Proyecto #4 de diseño
Lenguaje Ensamblador x86-32

I. Competencias a desarrollar:

Diseñar un programa básico en lenguaje ensamblador del procesador x86-32 que implemente el juego asignado.

II. Condiciones y fechas de entrega:

- El proyecto se realiza en grupos de 4 personas seleccionadas por los estudiantes.
- **Entrega y presentación:** semana 20
- **Presentación:** deben estar presentes todos los integrantes del grupo y demostrar el funcionamiento de su programa**.**
- **Material a entregar en Canvas** *(ver detalles en la evaluación)*
    - Programas fuente (.asm)
    - Documento del diseño en formato PDF que incluye todo lo necesario para explicar el trabajo realizado:
        i. Especificación del uso de los registros
        ii. diagrama de flujo del programa hecho en una herramienta de software o algoritmo narrativo, ordenado y con la simbología correcta
        iii. Conclusiones
        iv. Bibliografía citada de forma correcta y de al menos 2 sitios confiables.

III. Instrucciones

Su proyecto consiste en elaborar un juego en lenguaje ensamblador x86-32. El programa debe cumplir los requerimientos de funcionamiento, programación defensiva y uso de subrutinas propias.

Se requiere construir una interfaz gráfica (GUI) para interactuar con el usuario utilizando caracteres ASCII, el objetivo principal es que su juego funcione y sea divertido. **Además, se solicita utilizar la técnica de "*ASCII art*" para realizar alguna gráfica alusiva al juego, que usa caracteres ASCII para dibujar, consulte en los sitios:**

- http://www.glassgiant.com/ascii/
- http://picascii.com/

Todos los juegos deben incluir un menú que permita:

- Ingresar al juego
- Brindar instrucciones
- Salir del juego

IV. Temarios:

1. **Tragamonedas.** Su programa será una máquina tragamonedas que genera tres dígitos al azar, en donde cada dígito puede ser un número de 1 a 8. La máquina tragamonedas inicia con 20 monedas en su depósito (esta cantidad puede ser modificada), y el jugador inicia con 10 monedas.

Pasos para jugar:

i. El jugador debe ingresar la cantidad de monedas que desea apostar (menor o igual a la cantidad de monedas que posee) y luego "girar la palanca" de la máquina.
ii. La máquina genera los tres dígitos al azar y los muestra al jugador.
   - Si le salen los tres dígitos iguales entonces el jugador gana el DOBLE de la cantidad de monedas apostadas o la cantidad de monedas que aún existan en el depósito de la máquina (lo que sea menor).
   - Si el jugador pierde, entonces sus monedas ingresan al depósito de la máquina.
iii. El juego termina cuando el jugador o la máquina se queda sin monedas. Gana quien tenga las monedas. Se debe mostrar un mensaje indicando si el jugador o la máquina ha ganado.

2. **Carrera con obstáculos.**

El objetivo del juego es llegar a la meta. La meta está localizada a 50 pasos de la línea de inicio.

Pasos para jugar:

i. El jugador lanza dos dados (máximo 6 veces) y avanza la cantidad de pasos indicados por los dados.
   - Si al tirar los dados ambos son iguales, el jugador retrocederá 10 pasos
   - No puede haber retrocesos que nos lleven antes de la línea de inicio - el límite es la línea de inicio.

- Si llega a la meta o la sobrepasa antes de 6 intentos de lanzamientos de dados, entonces cruza la meta, y gana el juego. Mostrar mensaje de que ha ganado.

3. Minesweeper

El objetivo del juego es despejar todas las casillas de un tablero de 4x4, evitando minas colocadas al azar. Se inicia con dos minas colocadas en posiciones ocultas y aleatorias.

Pasos para jugar:

i. El jugador selecciona una posición por turno y el tablero se refresca con un número en las posiciones contiguas a la selección que indica la distancia horizontal, vertical o diagonal hacia la mina más cercana.

2	2	1	
0	#	1	
2	1	0	

Tablero mostrando posición elegida y distancia a las minas

ii. Si la posición vecina a la elección tiene una mina, mostrará distancia 0.
iii. Si la selección cae en una casilla sin minas, se marcará la posición con (#).
iv. Pero si existe una mina en esa posición, se marca la posición con (X) y se acaba el juego.

			X
	#		
		X	

Tablero habiendo elegido una mina

.

v. Si el jugador descubre todas las casillas sin mina, gana el juego y se muestra un mensaje en pantalla.

4. Fuga de letras. Este juego consiste en completar las letras que faltan en la palabra que se muestra en la pantalla. Se tienen dos jugadores que van jugando por turnos y empiezan con un puntaje de 0.

Pasos para jugar:

i. En cada juego se muestran un total de diez palabras diferentes:
   - Si el jugador acierta con las letras que forman la palabra, ganará 5 puntos.
   - Si no cierta, se le restarán 2 puntos (tomar en cuenta que, si aún está en 0 puntos, se mantiene igual).

ii. Gana el jugador que haya obtenido más puntos.

Ejemplo de ejecución:

```
***JUGADOR 1***
Palabra: igl_sia
Ingrese la vocal: u
***ERROR! Puntos jugador 1: 0***

***JUGADOR 2***
```

```
Palabra: plan_ta
Ingrese la vocal: e
***BIEN! Puntos jugador 2: 5***
```

IMPORTANTE: su programa deberá tener un banco de 40 palabras que serán seleccionadas aleatoriamente.

5. Par-impar

El objetivo del juego es llegar al final del tablero de 10 posiciones seleccionando si el dado caerá en número par o impar.

Pasos para jugar:

i. En cada turno el jugador elegirá su opción: par o impar.
ii. Luego de la selección, el juego genera un número aleatorio entre 1 y 6.
    - Si la selección del jugador coincide con el resultado del dado, el jugador avanza a la siguiente casilla,
    - En caso contrario, retrocede una casilla.
iii. El juego termina exitosamente cuando:
    - El jugador alcanza la última posición y logra elegir el resultado correcto
    - O termina si el jugador retrocede a la posición 1 y elige el resultado incorrecto.

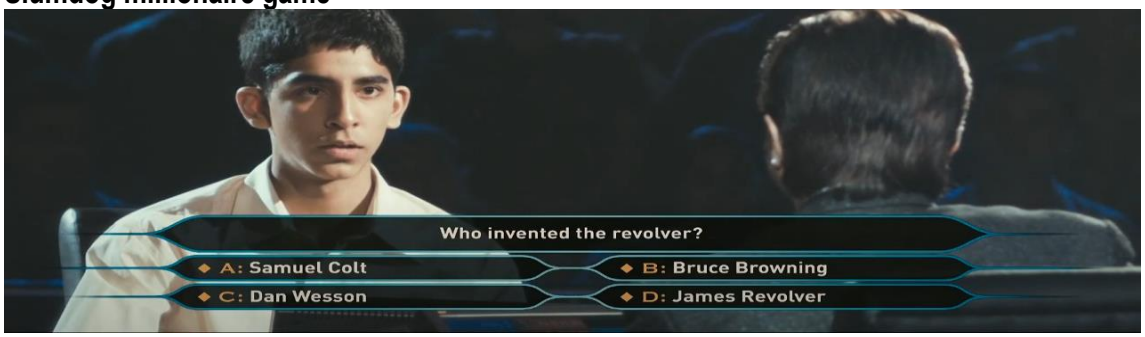
6. Salto de ranas. Este juego consiste en 7 posiciones, y 6 ranas, 6 ranas de la especie (b) están localizadas en la parte derecha y 3 ranas de la especie (a) están ubicadas en la parte izquierda de las posiciones, dejando así un único espacio vacío.

(a)	(a)	(a)	( )	(b)	(b)	(b)
(1)	(2)	(3)	(4)	(5)	(6)	(7)

El objetivo del juego es intercambiar las 3 ranas de cada lado, se debe tener en cuenta que:

i. Las ranas no pueden retroceder
ii. Solo pueden moverse un espacio a la vez
iii. Las ranas de distinta especie pueden saltarse otra rana si existe un espacio vació adelante
iv. Sin embargo, las ranas de la misma especie no pueden saltarse entre ellas.

7. **Slumdog millionaire game**



Contexto: https://www.youtube.com/watch?v=AIzbwV7on6Q
Diseñar y programar el funcionamiento de un juego de trivia similar al de la película Slumdog Millionaire. Usted tiene derecho a definir en memoria los siguientes parámetros:

- Cantidad de preguntas a realizar
- Preguntas a realizar
- 4 opciones de respuesta por pregunta
- Respuesta correcta
- Cantidad de dinero (En la moneda que desee) que el participante gana por pregunta, variar el monto según la dificultad de la pregunta.

Dar un mensaje de bienvenida y mostrar una pregunta, sus posibles respuestas y la cantidad que puede acumular a su cuenta si acierta.

Mientras el participante acierta una pregunta, la suma definida para esa pregunta se va acumulando a su cuenta y previo a responder la siguiente pregunta, se debe consultar si el participante desea retirarse con la suma actual o contestar a la siguiente pregunta para tener opción a ganar más dinero.

Si el participante responde bien a todas las preguntas realizadas, mostrar un mensaje de felicitación e indicar el monto que se ha ganado.

Si el participante falla una pregunta, mostrar un mensaje de juego terminado, indicar que se perdió la cantidad de dinero que llevaba acumulada, la pregunta que se falló, la respuesta seleccionada y la respuesta correcta.

Ejemplo de preguntas: https://www.youtube.com/watch?v=znj0_mrJaNg

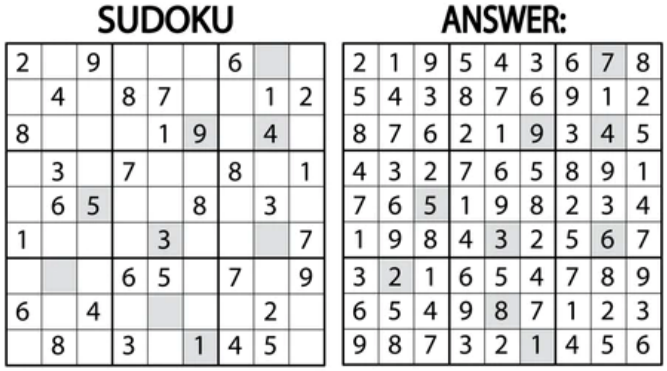
8. **Spotify simulation**

Diseñar y programar una simulación del reproductor de audio en línea Spotify. Para esto se deben de seguir al menos las siguientes funcionalidades:

- Opción de crear o eliminar canciones (para crear canciones debe de llenar los campos requeridos de cada archivo de audio).
- Cada archivo debe de tener:
    i. Nombre de artista o grupo
    ii. nombre de la canción.
    iii. Un pedazo de la letra que más le guste de la canción
    iv. Duración total.

- Lista general de canciones
- Contar con 2 listas de reproducción.
- Opción de agregar canciones a una lista o eliminarlas (Puede que haya canciones de la lista general que no estén asignadas a alguna lista de reproducción).
- Opción de seleccionar una lista de reproducción para poder visualizarla (Lista general o alguna de las listas que usted cree).
- Seleccionar una canción de la lista de reproducción actual y mostrarla como que se está reproduciendo (durante 30 segundos antes de pasar a la siguiente canción de la lista).
- Opción de parar la reproducción.

9. Sudoku



2		9				6		
	4		8	7			1	2
8				1	9		4	
	3		7			8		1
	6	5			8		3	
1				3				7
			6	5		7		9
6		4					2	
	8		3		1	4	5	

2	1	9	5	4	3	6	7	8
5	4	3	8	7	6	9	1	2
8	7	6	2	1	9	3	4	5
4	3	2	7	6	5	8	9	1
7	6	5	1	9	8	2	3	4
1	9	8	4	3	2	5	6	7
3	2	1	6	5	4	7	8	9
6	5	4	9	8	7	1	2	3
9	8	7	3	2	1	4	5	6

Diseñar y programar el funcionamiento de un juego de Sudoku.

- Se debe de contar con 3 opciones de tablero (Bajo, medio y alto nivel) y debe de mostrar en pantalla el estado actual del tablero.
    i. Puede rellenar los espacios del tablero vacíos o rellenarlos con “0”, “X“ o cualquier símbolo que le ayude a entender que el espacio está vacío y se logre mantener la forma del tablero.
    ii. Para identificar los números fijos, puede colocarlos entre corchetes [ ].
    iii. Utilizar cualquier otra impresión de líneas o puntos para que el tablero se vea más vistoso y ordenado.

- Debe de tener la opción de editar el valor de los cuadros del tablero escribiendo una coordenada y el valor que desea escribir.
- Tener la opción de salir del tablero hacia el menú principal de tableros sin la opción de guardar el avance.

- Si la coordenada contiene un número fijo, este no podrá editarse.Cuando se hayan llenado todos los espacios del tablero, preguntar al jugador si desea evaluar si el tablero se llenó correctamente para tomar una de las siguientes decisiones:
    i. Si el tablero se llenó correctamente, mostrar un mensaje de felicitaciones y volver al menú principal de tablero.
    ii. Si el tablero no se llenó correctamente, regresar a la pantalla en la que se está llenando el tablero.

10. Battlefield

Este temario consiste en la simulación del juego Battlefield.

- Para esto debe de poder imprimir 2 tableros con un rango de coordenadas horizontal de 1 a 9 y vertical de A a I.
- Ambos jugadores pueden seleccionar dónde colocar 3 barcos que ocupen 3 coordenadas de los 2 tableros disponibles (estas deben de estar juntas).
- Cuando el juego empieza, se van tomando turnos y cada jugador debe de seleccionar una coordenada para atacar las naves del tablero del oponente, se deben de imprimir 2 tableros extra para llevar el control de las coordenadas en las que cada jugador va atacando y gana quien termine de derribar todas las naves del oponente de primero.

11. Black jack

Este temario consiste en el diseño y simulación de un juego de black jack.

- Inicialmente, cada jugador cuenta con $3,000.00 y debe ingresar un monto a apostar.

- Seguidamente se seleccionan 2 cartas que quedan de manera oculta para la banca.
- Se entregan dos cartas al jugador.
- El jugador tiene la opción de solicitar más cartas para su mano.
- El objetivo del juego es derrotar a la banca al formar un número mayor que el que tiene, que no sea mayor a 21 o formar cualquier número si es que la banca se pasó de 21.
- Si el jugador forma un número mayor a 21, pierde automáticamente.
- Si el jugador gana, se le agrega el dinero de la apuesta que ganó y si pierde, se le descuenta.
- El jugador ya no puede seguir otra ronda si se queda sin dinero para apostar.

12. Acertijo del pastor.

Este temario consiste en el diseño de un juego de acertijos en el que un pastor tiene que atravesar a la otra orilla de un río con un lobo, una cabra y una lechuga. Dispone de una barca en la que solo caben él y una de las otras tres cosas. Si el lobo se queda solo con la cabra se la come, si la cabra se queda sola con la lechuga se la come. ¿Cómo debe hacerlo?

- Imprimir en pantalla el avance del juego.
- Cuando el acertijo se logre resolver, mostrar un mensaje de felicidades.

V. Evaluación

Cada miembro del grupo presentará una parte del proyecto, teniendo una nota individual, la cual servirá para **determinar el porcentaje de conocimiento que tenga de la totalidad de este**. *Es decir, si por ejemplo la nota obtenida en el proyecto es de 80 puntos, y el estudiante posee un 50% de conocimiento del código, la nota será de 40 puntos*.

Para acreditar la nota del proyecto es necesario: 1) entregar el informe y el programa realizado **en Canvas** y 2) Presentarse el día de la entrega del proyecto en el periodo acordado con el catedrático en la semana 20.

Criterios	Nivel 3 Experto	Nivel 2 Aprendiz	Nivel 1 Novato
Funcionamiento del programa **40%**	El programa funciona con todos sus requerimientos: ingreso de datos, despliegue de resultados y salida correcta al sistema operativo. **40%**	El programa funciona entre el 80% y 60% de sus requerimientos. **30%**	El programa funciona en menos del 50% de sus requerimientos. **12%**
Interfaz con el usuario **20%**	La interfaz indica claramente los avances del juego, es clara y amigable, está muy completa y se encuentra diseñada de forma creativa. Utiliza ASCIIart con una figura alusiva al juego **20%**	La interfaz Indica parcialmente los avances del juego, es medianamente amigable con el usuario, está incompleta o le falta creatividad. No utiliza ASCIIart. **12%**	La interfaz no indica los avances del juego, es poco amigable con el usuario y/o está muy incompleta. No utiliza ASCIIart. **4%**
Programación defensiva y ayuda del uso del programa **10%**	El programa tiene muy buena programación defensiva en todos los ingresos de datos, y proporciona mensajes oportunos ante situaciones inesperadas. El programa da instrucciones de ayuda para utilizar el juego. **10%**	El programa tiene programación defensiva en la mayoría de los ingresos de datos, y proporciona algunos mensajes oportunos ante situaciones inesperadas. El programa da una ayuda regular para utilizar el juego. **6%**	El programa tiene muy poca o ninguna programación defensiva en los ingresos de datos, y proporciona pocos o ningún mensaje oportuno ante situaciones inesperadas. El programa da muy poca ayuda para utilizar el juego. **2%**
Documentación y orden del programa **10%**	La documentación incluye encabezado y comentarios representativos en los bloques de código más importantes. Los nombres de las variables son significativos. La presentación del programa es muy clara y ordenada, y utiliza una tabulación adecuada. **10%**	Falta documentación en el encabezado o en bloques de código Los nombres de las variables son medianamente significativos. La presentación del programa es regularmente clara y ordenada. La tabulación es aceptable. **6%**	Falta gran parte de la documentación del código Los nombres de las variables no expresan ningún significado. La presentación del programa es confusa y desordenada. No hay tabulación de las instrucciones. **2%**

Uso de subrutinas y programación estructurada (no cuentan las subrutinas vistas en clase) **10%**	El programa es 100% estructurado con más de dos. Las subrutinas tienen en el encabezado el nombre del estudiante que las diseñó. **10%**	El programa es casi completamente estructurado con mínimo dos. **6%**	El programa no está estructurado con subrutinas **2%**
Documento de análisis y diseño **10%**	El documento incluye todo lo necesario para explicar el trabajo realizado: especificación del uso de los registros, diagrama de flujo del programa hecho en una herramienta de software o algoritmo narrativo, ordenado y con la simbología correcta, conclusiones, bibliografía citada de forma correcta y de al menos 2 sitios confiables. **10%**	El documento incluye entre el 80% y 60% de los aspectos necesarios para explicar el trabajo realizado. **6%**	El documento incluye menos del 50% de los aspectos para explicar el trabajo realizado. **2%**