Instituto Tecnológico de Costa Rica

Facultad de Ingeniería en Computación

Falling Code

Tarea 1

Fabrizio Alvarado Barquero
2017073935

Septiembre 2020

0.1 Introdución

El problema a solucionar es la implementación de un sistema operativo llamado MacOS, el cual nos va a ayudar a comprender de una mejor manera el funcionamiento del booteo de una computadora, como funciona este proceso en si y como está compuesto y estructurado en cuanto a los elementos de este proceso.

El sistema MacOS presenta una unica funcionalidad, la cual es la animación de 'Falling Code' tal como la que se presenta en la película The Matrix, donde el programa constantemente esta imprimiendo caracteres para no dejar de hacer la simulación de 'falling'. Además permite que el usuario ingrese caracteres deseados mediante el teclado y estos mismos son impresos con la animación previamente descrita.

0.2 Ambiente de desarrollo

- QEMU:
Sistema emulador de software.

- Geany:
Editor de texto.

- GitHub:
Control de versiones.

- Oracle Virtual Box:
Emulador para verificación del resultado final.

0.3 Estructuras de datos usadas y funciones

* Funciones

- start:
Función inicial del programa, esta activa el modo video, cambia el color de fondo y setea el cursor.

- code:
Función recursiva del programa, la cual revisa el input, en caso de haber algo dentro, llama a la función respectiva de imprimir, de no haber nada, llama al timer e imprime el caracter default y finalmente llama de nuevo a esta misma funcion para iniciar nuevamente el proceso.

- printcharacterDefault:
Imprime la secuencia de columnas a lo largo de la pantalla con caracteres default.

- imprimirDefault:
Imprime el caracter default utilizado, el cual es 1 ASCII (una carita felíz).

- printcharacter:
Imprime la secuencia de columnas a lo largo de la pantalla con el caracter ingresado.

- inputcharacter:
Lee el caracter ingresado.

- timer:
Realiza un delay de aproximadamente 2 segundos entre caracteres.

- scrollDown:
Realiza la función de bajar el scroll de la pantalla 3 renglones.

- imprimir:
Imprime el caracter previamente ingresado.

* Estructuras

- No hay estructuras como tal, el programa es muy lineal.

0.4 Instrucciones para ejecutar programa

- Generación de .bin desde el .asm:
nasm -f bin -o [elBinario].bin [elEnsamblador].asm

- Emulación del sistema en plataforma QEMU:
qemu-system-i386 -fda [elBinario].bin

-Configuración y generación del .iso:
nasm -f bin -o boot.bin boot.asm
dd if=/dev/zero of=floppy.img bs=1024 count=1440
dd if=boot.bin of=floppy.img seek=0 count=1 conv=notrunc
mkdir iso
cp floppy.img iso genisoimage -quiet -V 'MYOS' -input-charset iso8859-1 -o myos.iso -b floppy.img -hide floppy.img iso/

- Ejecutar el MacOS en VB:
Crear una maquina virtual y seleccionar el .iso

0.5 Actividades realizadas por estudiante

Viernes 25 de septiembre - 5 horas
-Instalación de herramientas como qemu y nasm
-Investigación sobre booteo MBR
-Se refresco el lenguaje ensamblador mediante un hello world
-Investigacion e implementacion de ejemplo de lectura y escritura de usuario

Problemas:No logre implementar ejemplo de lectura y escritura en el qemu, únicamente como programa corriente de assembly

Sábado 26 de septiembre - 5 horas
-Investigué sobre cómo funciona realmente el booteo ya que estaba trabajando sobre un programa de ensamblador "normal".
-Tuve que iniciar un documento nuevo ya que los tenía no funcionaban de manera "booteable"
-Logre implementar una lectura y escritura de letras, además del video mode y el color de las letras.
-Termine el día de trabajo "traveseando" la funcionalidad de mover cursor para acomodar las letras que se van digitando.

Problemas: No logro encontrar una fórmula o algo por estilo para lograr el efecto de "falling", además de otros detalles como ir eliminando los caracteres antiguos.

Lunes 28 de septiembre - 3 horas
-Pensé alguna manera de mejorar el efecto falling, pero no se me ocurre ninguna solución con notable mejora a la antigua.
-Mejoré algunos detalles como el color de fondo del DOS y ocultar el cursor.
-Investigue sobre la generación del ISO.
-Documentación de código.
-Documentación de tarea.

Problemas: No logre generar el iso correctamente.

Martes 29 de septiembre - 6 horas
-Se añade funcionalidad del efecto falling
-Se añade funcionalidad de un timer
-Se añade funcionalidad de que la pantalla haga scroll para abajo aun cuando el usuario no está digitando caracteres
-Investigue más sobre la generación del .iso

Problemas: Me falta poder generar correctamente el iso

Miércoles 30 de octubre - 3 horas:
-Investigué aún más sobre booteo ya que ninguna de las anteriores me funcionó.
-Logré encontrar una página de Stack Overflow donde me funcionó la generación del iso.
-Logré bootear correctamente el iso generado en VirtualBox.
-Finalización de documentación

0.6 Autoevaluación

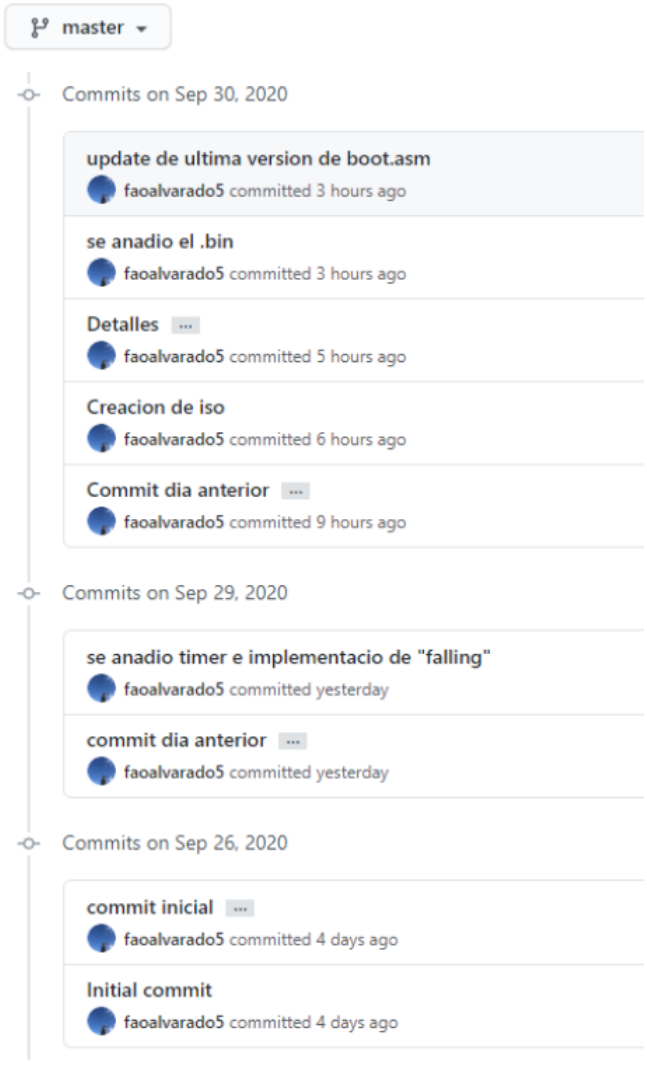
- Estado final:
El estado final del programa MacOS es tal como se pidió en la especificación, un sistema booteable que permite ver una animación de falling code.

- Problemas:
El unico problema existente es que no tiene el nivel de detalle que se veía en el video de ejemplo del programa en The Matrix, pero básicamente y en esencia es igual.

- Limitaciones:
Creo que no hubo ninguna limitación mas allá de mi pobre conocimineto en el lenguaje utilizado.

Evaluación:

Característica	Valor	Obtenido estimado
Arranque	30	30
MacOS	50	45
Documentación	20	15
Extra	10	0
Total	110	90



-Repositorio de GitHub:
https://github.com/faoalvarado5/tarea1Operativos

-Reporte de commits:
https://github.com/faoalvarado5/tarea1Operativos/commits?author=faoalvarado5

0.7 Lecciones aprendidas

En lo personal me gustó mucho la tarea, ya que por más dificil que fuera y demás, refresqué y reforcé mucho mi conocimiento sobre el lenguaje esamblador, debido a que desdichadamente mi curso de Arquitectura no fue el mejor.

Más puntualmente, aprendí el proceso de arranque de un sistema operativo y como este bootea desde que se presiona el botón de inicio de la computadora.

Finalmente, pero no menos importante, aprendí a organizarme y trabajar de una manera adecuada bajo presión, ya que en menos de una semana logré aprender un lenguaje, una funcionalidad vital de la computadora como el arranque y como esta está compuesta.

0.8 Bibliografía

- Gabrielececchetti.it. (2020). Retrieved 1 October 2020, from http://www.gabrielececchetti.it/Teaching/CalcolatoriElettronici/Docs/i8086$_{a}nd_{D}OS_{i}nterrupts.pdf$.

$-AssemblyLanguage.Vitaly_{f}ilatov.tripod.com.(2020).Retrieved1October2020, from$
$http://vitaly_{f}ilatov.tripod.com/ng/asm/.$

$-2-main_{m}enu-TechHelp!.Techhelpmanual.com.(2020).Retrieved1October2020, from$
$http://www.techhelpmanual.com/2-main_{m}enu.html.$

$-INT10H.En.wikipedia.org.(2020).Retrieved1October2020, from$
$https://en.wikipedia.org/wiki/INT_{1}0H.$

$-FLAGSregister.En.wikipedia.org.(2020).Retrieved1October2020, from$
$https://en.wikipedia.org/wiki/FLAGS_{r}egister.$

$-16-bitcomputing.En.wikipedia.org.(2020).Retrieved1October2020, from$
$https://en.wikipedia.org/wiki/16-bit_{c}omputing::text=A$

$-bootloader, C., Abdel-Rahman, M., Petch, M.(2020).$
$CreatingabootableISOimagewithcustombootloader.StackOverflow.Retrieved1October2020, from$
$https://stackoverflow.com/questions/34268518/creating-a-bootable-$
$iso-image-with-custom-bootloader.$