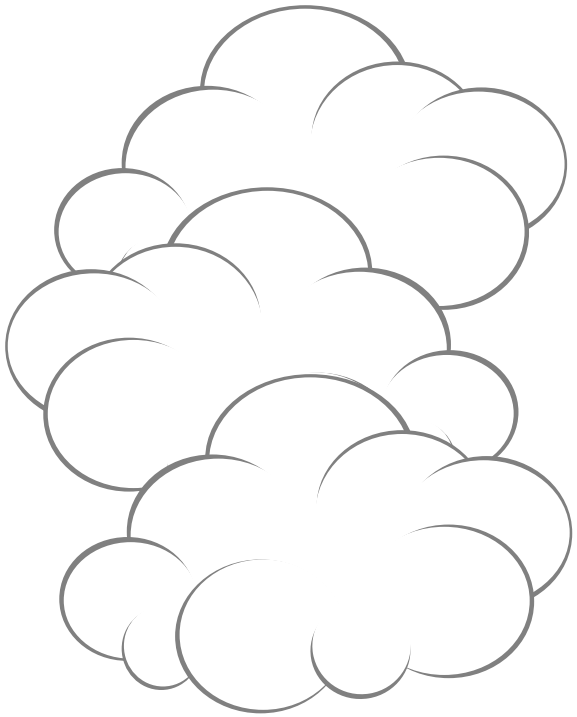
California State University, Sacramento
College of Engineering and Computer Science

Computer Science 35: Introduction to Computer Architecture

Fall 2021 – Lab 7 – *Apparition Testing*

Overview



In the ***Wizarding World of Harry Potter***, witches and wizards have many amazing powers. But, just like in the muggle world, wizards and witches often have to travel – whether it is to go shopping, go to work, or just visit friends.

Wizards and witches have access to the standard forms of transportation: flying brooms and the Floo Network (traveling through fireplaces). While these are safe and trusted forms of transportation, they are not a flexible in the destination. And, in the case of brooms, a tad slow.

One incredible talent is the ability "apparate" – where the witch or wizard can instantly teleport to any desired destination.

However, apparition has a problem – it's a tad dangerous. An unfortunate witch or wizard can "splinch" themselves – leaving a body part behind. Yes, that's quite horrifying! For this reason, there is Apparition Class, where students can learn the technique safely.

Your Task

For this class, you are going to use two features of processors – the Vector Table and subroutines.

At the beginning of the semester, you wrote a lab that implemented the classic Hello World program. For the entire semester, you relied on the CSC35.o library. This hid the details of the operating system from you. Well, it's time to do the hard work – talk to the operating system directly.

In fact, you are <u>not</u> allowed to use the library!

To simulate Apparition Testing, you are going to print some text to the screen – then call a subroutine (twice) that will print "Poof" (or something to that effect). Finally, you will print your plans for Winter Break.

So, your program will do the following:

1. Print your name to the screen.
2. Print some text to the screen like "here goes!"
3. Call a subroutine twice. It represents apparition to the subroutine, and then coming back. The subroutine will display text like "poof" (you will have to use a longer string).
4. Finally, print some text to the screen about your victory in apparating and your plans for Winter Break.
5. End the program

Sample Run

The following is a sample run of the program. *You must change the wording of the text.* You don't need the blank lines – I added those for readability.

```
My name is Harry Potter

I'm a bit nervous, I don't want to splinch myself.

Poof!

Poof!

Well, that was easy! I guess for Winter Break, I'll hunt for horcruxes!
```

Both were Printed by the subroutine

Talking to the Kernel

Let's start off by seeing if we output "Hello, World!" using UNIX Kernal calls. The basic calls, that you will need for this lab, are below.

Call	rax	rdi	rsi	rdx
Write	1	File Descriptor (1 = screen)	Source address	Total bytes to write.
Exit	60	Error Code *(0 = all okay)*	*none*	*none*

Tips

- You have to manually count the bytes for each string. The **\0** was for the benefit of the PrintCString subroutine. It used the null character to count the bytes for you.
- Like all labs, **build it in pieces**. Get the "Exit" call to work first before working on the writes.
- You must setup all registers – **each time** – before you call the kernal.
- Pay close attention to the order of your instructions. Syscall only after you have all the registers ready.

Linking the Object File

Since you are **not** using the CSC 35 library. When you link, you will only specify the object you created.

```
ld -o a.out lab7.o
```

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. Any lab using the csc35.o library will automatically receive a zero. The requirements are as follows:

1. Print your name to the screen
2. Print some text to screen before you apparate (must be at least 10 characters).
3. Call the subroutine twice (since you are apparating to the destination and coming back)
4. The subroutine should print some text (must be at least 10 characters).
5. Print some text about your Winter Break plans (must be at least 10 characters).
6. Exit your program

Submitting Your Lab

Please set the subject field of your e-mail to be:

CSc 35 – #

...where # is your lecture section number. This will help me sort your work.

To submit your lab, you must run Alpine by typing the following. You might have to re-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:

```
To      : dcook@csus.edu
```

Please give a descriptive subject for your e-mail. For example, the following is a good subject for a student in lecture Section 1.

```
Subject : CSC 35 – 1
```

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 *object* and *source* 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.