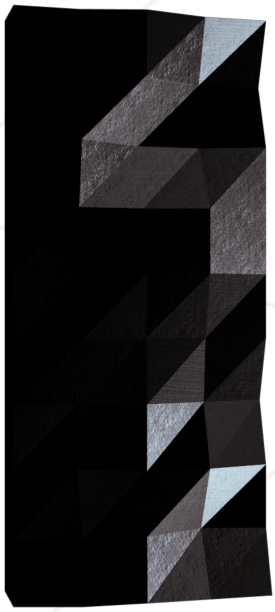

Recursion in Assembly

You must understand recursion to understand recursion.

Alexander Wang alwang@student.42.us.org
42 Staff pedago@42.fr

Summary: This project will help you learn the essentials of recursion in Assembly.

Contents

Chapter I

Foreword

War! The Republic is crumbling
under attacks by the ruthless
Sith Lord, Count Dooku.
There are heroes on both sides.
Evil is everywhere.
In a stunning move, the
fiendish droid leader, General
Grievous, has swept into the
Republic capital and kidnapped
Chancellor Palpatine, leader of
the Galactic Senate.
As the Separatist Droid Army
attempts to flee the besieged
capital with their valuable
hostage, two Jedi Knights lead a
desperate mission to rescue the
captive Chancellor....

Chapter II

Introduction

Welcome to the final project in the Assembly series. Here you will learning about recursion. Yes, yes I understand you hate recursion. But unfortunately certain things in programming is just significantly easier if you use recursion. Good examples for this would be the nifty factorial which you will be programming in this section.

Thankfully you will only need to do two exercises for this project and then you can move on to staring at vulnerabilities for awhile before moving on to the wonderful Capture-The-Flag at the end of this.

Chapter III

Goals

The goal of this project is quite simple. It is for you, the reader and the person doing this project, to have a basic understanding of recursion. While it is a, truthfully, relatively infrequent occurence in the wild it is critically important to notice when you are in one.

Chapter IV

Mandatory part

- The language you will be using is x86 Assembly
- You will compile everything using GAS or "as".
- Everything here will be done in a Linux 32-bit Virtual Machine
- The autograder has been tested only on a Ubuntu 32-bit system if one exists.
- As such, it is highly recommended you use Ubuntu 32-bit.
- Unless specified otherwise you **must** follow the C calling conventions.

Chapter V

Exercise 00: Even or Odds

	Even or Odd
Topics to study :	
Files to turn in : `evenOrOdd.s`	
Notes : `n/a`	

- Implement the following recursively

$$f(x) = \begin{cases} 1 \text{ if x is even} \\ 0 \text{ if x is odd} \end{cases}$$

- Save the program as `evenOrOdd.s`

It will look like the following:

$$f(x) = \begin{cases} \texttt{1 if x is 0} \\ \texttt{0 if x is 1} \\ \texttt{f(x - 2) all other} \end{cases}$$

Chapter VI

Exercise 01: Recursive Factorial

	Recursive Factorial
Topics to study :	
Files to turn in : `recurseFacts.s`	
Notes :	

- Implement factorial recursively:

$$x! = \begin{cases} 1 \text{ if x} = 0 \\ \text{(x - 1)! * x if x} > 0 \end{cases}$$

- Save your program in `recurseFacts.s`

Chapter VII

Bonus part

This will only be graded if the rest are perfect:

- Implement the following algorithm recursively as a function called "gcd":

$$gcd(a,b) = \begin{cases} \text{a if b} = 0 \\ \text{gcd(b, a) if a} < \text{b} \\ \text{gcd(a - b, b) else} \end{cases}$$

- As usual, you must follow the C calling convention.
- Output the answer as usual.
- Save the following as `gcd.s`

Chapter VIII

Turn-in and peer-evaluation

Turn your work in using your `GiT` repository, as usual. Only work present on your repository will be graded in defense.