PORTLAND STATE UNIVERSITY
CS201: COMPUTER SYSTEMS PROGRAMMING (SECTION 5)
WINTER 2020

HOMEWORK 3

DUE: FEBRUARY 26, 11:59 PM

1. Goals and Overview

1) Develop practical knowledge of x64 Assembly Language including Branches, Arithmetic and Logic Operations
2) Interface with C programs using the System V x64 calling convention
3) Access unidimensional arrays using Assembly

2. Development Setup

For this programming assignment you will work in the CS Linux Lab (linuxlab.cs.pdx.edu) located in FAB 88-09 and 88-10. If you don't already have an account, go to http://www.cat.pdx.edu/students.html for instructions.

For this Homework you must use x64 Assembly (AT&T Syntax) and Make. Please refrain from using C code, any language extensions or 3rd party libraries. We will use the GNU Assembler (AS), the GNU C Compiler (gcc) and Make already installed in the development machines in the lab.

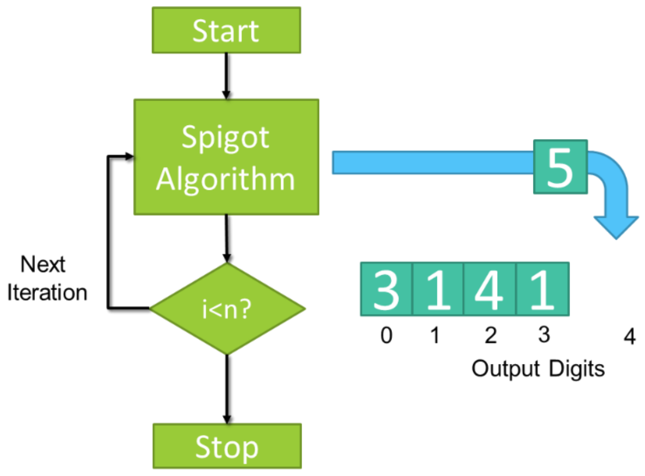
Grading will be done in this setup so please make sure that your code works under this conditions. Also please note that the System V calling convention is specific to Linux running on Intel 64-bit platforms so your code will not run on any other platforms!

3. Introduction

In this Homework we will implement the Spigot Algorithm for the computation of Euler's number (e=2.78128…) in Assembly. Spigot Algorithms are a class of algorithms designed to compute the value of a mathematical constant to an arbitrary precision.

One of the advantages of Spigot algorithms is that they only require a limited amount of intermediate storage for the computation, as the digits are generated one by one from left to right. This is a very desirable property as most architectures have limited size registers and are not very efficient performing arbitrary precision arithmetic.

A Spigot algorithm will run a determined function multiple iterations and during each iteration it will generate the following digit in the sequence.



4. Problem Description

4.1 Spigot Algorithm of e

Our implementation of the Spigot Algorithm of e will be implemented based on the following infinite series:

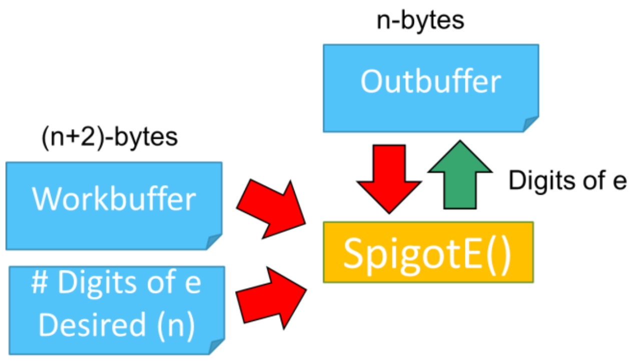
$$e = \sum_{k=0}^{\infty} \frac{1}{k!}$$

As a general ide the algorithm truncates the series to a fixed number of elements and then extracts each digit by dividing by 10. It is important to note that **you do not need to understand how the algorithm works. Instead we will translate the C algorithm provided into Assembly in similar manner as we have been doing in class!**

4.2 Assembly Implementation

You are provided with a complete solution of the Spigot Algorithm of e in C, located in the file SpigotE.c. This file contains the function SpigotE(). The provided solution also includes a main function that receives as command line parameter, the number of desired digits of e to compute. The function main() will allocate a working buffer of size n+2 and an output buffer of size n, where n is the number of digits of e to compute. The main function then will call SpigotE() with this parameters.

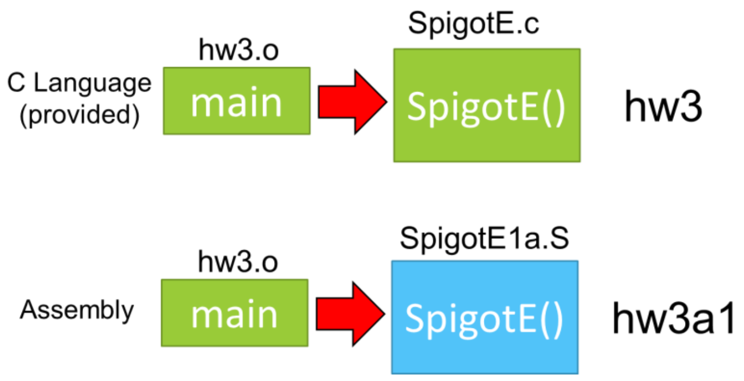
The SpigotE() function is responsible of computing the first n digits of e and will write each digit individually in the output array of characters. **Please note that the function SpigotE() does not return the values of the constant as ASCII characters but just as 8-bit integers**.



The main function is then responsible of converting each of the output digits of e returned by SpigotE() into ASCII characters and prints them on the screen.

Your job in this assignment is to write a replacement of SpigotE()in assembly (SpigotE1a.S). **The main C function provided in hw3.c should be able to link to the new object file assembled (SpigotE1a.o). The resulting linked executable file should be named hw3a1.**

You must not make any changes to the structure or implementation of hw3.c and you must follow all the calling conventions necessary to interface with the provide hw3.c file.



4.3 Makefile

As part of your program your solution you should augment the provided Makefile so that it automatically compiles both variants of the code: Executable using C Version of SpigotE() named hw3 (provided) and Executable using Assembly Version of SpigotE named hw3a1. Your Makefile must also provide a cleanup entry (make clean) to delete all the files generated by the compilation process (i.e. object files, executables, libraries, etc.)

5. Implementation Overview

These are a few key concepts you must address as part of your implementation:

1) You must create a text segment to write SpigotE

2) You must declare the name of SpigotE as a global symbols in your Assembly file so that the linker can find them and properly interface to them. Note: You can use the assembly "Hello World" program from one of the Assembly lectures to guide you on completing Step 1 and 2
3) Your functions must respect the System V calling conventions by preserving the values of the callee-saved registers.
4) Your function must follow the System V calling conventions for parameters passing and function return so that your code can interface with the function caller (hw3.c).
5) Both of your assembly implementations must return the exact same values as the provided implementation in C.

6. Hand-In

For submission, you should provide only source code (*.S, *.c, *.h) and a Makefile script as outlined in Section 4.3. More specifically you must provide the original unchanged SpigotE.c file, the assembly SpigotE version in SpigotE1a.S and the Makefile.

Please pack your files into a TAR file with the following filename structure CS201_HW3.tar.

We will use the D2L system for submission (https://d2l.pdx.edu/). Please remember that there is no late policy.

7. Rubric

Grading will be done according to the following Rubric with partial credit given for features partially implemented. **NO POINTS WILL BE GIVEN FOR SUBMITTING DISASSEMBLED CODE.**

Feature	**Possible Points**
First digit of e is initialized to 2	5
Text segment is created correctly and SpigotE is a global symbol	10
Each element of Workbuffer is initialized to 1	20
Inner and Outer loop of Spigot computation iterate correctly	25
Output and Workbuffer entries are computed correctly	25
Code does not causes segmentation fault	5
All callee-saved registers are preserved across the function calls	10
Total	***100***

8. References

Homework 3 Overview Class Slides - http://www.raoulrivas.net/content/upload/08b-Hw3.pdf

Assembler Manual - http://web.mit.edu/gnu/doc/html/as_toc.html

GNU Assembler Examples - http://cs.lmu.edu/~ray/notes/gasexamples/

GNU Make Manual (Look at Section 2.2 for a simple Makefile example) - http://www.gnu.org/software/make/manual/make.html