California State University, Sacramento
College of Engineering and Computer Science

Computer Science 35: Introduction to Computer Architecture

Spring 2021 – Lab 3 – *Gringotts Wizarding Bank*

Overview

Whether you are a student at the famous ***Hogwarts School of Witchcraft and Wizardry***, or a wizard working a humble job in Hogsmeade, you need to keep track of your finances.

The ***Gringotts Wizarding Bank*** is renowned throughout the Wizarding World is a the premier (and feared) banking firm.

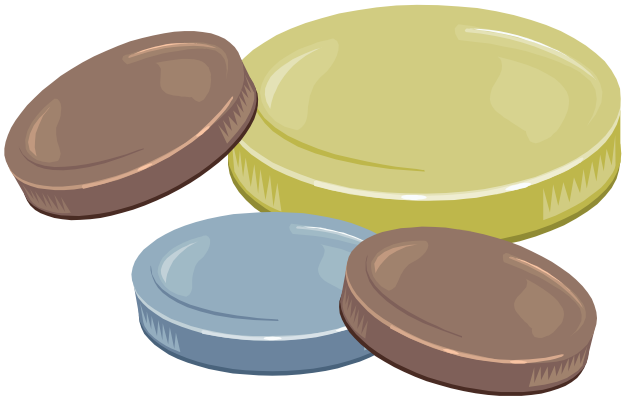
Like the muggle world, wizards and witches need to keep track of how much money you have. Debt is bad.

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.

For this program, you are going to input how much the user makes each month. Then, you are going to input, and subtract, all their debt-related expenses. Your program will then tell them how much they ***really*** make each month
You must think of a solution on your own.

Afterwards, your program will display text letting them know if they are fiscally stable. Gringotts does not like its customers to go into debt. So, this will be an If-Else Statement.



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 **red**. You don't have to make the text that color in your program.

Example 1

In this example, the wizard/witch is fiscally solvent (able to pay the bills). So, the problem will display a nice message from Gringotts.

```
Gringotts Wizarding Bank

How many galleons do you earn month?
2000

How much do you spend on potions?
150

How much do you spend on rent?
300

How much do you spend on food & medicine?
700

Well, your actual net income is $850

You are fiscally solvent. It would be wise deposit it.
```

Example 2

This example, the wizard/witch's outgo is larger than their income. As a result, they are not fiscally solvent and is in danger of going into further debt.

```
Gringotts Wizarding Bank

How many galleons do you earn month?
1200

How much do you spend on potions?
200

How much do you spend on rent?
500

How much do you spend on food & medicine?
650

Well, your actual net income is $-150

YOU ARE GOING TO OVERDRAW. MAKE AN APPOINTMENT WITH A GOBLIN ADVISOR.
```

Tips

Reading Integers

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

```
call ScanInt                #rbx = scanner.nextInt();
```

How to approach the problem

- Work in your program in parts. First get the math working correctly before working on the If-Statement. Incremental design is the best solution.
- The RBX register is used by the library to input data and output results. You will need to make use of other registers.
- It might be a good idea to select a register to store the galleons the user earns month. You can subtract the other values from it.
- Pay close attention to which operand is changed with the SUB instruction.

Requirements

Do **not** use direct storage (which we haven't covered yet). Any lab using direct storage will receive a zero.

You must think of a solution on your own. The requirements are as follows:

1. Input the initial (galleons earned) value *(5 points)*
2. Input at least 3 costs with helpful prompts. *(10 points)*
3. Compute how much they actually have left (the net). Display this information to the screen. *(10 points)*
4. Output a statement depending if their net is ≥ 0. This is an If-Else Statement. *(15 points)*

Submitting Your Lab

This activity may only be submitted in Intel Format. Using AT&T format will result in a zero.

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
```

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.