Instituto Tecnológico de Costa Rica

Computer Engineering

Introduction to Operating Systems (CE-4303)

Short Assignment 2

Creating a bootloader using ASM

Professor:

Diego Vargas González

Student:

Fabián Astorga Cerdas - 2014040808

Javier Sancho Marín - 2014159997

Óscar Josué Ulate Alpízar - 201229559

Introduction

The Bootloader is a piece of software that is executed every time the hardware device is powered up. It is executable machine code. It's quite heavily device-specific because its main task is to initialize all the low-level hardware details. The bootloader can be contained on a separate EEPROM or directly on flash storage (most common). Is located in the first sector of a non-volatile memory device and can be only 512 bytes in size (Master Boot Record).

A bootloader is not required to boot Linux. The use of one or several bootloaders in a row to chainload a Kernel is not a categorical necessity, it is merely a very crafty method to start an operating system [1]. In this case, an USB is used to boot a program made in assembler.

The boot process has a different component's like: The Bios, The Post Process and BIOS Boot Handoff. The BIOS is where hardware meets software for the first time, the BIOS code is baked into the motherboard of your PC, usually stored on what is called an EEPROM and is considerably hardware-specific. Once your PC has been powered on, the BIOS begins its work as part of the POST (Power-On Self Test) process. It bridges all the various parts of your PC together, and interfaces between them as required, setting up your video display to accept basic VGA and show it on the screen, initializing the memory banks and giving your CPU access to all the hardware [2].

In summary, your PC is turned on and the BIOS initializes the hardware, then the BIOS calls code stored in the MBR (Master Boot Record) at start of disk 0, next the MBR loads code from the bootsector of the active partition, finally the bootsector loads and run the bootloader from its filesystem.

Boot loader Creation & Considerations

In order to create a bootloader with x86 set instructions, you need to attend the following considerations [4]:

1. Stored in the Master Boot Record (MBR), on the first sector of the disk.

2. Size is exactly 1 sector, or 512 bytes.

3. Ends with a signature of 0x55 and 0xAA at bytes 511 and 512.

4. Loaded by the BIOS POST interrupt 0x19 at address 0x7C00.

5. Runs in 16 bits mode.

Program Description

The program the bootloader loads is a game based on Snake, where the player's objective is to control a long creature, similar to a snake, that wanders around a delimited space map, eating apples, trying to avoid hitting its own tail or the walls that surround the game map. Each time the snake eats an apple, the snake grows more and more, causing the difficulty of the game to increase. The user controls the direction of the snake's head and the body of the snake follows it.

Development Environment

- Atom: is a source code editor open source available in many operating systems like GNU/Linux, MacOS and Windows.
- Nasm: the Netwide Assembler, NASM, is an 80x86 and x86-64 assembler designed for portability and modularity. It supports a range of object file formats, including Linux and *BSD a.out, ELF, COFF, Mach O, Microsoft 16-bit OBJ, Win32 and Win64. It will also output plain binary files. Its syntax is designed to be simple and easy to understand, similar to Intel's but less complex. It supports all currently known x86 architectural extensions, and has strong support for macros [5].
- Qemu: is a generic and open source machine emulator and virtualizer. This tool was used for test generated boot programs [6].
- dd: is a command line utility from coreutils, copies a file (from standard input to standard output, by default) with a changeable I/O block size, while optionally performing conversions on it. In this project was used to copy programs in a device to make a booteable's USB sticks [7].
- Makefile: was written in order to simplify the compilation process, which used the dd utility to generate the image file and putting it in the USB drive.

Program Design Details

Code description

The bootloader was designed and developed according to [3][4], which was found to be like a standard implementation. It's responsible for jumping to the address where the Snake is stored.

The *snake.asm* file contains all the logic of the game. It starts with the menu of the game. All of the code was programmed based on the *nasm* compiler. The menu is very basic, it contains the titles, such as the name of the game, (SNAKE), and the three levels of difficulty. It also contains the logic of the keyboard interruptions to start the game.

- $draw_snake$: function which is in charge of drawing the snake.
- $draw_apple$: Function which is in charge of drawing the apple in some place of the game map.
- $CheckWallCollision$: Verify if the snake collides with the walls in the edges of game map.
- $CheckSelfCollision$: Verify if the snake collides with your own body.
- $CheckAppleCollision$: Verify if the snake collides with the apples and then grows because the collision.
- $ShiftArray$: Move the snake across the game board.
- $RandomNumber$: Generate a random number used to set the position of the next apple.
- $DrawPixel$: Generic function to paint pixels on the screen.
- $CheckLVL$: Set the speed of the snake according to the level the game is in.
- $InterpretKeypress$: This function interprets which arrow is being pressed and ignores any other key.
- $SetScreen$: It sets the initial game screen. It sets the game space after the main menu.

UML Diagram

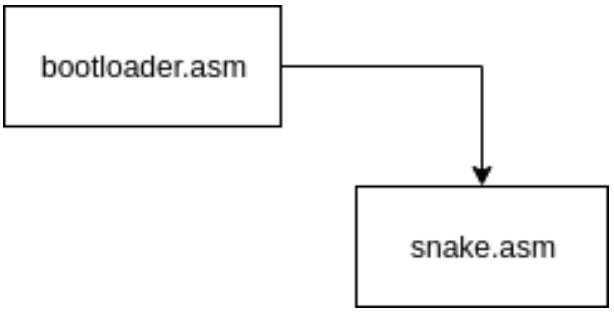

Figure 1. UML program diagram

How to use the Program

To use the snake bootloader, follow the next steps:

1. Insert an USB drive in your PC.
2. Go to the program directory, where the makefile and the others files are ubicated.
3. To generate the executables files, insert the following command: 'sudo make'
4. If the player wants to emulate the program, he can do it with Qemu (make sure you have this program installed). Insert this command: 'sudo make emul' and the snake will be start. If you don't want to emulate it, please skip this step.
5. To burn the .iso file into the USB, insert the following command: 'sudo make usb'
6. Reboot the machine and boot into the USB drive.
7. Game is ready to be played. Use the arrow keys to manage the snake.
8. The player can change the level if press key '1' to play level one, play level two if press key '2' and can play in level three if press key '3'.
9. If the player doesn't avoid its own tail or the walls that surround the game map, he will loose and immediatly starts another new game.

Student Activity Log

Task	Fabian Astorga	Javier Sancho	Oscar Ulate	Total
ASM research	7:00	12:30	10:00	30:30
Bootloader development	4:00	0:00	0:30	4:30
Setting second stage	9:00	1:00	1:00	11:00
Snake development	1:00	15:00	8:00	24:00
Fix game bugs	1:30	4:00	3:00	8:30
Makefile, management	2:00	0:30	0:30	3:00
Documentation	2:00	2:00	2:00	6:00

Table 1. Activity log

Project Final Status

Status

The project was almost completed succesfully. All of the requirements were completed. The working group didn't know many of the features used in the system, so a lot of research was needed to complete the game, especially the x86 instruction set and arquitecture.

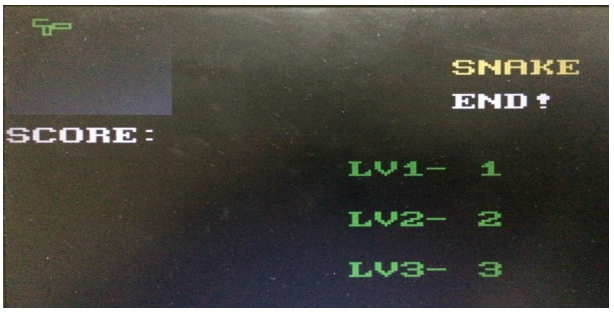

Figure 2. Screenshot of the game running.

Link to the project in github

https://github.com/josue2794/snake_jof.git

Issues

The project was completed succesfully. All of the requirements were completed. The working group didn't know many of the features used in the system, so a lot of research was needed to complete the game.

Conclusions

The bootloader is an important piece of software in the boot process of the computers operating systems, because it allows the boot option on internal or external devices. The boot process is fairly simple, but understanding its implementation in assembly language is not so easy. Because of its readability, it is hard to detect bugs by scanning the code. The bootloader loads all the sucessive stages of the booting process by jumping or calling to another physical address. The BIOS plays an important role in the initiation of a computer, because the connecting between hardware and low-level software pieces that allows the initializing user programs and others. The MBR (Master Boot Record) it's only reserved for using with boot devices, this space has only 512 bytes to assemble the bootloader.

Suggestions and Recommendations

- When programming in assembler, it's very important to have a clear understanding of the management of memory and the registers that we can use.
- Configuring the bootloader to load enough sectors for the second stage is also very important so that it behaves as intended.
- It's important to be careful when you write in disk sector, because you can write it in your system's disk.
- When testing, it's advisable to use a virtual machine so the programmer doesn't have to reboot their computer and spare some time.
- Emulators, such as QEMU, are good for making a general idea of the results, but not good enough to see how good the final product will be. They are inexact.

References

[1] R.A.Thomas. (2017, October). The Bootloader. 15/03/2018, OpenWrt website: https://wiki.openwrt.org/doc/

[2] A. Bhatter. (2015, April) Writing a boot loader in assembly and c - part 1. [Online]. Available: http://www.codeproject.com/Articles/664165/Writing- a-boot-loader-in-Assembly-and-C-Part (Accessed March 15, 2018).

[3] Hasani, F. (2017). Building Your Own Bootloader - Fisnik Hasani. [online] Fisnikhasani.com. Available at: http://fisnikhasani.com/building-your-own-bootloader/ (Accessed 15 March 2018).

[4] Youri Ackx. (2014, December). Hello World bootloader in assembly. 15/03/2018, Disqus Sitio website: http://blog.ackx.net/asm-hello-world-bootloader.html

[5] NASM Manual, The NASM development team, March 2016. [Online]. Available: http://www.nasm.us/doc (Accessed March 15, 2018).

[6] "Qemu wiki," 2015. [Online]. Available: http://wiki.qemu.org/Main Page (Accessed March 15, 2018).

[7] S. Kemp, D. MacKenzie and P. Rubin, dd - Linux Manual Page, Free Software Foundation, 2010. [Online]. Available: http://man7.org/linux/man- pages/man1/dd.1.html (Accessed March 15, 2018).