TP FINAL
Organización del Computador
"Juego OC Starship"

ASIGNATURA: Organización del computador
COMISIÓN: COM-03 (TURNO TARDE)
INTEGRANTES:
+ Manganiello Nereo Valentin (LEGAJO:44097141)
+ Oviedo Wanuffelen Luciana Daniela (LEGAJO: 44584069)
+ Rauschenberger Facundo Agustín (LEGAJO: 44788532)
FECHA DE ENTREGA: 14/06/23

INTRODUCCIÓN

En este trabajo final, nos adentraremos en el mundo del lenguaje ensamblador ARM, explorando sus fundamentos, estructuras de control, acceso a memoria y llamadas al sistema. A través de la codificación de un juego reconoceremos cómo el ensamblador permite el control directo sobre el hardware, brindando un nivel de optimización y personalización que va más allá de los lenguajes de alto nivel.

Durante nuestro recorrido, examinaremos las características clave de la arquitectura ARM, incluyendo su conjunto de instrucciones reducido, sus registros especializados y su modo de direccionamiento flexible. Demostrando los conocimientos aprendidos en clases tales como manipulación de bits, operaciones aritméticas y lógicas, así como también cómo interactuar con dispositivos periféricos y manejar interrupciones.

La herramienta que utilizaremos para la programación del juego es 'Putty' cuyo finalidad es compilar, depurar y ejecutar código ensamblador, permitiéndonos desarrollar y probar eficientemente nuestro proyecto.

DESARROLLO

Durante el desarrollo del trabajo existieron complicaciones con respecto al tiempo destinado al trabajo y llevar a cabo el resto de materias.

A la vez de enfrentar un nuevo lenguaje, en este caso de bajo nivel, en el cual es la primera vez que trabajamos y que es conocido por ser complicado a la hora de operar con él. Pero con ayuda de la teoría y del equipo pudimos finalizarlo. Los integrantes estuvimos en las mismas condiciones tanto de tiempo, como de capacidad y conocimiento.

Una problemática recurrente era el "mal" manejo de los registros tales como 'violacion de segmento'. En estos casos no queda más que buscar el error debugging o llevar a cabo otra solución. Por ejemplo, a la hora de querer reiniciar el juego, no lograbamos borrar los asteroides que quedaban en la pantalla tras finalizar la partida. Para solucionarlo tuvimos que implementar una subrutina **borrarc** que recorre la pantalla (que es una cadena establecida como *planeta*) desde la primera posición de la primer línea jugable e ir obteniendo carácter a carácter con un índice, si se detecta que es un asteroide, se reemplaza por un espacio en blanco. Aparte de ello, en el **main** implementamos otra subrutina que se encarga de restablecer todas las variables con sus datos originales.

A continuación mostraremos las subrutinas fundamentales tales como los movimientos de la nave y de los esteroides en forma de explicación.

Antes de los movimientos de la nave, necesitamos saber qué dirección quiere tomar el usuario. En el juego la nave se moverá con las clásicas teclas WASD, por lo mismo debemos leerlas y para ello usamos la subrutina **detectar_tecla**. Esta información se almacenará en un registro para luego hacer comparaciones tanto con mayúsculas como minúsculas.

A parte de las las teclas nombradas se le suman las teclas Q para dejar de jugar y la tecla P para volver a jugar.

Suponiendo que el usuario haya seleccionada 'd' por lo tanto la nave se moverá a la izquierda. ¿Pero cómo lo hace? pues en la subrutina **derecha** debemos tener cargada la

posición actual, que es un valor predeterminado y vamos sumando algún valor para crear el efecto de movimiento. A la vez, a la posición anterior la reemplazamos con un espacio vacío. Es de mucha ayuda las instrucciones de transferencia de datos push y pop.
Con esta misma lógica se confeccionaron los demás movimientos:
En la subrutina **izquierda** se resta a la posición actual.
En la subrutina **abajo** se suma a la posición actual.
En la subrutina **arriba** se resta a la posición actual.

Por otro lado, los esteroides tienen sus debidas cadenas, (tres distintas) para que las mismas se vayan alternando en su aparición, y posiciones. Aunque este último se modifica constantemente durante la partida, puesto que su función es que se mueven de abajo hacia arriba con el objetivo de colisionar con la nave. Y el objetivo del usuario es escabullirse por los espacios generados por estos y no perder vidas, o al menos no todas.
Una de las subrutinas empleadas en ellos, es **imprimir_esteroides** en ella cargamos el planeta y con ayuda de los registros y ciclos cargamos caracter por caracter en el planeta ¿como lo hacemos? pues lo hacemos ocupando espacios de la cadena *planeta* por la de asteroides. Lo logramos recorriendo la cadena de asteroides con un índice de inicio y el índice del largo de la cadena, luego por medio de comparaciones se acabará cuando el contador sea igual al largo de la cadena del esteroide.
Y para moverlos usamos una lógica parecida, implementando una subrutina **mover_asteroide** en cada iteración del ciclo en ella recorremos toda la cadena planeta buscando los asteroides, representados por asteriscos '*', cuando encuentra uno se resta un número inmediato a la posición original para dar la ilusión de movimiento hacia arriba (tal cual como lo hacemos con la nave), luego reemplazamos por un espacio vacío en la posición anterior. Luego de que se reemplazó el asterisco, suma uno al contador del índice para que se pueda recorrer la cadena.
Además de ello estas cadenas deben "desaparecer" tras llegar a la cima de la pantalla, para luego "aparecer" de nuevo. Lo hacemos por anidamientos de subrutinas y por

medio de comparaciones, estando en la subrutina **mover** cuando encontremos el carácter '_' que representan los límites de la pantalla vamos a dirigirnos a la subrutina **reinicio** donde nos lleva a una subrutina aparte. A esta última la nombramos **ultima_fila**, aquí lo que hacemos es usar registros para seguir el movimiento de los esteroides. Es decir, cuando el índice de inicio y el índice de fin sean iguales significa que llegamos al fin del recorrido de los esteroides y para ello se finaliza la rutina, aquí también debemos colocar en la posición anterior de los esteroides con un espacio vacío.
En caso de colisionar la nave con los asteroides va a restar una vida, durante la partida se cuenta con un total de tres y la posibilidad de obtener una extra. El objetivo es que llegue a la superficie de Argos antes de perder las vidas disponibles. En caso de perder todas las vidas se dará el juego por terminado y como consecuencia como perdido. Por otro lado, si el usuario llega con al menos una vida también terminará y se dará por ganado el juego.

CONCLUSIÓN

En resumen, el proyecto en lenguaje Assembler realizado en grupo resultó complicado principalmente por llevarlo a cabo en un nuevo lenguaje. Sin embargo, a pesar de los desafíos, logramos completarlo exitosamente dentro del plazo establecido.

La experiencia de trabajar en este proyecto nos enseñó importantes lecciones sobre este nuevo lenguaje que seguiremos explorando en próximas materias y la gestión del tiempo. Aprendimos a enfrentar y superar nuevos desafíos, adaptándonos a las circunstancias y tomando decisiones para alcanzar nuestros objetivos.

UBICACIÓN DEL TRABAJO: ~/TpOrga $ ./tpfinal