LAB 1: Introduction to Frances-A

Objectives:

a) Become familiar with the Frances-A tool.
b) Understand the components Frances-A.
c) Introduce some syntax of a low level language.
d) Understand memory allocation and register use.

1) Background: Frances-A is a tool designed to help teach computer architecture and assembly language. This tool graphically shows the relationships between high-level programming languages and machine languages. It also shows the machine state and allows the user to step through execution one instruction at a time. Color coding is used to show the relationship between high-level code constructs and assembly code (see legend at the bottom left of the tool's website). Additionally, color coding shows which components of the machine state have been accessed and modified by the latest instruction.
    i) Red- indicates changes to the content of a memory location;
    ii) Yellow – indicates memory access;
    iii) Green – indicates locations that change with every step.
2) Go to the Frances-A site at http://www.cs.iastate.edu/~sapha/tools/frances/frances-a/ and click on the link for the Web Tool. For most of the exercises you will use the C language for the high-level language. This is Frances-A's default language. Use it unless otherwise directed.
    a) The default is a program that does nothing. Click "Compile"
        i) The assembly code for the program is generated and is shown in the box to the right of the source code. Additionally, the registers and stack are shown to the far right of the screen
    b) Click "Execute next instruction". Notice that the first instruction in the assembly is now in the "Last Instruction" box, highlighted in green, above the registers and the "Next Instruction" box contains the second instruction. The "Last Instruction" was just executed and the "Next Instruction" is the next one to be executed.
    c) To the left of the "Execute next instruction" button is the "Last instruction" button. This button allows you to go backwards in the execution and re-execute the instruction in the "Last Instruction" box above the registers.
    d) Finally, the "Reset" button located to the left of the "Last Instruction" button allows you to enter new source code to be complied.
3) x86 AT&T Assembly Language. Just as there are various high-level languages there are various low-level languages. Examples of high-level languages are C, Java, C++ and COBOL. There are also various low-level languages such as AT&T and Intel. Here we will discuss the x86 AT&T assembly language.
    a) The format of a basic data movement instructions in the AT&T format is

        mnemonic *source, destination*

        Where *mnemonic* is the operation and the *source* and *destination* are the locations of the data being moved.
    b) Registers are prefixed by the % sign. For example, %esp, %ebx, %edi. Values are prefixed by a '$' sign. For example, $29 is the decimal value 29, $0x5 and $0xc are the hexadecimal values 5 and c (decimal 12), respectively. So the instruction,

mov $0x10, %eax moves the value 0x10 (decimal 16) into the eax register.

By placing a value in front of a register, it adds the value to the value in the register. As an example consider the first assembly instruction of the program:

lea 0x4(%esp), %ecx

lea is the command load effective address, which taking the address stored in the register *esp* adding 4 to it and loading it into the register *ecx*.

4) Exercises:
   a) Type the following code into Frances-A and answer the questions.

```
int main(){
    int x=5;
}
```

   i) What assembly code is the same as the default program?
   ii) What assembly code is different?
   iii) The assembly code from the default program prior to the new code is setting up the processor (registers) to process this program. That which appears after is restoring the processor (registers) to their previous state so that the computer may resume the process running prior to your program being called. Step through the program. Describe what is occurring in each of the new assembly lines?

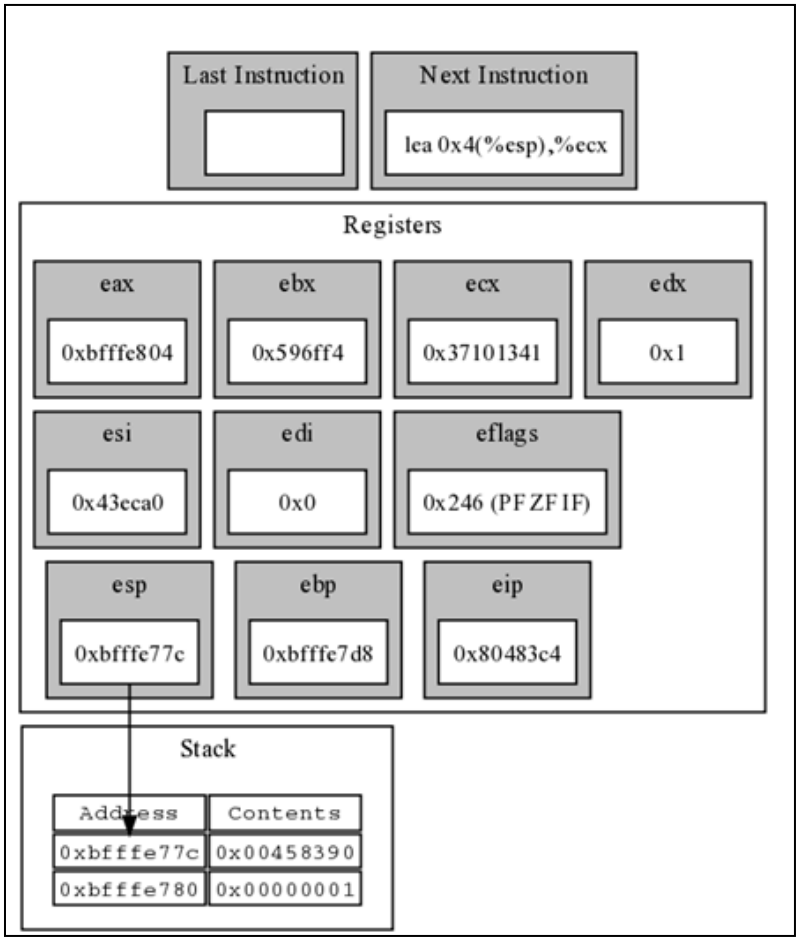

Figure 1

b) In Figure 1 are the registers as given in Frances-A. Put them in the following categories (some may go into multiple categories): General Purpose, Data, Address, Condition Codes.

c) Given the register values in Figure 1
   i) Near which address in memory would one find the assembly code for the program being executed?
   ii) After the next instruction executes which register(s) will change value?
   iii) After the next instruction executes what will be the value of the changed register(s)?

d) Given Figure 2
   i) At which memory location is the variable x stored? In hexadecimal? Relative to ebp?
   ii) At which assembly line is the value 2 put into this memory location?
   iii) Write the above C program in Frances-A. After execution of step 13 what are the values of eax and the stack? After step 14 what are these values?

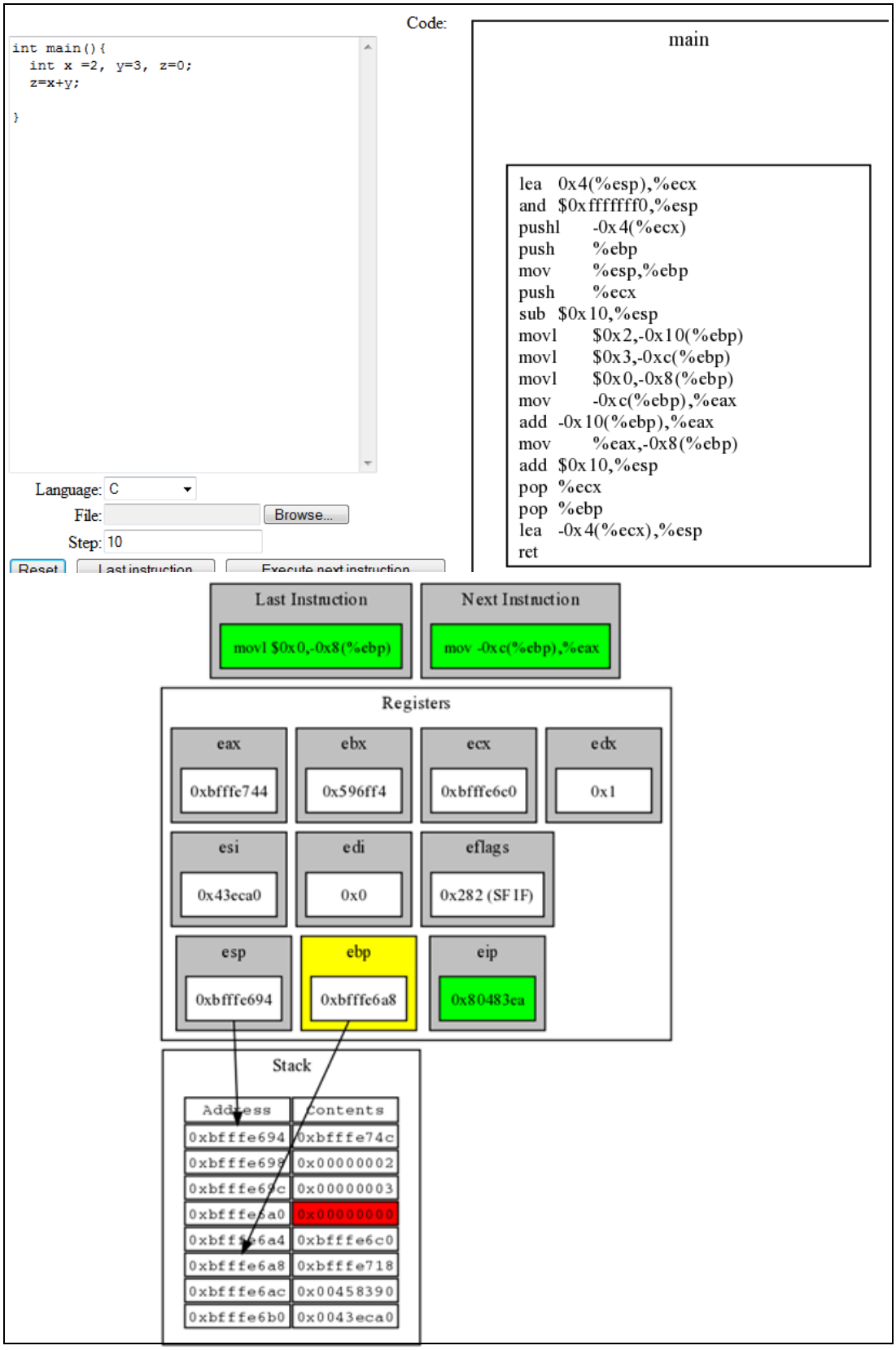
Code:
int main(){
int x =2, y=3, z=0;
z=x+y;
}
Language: C
File:
Browse...
Step: 10
Reset
Last instruction
Execute next instruction
main
lea 0x4(%esp),%ecx
and $0xfffffff0,%esp
pushl -0x4(%ecx)
push %ebp
mov %esp,%ebp
push %ecx
sub $0x10,%esp
movl $0x2,-0x10(%ebp)
movl $0x3,-0xc(%ebp)
movl $0x0,-0x8(%ebp)
mov -0xc(%ebp),%eax
add -0x10(%ebp),%eax
mov %eax,-0x8(%ebp)
add $0x10,%esp
pop %ecx
pop %ebp
lea -0x4(%ecx),%esp
ret
Last Instruction
movl $0x0,-0x8(%ebp)
Next Instruction
mov -0xc(%ebp),%eax
Registers
eax
0xbfffe744
ebx
0x596ff4
ecx
0xbfffe6c0
edx
0x1
esi
0x43eca0
edi
0x0
eflags
0x282 (SF IF)
esp
0xbfffe694
ebp
0xbfffe6a8
eip
0x80483ca
Stack
Address
Contents
0xbfffe694
0xbfffe74c
0xbfffe698
0x00000002
0xbfffe69c
0x00000003
0xbfffe6a0
0x00000000
0xbfffe6a4
0xbfffe6c0
0xbfffe6a8
0xbfffe718
0xbfffe6ac
0x00458390
0xbfffe6b0
0x0043eca0

Figure 2