Instituto Tecnológico de Estudios Superiores de Monterrey

Tarea 3 : Programación en ensamblador

Unidad Formativa:

Modelación de sistemas mínimos y arquitecturas computacionales

(TC1032.13)

Integrantes del equipo:

Axel Jarquín Morga A01636324

Ricardo González Leal A01639036

Juan Pablo García Malta A01639025

Carla Morales López A01639225

Campus:

Guadalajara

Entregable 1 (Realiza un programa que calcule: Y=X-2*A-3*B-4*C **)**

Código:

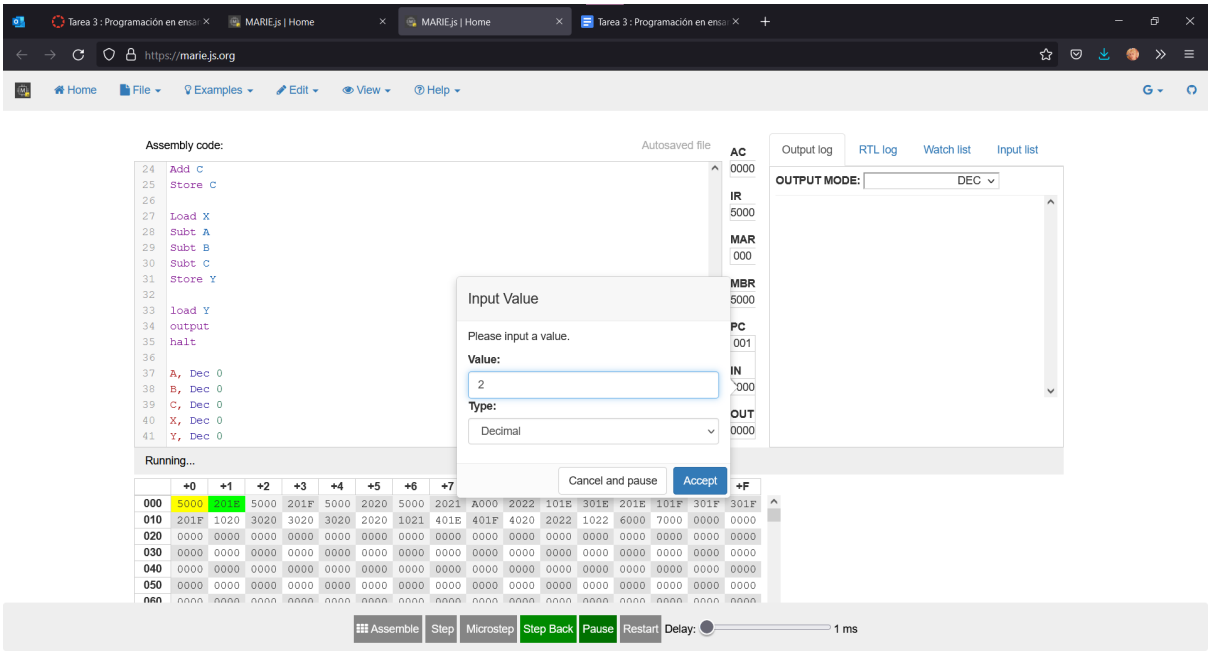
```
Input
Store A
Input
Store B
Input
Store C
Input
Store X
Clear
Store Y

load A
Add A
Store A

load B
Add B
Add B
Store B

load C
Add C
Add C
Add C
Store C

Load X
Subt A
Subt B
Subt C
```

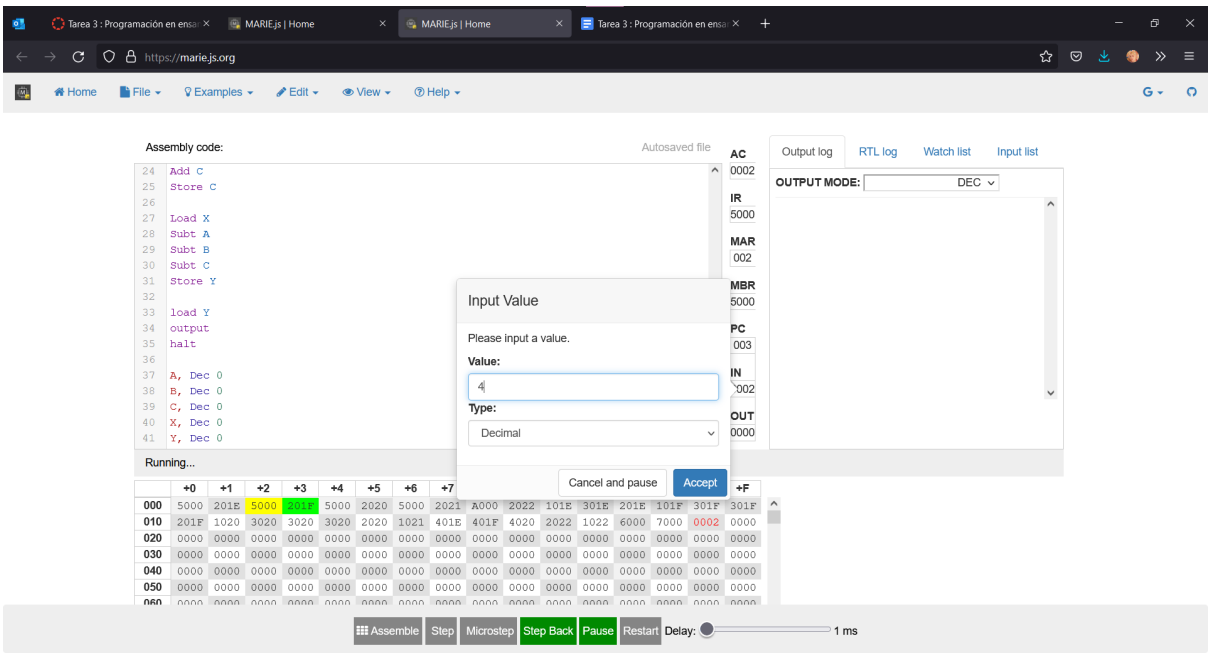
Store Y

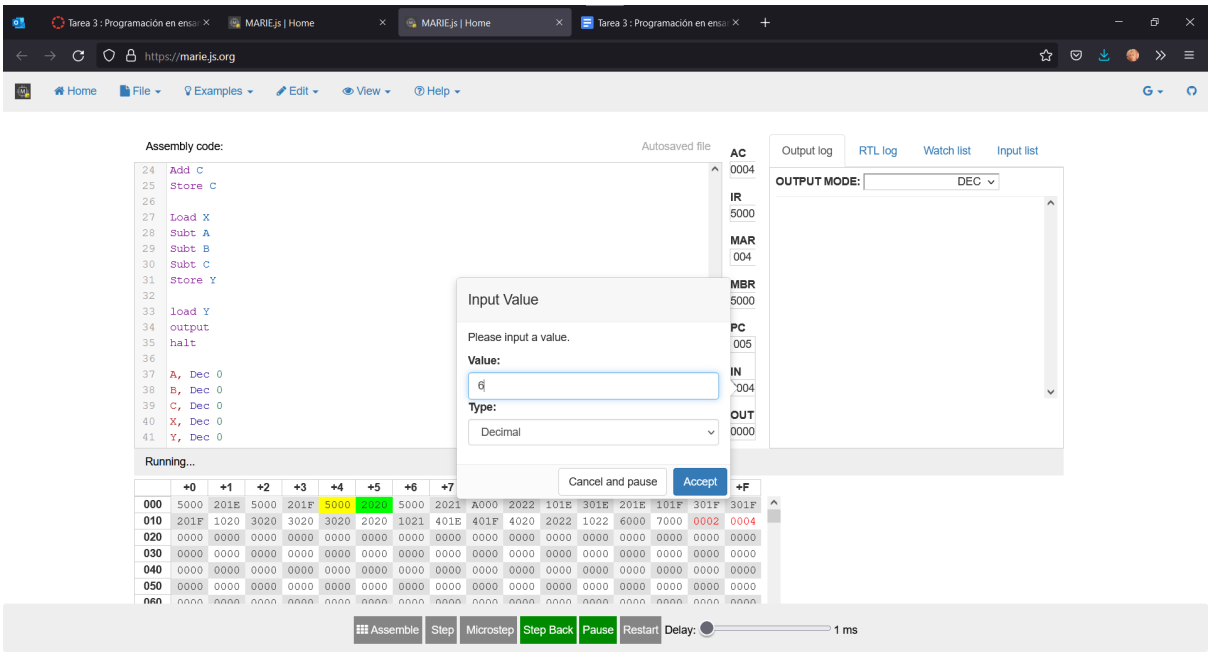
load Y
output
halt

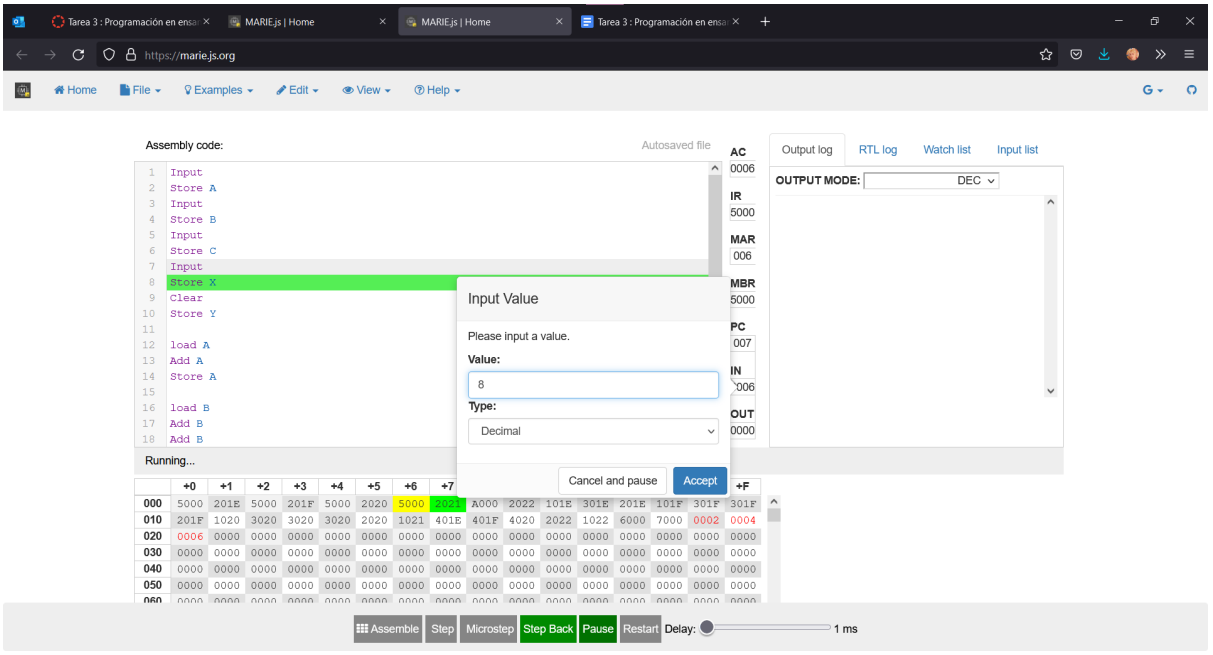
A, Dec 0
B, Dec 0
C, Dec 0
X, Dec 0
Y, Dec 0

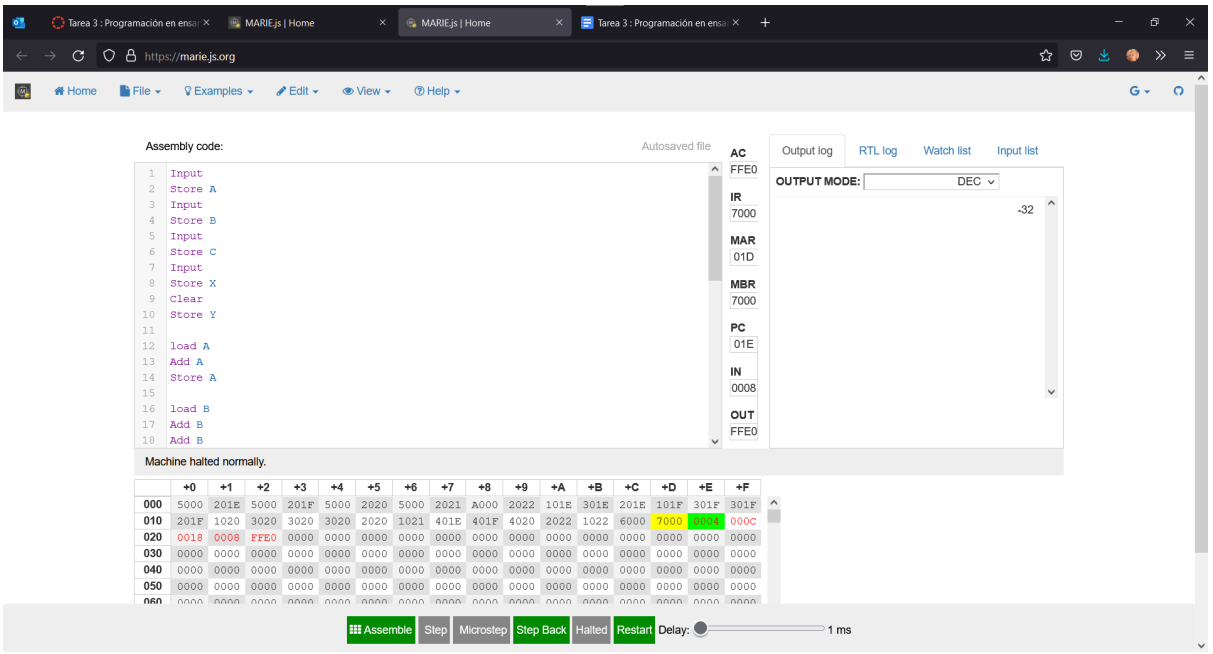
Capturas:




Tarea 3 : Programación en ensa
MARIE.js | Home
https://marie.js.org
Home
File
Examples
Edit
View
Help
Assembly code:
Autosaved file
Add C
Store C
Load X
Subt A
Subt B
Subt C
Store Y
load Y
output
halt
A, Dec 0
B, Dec 0
C, Dec 0
X, Dec 0
Y, Dec 0
Input Value
Please input a value.
Value:
4
Type:
Decimal
Cancel and pause
Accept
Output log
RTL log
Watch list
Input list
OUTPUT MODE:
DEC
AC 0002
IR 5000
MAR 002
MBR 5000
PC 003
IN 002
OUT 0000
Running...
Assemble
Step
Microstep
Step Back
Pause
Restart
Delay:
1 ms


Tarea 3 : Programación en ensa
MARIE.js | Home
https://marie.js.org
Home
File
Examples
Edit
View
Help
Assembly code:
Autosaved file
Add C
Store C
Load X
Subt A
Subt B
Subt C
Store Y
load Y
output
halt
A, Dec 0
B, Dec 0
C, Dec 0
X, Dec 0
Y, Dec 0
Input Value
Please input a value.
Value:
6
Type:
Decimal
Cancel and pause
Accept
Output log
RTL log
Watch list
Input list
OUTPUT MODE:
DEC
AC 0004
IR 5000
MAR 004
MBR 5000
PC 005
IN 004
OUT 0000
Running...
Assemble
Step
Microstep
Step Back
Pause
Restart
Delay:
1 ms





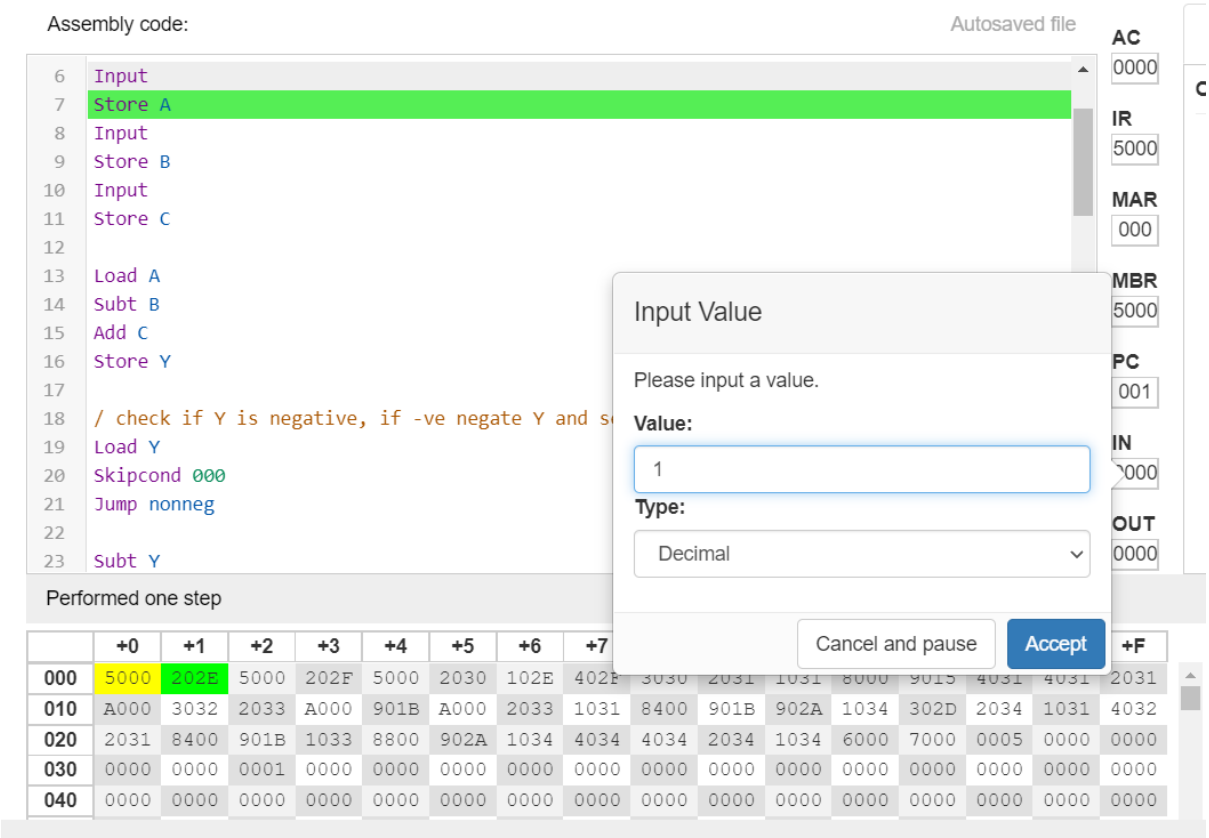
Entregable 2 (Realiza un programa que calcule: Y=5*(A-B+C) **)**

Código:

```
Input
Store A
Input
Store B
Input
Store C
```

```
Load A
Subt B
Add C
Store Y

/ check if Y is negative, if -ve negate Y and set negative flag
Load Y
Skipcond 000
Jump nonneg

Subt Y
Subt Y
Store Y
Clear
Add one
Store negflag
Clear
Jump loop

nonneg,  Clear
        Store negflag
        / check if Y is zero, if it is, then we jump to halt
        Load Y
        Skipcond 400
        Jump loop / false
        Jump halt / true

/ Loop for performing iterative addition
loop,    Load result
        Add X
        Store result

        Load Y
        Subt one
        Store Y

        Skipcond 400 / have we completed the multiplication?
        Jump loop / no; repeat loop
        / yes, so exit the loop

/ check for negative flag, if it is set, negate the result
Load negflag
Skipcond 800
```

```
Jump halt

/ negate result
Load result
Subt result
Subt result
Store result
/ run the next three instructions, which halts the program

/ Output result to user then halt program
halt,   Load result
        Output
        Halt

X, DEC 5
A, DEC 0
B, DEC 0
C, DEC 0
Y, DEC 0
one, DEC 1
negflag, DEC 0
result, DEC 0
```



Assembly code:

Autosaved file

```
6  Input
7  Store A
8  Input
9  Store B
10 Input
11 Store C
12
13 Load A
14 Subt B
15 Add C
16 Store Y
17
18 / check if Y is negative, if -ve negate Y and s
19 Load Y
20 Skipcond 000
21 Jump nonneg
22
23 Subt Y
```

AC 0001
IR 5000
MAR 002
MBR 5000
PC 003
IN 0001
OUT 0000

Input Value

Please input a value.

Value: 2

Type: Decimal

Cancel and pause | Accept

Performed one step

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	5000	202E	5000	202F	5000	2030	102E	402F	3030	2031	1031	8000	9015	4031	4031	2031
010	A000	3032	2033	A000	901B	A000	2033	1031	8400	901B	902A	1034	302D	2034	1031	4032
020	2031	8400	901B	1033	8800	902A	1034	4034	4034	2034	1034	6000	7000	0005	0001	0000
030	0000	0000	0001	0000	0000	0000	0000	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

Assembly code:

Autosaved file

```
6  Input
7  Store A
8  Input
9  Store B
10 Input
11 Store C
12
13 Load A
14 Subt B
15 Add C
16 Store Y
17
18 / check if Y is negative, if -ve negate Y and s
19 Load Y
20 Skipcond 000
21 Jump nonneg
22
23 Subt Y
```

AC 0002
IR 5000
MAR 004
MBR 5000
PC 005
IN 0002
OUT 0000

Input Value

Please input a value.

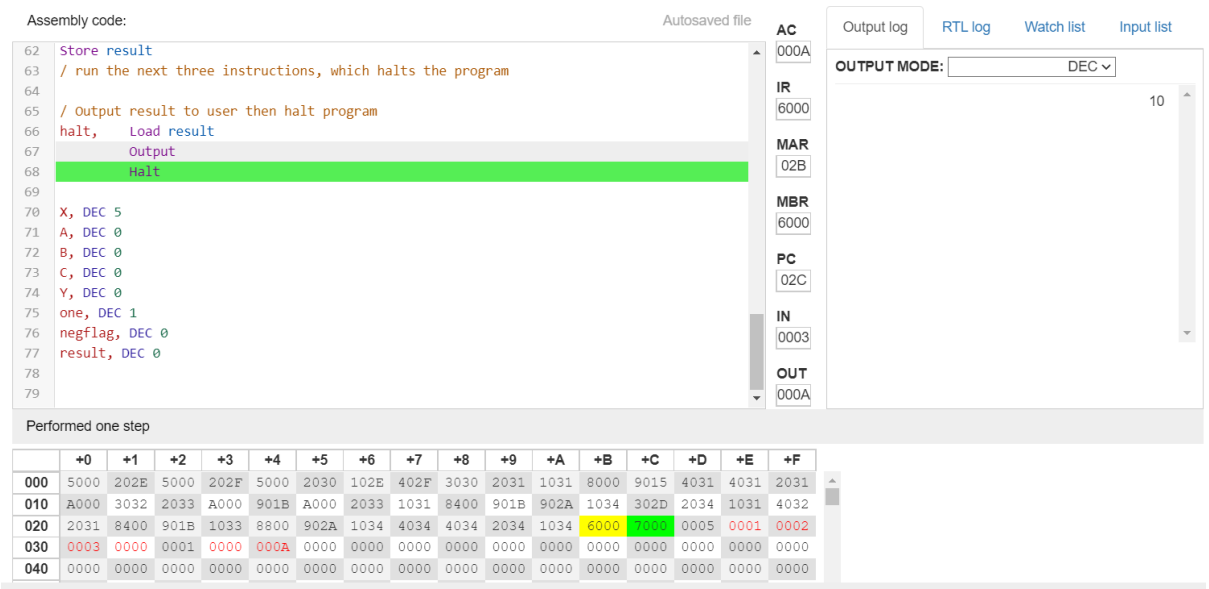
Value: 3

Type: Decimal

Cancel and pause | Accept

Performed one step

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	5000	202E	5000	202F	5000	2030	102E	402F	3030	2031	1031	8000	9015	4031	4031	2031
010	A000	3032	2033	A000	901B	A000	2033	1031	8400	901B	902A	1034	302D	2034	1031	4032
020	2031	8400	901B	1033	8800	902A	1034	4034	4034	2034	1034	6000	7000	0005	0001	0002
030	0000	0000	0001	0000	0000	0000	0000	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



Entregable 3 (Realiza un programa que invierta 4 localidades de memoria **)**

Código:

/ Programa que invierte 4 localidades de memoria.

LOOP, Clear
/ output Hello
LoadI PTR
StoreI TEMP
Output

/ increment pointer
Load PTR
Add ONE
Store PTR
Load TEMP
Add ONE
Store TEMP

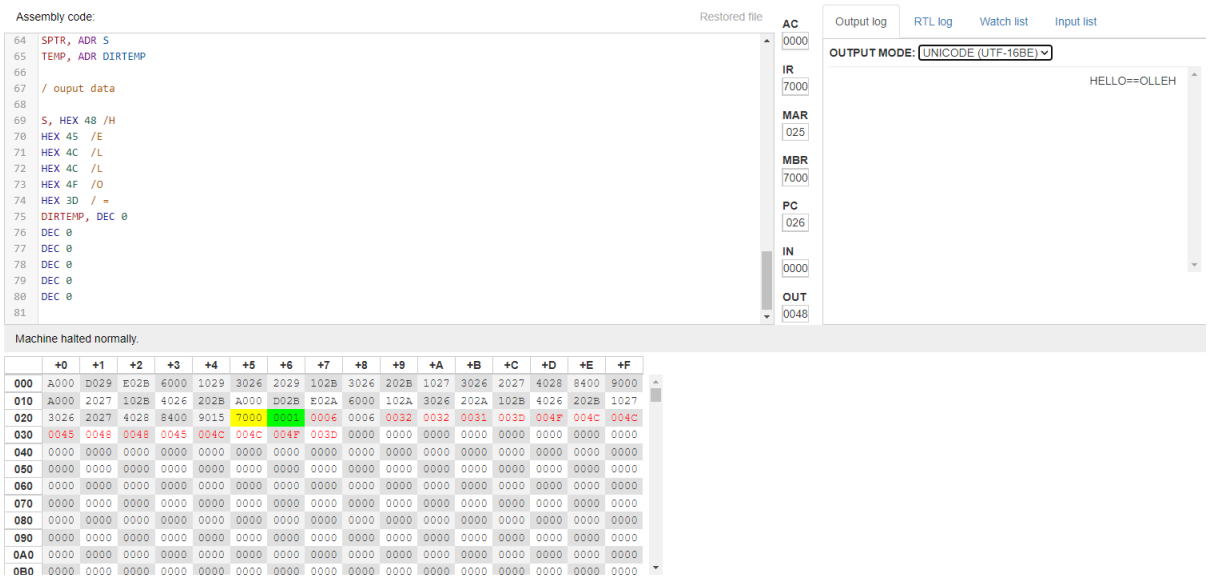
/ increment counter
Load CTR
Add ONE
Store CTR

```
/ Comprueba si se han procesado todas las palabras y se detiene si es así
Subt WORDS
Skipcond 400
Jump LOOP

/Inicia el proceso de invertir
Clear
Store CTR
Load TEMP
Subt ONE
Store TEMP
LOOP2, Clear
/ output Hello
LoadI TEMP
StoreI SPTR
Output

/ increment pointer
Load SPTR
Add ONE
Store SPTR
Load TEMP
Subt ONE
Store TEMP

/ increment counter
Load CTR
Add ONE
Store CTR

/ Comprueba si se han procesado todas las palabras y se detiene si es así
Subt WORDS
Skipcond 400
Jump LOOP2
Halt

ONE, DEC 1
CTR, DEC 0
WORDS, DEC 6

/ Apuntadores a palabras
PTR, ADR S
SPTR, ADR S
```

TEMP, ADR DIRTEMP

/ ouput data

S, HEX 48 /H
HEX 45 /E
HEX 4C /L
HEX 4C /L
HEX 4F /O
HEX 3D / =
DIRTEMP, DEC 0
DEC 0
DEC 0
DEC 0
DEC 0
DEC 0



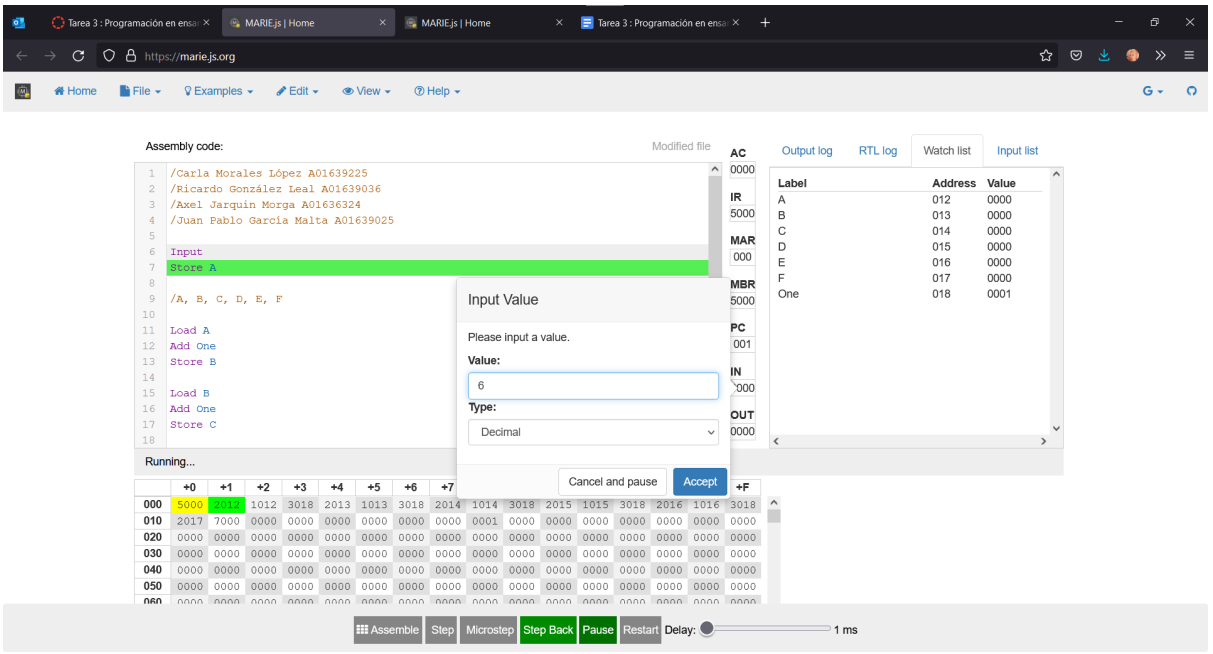
	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	A000	D029	E02B	6000	1029	3026	2029	102B	3026	202B	1027	3026	2027	4028	8400	9000
010	A000	2027	102B	4026	202B	A000	D02B	E02A	6000	102A	3026	202A	102B	4026	202B	1027
020	3026	2027	4028	8400	9015	7000	0001	0006	0006	0032	0032	0031	003D	004F	004C	004C
030	0045	0048	0048	0045	004C	004C	004F	003D	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
070	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
080	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
090	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0A0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0B0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

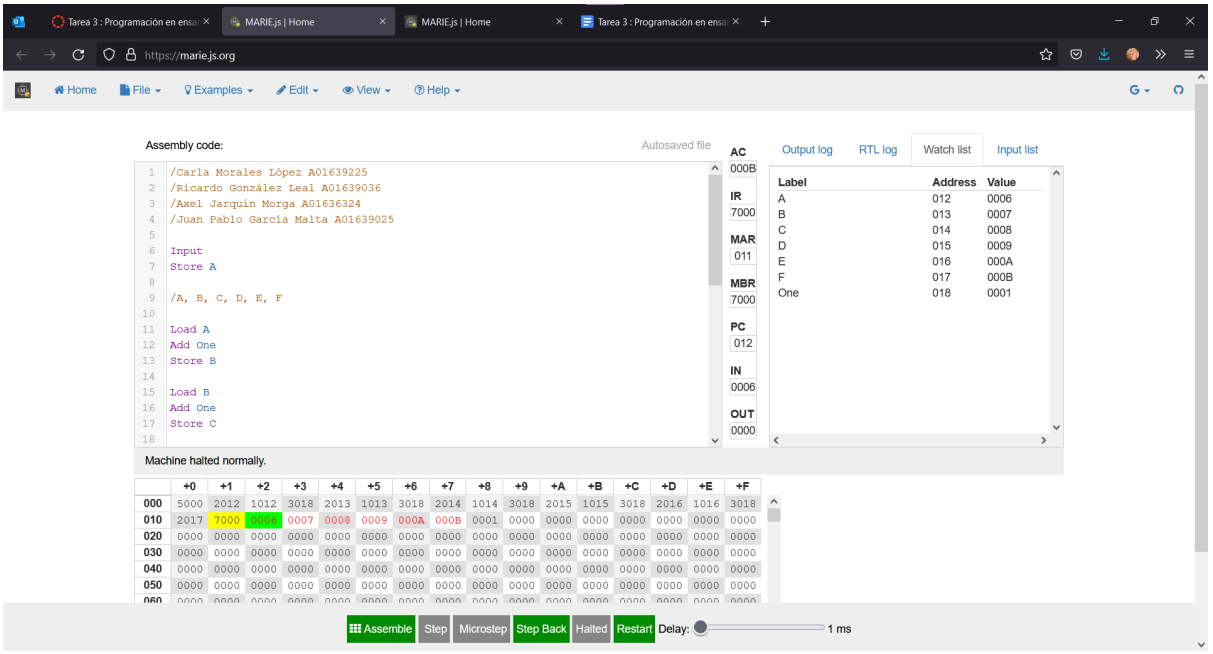
Entregable 4 (Realiza un programa que tome la primera localidad de memoria de una secuencia de 6 números, llenando las restantes cinco con en valor de la anterior más uno**.)**

Código:

```
Input
Store A

/A, B, C, D, E, F

Load A
Add One
Store B

Load B
Add One
Store C

Load C
Add One
Store D

Load D
Add One
Store E

Load E
Add One
Store F

Halt

A, Dec 0
B, Dec 0
C, Dec 0
D, Dec 0
E, Dec 0
F, Dec 0
One, Dec 1
```

Capturas:





Entregable 5 (Realiza un programa que tome una serie de cinco localidades de memoria, y a cada una sume un valor, a la primera 1, a la segunda 2, etc. **)**

Código:

```
Input
Store A
```

```
Input
Store B
Input
Store C
Input
Store D
Input
Store E

LOOP, Clear
LoadI PTR //loads the value to which the pointer refers
Add COUNTER
StoreI PTR //update the value to which the pointer refers
Output
Load COUNTER
Add ONE
Store COUNTER //updates the counter

/Increment Pointer to +1
Load PTR
Add ONE
Store PTR

/check if all the memory locations have been modified.
Load COUNTER
Subt SIX
Skipcond 400
Jump LOOP
Halt

ONE, DEC 1
SIX, DEC 6
COUNTER, DEC 1
//pointer to words
PTR, ADR A
/words
A, DEC 1
B, DEC 1
C, DEC 1
D, DEC 1
E, DEC 1
```

Assembly code:

Autosaved file

```
1  Input
2  Store A
3  Input
4  Store B
5  Input
6  Store C
7  Input
8  Store D
9  Input
10 Store E
11 Output
12
13 LOOP, Clear
14 LoadI PTR //loads the value to which the pointer refers
15 Add COUNTER
16 StoreI PTR //update the value to which the pointer refers
17 Load COUNTER
18 Add ONE
```

AC 0005

IR 2022

MAR 022

MBR 0005

PC 00A

IN 0005

OUT 0000

Performed one step

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	5000	201E	5000	201F	5000	2020	5000	2021	5000	2022	6000	A000	D01D	301C	E01D	101C
010	301A	201C	101D	301A	201D	101C	401B	8400	900B	7000	0001	0006	0001	001E	0005	0005
020	0005	0005	0005	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Assembly code:

Autosaved file

```
6  Store C
7  Input
8  Store D
9  Input
10 Store E
11
12 LOOP, Clear
13 LoadI PTR //loads the value to which the pointer refers
14 Add COUNTER
15 StoreI PTR //update the value to which the pointer refers
16 Output
17 Load COUNTER
18 Add ONE
19 Store COUNTER //updates the counter
20
21 /Increment Pointer to +1
22 Load PTR
23 Add ONE
```

AC 0006

IR 6000

MAR 00E

MBR 6000

PC 00F

IN 0005

OUT 0006

Output log | RTL log | Watch list | Input list

OUTPUT MODE: DEC

6

Performed one step

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	5000	201E	5000	201F	5000	2020	5000	2021	5000	2022	A000	D01D	301C	E01D	6000	101C
010	301A	201C	101D	301A	201D	101C	401B	8400	900A	7000	0001	0006	0001	001E	0006	0005
020	0005	0005	0005	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Assembly code: Autosaved file

```
16  Output
17  Load COUNTER
18  Add ONE
19  Store COUNTER //updates the counter
20
21  /Increment Pointer to +1
22  Load PTR
23  Add ONE
24  Store PTR
25
26  /check if all the memory locations have been modified.
27  Load COUNTER
28  Subt SIX
29  Skipcond 400
30  Jump LOOP
31  Halt
32
33  ONE, DEC 1
```

AC FFFD
IR 8400
MAR 017
MBR 8400
PC 018
IN 0005
OUT 0007

Output log | RTL log | Watch list | Input list

OUTPUT MODE: DEC

6
7

Performed one step

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	5000	201E	5000	201F	5000	2020	5000	2021	5000	2022	A000	D01D	301C	E01D	6000	101C
010	301A	201C	101D	301A	201D	101C	401B	8400	900A	7000	0001	0006	0003	0020	0006	0007
020	0005	0005	0005	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Assembly code: Autosaved file

```
26  /check if all the memory locations have been modified.
27  Load COUNTER
28  Subt SIX
29  Skipcond 400
30  Jump LOOP
31  Halt
32
33  ONE, DEC 1
34  SIX, DEC 6
35  COUNTER, DEC 1
36  //pointer to words
37  PTR, ADR A
38  /words
39  A, DEC 1
40  B, DEC 1
41  C, DEC 1
42  D, DEC 1
43  E, DEC 1
```

AC 0000
IR 8400
MAR 017
MBR 8400
PC 019
IN 0005
OUT 000A

Output log | RTL log | Watch list | Input list

OUTPUT MODE: DEC

6
7
8
9
10

Performed one step

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	5000	201E	5000	201F	5000	2020	5000	2021	5000	2022	A000	D01D	301C	E01D	6000	101C
010	301A	201C	101D	301A	201D	101C	401B	8400	900A	7000	0001	0006	0006	0023	0006	0007
020	0008	0009	000A	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Entregable 6 (Realiza un programa que nos diga la longitud de un String, del cual conoces la dirección donde inicia, y termina cuando encuentra el NULL en el contenido de memoria; asume que el string está en ASCII. **)**

```
LOOP, Load PTR
/ output word
LoadI PTR
Output

//Longitud
Load Longitud
Add ONE
Store Longitud

/ increment pointer
Load PTR
Add ONE
Store PTR

/ increment counter
Load CTR
Add ONE
Store CTR

/ check whether all words have been processed and halt if so
Subt WORDS
Skipcond 400
Jump LOOP
Load Longitud
Output
Halt

ONE, DEC 1
CTR, DEC 0
WORDS, DEC 19

/ initialize pointer to words
PTR, ADR S

/ ouput data
/ "Hello World!"
S, HEX 45
```

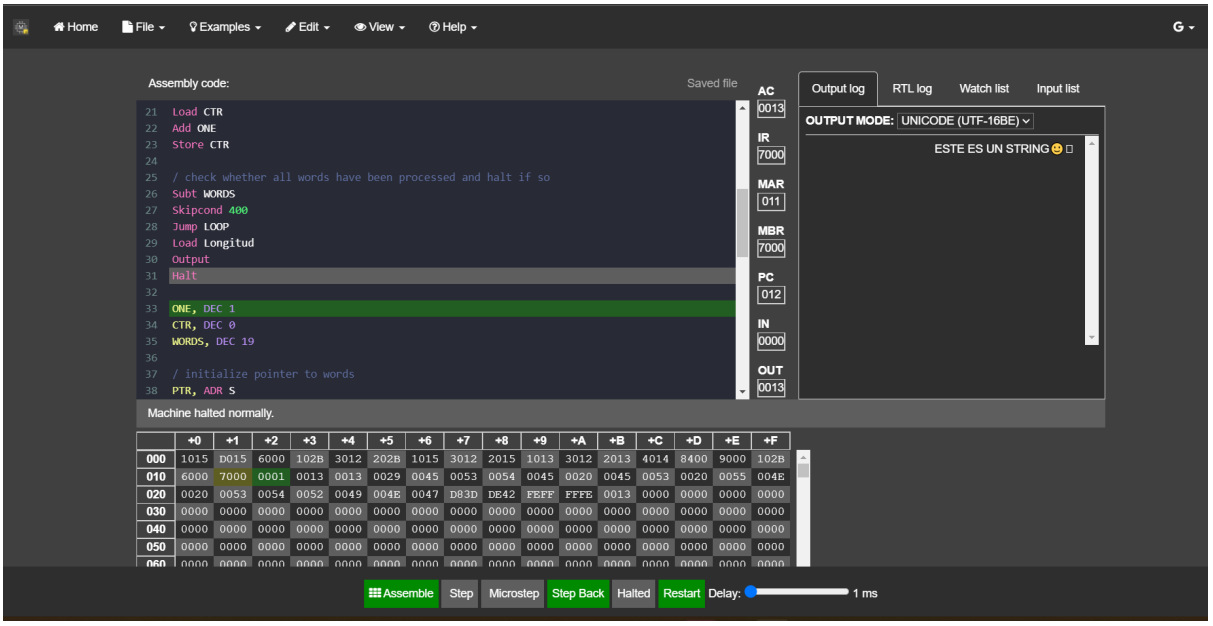
HEX 53
HEX 54
HEX 45
HEX 20
HEX 45
HEX 53
HEX 20
HEX 55
HEX 4E
HEX 20
HEX 53
HEX 54
HEX 52
HEX 49
HEX 4E
HEX 47

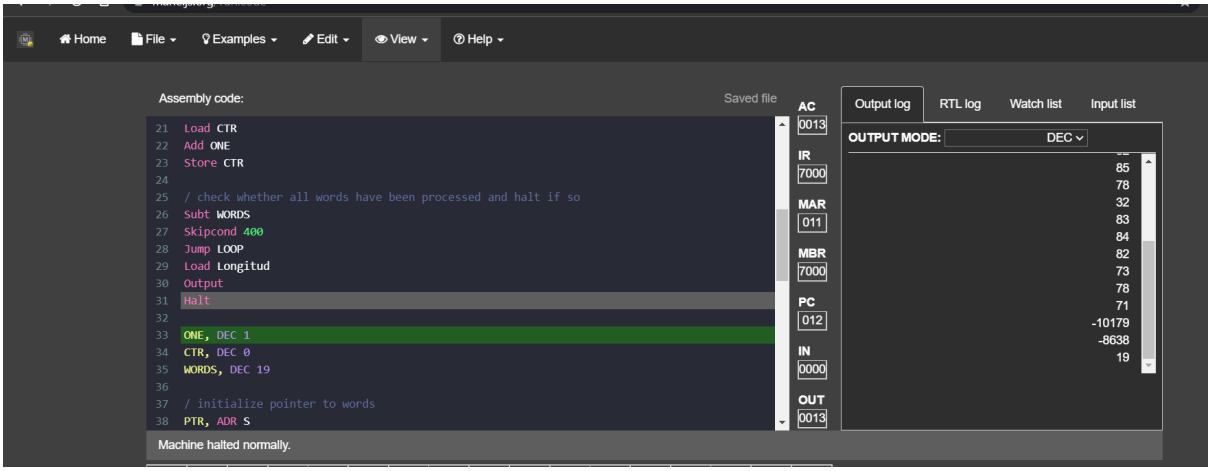
/ Smiley (outside of the Basic Multilingual Plane)
HEX D83D
HEX DE42

/ Byte Order Mark, big-endian (ignored)
HEX FEFF

/ Byte Order Mark, little-endian (ignored)
HEX FFFE

Longitud, DEC 0


Home
File
Examples
Edit
View
Help
Assembly code:
Saved file
21 Load CTR
22 Add ONE
23 Store CTR
24
25 / check whether all words have been processed and halt if so
26 Subt WORDS
27 Skipcond 400
28 Jump LOOP
29 Load Longitud
30 Output
31 Halt
32
33 ONE, DEC 1
34 CTR, DEC 0
35 WORDS, DEC 19
36
37 / initialize pointer to words
38 PTR, ADR S
AC 0013
IR 7000
MAR 011
MBR 7000
PC 012
IN 0000
OUT 0013
Output log
RTL log
Watch list
Input list
OUTPUT MODE: UNICODE (UTF-16BE)
ESTE ES UN STRING
Machine halted normally.
+0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
000 1015 D015 6000 102B 3012 202B 1015 3012 2015 1013 3012 2013 4014 8400 9000 102B
010 6000 7000 0001 0013 0013 0029 0045 0053 0054 0045 0020 0045 0053 0020 0055 004E
020 0020 0053 0054 0052 0049 004E 0047 D83D DE42 FEFF FFFE 0013 0000 0000 0000 0000
030 0000 0000 0000 0000 0000 0000 0000 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
Assemble
Step
Microstep
Step Back
Halted
Restart
Delay:
1 ms


Home
File
Examples
Edit
View
Help
Assembly code:
Saved file
21 Load CTR
22 Add ONE
23 Store CTR
24
25 / check whether all words have been processed and halt if so
26 Subt WORDS
27 Skipcond 400
28 Jump LOOP
29 Load Longitud
30 Output
31 Halt
32
33 ONE, DEC 1
34 CTR, DEC 0
35 WORDS, DEC 19
36
37 / initialize pointer to words
38 PTR, ADR S
AC 0013
IR 7000
MAR 011
MBR 7000
PC 012
IN 0000
OUT 0013
Output log
RTL log
Watch list
Input list
OUTPUT MODE: DEC
85
78
32
83
84
82
73
78
71
-10179
-8638
19
Machine halted normally.