Tecnológico de Costa Rica

Primera Tarea Corta

Tutormec

Autor:
Yeison Cruz León

Profesor:
Ing. Kevin Moraga, MSc.

4 de agosto de 2015

1 Introducción

Desde tiempos antiguos, el escribir rápido en una computadora o bien una máquina de escribir ha sido una necesidad latente para casi cualquier usuario de alguna de estas; realizar trabajos, ensayos, resúmenes, todo se vuelve más sencillo cuando tienes una velocidad de escritura alta, puesto que se consume menos tiempo digitando lo que se piensa y más tiempo pensando lo que digitas.

El siguiente programa trata de enseñar precisamente eso, escribir de una forma veloz usando una computadora, para ello se utilizaron los siguientes requerimientos de software:

- Programar en ensamblador el booteo desde una unidad USB.
- Una vez que bootee desde el USB, cargará única y exclusivamente un programa llamado: **Tutormec**. Este programa consiste en un tutor de mecanografía.
- Se le presenta al usuario una interfaz donde las letras que se tienen que presionar se mueven por la pantalla utilizando las primeras 10 líneas.
- En el centro de la pantalla se encuentran bloques de la línea 10 a la línea 15 con una separación entre ellos.
- El usuario deberá lograr capturar las letras cuando estas se encuentren por la separación de los bloques.
- Correrá una animación de la letra pasando por en medio de los bloques.
- En el lado inferior de la pantalla usando las líneas de la 15 a la 25 se le deberá indicar de alguna forma al usuario que dedo debería usar para capturar la letra.
- Para todo lo anterior mencionado se debe utilizar la arquitectura x86.

2 Ambiente de desarrollo

Para realizar lo anterior mencionado se utilizarón diversas herramientas que facilitaron su desarollo, entre ellas se encuentran las siguientes:

- **Sublime Text 2:** Esta herramienta es un editor de texto muy popular centrado principalmente en código, el mismo soporta Snippets, plugins varios y una amplía gama de sistemas de construcción de código, entre ellos algunos de los más conocidos como Python, Java, PHP, C, C++ y por supuesto ASM.

- **The Netwide Assembler (NASM):** Es un ensamblador y desensamblador para la arquitectura x86 de Intel, utilizado para escribir programas en 16, 32 y 64 bits, considerado uno de los más populares ensambladores para la mayoría de sistemas operativos.

- **VMware:** Software de virtualización que simula un sistema físico con unas características dadas por el usuario, en el caso de este programa fue útil para bootear desde el USB y no tener la necesidad de estar reiniciando el computador en cada ocasión que se quería probar una nueva característica implementada.

- **Plop Boot Manager:** Esta herramienta nos brinda un .ISO el cual puede ser cargado utilizando **VMware** con el fin de mostrarnos una interfaz muy simple en la cual podemos seleccionar el dispositivo desde el cual deseamos realizar un booteo, para este caso, desde nuestro dispositivo USB.

- **HexEdit:** Herramienta que nos da la posibilidad de explorar un dispositivo USB y observar todos sus datos internos codificados en binario, muy útil para saber si nuestros datos se están copiando en los sectores del disco que necesitamos o hemos establecido.

- **GNU C And C++ Compiler (GCC):** Compilador de los lenguajes C y C++ utilizado en este caso para compilar un archivo .c que fue utilizado en el desarrollo del programa.

3 Estructuras de datos usadas y funciones

3.1 Estructuras de datos

En el desarrollo del programa se utilizaron las siguientes estructuras de datos:

- **Arreglo:** Un arreglo es un conjunto de datos que se almacenan en memoria de manera continua, para diferenciar sus elementos se utilizan índices donde cada dato del arreglo tiene asignado un índice distinto iniciando generalmente en cero.

3.2 Funciones principales

Dentro del código del programa se pueden encontrar las siguientes funciones o rutinas principales:

3.2.1 Bootloader

- **loadSector:** Esta rutina es la encarga de colocar en memoria RAM el Sistema Operativo creado así como su filesystem también.
- **print:** Rutina utilizada con el fin de imprimir cualquier mensaje necesario en pantalla, así mismo, todas las funciones cuyo nombre sea el ya establecido serán utilizadas para lo mismo.

3.2.2 OS

- **clearScreen:** Rutina cuyo único objetivo es limpiar toda la pantalla de cualquier caracter que se encuentra en ella.
- **moveCursor:** Rutina encargada de situar el cursor en las coordenadas especificadas por el usuario.
- **readExecutableName:** Encargada de leer todo lo que el usuario digite en la pantalla para su posterior verificación.
- **validateExecutableName:** Verifica si lo digitado por el usuario corresponde a un programa válido el cual pueda ejecutar el Sistema Operativo.
- **findExecutableSector:** En caso que el programa digitado por el usuario sea uno válido es la rutina encargada de buscar en cual sector

del dispositivo USB se encuentra el mismo para su posterior carga en memoria RAM.

- **waitKey:** Detiene la ejecución del programa y permanece a la espera que el usuario digite cualquier tecla.

3.2.3 Tutormec

- **gamePrincipalLoop:** Es el ciclo principal del juego, encargado de llamar a refrescar todos los datos que aparecen en pantalla así como de llamar a mover las letras y el mensaje inferior.
- **exitGame:** En el caso que el jugador presione la tecla "esc" es el encargado de volver a la dirección en la que se encuentra el Sistema Operativo.
- **init:** Es el encargado de mostrar la pantalla de inicio del juego así como de inicializar las variables (arreglos) necesarios para la ejecución del mismo.
- **refreshScreen:** Rutina principal encargada de refrescar todos los datos visibles en pantalla.
- **detectKeyPress:** Detecta si una tecla ha sido presionada para guardarla en un buffer de teclas y de esta forma saber cual letra en pantalla debe ser capturada.
- **intToStringConversion:** Rutina encargada de convertir un numero entero sin signo a una cadena de caracteres con el fin de poder imprimirla en pantalla.
- **drawStartScreenBackground:** Dibuja el fondo de la pantalla de bienvenida.
- **startScreen:** Muestra la pantalla de bienvenida principal del juego.
- **drawLettersBlock:** Dibuja el fondo blanco/gris donde se estan movimiento las letras.
- **drawImpactBlockBackground:** Dibuja el fondo rojo donde se encuentran los bloques de impacto.
- **drawImpactBlock**: Dibuja los bloques negros de impacto.

- **drawHandBackground:** Dibuja el fondo color cian de la parte inferior de la pantalla.
- **drawBlock:** Encargada de dibujar un bloque de cualquier tamaño y en cualquier posición de la pantalla.
- **hideCursor:** Esconde el cursor en la pantalla para que no sea visible.
- **verifyCapturedLetter:** Verifica si se ha indicado a una letra que sea capturada (comience a descender).
- **moveMessage:** Rutina encargada de mover el mensaje que se encuentra en el inferior de la pantalla.
- **moveLetters:** Rutina encargada de mover las letras que se encuentran en la parte superior de la pantalla.
- **makeADelay:** Rutina encargada de realizar un retraso especificado por el usuario en el programa con el fin que las letras no sean actualizadas tan rápidamente y se observe un movimiento fluido.

4 Instrucciones para ejecutar el programa

Nota: Se asume que el usuario ya tiene instalado el compilador GCC así como también el ensamblador NASM, además de encontrarse en un ambiente Linux.

1. Conecte el dispositivo USB a un puerto USB válido.
2. Realice un borrado de bajo nivel al dispositivo USB conectado, esto con el fin de limpiar cualquier posible basura que pudiera tener dentro.
3. Abra una consola y desplácese hasta la carpeta que contiene los archivos .asm, .c y el Makefile.
4. Ejecute en la consola la instrucción "fdisk -l" y busque la dirección del dispositivo USB conectado.
5. Abra el archivo "Makefile" con su editor de texto favorito y modifique la instrución "dev = /dev/sdb" colocando despues del guión la dirección de su dispositivo previamente buscada.
6. Guarde y cierre el archivo Makefile.
7. Diríjase a la consola nuevamente y digite la instrucción "make".
8. Espere que termine de ejecutarse hasta que indique que el dispositivo USB se encuentra listo.
9. Reinicie su computador y configure el booteo para iniciar desde el dispositivo USB.
10. Reinicie nuevamente.

5 Actividades realizadas por el estudiante

- **24/07/15 - 5 minutos:** Buscar el ensamblador NASM.
- **24/07/15 - 2 horas:** Buscar un libro que me ayudara con el lenguaje ensamblador.
- **24/07/15 - 3 horas:** Como crear un bootloader en ensamblador, así como también el funcionamiento del mismo.
- **24/07/15 - 2 horas:** Preparar un entorno de programación con herramientas que me facilitaran el probar nuevo código escrito.
- **25/07/15 - 23 horas (6:00 a.m. to 5 a.m.):** Iniciando el aprendizaje de ensamblador.
- **26/07/15 - 8 horas:** Retomar el aprendizaje de ensamblador.
- **27/07/15 - 2 horas:** Retomar el aprendizaje de ensamblador.
- **28/07/15 - 12 horas (3:00 p.m. to 3:00 a.m.):** Iniciar la construcción de Tutormec.
- **29/07/15 - 10 horas (6:00 p.m. to 4:00 a.m.):** Comprender a fondo el uso de los registro del procesador, pues seguía sin tenerlo claro, así como también el uso de las distintas funciones aritméticas que ensamblador ofrece.
- **31/07/15 - 12 horas (7:p.m. to 7:00 a.m.):** Continuar con la construcción de Tutormec.
- **01/08/15 - 11 horas con 30 minutos (3:00 p.m. to 2:30 a.m.):** Se termina la construcción del programa Tutormec así como el software en general.

6 Comentarios finales (estado del programa)

6.1 Estado final del programa

El programa cumple con el 100% de los requerimientos técnicos y funcionales requeridos.

6.2 Problemas encontrados

Se encuentran una diversidad de problemas, se exponen algunos cuantos de ellos, principalmente los más significativos.

- **Cantidad de sectores cargados:** Cuando se inicia la construccón del programa "Tutormec" pesaba menos de 512 bytes, sin embargo, con el tiempo fue creciendo hasta superar los 512 bytes, en el archivo del Sistema Operativo se indicaba que debía cargar un sector únicamente, dado que el archivo pesaba más que eso al cargar sólo uno todo se corrompía y nada servía.

- **Multiplicación:** Al ejecutar la instrucción "mul X" esta realizada a lo interno una multiplicación de "X" por "al", sin embargo, en las **TRES** páginas que investigué todas indicaban que la multiplicación se realizaba entre "X" y "dh" por tanto es claro que jamás iba a servir, no fue hasta que leí el manual de Intel que me enteré del error.

- **Salto que no se ejecutaba:** En determinado segmento del código se tenía una instrucción del estilo "cmp X, 0" seguida de "inc Y" y "je segmento", por alguna razón que desconocía a pesar que estaba **SEGURO** que la comparación daba positiva el salto no se ejecutaba y pasaba de Ã©l como si nada... Llevó varias horas de trabajo identificar que la instrucción "inc" modificaba el flag "ZF" a "0" en caso que al realizar la incrementación el resultado no fuera cero, en otras palabras la instrucción "cmp" colocaba el flag a uno pero luego "inc" lo colocaba a cero y por tanto "je" no se ejecutaba.

- **Falta de registros:** Dentro del programa en muchas ocasiones se tuvo que realizar direccionamiento indexado, para ello utilizaba el registro "bx" el cual permite hacerlo, sin embargo, en muchas ocasiones necesitaba direccionar a más de un lugar a la vez y pude hacerlo usando la pila y "bx", pero en un determinado momento tenía la pila con datos importantes y el registro "bx" ocupado, necesitaba otro registro y no sabía que hacer, fue hasta que investigué más a fondo los registro del

procesador que me enteré que tanto el registro "si" como el registro "di" también permitian realizar direccionamiento, un mundo de posibilidades se abrió.

- **El programa se detiene:** En muchas ocasiones y de forma totalmente esporádica el programa se detenía sin motivo alguno, la pantalla se congelaba y nada servía, similar al "pantallazo azul de la muerte", tomó bastante tiempo darme cuenta que en una función que usaba con regularidad si se daban "ciertas" condiciones el flujo se iba por cierto lugar donde realizaba un "push" a la pila pero en algunas ocasiones el flujo se salía sin pasar por donde se realizaba el "pop", por tanto cuando se salía de la rutina mediante el "ret" lo que sacaba de la pila era lo que yo había introducido con el "push" y no la dirección de retorno, problema bastante difícil de detectar.

- **Posicion 0,11 como zona de impacto**: Esta más que claro que la posición X = 0 **NO** puede ser una posición de impacto pues en esa zona no se encuentra ningún bloque de impacto, sin embargo, cuando bajaba una letra por esa zona se detectaba como un choque, lo cual no tenía sentido alguno, detectar el error fue bastante difícil, en algún momento del código se realiza una comparación con un arreglo para saber si ya ha validado todas las posiciones de impacto existentes, el arreglo tiene un tamaño de 30 posiciones de impacto, es decir, si llega a la posición 30 y no detecto ningún impacto es porque no lo hubo, sin embargo, en la comparación se verificaba contra 31 y no contra 30, lo que ocasionaba esto era que el puntero del arreglo se devolviera al inicio del mismo detectando así la posición X = 0 como una zona de impacto.

6.3 Limitaciones adicionales

Ninguna.

7 Conlusiones y recomendaciones del proyecto

7.1 Conclusiones

- Ensamblador es un lenguaje muy difícil de aprender pero entretenido de programar.
- El acceso a disco puede ser complicado sino se está seguro de donde están los datos.
- Un sólo byte de menos o de más en el bootloader puede ocasionar que el BIOS no lo reconozca como uno válido.
- Un byte de más en un registro puede ocasionar que se corrompa el sistema completo.

7.2 Recomendaciones del proyecto

- No pierda el tiempo, empiece a trabajar desde el primer día que le entregaron la asignación o no le dará tiempo suficiente para terminarlo.
- Tenga cuidado con respecto a donde escribe los datos de sus archivos .bin, una mala asignación del dispositivo puede ocasionar que los escriba en su disco duro principal corrompiendo el MBR.
- Lea mucho código ajeno, entiendalo hasta la última línea, si a mí me funcionó puede funcionarle a usted.
- Monte un entorno de trabajo, no piense en reiniciar la computadora cada vez que quiera probar algo, pues eso además de ser bastante tedioso le hará perder mucho tiempo valioso.
- Guarde todas las páginas que visita las cuales tengan información valiosa que le fue de utilidad, puede que la necesite luego una vez más y así no perderá tiempo volviendola a buscar.
- Raye, y raye mucho, tome lápiz y papel y haga ejecuciones "a pie", es una de las mejores maneras de entender por donde se está yendo su flujo de datos.

8 Bibliografía

Alpern, D. A. (02 de 08 de 2015). Sitio Web de Dario Alpern. Recuperado el 25 de 02 de 2015, de Sitio Web de Dario Alpern: http://www.alpertron.com.ar/INST8088.HTM

Appusajeev. (02 de 08 de 2015). Curiosity, Experimentation. Recuperado el 25 de 07 de 2015, de Curiosity, Experimentation: https://appusajeev.wordpress.com/2011/01/27/ writing-a-16-bit-real-mode-os-nasm/

Desconocido. (02 de 08 de 2015). La Web de Sistemas Operativos . Recuperado el 26 de 07 de 2015, de La Web de Sistemas Operativos : http://labsopa.dis.ulpgc.es/prog_c/FICHER.HTM

Desconocido. (02 de 08 de 2015). MasterHacks. Recuperado el 01 de 08 de 2015, de MasterHacks: http://masterhacks.net/programacion/instrucciones-para-ciclos-loop-en-ensamblador/

Desconocido. (02 de 08 de 2015). Viralpatel.net. Recuperado el 30 de 07 de 2015, de Viralpatel.net: http://viralpatel.net/taj/tutorial/hello_world_bootloader.php

Ferreira, A. (02 de 08 de 2015). Aprendiendo Ensamblador. Recuperado el 25 de 07 de 2015, de Aprendiendo Ensamblador: http://learnassembler.com/log.html

Hurtado, A. V. (02 de 08 de 2015). abreojosensamblador.net. Recuperado el 31 de 07 de 2015, de abreojosensamblador.net: http://www.abreojosensamblador.net

Hyde, R. (02 de 08 de 2015). The Art Of Assembly. Recuperado el 25 de 07 de 2015, de The Art Of Assembly: http://www.plantation-productions.com/Webster/

Kilian. (02 de 08 de 2015). es.ccm.net. Recuperado el 31 de 07 de 2015, de es.ccm.net: http://es.ccm.net/faq/3284-compilar-un-programa-ensamblador-con-nasm

Mike. (02 de 08 de 2015). MikeOS. Recuperado el 28 de 07 de 2015, de MikeOS: http://mikeos.sourceforge.net/

Perez, L. U. (02 de 08 de 2015). Instituto Tecnologico de Tuxtepec. Recuperado el 29 de 07 de 2015, de Instituto Tecnologico de Tuxtepec: http://www.ittux.edu.mx/sites/default/files/MICROCOMPUTADORAS_AL_DETALLE.pdf