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 cheescake

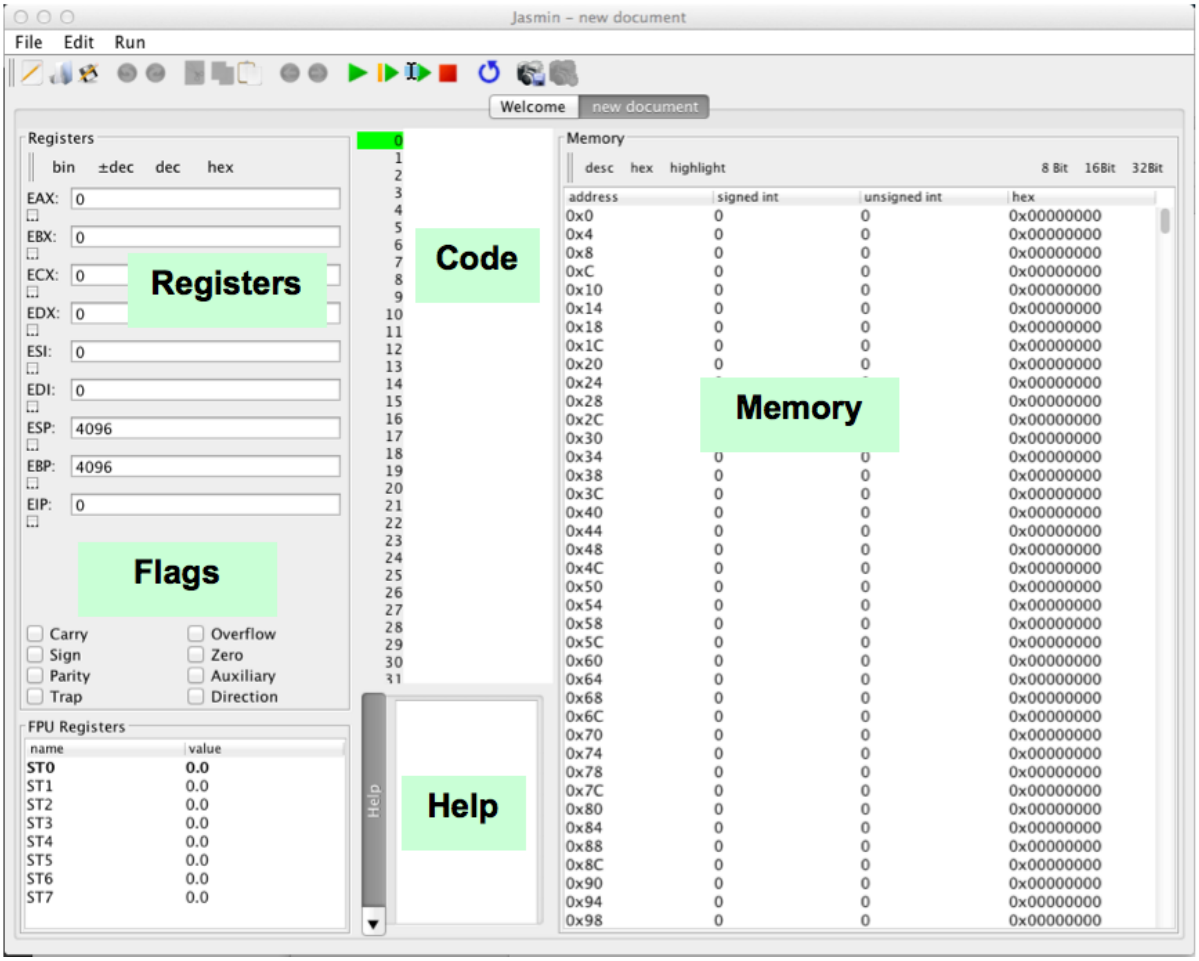
Understanding the Jasmin Window

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.

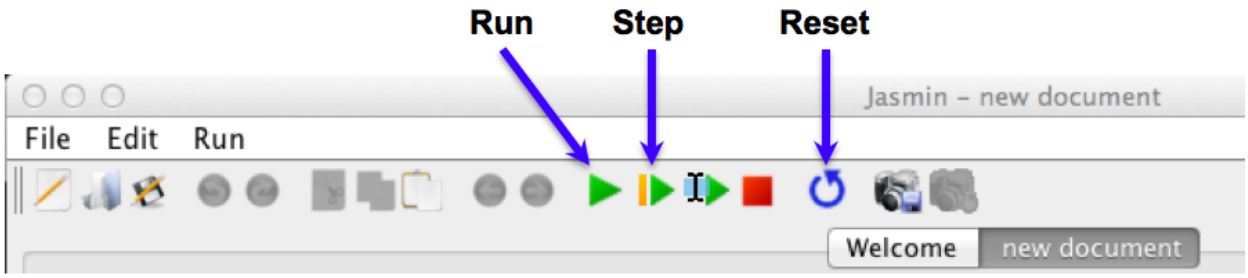
Using mov Instructions

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.

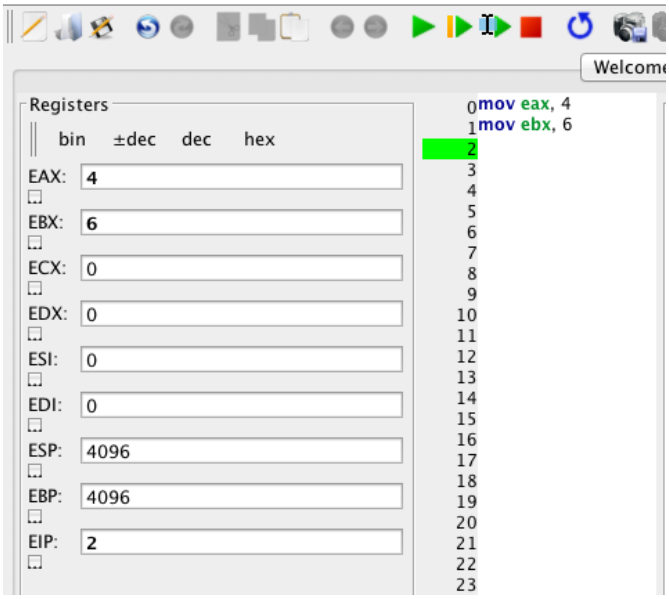
At the top of the Jasmin window, click the green **Run** button, as shown below.



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:

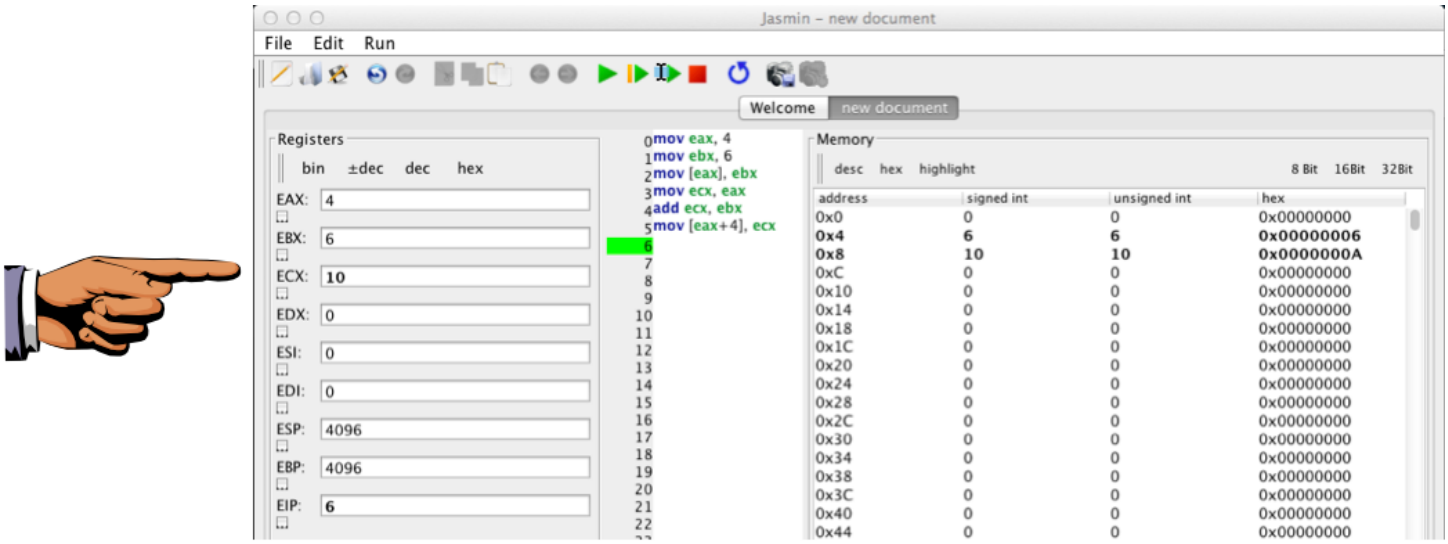
```
mov eax, 4
mov ebx, 6
mov [eax], ebx
mov ecx, eax
add ecx, ebx
mov [eax+4], ecx
```

Here's what these instructions do:

`mov eax, 4`	Move the value 4 into eax
`mov ebx, 6`	Move the value 6 into ebx
`mov [eax], ebx`	Move the value in ebx (which is 6) into the memory location pointed to by eax (memory location 4)
`mov ecx, eax`	Move the value in eax (which is 4) into ecx
`add ecx, ebx`	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



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

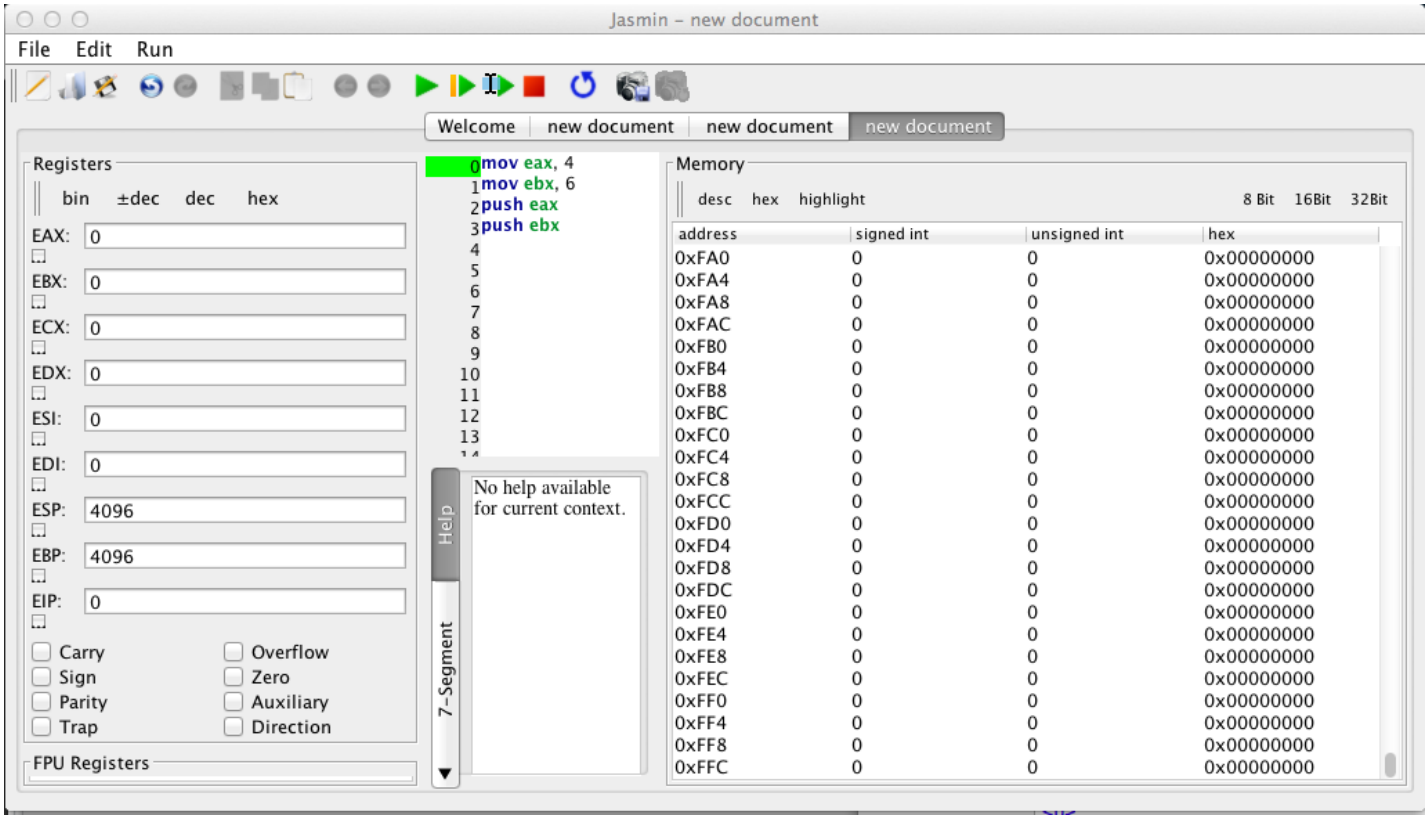
In Jasmin, click **File**, **New**.

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.

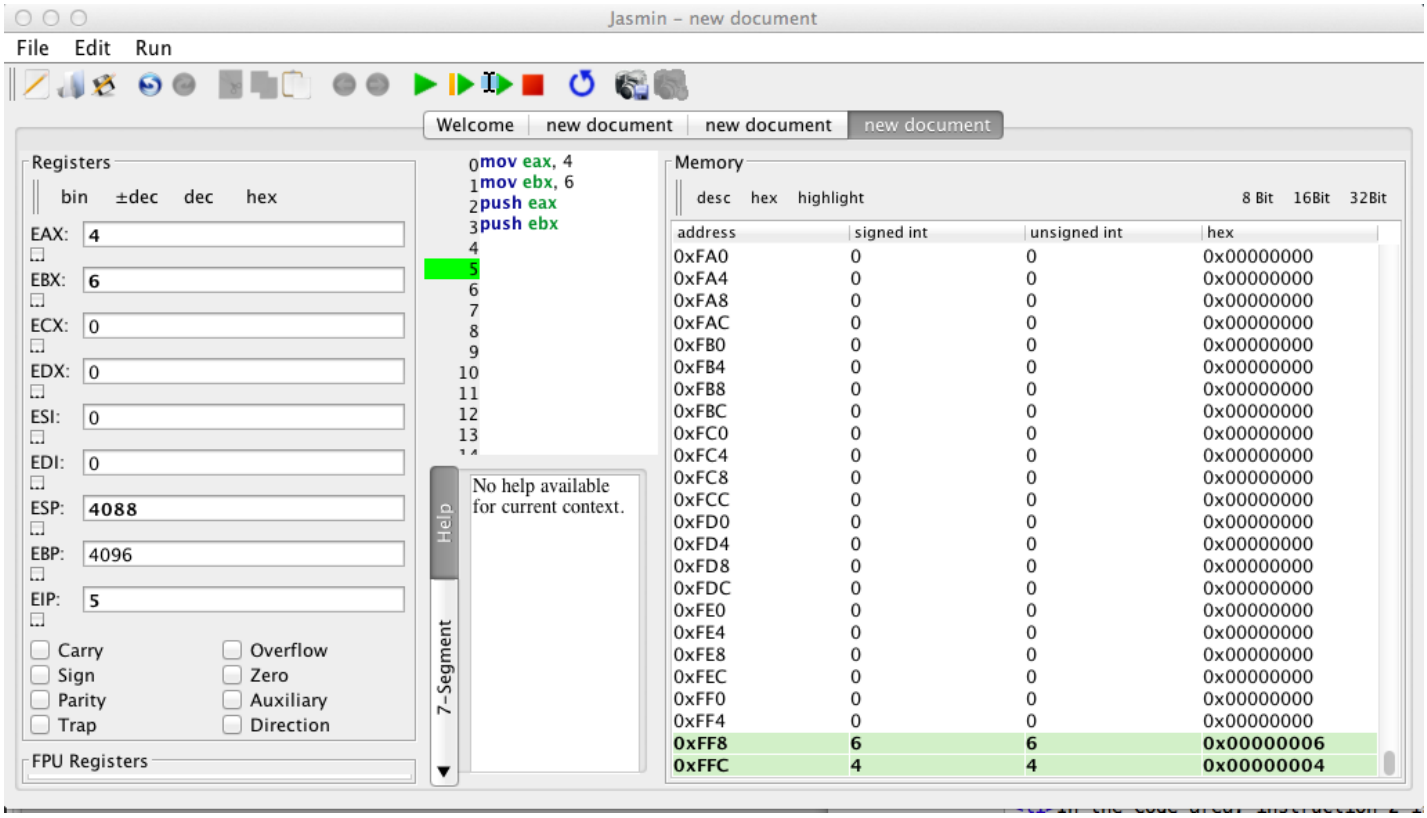
Understanding Push

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

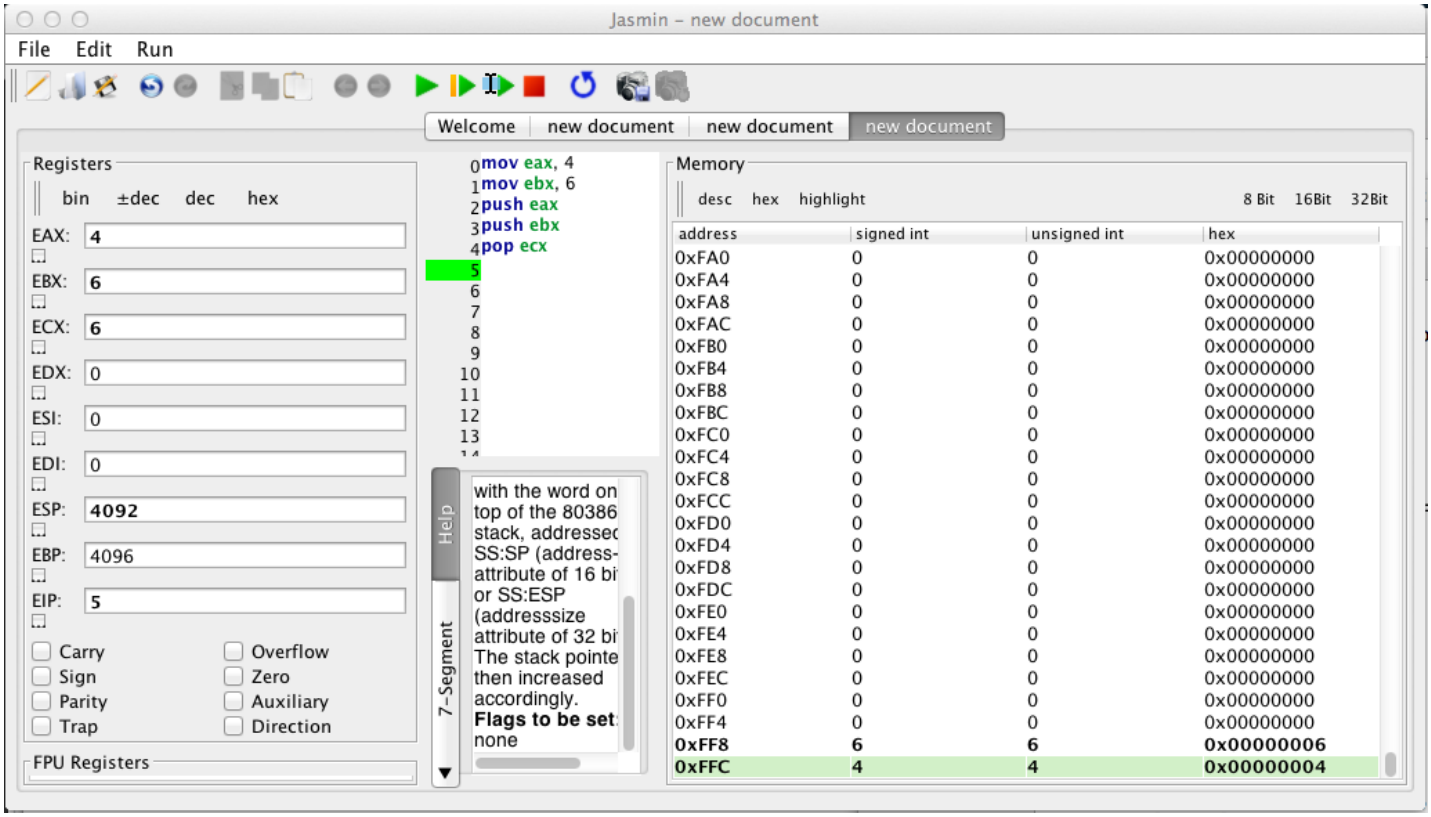
Add a pop instruction to your code, so it now looks like this:

```
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**.

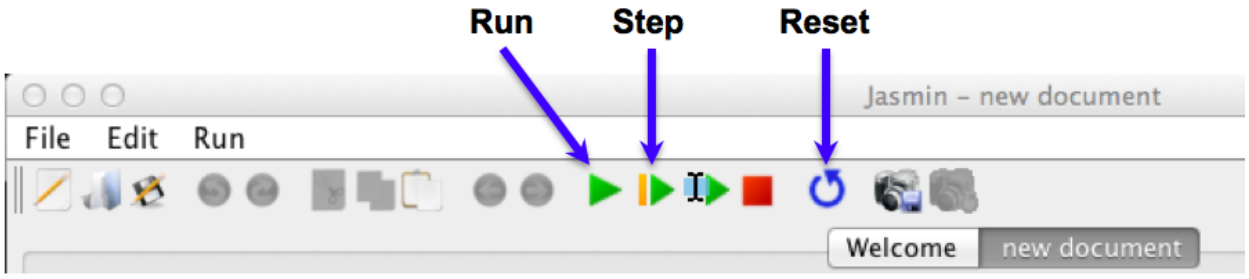
In the Code section, type in these instructions.

```
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
```

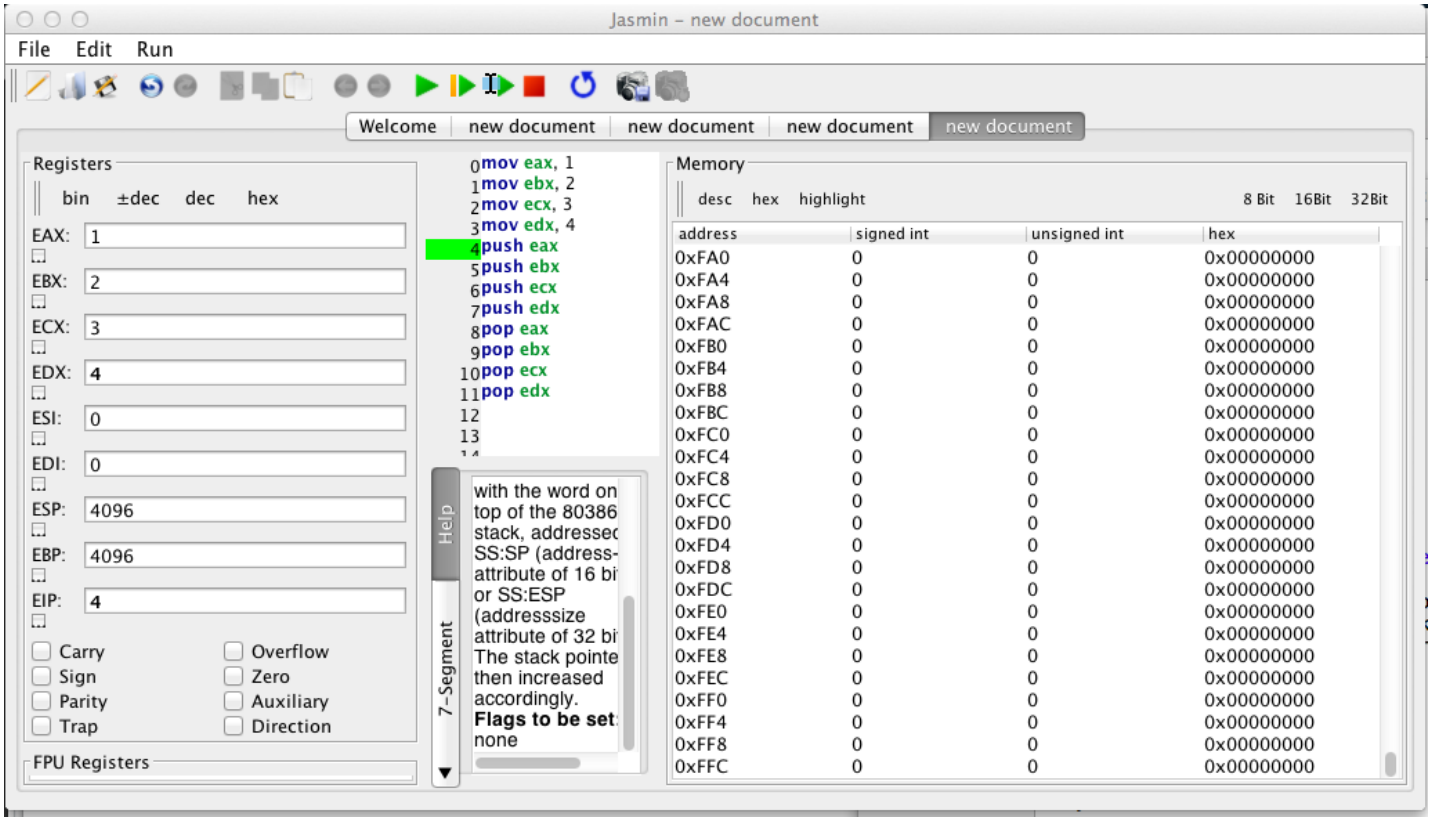
These instructions load values into the four registers, push them onto the stack in order, and pop them off the stack in order.

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:

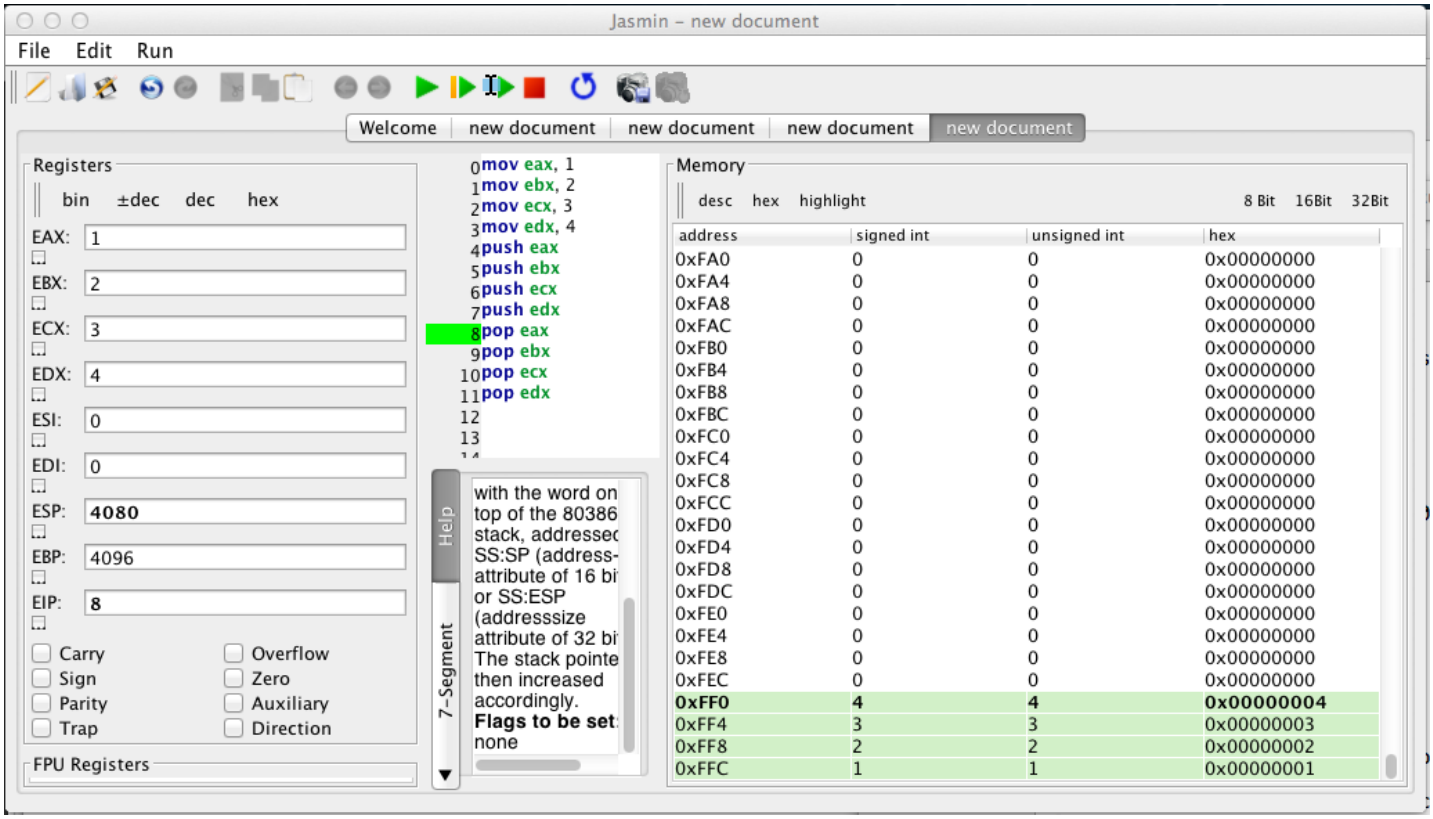


You see the values 1, 2, 3, and 4 loaded into the EAX, EBX, ECX, and EDX registers, as shown below.



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.

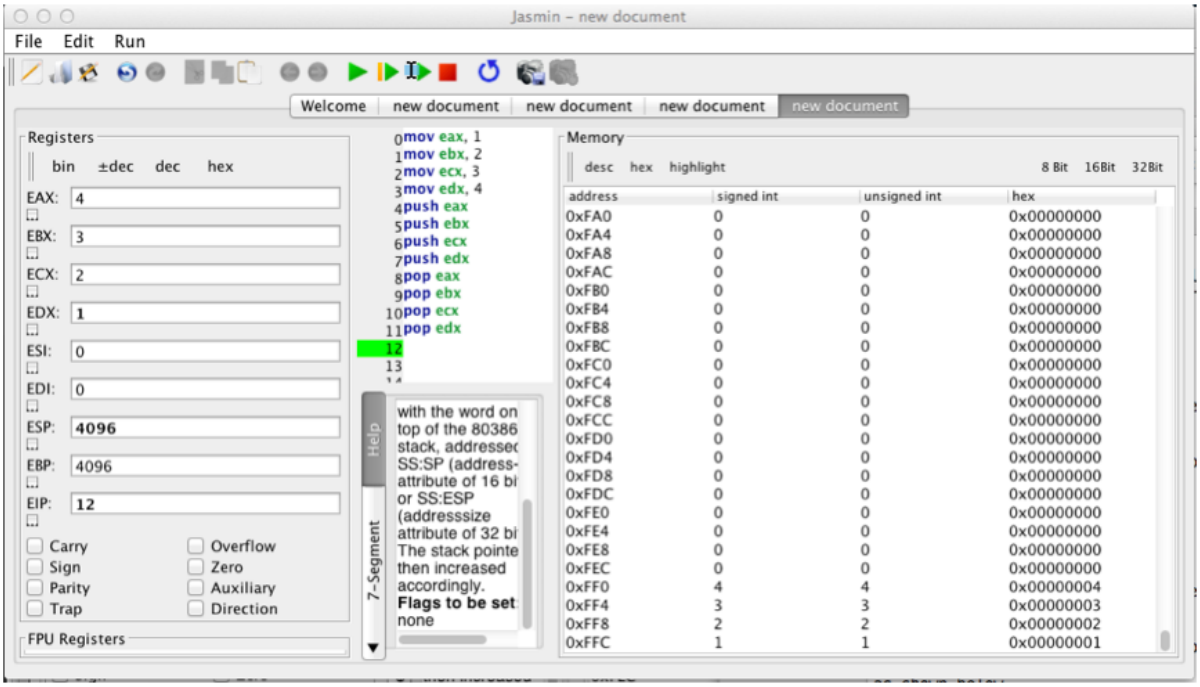


Push the **Step** four more times to execute the remaining four instructions.

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