Assignment #8 – External Functions/Command Line Arguments

CS 218 Fall 2020

Purpose: This assignment will explore the use of accessing functions from a library and processing command line arguments.

You will need to write the program from scratch. Although we will only be working with one file, we will use the gcc compiler to link our program with the C standard library. If you do not have gcc installed, see the virtual machine installation guide.

The program will be using a command line argument to get the diameter of a half sphere and outputting its volume. These values will be double precision floating points. Instead of writing the code to convert to these formats, we will utilize the C standard library files stdlib.h and stdio.h to handle it.

Specifications

The program will expect to receive 3 command line arguments. The first is just the name of the program, the second will be the tag “-d”, the third will be a floating point format value.

```
./a.out -d 42.1
```

We will be using the C standard library function atof to convert the third argument into a double precision floating point.

Ensure that the -d tag is properly written (including the NULL character) and that exactly 3 command line arguments are used to start the program. Otherwise, print out an appropriate error. After converting the floating point value, ensure it is not an invalid string (see below). Print an error if it is.

A negative value for the diameter is acceptable.

Write a function to calculate and return the volume of a half sphere using the following formula and print the result using printf() (see below):

radius = diameter / 2
PI = 3.14159
halfSphereVolume = 2 / 3 x PI x radius^{3}

You may use variables to declare the values of PI and the error result for atof.

All instructions and values should be done using double precision floating points.

Setting up external function calls

Make sure to place this just before your text section:

```
extern atof, printf
```

Linking using GCC

In order to use the gcc compiler to link our program with the standard library, you will need to change your commands to assemble/link your program slightly.

```
yasm -g dwarf2 -f elf64 programName.asm
gcc -no-pie programName.o
```

In addition, instead of using “_start”, change your starting function name to “main”.

```
global main
main:
```

Standard Library Functions

double atof(&string)
This function will determine the double precision floating point value of the provided string argument. In the case that an invalid string is provided, the value 0.0 will be returned.

int printf(&formatString, <variable # of arguments>)
The format string contains the message to print to the console along with special format marks to insert values into the output. For example, to insert a floating point value you would put “%f” into the format string.

```
    "Halfsphere Volume: %f", LINEFEED, NULL
```

Note: printf requires that the rax register be set to the number of floating point registers being used as arguments. In the case of printing a single floating point value, it must be set to 1 prior to calling the function.

printf is known as a variadic function, meaning that it can take a variable number of arguments. In this case, the number of arguments required are determined from its format string.

Before calling printf, the stack must be adjusted to be aligned to values of 16. If it is causing a segmentation fault, subtract 8 from the stack pointer and restore it after the function call.

The return value can be ignored.

Submission

Once you are satisfied with the program, upload the assembly source code (.asm) file to the assignment #2 page on the class website.

Expected Outputs

```
$ ./a.out
Enter -d <number> after the program name.
$ ./a.out -f 14
Invalid diameter tag.
$ ./a.out -d12 14
Invalid diameter tag.
$ ./a.out -d 14.5
Halfsphere Volume: 798.127484
$ ./a.out -d Four
Invalid numeric format for diameter.
$ ./a.out -d 13.2e2
Halfsphere Volume: 602129705.760000
$ ./a.out -d 1320
Halfsphere Volume: 602129705.760000
$ ./a.out -d -0.9E-1
Halfsphere Volume: -0.000191
```