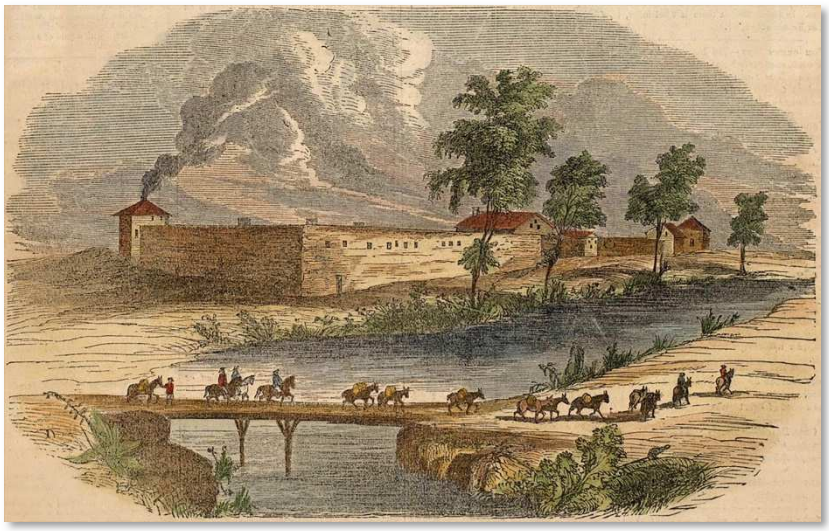
California State University, Sacramento
College of Engineering and Computer Science

Computer Science 35: Introduction to Computer Architecture

Spring 2022 – Lab 3 – *Sutter's Fort*

Overview



The year is 1847. After months of traveling over the vast wilderness of North America, you recently arrived in Sacramento, California. The long trail was worth it, the beauty of California has taken your breath away and your future looks both adventurous and bright.

Alas, you need to get a job! But, fortunately, Sutter's Fort is hiring.

This fort was created in 1839 by John Sutter in his hopes of creating a new civilization. Thanks to his efforts, the Fort has become a destination for commerce, trade, and the meeting place of adventurers from all over the world.

Your Task

You are going to create a program that does one of the things computers were designed to do – math! Though you might not know yet, but most of your life will be devoted to balancing your books. Yes, it sounds sad, but you are going to have to keep track of how much money you have. Your program will help you with your checks and bills.

For this program, you are going to input how much the fort brings in each month. Then, you are going to input, and subtract, all their debt-related expenses. Your program will then tell John Sutter how the Fort is doing.

Cash Flow = Income – Expenses

You are going to write a program that inputs the Fort's income, 2 expenses, and then display the cash flow. Basically, you are going to input 3 values and store them in memory (using direct storage). You must think of a solution on your own.

Sample Output

Your program does not have to look exactly like this, but this is an example of a working solution. The user's input is printed in **blue**. The data outputted from your calculations is printed in **green**. You don't have to make the text that color in your program.

```
Sutter's Fort Finance Calculator

How much did the Fort collect?
1800

How much was spent on employee salaries?
525

How much was spent on repairs?
150

Mr. Sutter, your cash flow is $1125
```

Necessary Instructions

Adding and Subtracting

For this lab, you *may* need to use two additional Intel x64 instructions: Add and Sub. These will allow you to add up the results of the inputted values.

```
add target, value
```
Java equivalent: `target += value;`

```
sub target, value
```
Java equivalent: `target -= value;`

The operands can have (almost) any combination of sources and targets. However, you cannot do *memory*, *memory*. So, do not use labels in both operands. You will have to **mov** a value into a register, first, and then use it.

Reading Integers

The CSC 35 Library has a subroutine called "ScanInt" that will read a value from the keyboard and store it into **rcx**. This is equivalent to the Java Scanner class method "nextInt".

```
call ScanInt
```
Java equivalent: `rcx = scanner.nextInt();`

Tips

- Work in your program in parts
- The **rcx** register is used by the library to input data and output results. Use direct storage store the values. In fact, you **must** use direct storage for credit.
- Pay close attention to which operand is changed with the SUB and ADD instructions.
- Intel x64 does **not** allow both operands to be address (labels).

Assemble, Link, and Execute

Type the following to assemble your program. **Warning:** If you list your **.asm** file after the **-o**, it will be destroyed.

```
as -o lab3.o lab3.asm
```

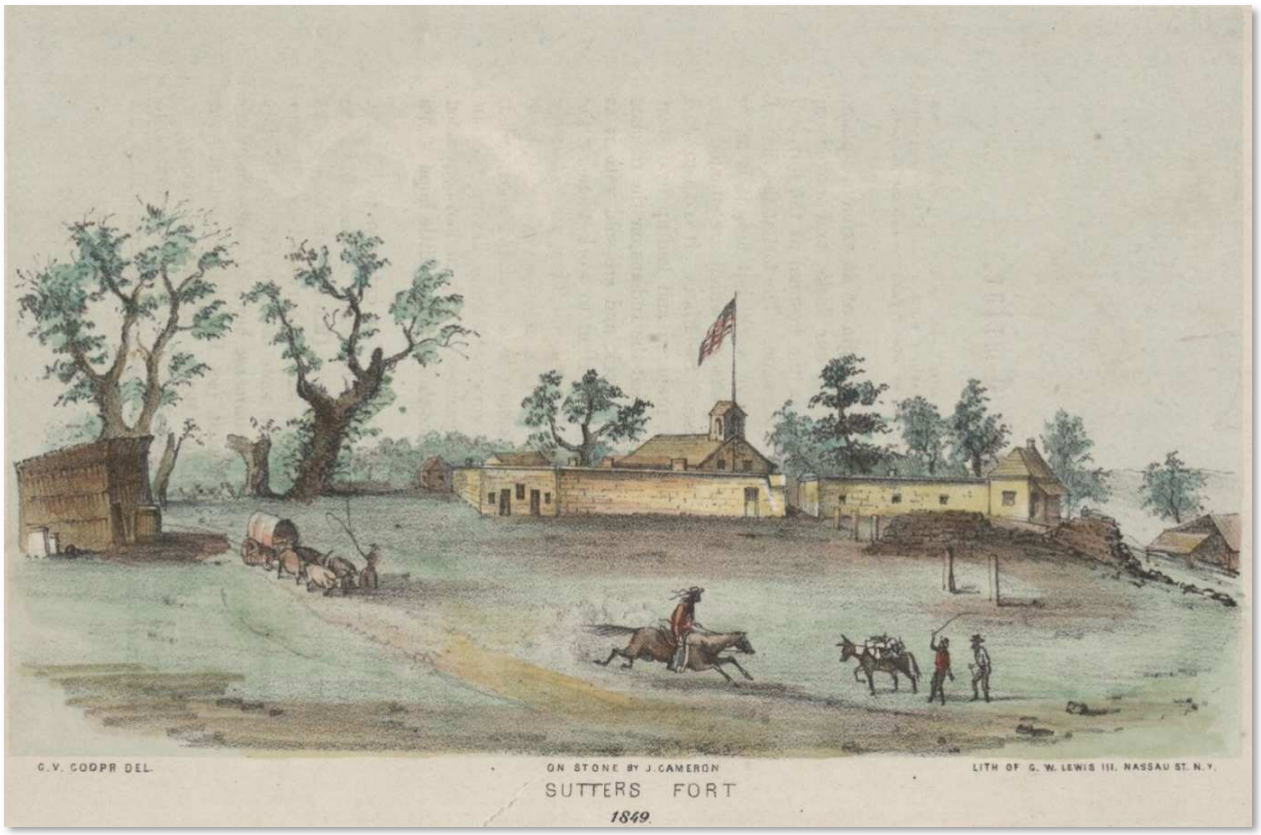
If you have any typos in your program, the assembler will list the errors. If that happens, open your existing program (type **nano lab3.asm**) and fix them.

To link your new object to the CSC 35 library, type the following. Don't forget to type **a.out** after **-o**. The first character is a lowercase L; not a 1.

```
ld -o a.out lab3.o csc35.o
```

If you have any mistakes with your labels, you will receive an error. Often, this is the result of a simple typo. You can execute your new program by simply typing its name and pressing the Enter Key.

```
./a.out
```



Requirements

- You must think of a solution on your own.
- Any lab not using direct storage will receive a zero.
- Any lab containing conditional logic will receive a zero (we have not covered it yet).

The requirements are as follows:

1. Input how much is collected with a helpful prompt.
2. Input **two** expenses with helpful prompts.
3. Compute and display the cash flow with a helpful sentence.

Submitting Your Lab

To submit your lab, you must run Alpine by typing the following and, then, enter your username and password.

```
alpine
```

To submit your lab, send the assembly file (do not send the a.out or the object file to:

```
dcook@csus.edu
```

This activity may only be submitted in Intel Format.

Using AT&T format will result in a zero. Any work from a prior semester will receive a zero.

UNIX Commands

Editing

Action	Command	Notes
Edit File	**nano** *filename*	"Nano" is an easy to use text editor.
E-Mail	**alpine**	"Alpine" is text-based e-mail application. You will e-mail your assignments it.
Assemble File	**as -o** *object source*	Don't mix up the *objectfile* and *asmfile* fields. It will destroy your program!
Link File	**ld -o** *exe object(s)*	Link and create an executable file from one (or more) object files

Folder Navigation

Action	Command	Description
Change current folder	**cd** *foldername*	"Changes Directory"
Go to parent folder	**cd ..**	Think of it as the "back button".
Show current folder	**pwd**	Gives the current a file path
List files	**ls**	Lists the files in current directory.

File Organization

Action	Command	Description
Create folder	**mkdir** *foldername*	Folders are called directories in UNIX.
Copy file	**cp** *oldfile newfile*	Make a copy of an existing file
Move file	**mv** *filename foldername*	Moves a file to a destination folder
Rename file	**mv** *oldname newname*	Note: same command as "move".
Delete file	**rm** *filename*	Remove (delete) a file. There is **no** undo.