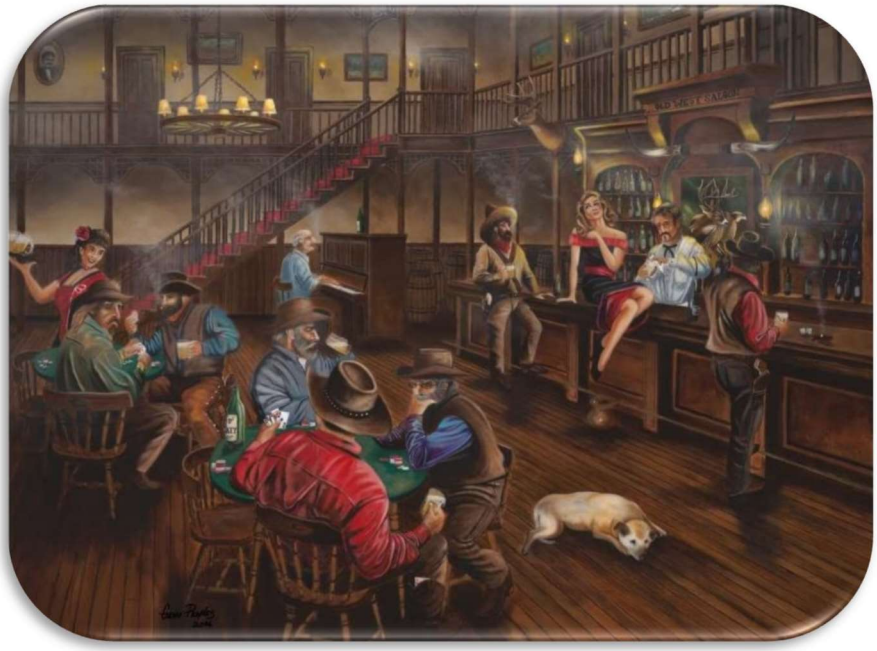
**California State University, Sacramento**
**College of Engineering and Computer Science**

**Computer Science 35: Introduction to Computer Architecture**

**Spring 2022 – Lab 7 – *Saloon***

## Overview



The blaze of the Gold Rush has set the mountains of the Sierra Nevada on fire. Hundreds of camps and towns have been built – nearly overnight – next to every river, stream or any other site yielding the precious gold.

Naturally, mining for gold is hard work. The hours are long, the work tedious, but the yield is quite amazing.

Your pockets are full of your spoils. Local metal-smiths eagerly take your gold-flakes and granules and smelts them into (much easier to carry) coins. Naturally, they take a percentage… but that's the beauty and strength of the free market.

Now that Sunday approaches, you and your friends are planning a trip into camp to enjoy a warm meal and some good conversation.

After arriving in camp, you party decides to have dinner. To your amazement, you can see that dozens of saloons have appeared overnight – as if they sprung out of the ground like dandelions.

## Your Task

Of course, your party is going to tally up quite a sizeable bill. Your program is going to help them, split the check. You are also going to use tables, rather than switch statements for your logic.

## Sample Run

The user's input is printed in **blue**. The data outputted from your calculations is printed in **red**.

```
Welcome to The Gold Bug Saloon!

1. Fresh elk steaks (323 cents)
2. Not-so-fresh steaks (152 cents)
3. Chicken sandwiches (215 cents)
4. Milk and mush bowls (97 cents)
5. Big-o-barrel-of-beer (737 cents)

What is your order?
1

Your party enjoyed:
1. Fresh elk steaks (323 cents)

How many people are splitting the bill?
3

Okay, witches and wizards, give 107 cents each.
```

A simple divide.

## Tips for the Data Section

Here a few tips on how to structure your program.

- Create strings for each menu item.
- Create a table of those addresses (to lookup the purchased item).
- Also create a table of costs.

The following contains an example of how to make a table of addresses (which are quads) and table of values (also quads). Please feel free to change your labels.

```
Steaks:
   .ascii "1. Fresh elk steaks (323 cents)\n\0"

OldSteaks:
   .ascii "2. Not-so-fresh steaks (152 cents)n\0"

   …

Items:
    .quad Steaks
    .quad OldSteaks
    …

Costs:
   .quad 323
   .quad 152
   …
```

## Tips for the Text Section

- **Work in your program in parts – incremental design!**
- The user is entering a 1-indexed number. This means the first item (for the customer) is 1. However, this is the index 0 in the table. **Subtract 1**.
- Remember to use the correct scale for quads!
- Since the `Items` table contains addresses (of string buffers), you should use MOV rather than LEA. You want the address, not the address of the address. So, use the following:

## Requirements

**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.**

You must think of a solution on your own. **You can come up with your own theme**. You don't have to use mine. The requirements are as follows:

1. Display a menu of items and costs. You must have (at least) five
2. Input their selection
3. Output the item they bought to the screen. You must use a table.
4. Input the number of guests
5. Calculate and output the correct split.

## 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. |