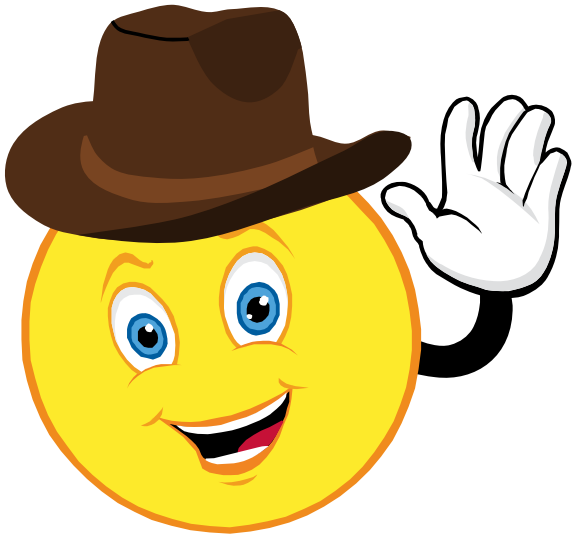
California State University, Sacramento
College of Engineering and Computer Science

Computer Science 35: Introduction to Computer Architecture

Spring 2022 – Lab 2 – *Howdy World*

Overview



It is a long-standing tradition in computer science that your first program displays “Hello World” to the screen. This dates back to the first message sent over ARPANET – the predecessor to the Internet.

This week, you will basically get your feet wet with assembly programming. On the next page of this handout, there is a very basic "Hello World" program. Essentially, your only task for this lab is to print the traditional "Hello World!" to the screen followed by some other information.

But let's make this lab a tad more "Sacramento"

In particular, let's place this lab within the pages of history The year is 1847. This is a couple years before the California Gold Rush, and an influx of settlers were coming to Sacramento over the long, and perilous, "California Trail". You have, after months upon months of grueling walking, have finally arrived in Sacramento. So, let's do a Howdy, World lab.

Part 1. Connecting to the Server

There are two options available to write your lab: MobaXTerm and PuTTY. Both are excellent applications which will allow you to create "secure shell" and run commands on a UNIX server. Of the two, MobaXTerm is a tad easier to use, and I recommend you try this one first.

MobaXTerm Instructions

1. Open MobaXTerm.
2. Click on the "Session" button on the top-left corner of the screen.
3. Click on SSH (it stands for Secure Shell).
4. Enter the following in the Remote Host box:

```
coding1.ecs.csus.edu
```

5. Click the "Ok" button.
6. Enter your Saclink username and password. Please note: when you type a password in UNIX, it doesn't display any text to the screen. Don't worry, UNIX is listening.

PuTTY Instructions

1. Open PuTTY.
2. Enter the following in the Host Name box:

```
coding1.ecs.csus.edu
```

3. Make sure the SSH checkbox is selected. (SSH stands for Secure Shell).
4. Click the "Open" button.
5. Enter your Saclink username and password. Please note: when you type a password in UNIX, it doesn't display any text to the screen. Don't worry, UNIX is listening.

Part 2. Getting the Course Library (csc35.o)

This course uses a library object file. The library contains a large number of utility functions that will allow you to easily print integers, strings, and other useful tasks. But, to use the library, you first need to obtain it.

You need to use a UNIX command called "curl" (Copy from URL). Please type the following at the command-line and press the Enter Key. It will download the library from my website and save it to your account.

```
curl devincook.com/csc/35/csc35.o > csc35.o
```

UNIX will display information during the download. You can ignore this. It means it worked correctly.

```
[dcook@ecs-pa-coding1 ~]$ curl devincook.com/csc/35/csc35.o > csc35.o
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  6624  100  6624    0     0   5434      0  0:00:01  0:00:01 --:--:--  5433
```

You can check that the file downloaded by typing **ls** and pressing the Enter Key. You should see the library file listed.

Part 3. Creating / Editing a File

Now that you have the library, it's time to write your first assembly program. First, you need to create your source code. In UNIX, there a number of excellent text editors. We are going to use an application called *Nano* – which is easy-to-use and quite user-friendly.

To create a new file (or edit an existing file), type "nano" followed by the filename and press the Enter Key.

```
nano lab2.asm
```

Once it opens, notice that all the options are listed on the bottom of the screen. In UNIX, the carrot symbol **^** represents pressing *and holding* the **Control** key. So, for example, to exit your program you will press **Control** and **X**.

The following is the solution for the classic "Hello, World' program. This solution makes use of the CSC 35 library. Please type it verbatim into the text editor. Pay very close attention where there are spaces.

You don't have to type the comments, but it might be good idea. You need to understand what each line does.

```
# lab2.asm
# YOUR NAME HERE
#
# 1. Assemble : as -o lab2.o lab2.asm
# 2. Link     : ld -o a.out lab2.o csc35.o
# 3. Execute  : ./a.out

.intel_syntax noprefix              #Use Intel formatting
.data                               #Start the data section
Greeting:                           #Message is an address
    .ascii "Howdy, world!\n\0"      #Create a buffer of ASCII

.text                               #Start the text section
.global _start                      #Make the _start label public

_start:                             #UNIX starts here
    lea  rcx, Greeting              #Put address into rcx
    call PrintStringZ               #Execute the csc35.o subroutine

    call Exit                       #Execute the csc35.o subroutine
```

Once you have typed out the program, press **Control** and **X** to exit Nano. Remember to select "yes" to save the buffer (that's the memory which is storing the text of your program).

Part 4. Assemble the Program

If you type **`ls`** and press Enter, you should be able to see your new file listed. Now that it is created, let's assemble it into an object file. Type the following and, then, press the Enter Key. Don't forgot the **`-o`**. It specifies that the filename listed – after it – is the output.

Warning: If you list your **`.asm`** file after the **`-o`**, it will be destroyed.

```
as -o lab2.o lab2.asm
```

If you have any typos in your program, the assembler will list the errors. If that happens, open your existing program (type **`nano lab1.asm`**) and fix them.

Part 5. Link the Objects

If you didn't receive any error messages from Step 3, then the object file was created. Now let's link your object file to the CSC 35 library and create a program. Let's use a default program name of "a.out".

Type the following. Don't forget to type **a.out** after **-o**. The first character is an lowercase L; not a 1.

```
ld -o a.out lab2.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.

Part 6. Execute

You can execute your new program by simply typing its name and pressing the Enter Key.

```
./a.out
```

Requirements

Now work on each of the requirements below one at a time. You will turn in the final program, but incremental design is best for labs. You must think of a solution on your own. The requirements are as follows:

1. Put your name and section # in a comment at the very top of your program. *(5 points)*

2. Print "Howdy, world" to the screen. You can make it "Greetings" or any type of text you like. This was pretty much done for you already in the sample code. *(5 points)*

3. Print the text "My name is" and your full name to the screen. *(5 points)*

   Do **not** use the same string as in #1. Create a new label and ASCII text.

4. You have just traveled over the entire continent to get to California. You probably have some advice for your new neighbors. Print that to the screen. *(5 points)*

   Do **not** use the same string as in #1, #2 or #3. Create a new label and ASCII text.

5. Finally, let's print off another historical year that is interesting to you. For example: "In the year 1947 Sacramento State was founded!" The sentence must have text before and after the year. *(10 points)*

   Here's the tricky part: the year must be printed using the **PrintInt** subroutine. So, put the immediate value (the year) into **rcx**. Make sure to use **mov** rather than **lea**.

   There is no concatenation in assembly. For the final sentence, you will need three prints – two PrintStringZ and one PrintInt.

Submitting Your Lab

Afterwards, run Alpine by typing the following and, then, enter your username and password.

```
alpine
```

Please send an e-mail to yourself (on your Outlook, Google account) to check if Alpine is working. To submit your lab, send the assembly file (not a.out or the object file) to:

```
dcook@csus.edu
```

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.