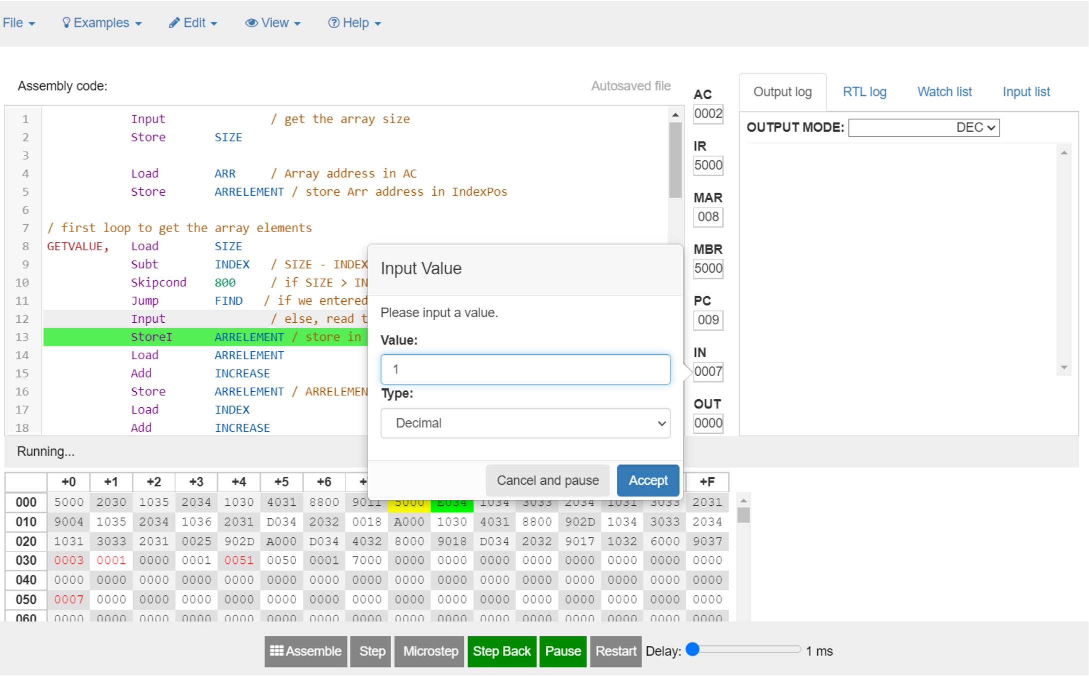
PC 009

IN 0003

OUT 0000

OUTPUT MODE: DEC

	+0	+1	+2	+3	+4	+5	+6	+								+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	A000	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0000	0000	0001	0050	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000



Assembly code:

Autosaved file

```
          Input                 / get the array size
          Store       SIZE

          Load        ARR       / Array address in AC
          Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
          Subt        INDEX     / SIZE - INDEX
          Skipcond    800       / if SIZE > IN
          Jump        FIND     / if we entered
          Input                 / else, read t
          StoreI      ARRELEMENT / store in
          Load        ARRELEMENT
          Add         INCREASE
          Store       ARRELEMENT / ARRELEMEN
          Load        INDEX
          Add         INCREASE
```

Running...

Input Value

Please input a value.

Value: 1

Type: Decimal

Cancel and pause | Accept

AC 0002

IR 5000

MAR 008

MBR 5000

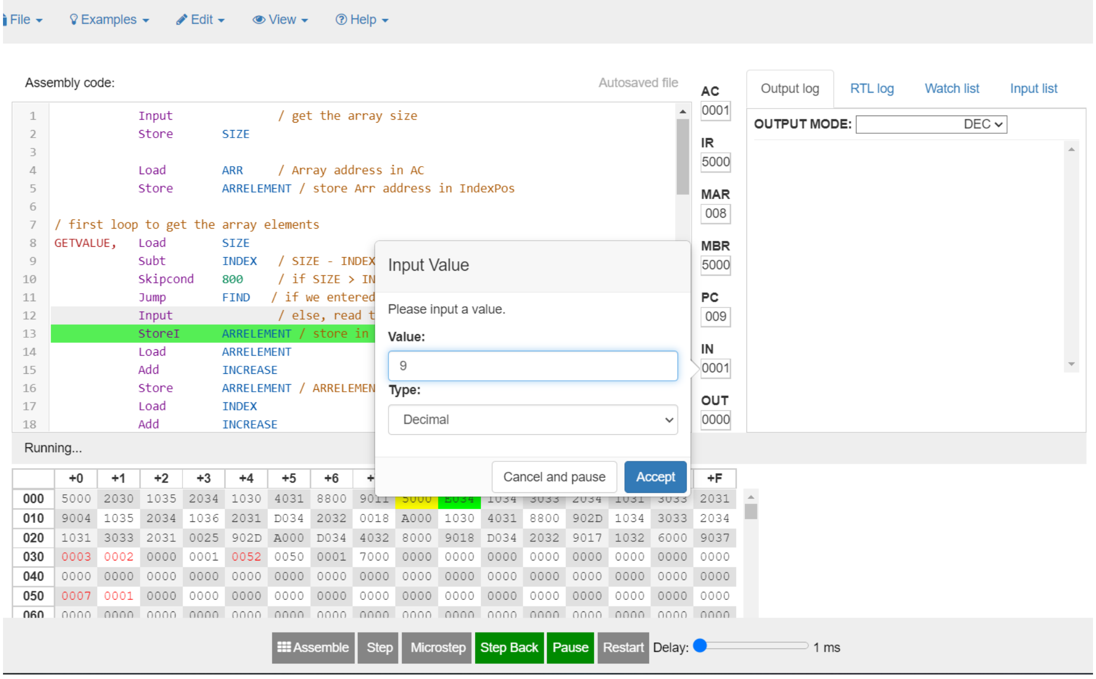
PC 009

IN 0007

OUT 0000

OUTPUT MODE: DEC

	+0	+1	+2	+3	+4	+5	+6	+								+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	A000	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0001	0000	0001	0051	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0007	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000



Assembly code: Autosaved file

```
            Input                  / get the array size
            Store       SIZE

            Load        ARR      / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX    / SIZE - INDEX
            Skipcond    800      / if SIZE > IN
            Jump        FIND    / if we entered
            Input                  / else, read t
            StoreI      ARRELEMENT / store in
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMEN
            Load        INDEX
            Add         INCREASE
```

Input Value

Please input a value.

Value: 9

Type: Decimal

Cancel and pause | Accept

AC 0001
IR 5000
MAR 008
MBR 5000
PC 009
IN 0001
OUT 0000

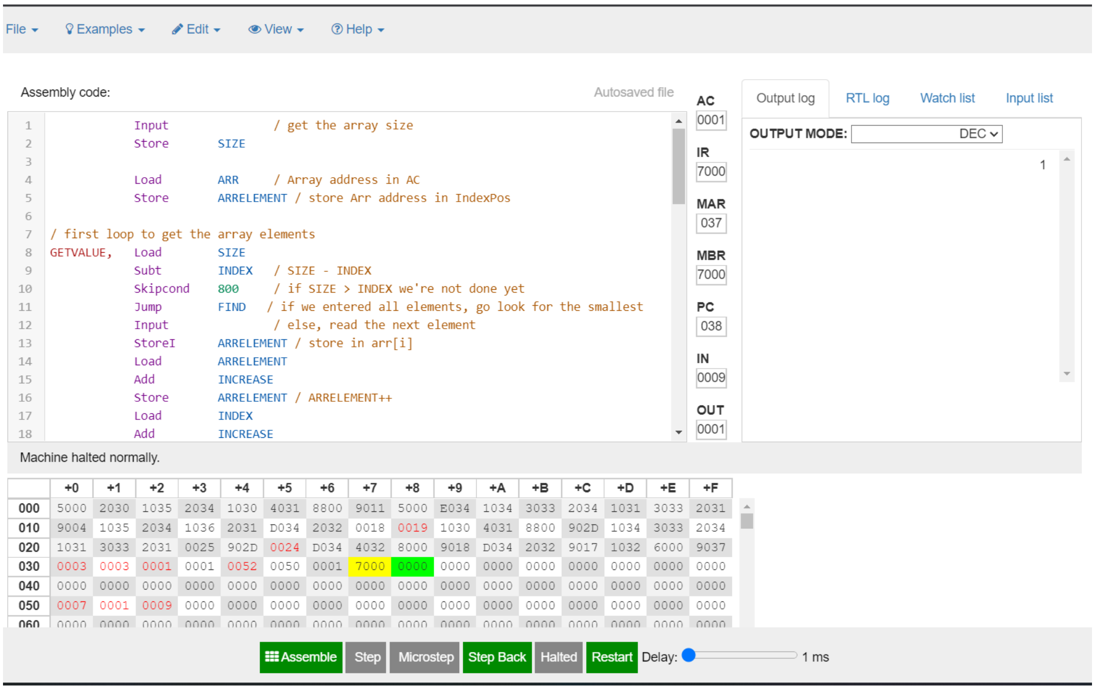
OUTPUT MODE: DEC

Running...

	+0	+1	+2	+3	+4	+5	+6	+								+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	A000	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0002	0000	0001	0052	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0007	0001	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Delay: 1 ms

Output:



Assembly code: Autosaved file

```
            Input                  / get the array size
            Store       SIZE

            Load        ARR      / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX    / SIZE - INDEX
            Skipcond    800      / if SIZE > INDEX we're not done yet
            Jump        FIND    / if we entered all elements, go look for the smallest
            Input                  / else, read the next element
            StoreI      ARRELEMENT / store in arr[i]
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMENT++
            Load        INDEX
            Add         INCREASE
```

AC 0001
IR 7000
MAR 037
MBR 7000
PC 038
IN 0009
OUT 0001

OUTPUT MODE: DEC

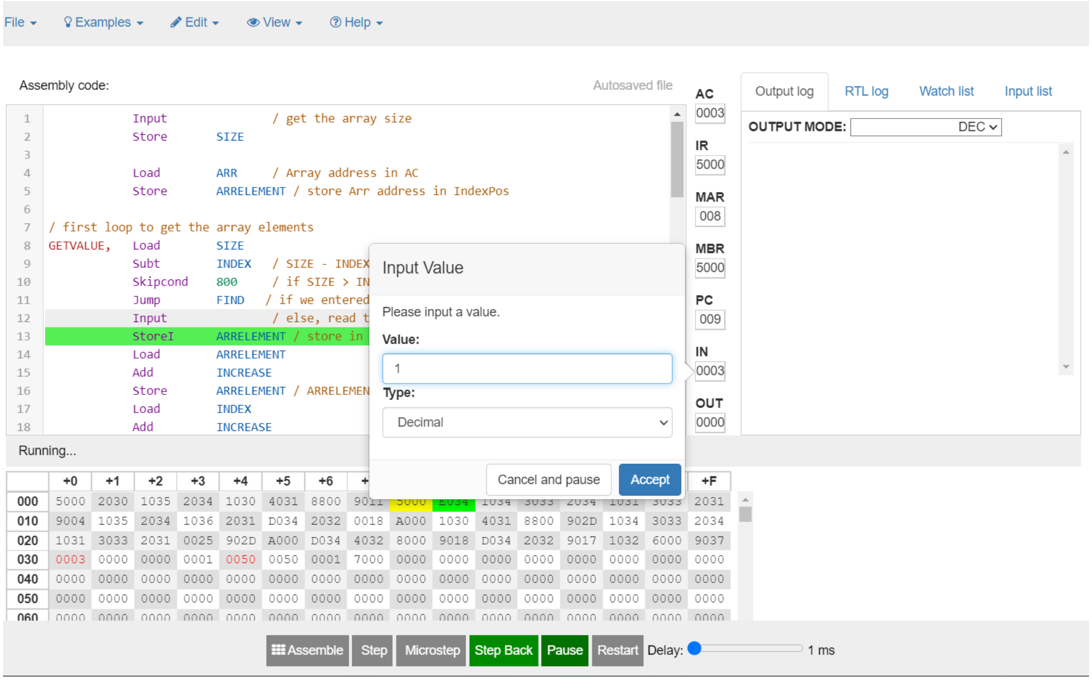
```
1
```

Machine halted normally.

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	0019	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	0024	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0003	0001	0001	0052	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0007	0001	0009	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Delay: 1 ms

Case 7: array of the same number [1, 1, 1]



Assembly code: Autosaved file

```
            Input               / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > IN
            Jump        FIND   / if we entered
            Input               / else, read t
            StoreI      ARRELEMENT / store in
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMEN
            Load        INDEX
            Add         INCREASE
```

Running...

AC 0003
IR 5000
MAR 008
MBR 5000
PC 009
IN 0003
OUT 0000

OUTPUT MODE: DEC

Input Value

Please input a value.

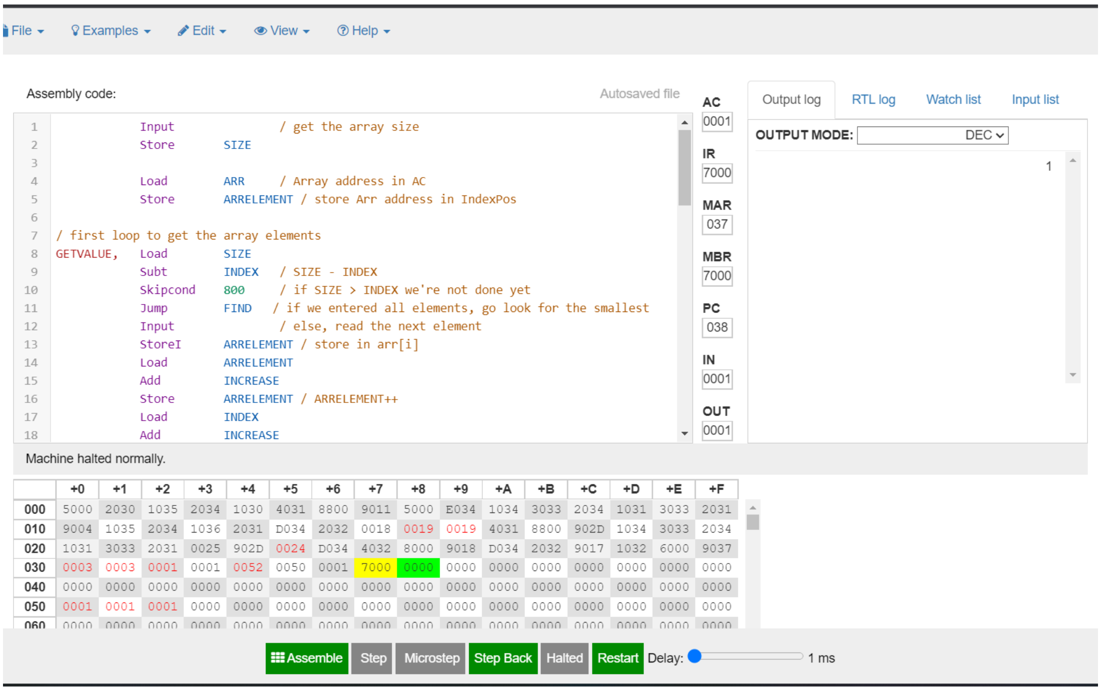
Value: 1

Type: Decimal

Cancel and pause | Accept

```
      +0   +1   +2   +3   +4   +5   +6   +                                  +F
000  5000 2030 1035 2034 1030 4031 8800 9011 5000 E034 1034 3033 2034 1031 3033 2031
010  9004 1035 2034 1036 2031 D034 2032 0018 A000 1030 4031 8800 902D 1034 3033 2034
020  1031 3033 2031 0025 902D A000 D034 4032 8000 9018 D034 2032 9017 1032 6000 9037
030  0003 0000 0000 0001 0050 0050 0001 7000 0000 0000 0000 0000 0000 0000 0000 0000
040  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
050  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
060  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
```

Output:



Assembly code: Autosaved file

```
            Input               / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > INDEX we're not done yet
            Jump        FIND   / if we entered all elements, go look for the smallest
            Input               / else, read the next element
            StoreI      ARRELEMENT / store in arr[i]
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMENT++
            Load        INDEX
            Add         INCREASE
```

Machine halted normally.

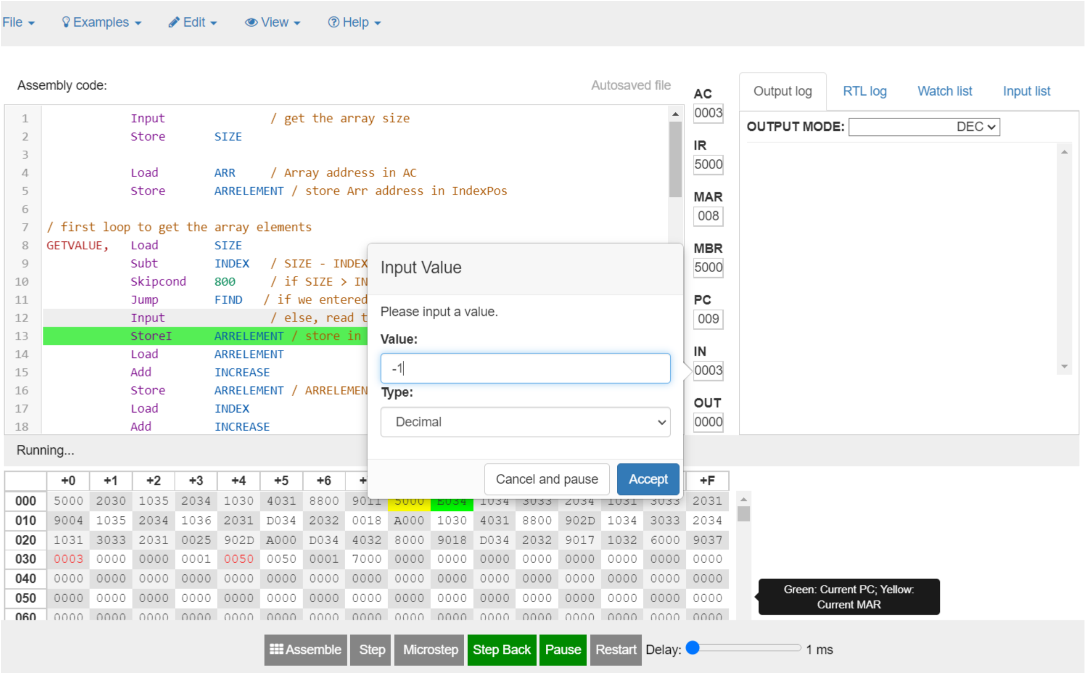
AC 0001
IR 7000
MAR 037
MBR 7000
PC 038
IN 0001
OUT 0001

OUTPUT MODE: DEC

```
1
```

```
      +0   +1   +2   +3   +4   +5   +6   +7   +8   +9   +A   +B   +C   +D   +E   +F
000  5000 2030 1035 2034 1030 4031 8800 9011 5000 E034 1034 3033 2034 1031 3033 2031
010  9004 1035 2034 1036 2031 D034 2032 0018 0019 0019 4031 8800 902D 1034 3033 2034
020  1031 3033 2031 0025 902D 0024 D034 4032 8000 9018 D034 2032 9017 1032 6000 9037
030  0003 0003 0001 0001 0052 0050 0001 7000 0000 0000 0000 0000 0000 0000 0000 0000
040  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
050  0001 0001 0001 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
060  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
```

Case 8: array of integers [-1, 1, 0]



Assembly code:

```
            Input               / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > IN
            Jump        FIND   / if we entered
            Input               / else, read t
            StoreI      ARRELEMENT / store in
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMEN
            Load        INDEX
            Add         INCREASE
```

Running...

AC 0003

IR 5000

MAR 008

MBR 5000

PC 009

IN 0003

OUT 0000

OUTPUT MODE: DEC

Input Value

Please input a value.

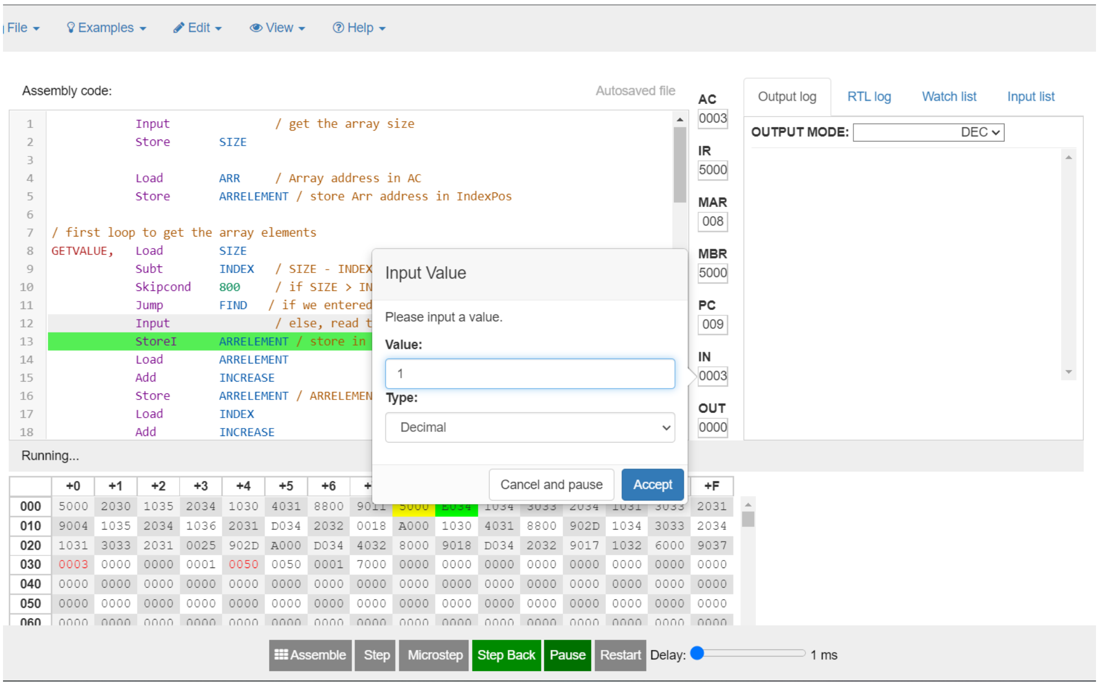
Value: -1

Type: Decimal

Cancel and pause · Accept

	+0	+1	+2	+3	+4	+5	+6	+								+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	A000	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0000	0000	0001	0050	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Delay: 1 ms



Assembly code:

```
            Input               / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > IN
            Jump        FIND   / if we entered
            Input               / else, read t
            StoreI      ARRELEMENT / store in
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMEN
            Load        INDEX
            Add         INCREASE
```

Running...

AC 0003

IR 5000

MAR 008

MBR 5000

PC 009

IN 0003

OUT 0000

OUTPUT MODE: DEC

Input Value

Please input a value.

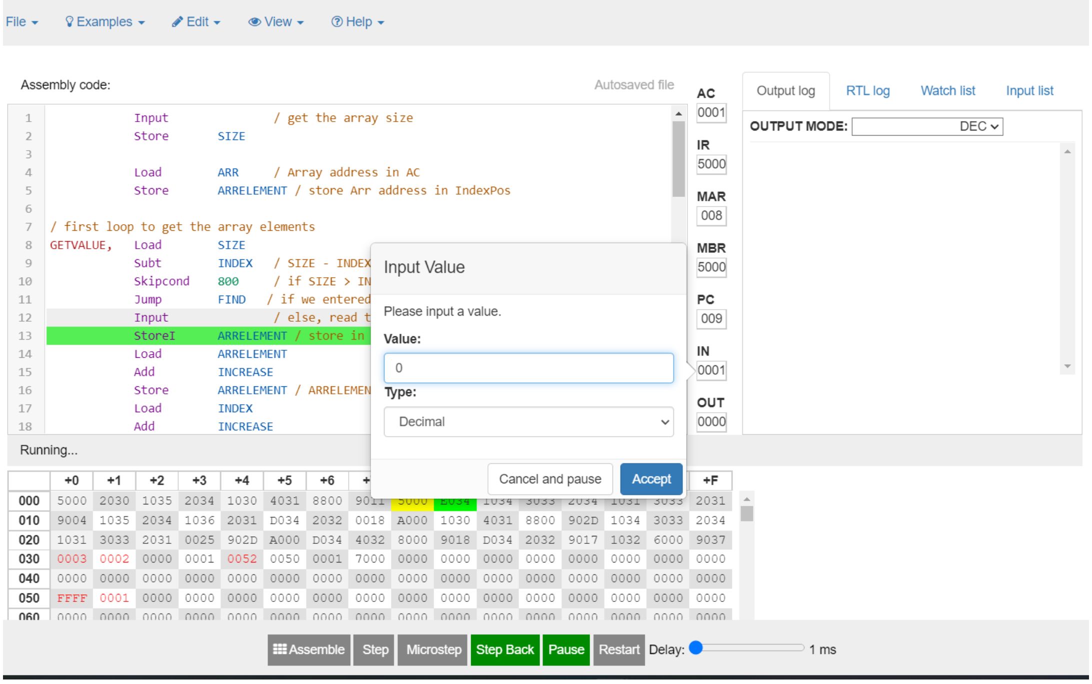
Value: 1

Type: Decimal

Cancel and pause · Accept

	+0	+1	+2	+3	+4	+5	+6	+								+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	A000	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0000	0000	0001	0050	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Delay: 1 ms



Assembly code:

```
            Input              / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > IN
            Jump        FIND   / if we entered
            Input              / else, read t
            StoreI      ARRELEMENT / store in
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMEN
            Load        INDEX
            Add         INCREASE
```

Running...

Input Value

Please input a value.

Value: 0

Type: Decimal

Cancel and pause | Accept

AC 0001

IR 5000

MAR 008

MBR 5000

PC 009

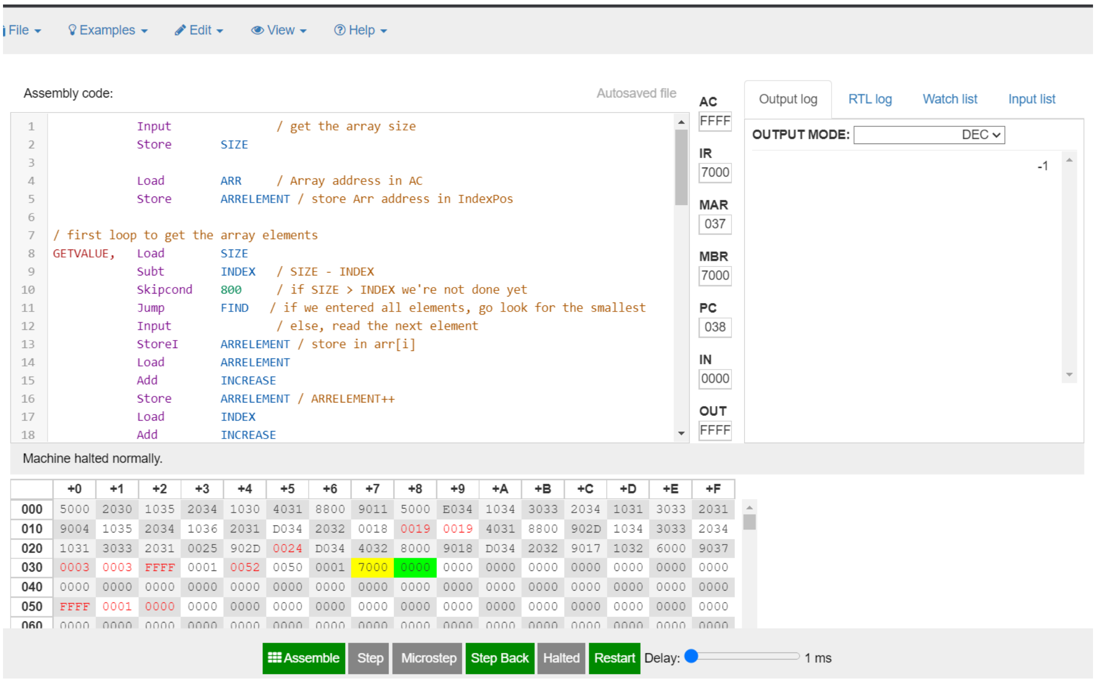
IN 0001

OUT 0000

OUTPUT MODE: DEC

	+0	+1	+2	+3	+4	+5	+6	+								+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	A000	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0002	0000	0001	0052	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	FFFF	0001	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Output:



Assembly code:

```
            Input              / get the array size
            Store       SIZE

            Load        ARR     / Array address in AC
            Store       ARRELEMENT / store Arr address in IndexPos

/ first loop to get the array elements
GETVALUE,   Load        SIZE
            Subt        INDEX   / SIZE - INDEX
            Skipcond    800     / if SIZE > INDEX we're not done yet
            Jump        FIND   / if we entered all elements, go look for the smallest
            Input              / else, read the next element
            StoreI      ARRELEMENT / store in arr[i]
            Load        ARRELEMENT
            Add         INCREASE
            Store       ARRELEMENT / ARRELEMENT++
            Load        INDEX
            Add         INCREASE
```

Machine halted normally.

AC FFFF

IR 7000

MAR 037

MBR 7000

PC 038

IN 0000

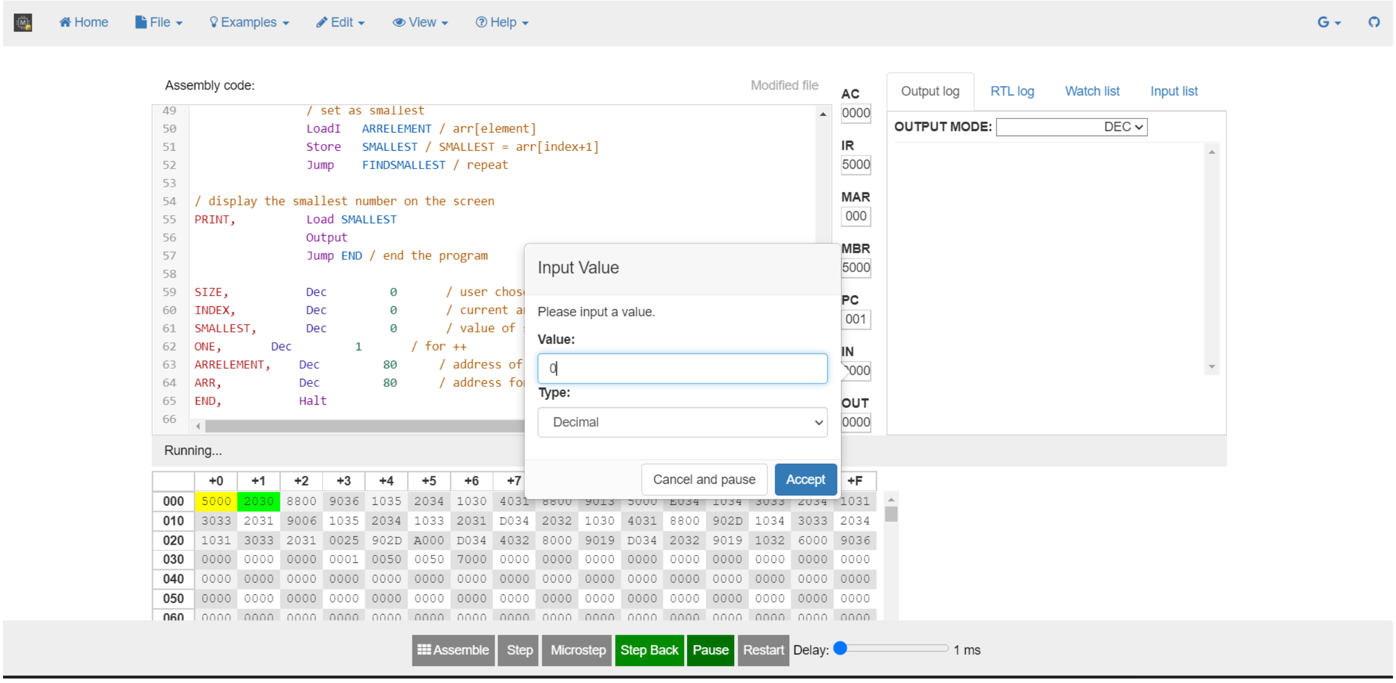
OUT FFFF

OUTPUT MODE: DEC

```
-1
```

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	5000	2030	1035	2034	1030	4031	8800	9011	5000	E034	1034	3033	2034	1031	3033	2031
010	9004	1035	2034	1036	2031	D034	2032	0018	0019	0019	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	0024	D034	4032	8000	9018	D034	2032	9017	1032	6000	9037
030	0003	0003	FFFF	0001	0052	0050	0001	7000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	FFFF	0001	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Case 9: size = 0



Assembly code:

Modified file

```
                / set as smallest
                LoadI   ARRELEMENT / arr[element]
                Store   SMALLEST / SMALLEST = arr[index+1]
                Jump    FINDSMALLEST / repeat

/ display the smallest number on the screen
PRINT,          Load SMALLEST
                Output
                Jump END / end the program

SIZE,           Dec         0       / user chos
INDEX,          Dec         0       / current a
SMALLEST,       Dec         0       / value of
ONE,       Dec         1       / for ++
ARRELEMENT,     Dec         80      / address of
ARR,            Dec         80      / address fo
END,            Halt
```

Running...

AC 0000

IR 5000

MAR 000

MBR 5000

PC 001

IN 000

OUT 0000

OUTPUT MODE: DEC

Input Value

Please input a value.

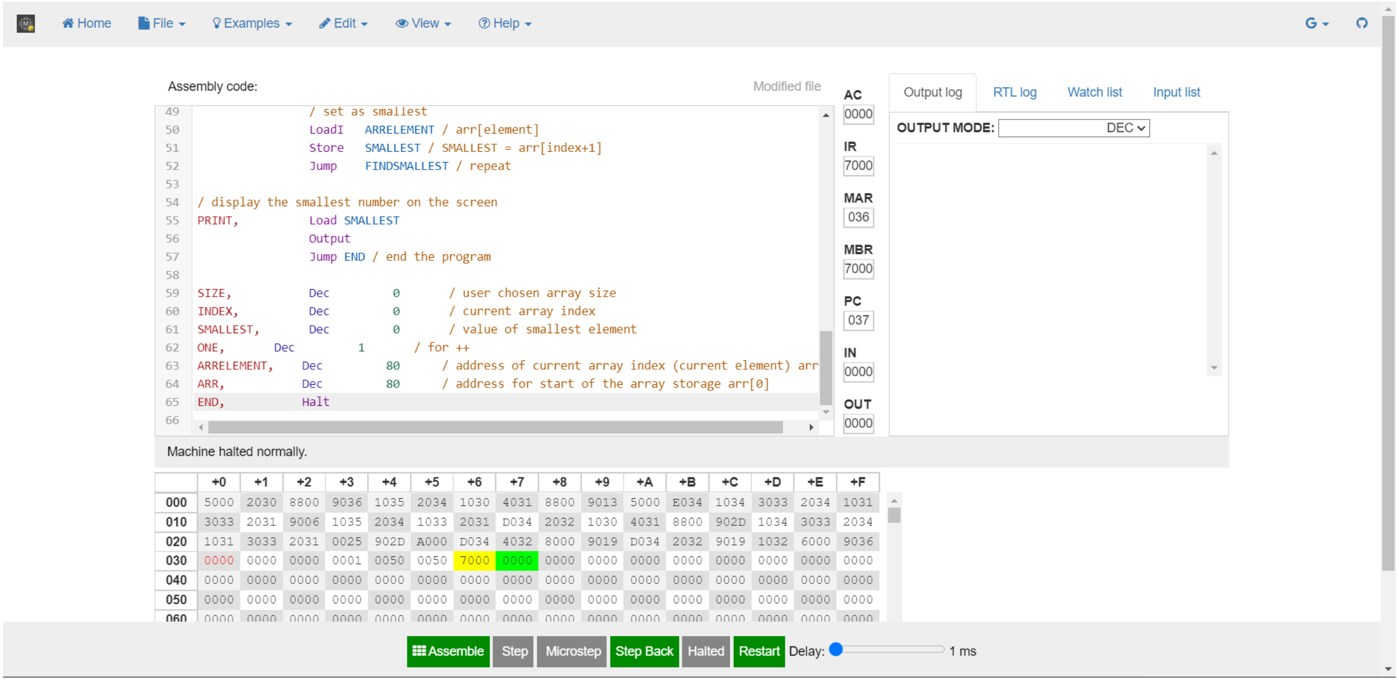
Value: 0

Type: Decimal

	+0	+1	+2	+3	+4	+5	+6	+7								+F
000	5000	2030	8800	9036	1035	2034	1030	4031	8800	9013	5000	E034	1034	3033	2034	1031
010	3033	2031	9006	1035	2034	1033	2031	D034	2032	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9019	D034	2032	9019	1032	6000	9036
030	0000	0000	0000	0001	0050	0050	7000	0000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Delay: 1 ms

Output:



Assembly code:

Modified file

```
                / set as smallest
                LoadI   ARRELEMENT / arr[element]
                Store   SMALLEST / SMALLEST = arr[index+1]
                Jump    FINDSMALLEST / repeat

/ display the smallest number on the screen
PRINT,          Load SMALLEST
                Output
                Jump END / end the program

SIZE,           Dec         0       / user chosen array size
INDEX,          Dec         0       / current array index
SMALLEST,       Dec         0       / value of smallest element
ONE,       Dec         1       / for ++
ARRELEMENT,     Dec         80      / address of current array index (current element) arr
ARR,            Dec         80      / address for start of the array storage arr[0]
END,            Halt
```

Machine halted normally.

AC 0000

IR 7000

MAR 036

MBR 7000

PC 037

IN 0000

OUT 0000

OUTPUT MODE: DEC

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	5000	2030	8800	9036	1035	2034	1030	4031	8800	9013	5000	E034	1034	3033	2034	1031
010	3033	2031	9006	1035	2034	1033	2031	D034	2032	1030	4031	8800	902D	1034	3033	2034
020	1031	3033	2031	0025	902D	A000	D034	4032	8000	9019	D034	2032	9019	1032	6000	9036
030	0000	0000	0000	0001	0050	0050	7000	0000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Delay: 1 ms

Comparison between the C++ program (High level language) and Marie Program(Low level language):

	C++	***Marie***
Variables	We declared them with the data type and the variable name.	The variables will be declared using tags and its value.
Storing the elements	The elements will be stored in the array we declared and each time we will go to the next index using for loop, we will increase the index i by one each time (i++) and check if (i<size). Then, we read and store the element in arr[i]. this will be repeated until we reach the end of the array.	The elements will be stored in the array address and each time we will go to the next address using jump, and each time we will go to the next array address by increasing the variable ARRELEMENT by adding one and storing it in the same variable (ARRELEMENT), and while we did not reach the end of the array, we will get the element as an input and store it in the address in ARRELEMENT. we will increase the index by adding one to it and storing it in the same variable (INDEX) each time we get an input. This will be repeated as long as the index is less than size, which we tested be (SIZE-INDEX) and then Skipcond 800 to make sure we didn't reach the end of the array.
find the smallest	we used a for loop and the counter i which was increased by one each time, and the loop will only stop if i is not less than the size. To find the smallest, we started by setting the smallest to the first element in the array arr[0]. Then in the loop, we tested if the next element is less than the smallest by using the logical operator less than (arr[i]<smallest). If the condition was correct, it will set the array element to be the smallest and repeat, (if the element was not smaller, it will repeat without setting it to smallest)	To start searching for the smallest, we started from the first element in the array by loading the address to the AC (variable ARR) and storing it in the variable ARRELEMENT, then we loaded the number 1 to the AC (variable ONE) and stored it in INDEX to start from index one, then we loaded the data in the array address (variable ARRELEMENT) which represents the first element in the array, and stored it in the variable SMALLEST. We tagged this set of instructions as FIND. Then, to find the smallest, we jump to FUNCTION and store the address of the next instruction (which is print)

		After FUNCTION, we will start the loop from FINDSMALLEST, here we will repeat the following instructions as long as INDEX<SIZE. The index will be increased by loading it to the AC and adding one and storing the result in the same variable (INDEX), it will stop (jump back to the stored address in FUNCTION) if INDEX<SIZE (we tested that condition by size – index and then Skipcond 800). Else, it will continue, also, we added one to ARRELEMENT to go to the next element. This set of instructions was tagged as FINDSMALLEST. So, while we did not go through the whole array, we jumped to SUBT tag, which tested if the element is less than the SMALLEST by loading the element in the address ARRELEMENT and subtracting the SMMALLEST, if the result was less than zero, it means the element is less than SMALLEST, this was tested using Skipcond 000. If the element was smaller, it will be loaded in the AC and stored in the variable SMALLEST, then it will jump back to FINDSMALLEST to check the next element. (if the subtraction result was not less than 0, it will jump to FINDSMALLEST without setting the element to SMALLEST) After checking all elements, we will jump back to the stored address in FUNCTION which will take us to the instruction (Jump PRINT)
Output	Using cout to display the variable "smallest"	Using load the variable smallest to AC then outputting it then jumping to END
End	The program will stop when it reaches return 0 in the main function	The program will stop when it reaches the instruction halt.

Conclusion:

This project was extremely beneficial for each one of us, although we faced some complex difficulties our teamwork helped us through them all. Marie isn't the easiest coding language yet we managed to create a program that finds the smallest element from all inputs by the user. After finishing this project, we also understood the difference between C++ "high level language" and Marie "low-level language"; Assembly language is used to write a code that interacts with the hardware, while C++ is a language with a user interface and much simpler commands to get the same result.