

Project 5: Using Jasmin to run x86 Assembly Code (15 points)

What You Need for This Project

- Any computer, running any OS

Purpose

To practice writing and running basic x86 assembly code, using the Jasmin interpreter.

Install Java

Open a Web browser and go to

<http://java.com>

Click the "**Do I have Java?**" link. Follow the instructions on your screen to download and run a Java applet. If you don't have Java, download and install it.

Note: If you are using the Mac, you should use Safari, not Chrome.

Download Jasmin

In a Web browser, go to

<http://sourceforge.net/projects/tum-jasmin/files/>

On the "Looking for the latest version?" line, click the link to download the latest version of Jasmin. When I did it, it was Jasmin-1.5.8.jar.

If you don't want the drawing of a partially undressed woman on the splash screen, use this version:

[Download politically correct Jasmin without the cheesecake](#)

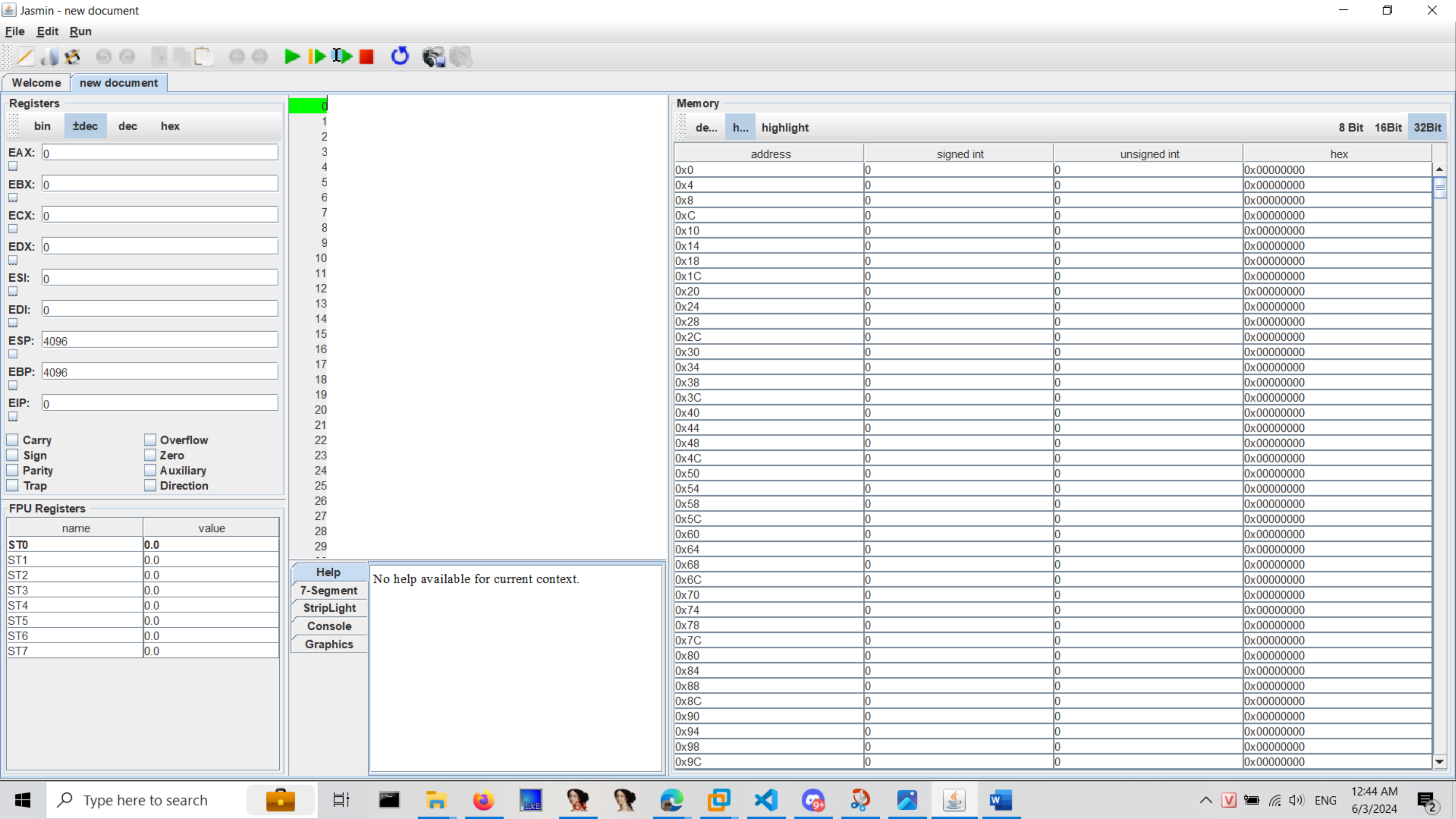
Understanding the Jasmin Window

Double-click the **Jasmin-1.5.8.jar** file you downloaded.

Jasmin launches, with a cringe-worthy pinup on it.

Click the "**New File**" button.

Look over the window, referring to the diagram below:



Find and examine these sections:

Registers

Data used during processing is stored in the registers **EAX**, **EBX**, **ECX**, and **EDX**.

The **ESP** (Extended Stack Pointer) contains the address of the top of the Stack.

The **EIP** (Extended Instruction Pointer) contains the address of the the next instruction to be processed.

Flags

These one-bit values that are used for branching. For example the JZ instruction will jump if the Zero flag is 1 (set), and the JNZ instruction will jump if the Zero flag is 0 (cleared).

Code

This is where you type in commands, such as `mov eax,4`

Help

Help messages appear here.

Memory

This processor has $0x1000 = 4096$ bytes of RAM, which is not enough to run complete modern programs, but plenty for running little assembly programs for learning purposes.

With the Memory pane scrolled to the top, as shown in the image above, you see memory that the program will use to store data during processing.

Scroll this pane to the bottom to see the Stack, which starts at address `0x1000` and grows downward.

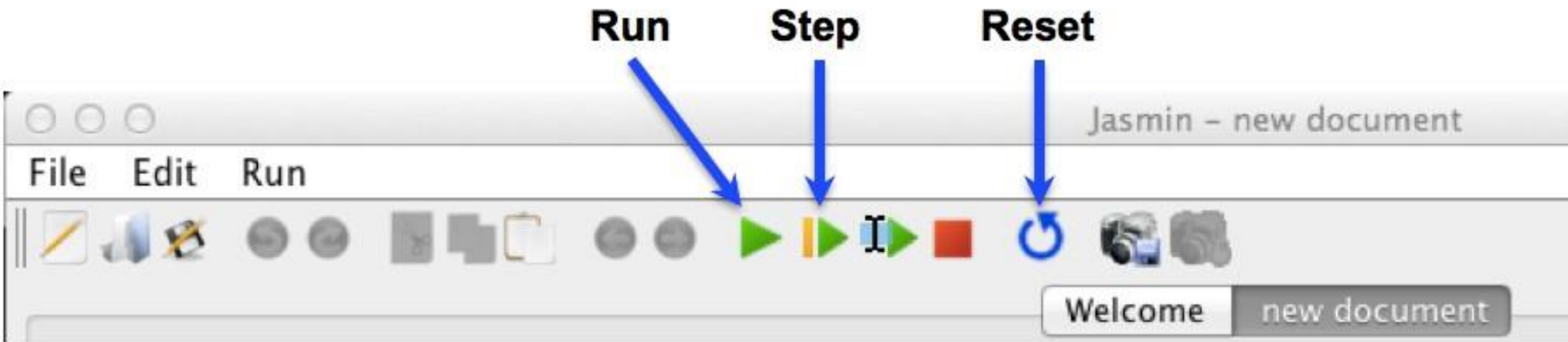
Using mov Instructions

In the Code section, type in these instructions.

```
mov eax, 4
mov ebx, 6
```

These instructions move the number 4 into `eax`, and the number 6 into `ebx`.

At the top of the Jasmin window, click the green **Run** button, as shown below.

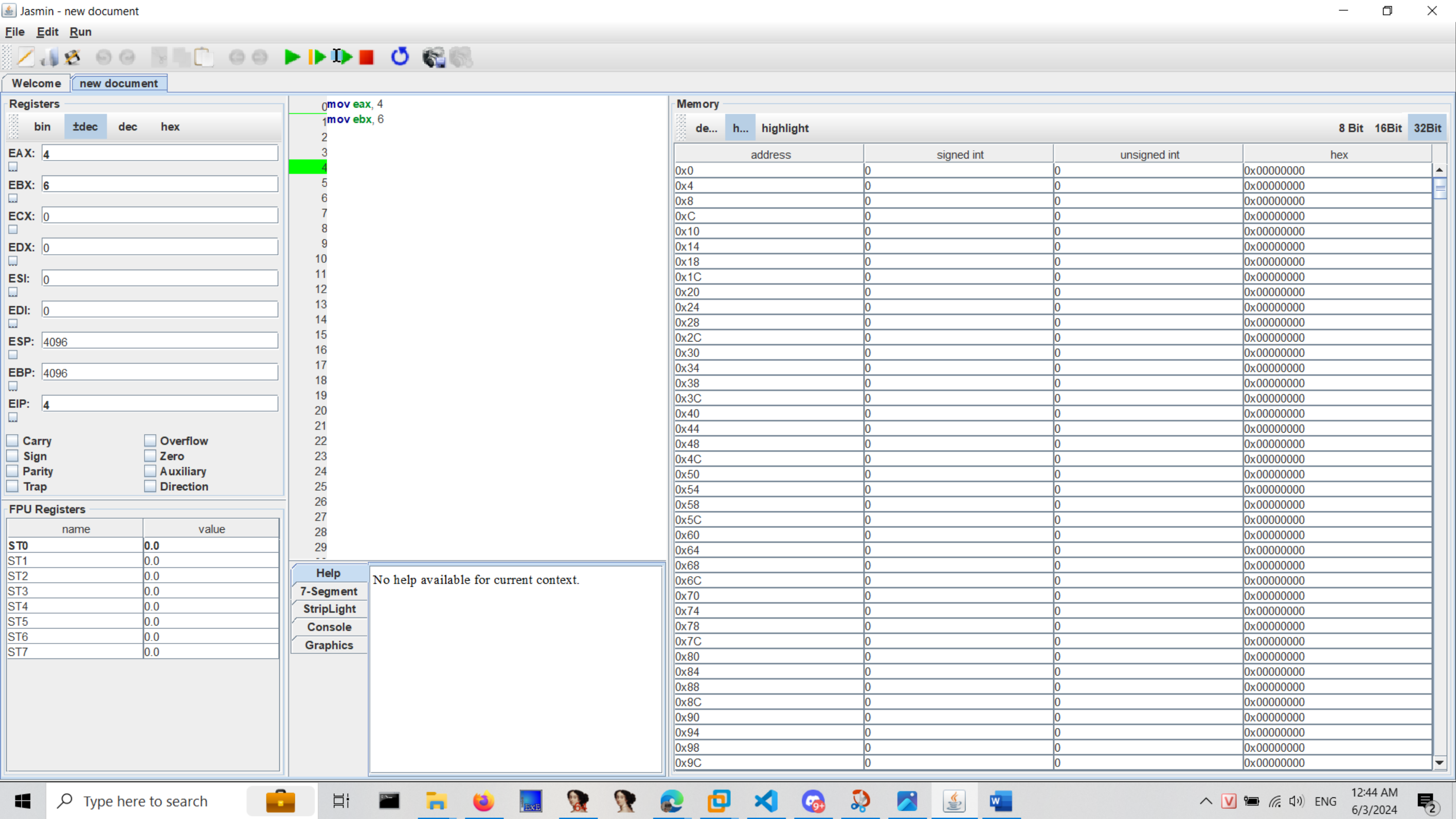


The program runs. When it stops, notice these things, as shown below:

- EAX contains 4
- EBX contains 6
- EIP contains 2, because instructions 0 and 1 have been executed
- In the Code area, instruction 2 is highlighted in green, indicating that it is the next instruction to be processed.

Troubleshooting

If you make an error in an instruction, the program will stop prematurely. Fix the instruction, and click the Reset button. Then you can run it again.



Storing Results in Memory

Add more lines to your Code section to make your program look like this:

```
mov eax, 4
mov ebx, 6
mov [eax], ebx
mov ecx, eax
add ecx, ebx
mov [eax+4], ecx
```

Here's what these instructions do:

- | | |
|-----------------------------|--|
| <code>mov eax, 4</code> | Move the value 4 into eax |
| <code>mov ebx, 6</code> | Move the value 6 into ebx |
| <code>mov [eax], ebx</code> | Move the value in ebx (which is 6) into the memory location pointed to by eax (memory location 4) |
| <code>mov ecx, eax</code> | Move the value in eax (which is 4) into ecx |
| <code>add ecx, ebx</code> | Add the value in ebx (which is 6) to the value in ecx (which is 4), and put the result into ecx (the result is 10) |

mov [eax+4], ecx

Move the value in ecx (which is 10) into the memory location four past the location pointed to by eax (memory location 8)

Run the program. When it completes, you should see these results, as shown below:

- EAX = 4
- EBX = 6
- ECX = 10
- Memory location 0x4 contains 6
- Memory location 0x8 contains 10

Jasmin - new document

File Edit Run

Welcomenew document

Registers

bin±decdechex

EAX: 4

EBX: 6

ECX: 6

EDX: 0

ESI: 0

EDI: 0

ESP: 4096

EBP: 4096

EIP: 7

Carry

Sign

Parity

Trap

Overflow

Zero

Auxiliary

Direction

FPU Registers

name	value
ST0	0.0
ST1	0.0
ST2	0.0
ST3	0.0
ST4	0.0
ST5	0.0
ST6	0.0
ST7	0.0

0mov eax, 4

1mov ebx, 6

2mov [eax], ebx

3mov ecx, eax

4add ecx, ebx

5mov [eax+4], ecx

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

--

Help

7-Segment

StripLight

Console

Graphics

memory location; (two memory operands cannot be used)

Usage: ADD DEST, SRC

Effects: DEST := DEST + SRC;

Flags to be set: The OF, SF, ZF, AF, CF, and PF flags are set according to the result.

Misc: No difference between signed and unsigned operands. OF and CF flags indicate carry in case of unsigned and signed values, SF indicates the sign in case of signed values.

Memory

de...h...highlight

8 Bit

16Bit

32

address	signed int	unsigned int	hex
0x0	0	0	0x00000000
0x4	0	0	0x00000000
0x8	6	6	0x00000006
0xC	0	0	0x00000000
0x10	0	0	0x00000000
0x14	0	0	0x00000000
0x18	0	0	0x00000000
0x1C	0	0	0x00000000
0x20	0	0	0x00000000
0x24	0	0	0x00000000
0x28	0	0	0x00000000
0x2C	0	0	0x00000000
0x30	0	0	0x00000000
0x34	0	0	0x00000000
0x38	0	0	0x00000000
0x3C	0	0	0x00000000
0x40	0	0	0x00000000
0x44	0	0	0x00000000
0x48	0	0	0x00000000
0x4C	0	0	0x00000000
0x50	0	0	0x00000000
0x54	0	0	0x00000000
0x58	0	0	0x00000000
0x5C	0	0	0x00000000
0x60	0	0	0x00000000
0x64	0	0	0x00000000
0x68	0	0	0x00000000
0x6C	0	0	0x00000000
0x70	0	0	0x00000000
0x74	0	0	0x00000000
0x78	0	0	0x00000000
0x7C	0	0	0x00000000
0x80	0	0	0x00000000
0x84	0	0	0x00000000
0x88	0	0	0x00000000
0x8C	0	0	0x00000000
0x90	0	0	0x00000000
0x94	0	0	0x00000000
0x98	0	0	0x00000000
0x9C	0	0	0x00000000

Windows taskbar

Type here to search

Taskbar icons

System tray

12:45 AM 6/3/2024

Saving a Screen Image

Make sure the five items listed above are visible.

Press the **PrintScrn** key to copy the whole desktop to the clipboard.

YOU MUST SUBMIT A FULL-SCREEN IMAGE FOR FULL CREDIT!

Paste the image into Paint.

Save the document with the filename "**YOUR NAME Proj 5a**", replacing "YOUR NAME" with your real name.

Using the Stack

In Jasmin, click **File, New**.

In the Code section, type in these instructions.

```
mov eax, 4
mov ebx, 6
push eax
push ebx
```

Before running the program, notice the ESP: it contains 4096, as shown below.

4096 is 0x1000 in hexadecimal--this is where the Stack ends.

Scroll down in the Memory pane to see the last values. As show above, the last location is at 0xFFC. This value is 32 bits long, so it contains four bytes, at locations 0xFFC, 0xFFD, 0xFFE, and 0xFFF. The ESP points to the next byte, 0x1000.

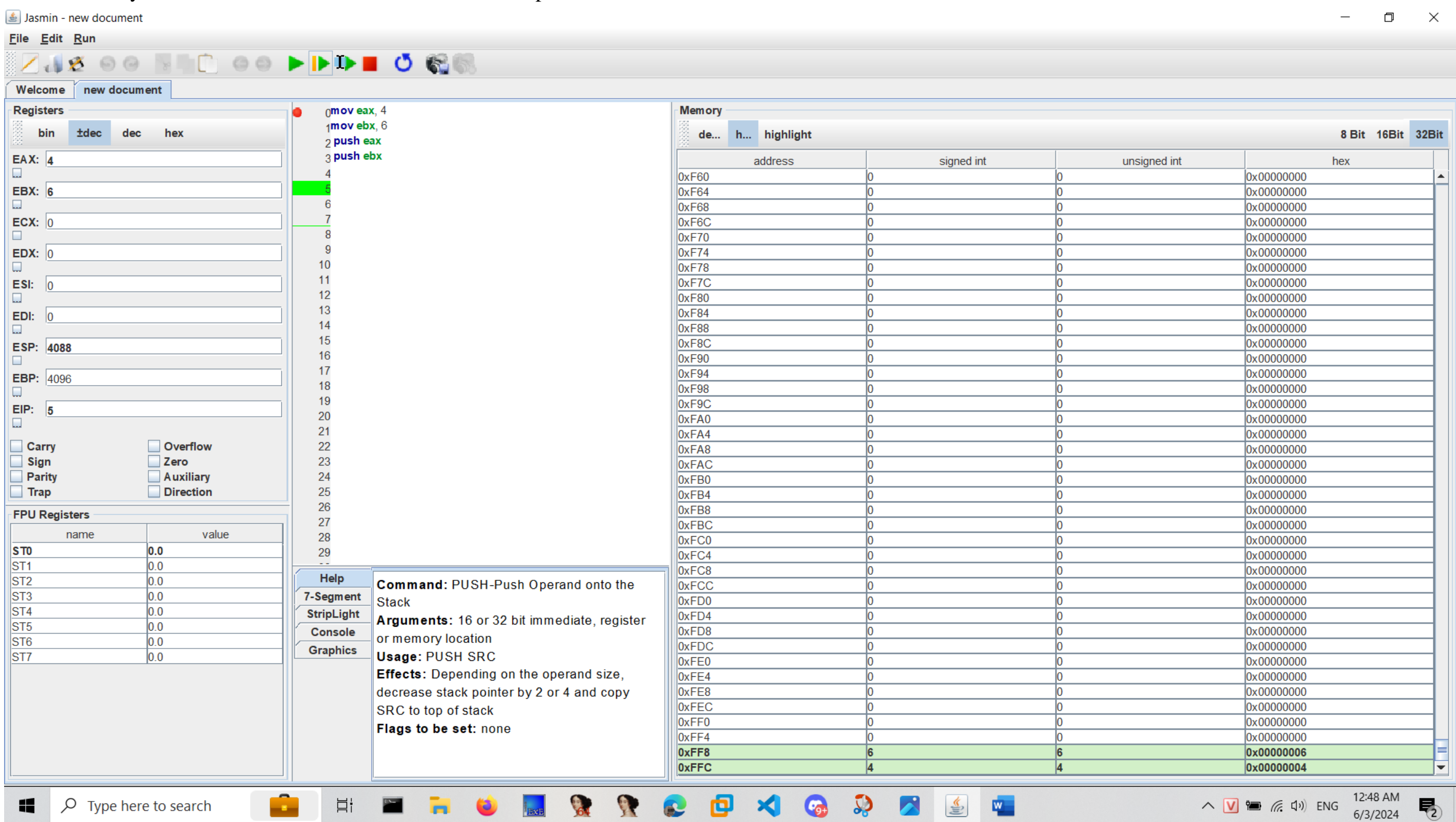
Understanding Push

At the top of the Jasmin window, click the green **Run** button.

These instructions move the number 4 into `eax`, and the number 6 into `ebx`. Then both values are pushed onto the stack.

Notice these things, as shown below:

- EAX contains 4
- EBX contains 6
- ESP contains 4088, which is 0xFF8, the new top of the stack.
- Memory location 0xFFC contains 4, the first value pushed onto the stack.
- Memory location 0xFF8 contains 6, the second value pushed onto the stack.



Understanding Pop

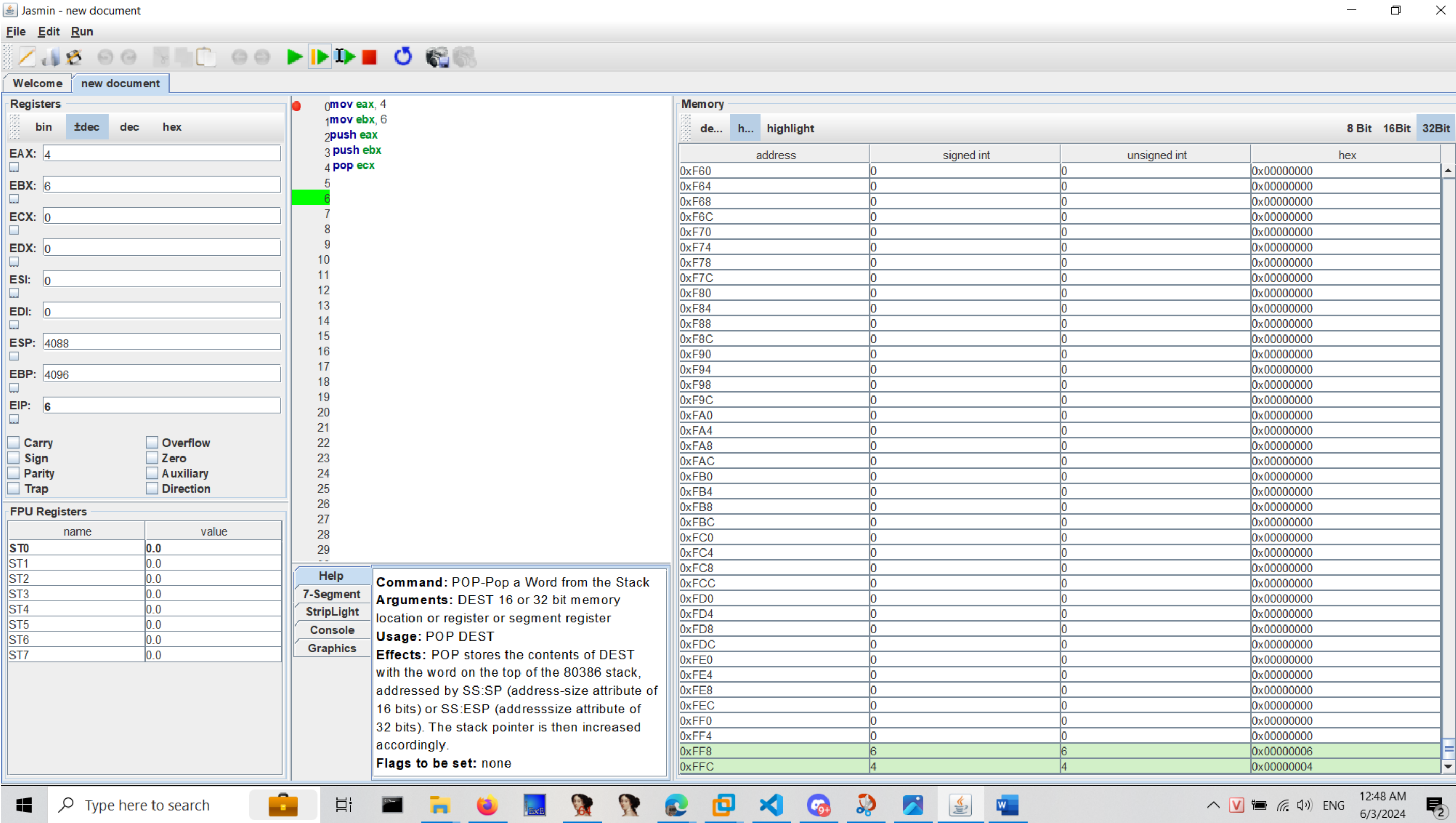
Add a pop instruction to your code, so it now looks like this:

```
mov eax, 4
mov ebx, 6
push eax
push ebx
pop ecx
```

Run the code.

Notice these things, as shown below:

- ECX contains 6, the value popped off the top of the stack.
- ESP contains 4092, which is 0xFFC, the new top of the stack.
- Memory location 0xFFC contains 4, the first value pushed onto the stack.
- Memory location 0xFF8 contains 6, which is now the top value on the stack.



Reversing a Sequence

In Jasmin, click **File, New**.

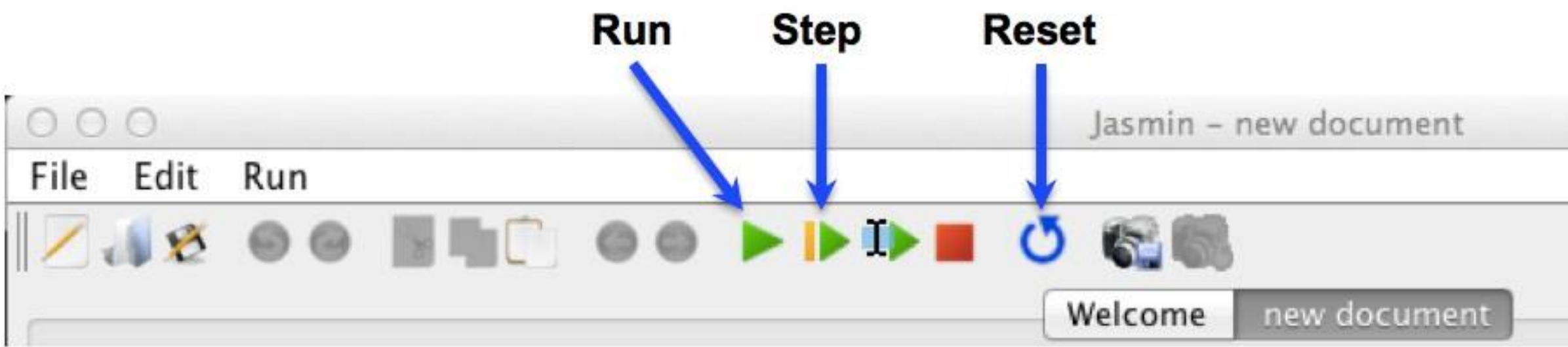
In the Code section, type in these instructions.

```
mov eax, 1
mov ebx, 2
mov ecx, 3
mov edx, 4
push eax
push ebx
push ecx
push edx
pop eax
pop ebx
pop ecx
pop edx
```

These instructions load values into the four registers, push them onto the stack in order, and pop them off the stack in order.

However, since the stack is a FILO (First In, Last Out) structure, this reverses the order of the values.

Push the **Step** four times to execute only the first four instructions, as shown below:



You see the values 1, 2, 3, and 4 loaded into the EAX, EBX, ECX, and EDX registers, as shown below.

Jasmin - new document

File Edit Run

Welcome new document

Registers

bin ±dec dec hex

EAX: 0

EBX: 0

ECX: 6

EDX: 4

ESI: 0

EDI: 0

ESP: 4096

EBP: 4096

EIP: 13

Carry Sign Parity Trap

Overflow Zero Auxiliary Direction

FPU Registers

name	value
ST0	0.0
ST1	0.0
ST2	0.0
ST3	0.0
ST4	0.0
ST5	0.0
ST6	0.0
ST7	0.0

```
0mov eax, 1
1mov ebx, 2
2mov ecx, 3
3mov edx, 4
4push eax
5push ebx
6push ecx
7push edx
8pop eax
9pop ebx
10pop ecx
11pop edx
12
13
14
```

Help

7-Segment

StripLight

Console

Graphics

Command: POP-Pop a Word from the Stack

Arguments: DEST 16 or 32 bit memory location or register or segment register

Usage: POP DEST

Effects: POP stores the contents of DEST with the word on the top of the 80386 stack, addressed by SS:SP (address-size attribute of 16 bits) or SS:ESP (addresssize attribute of 32 bits). The stack pointer is then increased accordingly.

Flags to be set: none

Memory

de... h... highlight

8 Bit 16Bit 32Bit

address	signed int	unsigned int	hex
0xF60	0	0	0x00000000
0xF64	0	0	0x00000000
0xF68	0	0	0x00000000
0xF6C	0	0	0x00000000
0xF70	0	0	0x00000000
0xF74	0	0	0x00000000
0xF78	0	0	0x00000000
0xF7C	0	0	0x00000000
0xF80	0	0	0x00000000
0xF84	0	0	0x00000000
0xF88	0	0	0x00000000
0xF8C	0	0	0x00000000
0xF90	0	0	0x00000000
0xF94	0	0	0x00000000
0xF98	0	0	0x00000000
0xF9C	0	0	0x00000000
0xFA0	0	0	0x00000000
0xFA4	0	0	0x00000000
0xFA8	0	0	0x00000000
0xFAC	0	0	0x00000000
0xFB0	0	0	0x00000000
0xFB4	0	0	0x00000000
0xFB8	0	0	0x00000000
0xFBC	0	0	0x00000000
0xFC0	0	0	0x00000000
0xFC4	0	0	0x00000000
0xFC8	0	0	0x00000000
0xFCC	0	0	0x00000000
0xFD0	0	0	0x00000000
0xFD4	0	0	0x00000000
0xFD8	0	0	0x00000000
0xFDC	0	0	0x00000000
0xFE0	0	0	0x00000000
0xFE4	0	0	0x00000000
0xFE8	0	0	0x00000000
0xFEC	0	0	0x00000000
0xFF0	0	0	0x00000000
0xFF4	0	0	0x00000000
0xFF8	6	6	0x00000006
0xFFC	4	4	0x00000004

Push the **Step** four more times to execute only the next four instructions.

You see the values 1, 2, 3, and 4 pushed onto the stack, as shown below.

Jasmin - new document

File Edit Run

Welcome new document

Registers

bin ±dec dec hex

EAX: 4

EBX: 3

ECX: 2

EDX: 1

ESI: 0

EDI: 0

ESP: 4096

EBP: 4096

EIP: 12

Carry Sign Parity Trap

Overflow Zero Auxiliary Direction

FPU Registers

name	value
ST0	0.0
ST1	0.0
ST2	0.0
ST3	0.0
ST4	0.0
ST5	0.0
ST6	0.0
ST7	0.0

```
0mov eax, 1
1mov ebx, 2
2mov ecx, 3
3mov edx, 4
4push eax
5push ebx
6push ecx
7push edx
8pop eax
9pop ebx
10pop ecx
11pop edx
12
13
14
```

Help

7-Segment

StripLight

Console

Graphics

Command: POP-Pop a Word from the Stack

Arguments: DEST 16 or 32 bit memory location or register or segment register

Usage: POP DEST

Effects: POP stores the contents of DEST with the word on the top of the 80386 stack, addressed by SS:SP (address-size attribute of 16 bits) or SS:ESP (addresssize attribute of 32 bits). The stack pointer is then increased accordingly.

Flags to be set: none

Memory

de... h... highlight

8 Bit 16Bit 32Bit

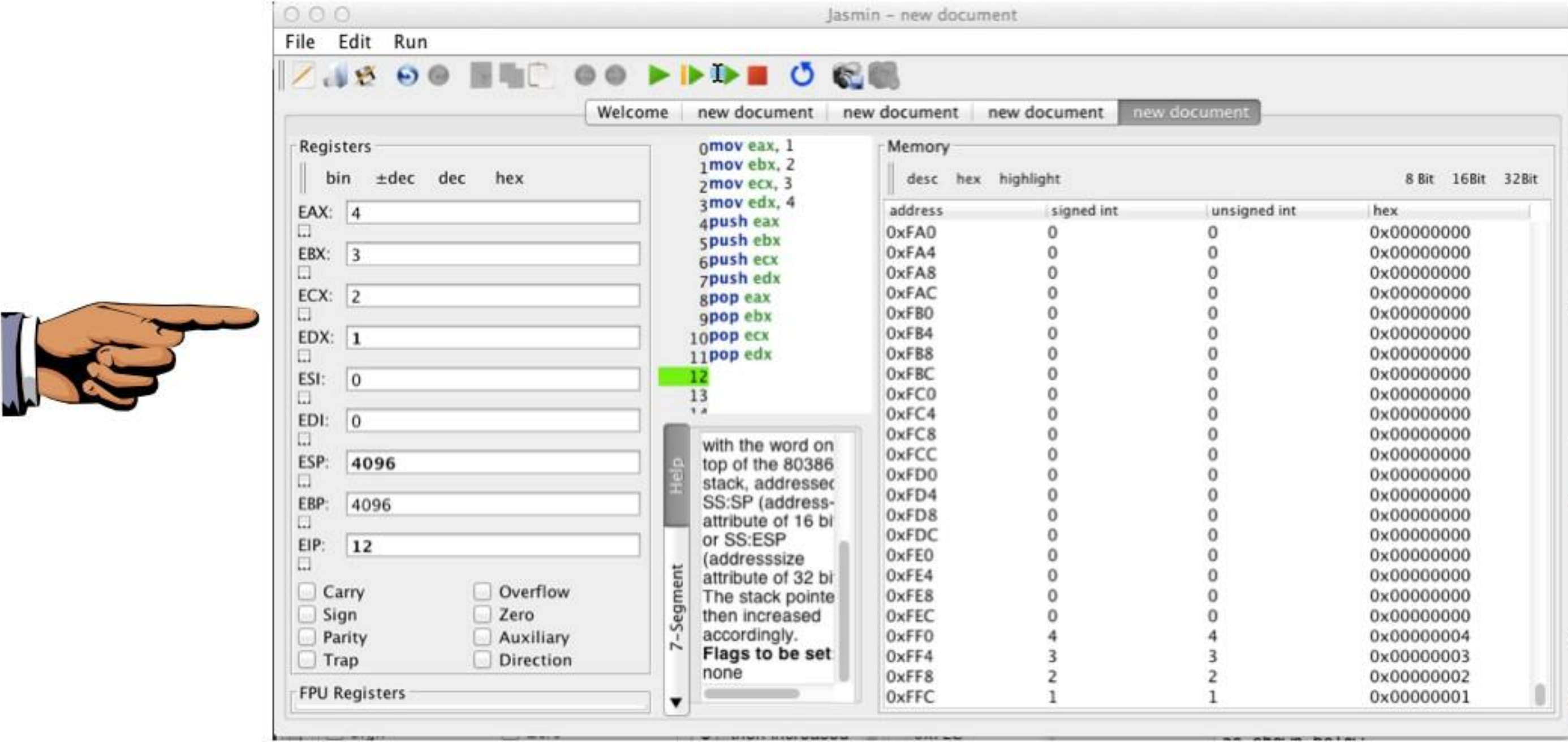
address	signed int	unsigned int	hex
0xF60	0	0	0x00000000
0xF64	0	0	0x00000000
0xF68	0	0	0x00000000
0xF6C	0	0	0x00000000
0xF70	0	0	0x00000000
0xF74	0	0	0x00000000
0xF78	0	0	0x00000000
0xF7C	0	0	0x00000000
0xF80	0	0	0x00000000
0xF84	0	0	0x00000000
0xF88	0	0	0x00000000
0xF8C	0	0	0x00000000
0xF90	0	0	0x00000000
0xF94	0	0	0x00000000
0xF98	0	0	0x00000000
0xF9C	0	0	0x00000000
0xFA0	0	0	0x00000000
0xFA4	0	0	0x00000000
0xFA8	0	0	0x00000000
0xFAC	0	0	0x00000000
0xFB0	0	0	0x00000000
0xFB4	0	0	0x00000000
0xFB8	0	0	0x00000000
0xFBC	0	0	0x00000000
0xFC0	0	0	0x00000000
0xFC4	0	0	0x00000000
0xFC8	0	0	0x00000000
0xFCC	0	0	0x00000000
0xFD0	0	0	0x00000000
0xFD4	0	0	0x00000000
0xFD8	0	0	0x00000000
0xFDC	0	0	0x00000000
0xFE0	0	0	0x00000000
0xFE4	0	0	0x00000000
0xFE8	0	0	0x00000000
0xFEC	0	0	0x00000000
0xFF0	4	4	0x00000004
0xFF4	3	3	0x00000003
0xFF8	2	2	0x00000002
0xFFC	1	1	0x00000001

Push the **Step** four more times to execute the remaining four instructions.

Now the registers contain these values:

- EAX = 4
- EBX = 3
- ECX = 2
- EDX = 1

as shown below.



Saving a Screen Image

Make sure the four items listed above are visible.

Press the **PrintScrn** key to copy the whole desktop to the clipboard.

YOU MUST SUBMIT A FULL-SCREEN IMAGE FOR FULL CREDIT!

Paste the image into Paint.

Save the document with the filename "**YOUR NAME Proj 5b**", replacing "YOUR NAME" with your real name.

Turning in your Project

Email the images to cnit.126sam@gmail.com with the subject line: **Proj 5 from YOUR NAME**

Source

http://www.lrr.in.tum.de/~jasmin/tutorials_basic.html#moving

Last modified 2-16-15