Tic-Tac-What

Tic-Tac-Toe in Assembly

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

*Summary: In this project, you will be writing a fully functioning Tic-Tac-Toe game in x86 Assembly with **many** restrictions on it.*

Contents

Chapter I

Foreword

Four thousand years before the rise of the Galactic Empire, the Republic verges on collapse. DARTH MALAK, last surviving apprentice of the Dark Lord Revan, has unleashed an invincible Sith armada upon an unsuspecting galaxy.
Crushing all resistance, Malak's war of conquest has left the Jedi Order scattered and vulnerable as countless Knights fall in battle, and many more swear allegiance to the new Sith Master.
In the skies above the Outer Rim world of Taris, a Jedi battle fleet engages the forces of Darth Malak in a desperate effort to halt the Sith's galactic domination....

Chapter II

Introduction

Welcome to the bonus project in x86 Assembly. For this project you will be implementing Tic-Tac-Toe in x86 Assembly. This is where I say you should back away now because only those who really like Assembly should try this project or know what they are doing.

In this project, unlike the ones earlier and later on. There will be *many* restrictions. All of them will be verified by an autograder. You have been warned.

`If you decide to complete this project, you will learn whether or not you would enjoy doing certain aspects of embedded programming (a possible career path in programming)`

Chapter III

Goals

The goal of this project is a fully functioning x86 Assembly Tic-Tac-Toe game complete with error messages and other things.

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.
- You **must** follow the C calling conventions.
- You **must** use every single possible jump at least once.
- You **cannot** use the *call* command.
- All functions must be written by you.
- Must print out a relevant error message when an error has been made
- Your program cannot crash in any shape or form for any reason outside of intended functionality.
- You must print out the board to the terminal.
- User input will be through the terminal.
- In order to maintain neatness you may use as many files as you want
- If there is multiple files must tell the grader how to compile in a text file.

Chapter V

Exercise 00: Tic-Tac-Toe

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

Implement a fully working Tic-Tac-Toe game in x86 Assembly.

Chapter VI

Bonus part

This will only be graded if the rest are perfect.

The sky is the limit but here are some suggestions

- Color the output of the users.
- Do not use the .data section at all.
- Convince a mentor to play it.

Chapter VII

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.