Proyecto TETRIS - Grupo 2

Angel Nahum Merino Ortega
Rodrigo Gael Guzmán Alburo

Introducción:

El proyecto consiste en realizar un juego de Tetris en lenguaje ensamblador. El juego de tetris debe funcionar como cualquier otro, pero además debe contar con un botón de stop, uno de pausa y otro de play. Al igual que el juego debe contar las líneas que el usuario logra completar, su puntaje, el puntaje máximo alcanzado, el nível en el que se encuentra dentro del juego y la pieza siguiente que caerá al tablero. Como base del proyecto se utilizó un código de ensamblador, creado y repartido por el profesor Luis Sergio Durán Arenas, donde se tenía ya implementada la interfaz gráfica del juego, una gran cantidad de macros y procedimientos tanto para la activación del juego, funcionamiento de botones, establecer límites del tablero y dibujar las distintas formas de las piezas dentro del tablero.
Se usaron conceptos básicos del lenguaje ensamblador, como son los distintos tipos de operaciones, saltos relativos, declaración de procedimientos y una variedad de interrupciones para distintos propósitos.

Desarrollo:

Lo que le hace falta al archivo base proporcionado por el profesor es lo siguiente:

- Funcionamiento de los botones
- Movimiento de las piezas hacia abajo en un cierto intervalo de tiempo
- Movimiento de las piezas con el teclado
- Establecer las colisiones entre las piezas y los bordes del tablero
- Contar tanto las líneas generadas, la puntuación del usuario y el nível en el que se encuentra dentro del juego
- Perder el juego si la pieza llega hasta arriba del tablero

Explicación funciones principales:

Para implementar muchas de estas funcionalidades se utilizarán coordenadas del tablero, de manera similar a como se utilizan para dibujar las piezas y gran parte de la interfaz. Tanto el funcionamiento de los botones, el movimiento de las piezas y las

colisiones funcionan en gran parte con las coordenadas, almacenando algunas, sumando, multiplicando, etc. Incluso con las puntuaciones se toman en cuenta las coordenadas, aunque no para el cálculo de estas, sino que para escribir en la pantalla se utilizan las coordenadas para saber exactamente donde mostrarlas.

- Funcionamiento Botones:

Dentro del código base del proyecto se tiene un ejemplo de cómo funciona el botón [x], utilizado para salir de la ejecución del juego. En este ejemplo, primero se realiza la conversión de la posición del mouse a la resolución de la pantalla de juego (80x25). Una vez teniendo la posición del mouse ajustada, se procede a verificar si se presiona el botón izquierdo del mouse. Para esto se hace uso de la interrupción 33h con AX=0003h, donde podemos verificar la posición del click con los registros CX y DX. Teniendo esta información podemos verificar si se hizo un click en algún botón a partir de sus coordenadas. Haciendo uso de saltos condicionales ‘jg’ y/o ‘jl’ se puede establecer un intervalo tanto para columnas (CX) y los renglones (DX) y confirmar si se hizo click en alguno de los 4 botones disponibles.

- Movimiento de las piezas hacia abajo

Si se confirma que se hizo click en el botón play, dentro del código se realiza un salto incondicional a la etiqueta ‘empieza:’. Dentro de esta etiqueta se realiza el llamado al procedimiento ‘PIEZAMBAJA’. Este procedimiento busca la pieza que se encuentra más bajo en vertical, es decir, cual es el renglón más bajo de la pieza, este valor se almacena en el registro AH. De este punto se retorna a la etiqueta ‘empieza’, donde se obtiene la diferencia entre el punto más bajo de la pieza y ‘lim_inferior’, para saber la distancia restante que debe bajar la pieza, esta distancia se almacena en el registro CX para posteriores comprobaciones.
Se entra a la etiqueta ‘uptfps’, donde la acción siguiente es verificar si no hay una pieza abajo de la actual, es decir, si hay una colisión, para esto se llama al proc ‘BUSCAABAJO’. En este procedimiento se marca a BX=1 si es que existe una colisión, de lo contrario BX=0. En el caso de que no haya una colisión, se llama a ‘DIBUJA_UPDT’, el cual se encarga de redibujar la pieza con un desplazamiento abajo. Se mueve a una variable llamada ‘dkb’ el valor de 50, que son las veces que se actualiza el programa para saber si se presionó una tecla.

Posteriormente se verifica si se hizo click en el botón de stop o de pausa, siguiendo el mismo procedimiento para detectar actividad en los botones. Si no se detecta actividad en el mouse, se hace un salto a la etiqueta ‘teclado’ para verificar actividad en el teclado. Una vez se terminan las acciones del teclado, se realiza la espera de 16 milisegundos. Una vez se completa el tiempo se procede a repetir el procedimiento dentro de la etiqueta ‘repite:’.
Para la repetición, primero se resta una unidad al registro CX, nuestro valor de la distancia que la pieza debe recorrer, y se compara con 0, si el valor resulta mayor, entonces se hace un salto a la etiqueta ‘uptfps’ para que la pieza llegue al fondo del tablero. Si CX = 0, esto significa que la pieza llegó al fondo, por lo tanto se llama al procedimiento ‘GUARDAPIEZA’ para almacenar su posición y tenerla en cuenta en futuras colisiones. Después se procede a verificar si se ha completado una línea con la pieza que acaba de llegar al fondo, para esto se llama al procedimiento ‘BUSCCAM’. Luego de esto se llama a ‘REDIBUJA’, en donde se redibuja el tablero una posición más abajo por si se elimino una fila. Se procede a borrar la pieza actual para reemplazarla con la pieza indicada por ‘pieza_next’ y se calcula cuál será la siguiente pieza que se muestre en ‘next’, todo esto sucede dentro de ‘BORRA_PIEZA_ACTUAL’. El siguiente procedimiento a llamar es ‘BORRA_NEXT’, donde se elimina la pieza que se muestra en la sección de ‘next’. Para dibujar la siguiente pieza ‘next’ calculada en ‘BORRA_PIEZA_ACTUAL’ se realiza dentro del proc ‘DIBUJA_ACTUAL’.
Una vez terminado este proceso se realiza un salto a la etiqueta ‘empieza’, repitiendo así todo el proceso.

- Establecer las colisiones entre las piezas

El procedimiento ‘GUARDAPIEZA’ guarda las coordenadas de una pieza (renglón,columna) en dos matrices, ambas de 720 elementos. La primera matriz almacena la posición de cada pieza, y la segunda matriz almacena el color actual de la pieza, en las mismas posiciones.

- Movimiento de las piezas con el teclado

En el juego se pueden presionar las teclas ‘a’ y ‘d’ para mover la pieza hacia la izquierda o a la derecha. Las tecla ‘s’ para mover la pieza una posición hacia abajo y las teclas ‘q’, para hacer un giro a la derecha, y la tecla ‘e’, para hacer un giro a la

izquierda. Dependiendo de la tecla presionada se hace el llamado a distintos procedimientos, ‘IZQ’ y ‘DER’ para mover la pieza horizontalmente, ‘GIRO_DER’ y ‘GIRO_IZQ’ para realizar ambos giros, y finalmente si se presiona la tecla ‘s’, solamente se le resta un 2 a la variable ‘dkb’, para bajar la pieza dos posiciones. Una vez se termina de realizar las acciones presionadas por las teclas ingresadas por el usuario, o si no se presiona tecla alguna, se procede a borrar el buffer del teclado para evitar movimientos extras. Esto se logra usando la interrupción 21h con la opción 0Ch.

- Tiempo de espera entre movimientos de pieza

Para lograr que el programa espere cierto tiempo para que una pieza baje, se hace uso de la interrupción 15h con opción 86h. Esta interrupción detiene la ejecución un cierto tiempo en microsegundos. Para definir los microsegundos se utilizan los registros CX y DX, donde CX es la parte alta y DX la parte baja del dígito. Dentro del programa se hace una espera de 16 milisegundos estableciendo los valores de CX = 0000h y DX = 3E80h.

- Eliminar las líneas formadas

Para saber si una línea está completamente formada usamos la función BUSCCAM la cual busca en todos los renglones que están llenos. Para saber si hay una pieza guardada en cada columna del renglón se usa la función BUSCAPOS la cual devuelve en bx 1 si se encontró un dato o 0 si no se encontró nada, para saber si un renglón está completo se repite la búsqueda 30 veces ya que son 30 columnas por renglón y con un acumulador en este caso AX si llega al valor de 30 realiza la eliminación con la función ELIMINA, para esto en di se guarda la posición inicial de las columnas. En la función ELIMINA se recorre la matriz 30 posiciones con ayuda de di que indica donde inicia la columna que está llena. Para lograr mostrar el cambio en pantalla después de eliminar las líneas debe llamar a REDIBUJA para que vuelva a dibujar el área de juego con ayuda de la matriz.

- Girar la pieza con el teclado

Para girar una pieza a la izquierda se presiona la tecla ‘q’, y para girar a la derecha la tecla ‘e’. Ambas funcionan con un mismo procedimiento, ‘REVISAG’, aunque cada una tiene su procedimiento intermedio, ‘GIRO_DER’ y ‘GIRO_IZQ’. Ambas tienen un

funcionamiento muy similar, ambas usan la variable ‘giro’ y ‘giro_aux’. En ‘GIRO_DER’, ambas variables tienen un valor de 1, y en ‘GIRO_IZQ’ ambas tienen un valor de 3. Esto para distinguir qué tipo de giro se realizará dentro de ‘REVISAG’. En este procedimiento lo primero que realiza es buscar y guardar las posiciones de los extremos horizontales, es decir, la posición más a la izquierda y la posición más a la derecha de la pieza a girar. Después se utiliza el procedimiento ‘BORRA_PIEZA’ para eliminar la pieza actual del tablero. Una vez eliminada, se buscan las posiciones más altas y más bajas de la pieza, esto para identificar qué tipo de pieza es. Esto se logra restando la parte más baja menos la parte más alta, si el resultado da 1, significa que la altura de la pieza es de 2, para invertirlo solo tenemos que invertir la pieza y girar los ejes, si el resultado da 2, significa que la altura de la pieza es de 3, y si da un resultado distinto, entonces la pieza solo se debe de girar. El proceso para girar es prácticamente hacer las operaciones para rotar una matriz de 4x4. Posteriormente se tienen bloques de código para evitar que las piezas traspasen los límites o colisiones y que los renglones se mantengan en [pieza_rens] y las columnas en [pieza_cols]. Primero se realiza lo último mencionado, los renglones pasan a ser columnas, y las columnas pasan a renglones. Posteriormente se dibuja la pieza teniendo precaución que no se traslape en los marcos del tablero ni las otras piezas, si eso sucede, se suma un valor a renglón o columna según sea el marco o pieza que sobrepasa.

- Contar líneas generadas, puntuación y puntuación generada máxima del usuario

Para contar y mostrar todas las puntuaciones y las líneas generadas, se hace uso del procedimiento ‘BUSCCAM’, el encargado de eliminar las líneas del tablero. En este, al final, se incrementa en uno al contador ‘lines_score’ y se añaden 10 puntos a la variable ‘hiscore’. Cada vez que se genera una línea se añaden 10 puntos. Posteriormente se llama al procedimiento ‘SUBIR_NIVEL’, en este se verifica cuantos puntos se han generado. Para subir de nivel se necesitan 20 puntos, por lo que en ‘SUBIR_NIVEL’ se comprueba que los puntos sean un múltiplo de 20, si esto es cierto, se incrementa en uno el contador ‘level’ y además se disminuye el tiempo en que bajan las piezas, aumentando la dificultad del juego.

- Se pierde el juego si las piezas llegan hasta arriba del tablero

Para lograr esto se utiliza el procedimiento ‘PERDIO’, su funcionamiento es sencillo. Se utiliza el procedimiento ‘BUSCAABAJO’, donde se busca si abajo de la pieza actual generada existe una pieza. Como el procedimiento ‘PERDIO’ se llama justo al momento de llamar al procedimiento ‘DIBUJA_ACTUAL’, esto verifiqua la condición de perder en el tetris.

Diagramas de flujo:

Diagrama de flujo al ejecutar el programa:

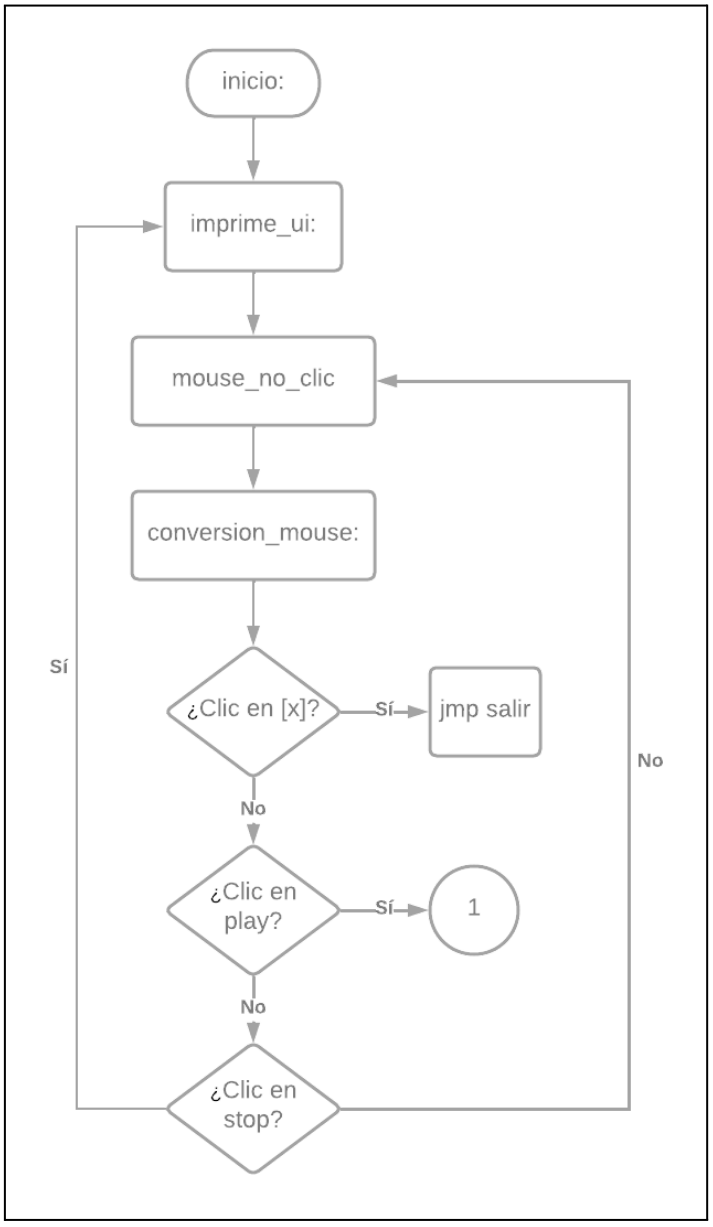
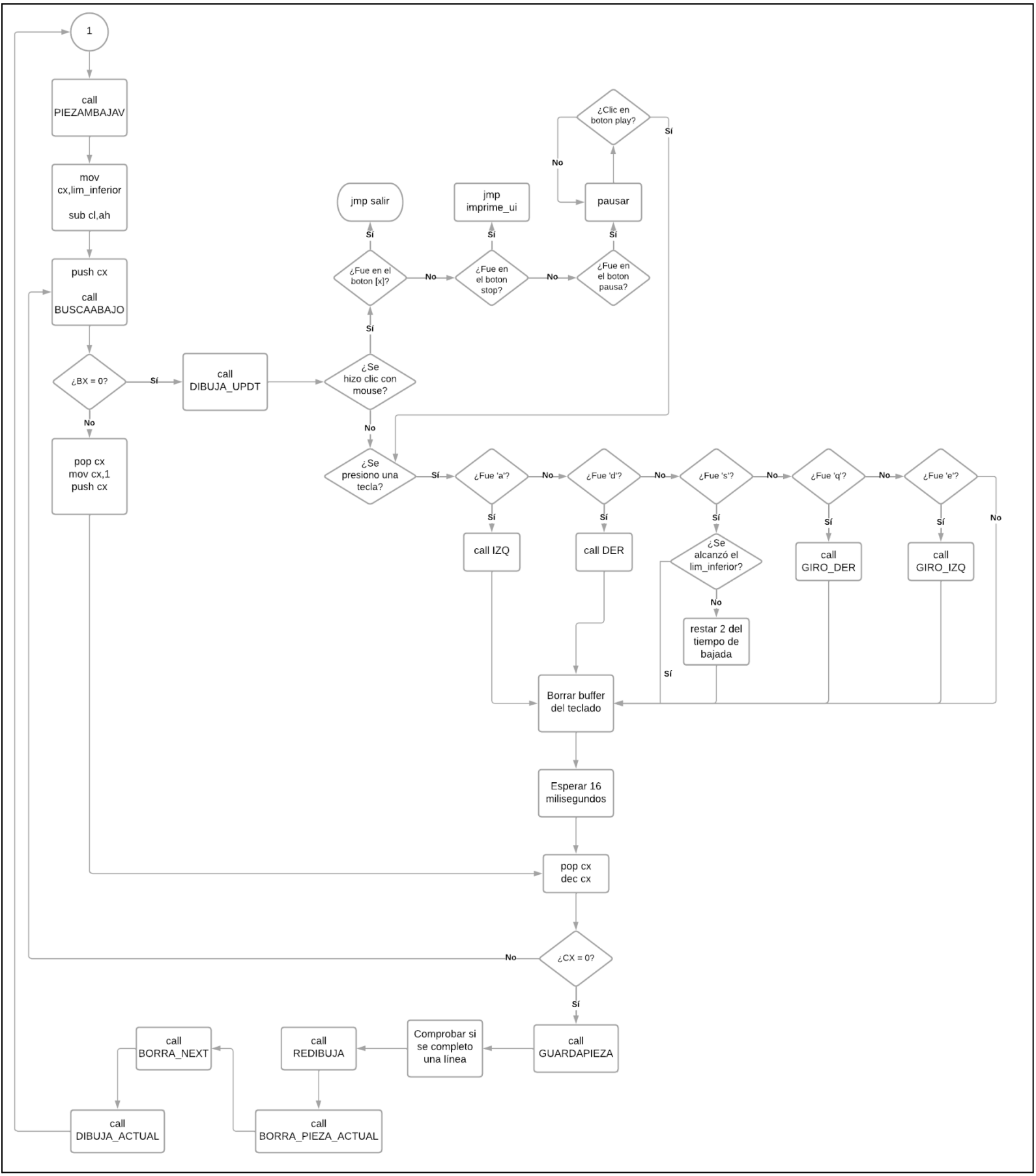


Diagrama de flujo al presionar play por primera vez:



Prueba de escritorio:

Se hará la prueba de escritorio en el caso de que se ejecutó el programa, pero aún no se ha hecho clic en ningún botón. Se realizará todo el proceso marcado por ambos diagramas presentados en este documento.

Se marcará por números cada uno de los pasos seguidos por el programa.

1. Se presiona el botón play
2. call PIEZAMBAJAV
3. AH = 2
4. CX = 23
5. CL = 23 - 2 = 21
6. push CX, CX = 0021h
7. call BUSCAABAJO
8. bx = 0
9. call DIBUJA_UPDT
10. No se hizo clic con el mouse
11. Se presiono la tecla ‘a’
12. call IZQ
13. (Se dibuja la pieza con un desplazamiento de 1 a la izquierda)
14. Borra buffer del teclado
15. Espera 16 milisegundos
16. pop CX, CX = 0021h
17. dec CX, CX = 0020h
18. CX ≠ 0
19. push CX, CX = 0020h
20. call BUSCAABAJO
21. bx = 0
22. call DIBUJA_UPDT
23. Se hizo clic con el mouse
24. Fue en el botón stop
25. jmp imprime_ui
26. mouse_no_clic
27. conversion_mouse
28. Clic en [x]
29. Salir

Conclusiones:

Angel Nahum Merino Ortega:

Durante la realización de este proyecto logré comprender muchos de los conceptos vistos en clase. Algunos de los conceptos más importantes requeridos para la realización del proyecto son: el uso de la pila, las interrupciones, las comparaciones lógicas, el uso de macros y procedimientos, el uso de diferentes tipos de saltos y los modos de direccionamiento existentes. Además, gracias a conceptos vistos en materias anteriores como EDA I, EDA II y fundamentos de programación me facilitaron la comprensión y realización del programa.

Lo más complicado de este proyecto a mi parecer es comprender el código base proporcionado por el profesor ya que aunque estaba bastante bien comentado requería comprender muy bien las macros, procedimientos y variables que eran usadas. Una vez logré comprender el código proporcionado se me hizo más fácil el pensar en cómo crear las funciones necesarias para la realización del proyecto. Otra de las cosas más complicadas durante la realización de este proyecto fue la eliminación de las filas ya que para esto se requiere un desplazamiento en una matriz lo cual fue algo complicado de realizar ya que tenía que ser un desplazamiento de 30 lugares y tenía que buscar la forma de saber como realizar este movimiento en la matriz evitando errores o salirse de la matriz causando que el programa se quede colgado y se detenga. Al final tuve que realizar 2 funciones una que realice el desplazamiento dentro de la matriz y otra que haga el cálculo de en qué dirección tenía que empezar el desplazamiento si es que estaba llena un renglón. También para lograr que se viera ese movimiento de la matriz en pantalla tuve que crear una función que limpie la pantalla y dibuje los cambios realizados.

Este proyecto fue razón de muchos desvelos y mucho tiempo pensando en como realizar las funciones y solucionar errores, pero aun así fue una grata experiencia. Me ayudó a comprender de mejor manera todo lo visto en clase y gracias a proyectos anteriores en diferentes lenguajes logré adaptarme a ensamblador más rápido de lo que esperaba. De forma personal siempre había querido intentar hacer un juego en algún lenguaje de programación, pero nunca tuve el tiempo o la

motivación adecuada para realizarlo. Por lo que este proyecto fue excelente y gracias a esto comprendí cómo sería programar alguno, siento que aprendí mucho y aun así me falta mucho más por aprender.

Guzmán Alburo Rodrigo Gael: En este proyecto pude poner en práctica todos los conceptos vistos durante el tema 2, al igual que en todos los ejercicios y tareas de codificación. Especialmente me ayudó para poner en práctica mi pensamiento abstracto, tanto para entender cómo se construyó el archivo base, como para ir añadiendo las distintas funciones del juego.El proyecto fue complicado de completar, Algo que se me complicó durante el proyecto fueron las colisiones entre piezas. En nuestro caso usamos una matriz de 24 renglones por 30 columnas para representar el tablero. Pero como en ensamblador no se pueden definir como tal matrices de dos dimensiones, usamos una matriz de 720 elementos, en donde cada 30 elementos representaba un renglón nuevo. Llegar a esta conclusión me tomó tiempo pero nos facilitó el implementar distintas funciones. Como lo fue eliminar la fila donde se creaba una línea. Esto fue sencillo de implementar teniendo la matriz de coordenadas, ya que solo se debe verificar que una fila esté llena de unos, es decir, que hay piezas en cada una de las columnas de dicha fila, pero esto traía complicaciones, ya que al eliminar la fila, las demás piezas debían de bajar, por lo que teníamos que realizar una traslación de la matriz. Otra funcionalidad del programa que a mí se me complicó fue el girar las piezas, esto lo realizó principalmente mi compañero, por lo que no fue problema para el proyecto en general, pero para mí comprendimiento sí. Se me complicó principalmente por todas las operaciones con los renglones y columnas que se realizan en nuestra implementación del giro, por lo que me costaba entender que se le hacía a qué renglón, o columna. Pero la forma en que finalmente logré entenderlo fue realizando varias pruebas de escritorio, para distinguir las variables y que valores iban tomando. En general el proyecto definitivamente fue un reto, tanto para mi conocimiento teórico y práctico, como para mi paciencia y habilidad para buscar información e ideas. Pero de igual manera fue entretenido programar un videojuego, muy sencillo y corto, pero de todas maneras me da una idea de la profundidad que algo tan sencillo puede tener al programarlo, y especialmente en lenguaje ensamblador. Una vez terminado me siento más preparado para hacer otros programas en mis próximas materias, y en mi tiempo libre también.