Proyecto #3 de diseño: lenguaje ensamblador ARM

I. Competencias a desarrollar:

Diseñar un programa básico en lenguaje ensamblador del procesador ARM que implemente el juego asignado.

II. Condiciones y fechas de entrega:

- El proyecto se realiza en parejas seleccionadas por los estudiantes.
- **Entrega y presentación:** viernes 26 de abril
- **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 (.s)
    - Documento del diseño en formato PDF que 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.

III. Instrucciones

Su proyecto consiste en elaborar un juego en lenguaje ensamblador de ARM. Deberá permitir jugar a dos personas como mínimo. Uno de los jugadores puede ser la computadora. 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/

IV. Temarios:

1. **Salto de ranas.** Este juego consiste en 11 posiciones, y 8 ranas, 4 ranas de la especie (b) están localizadas en la parte derecha y 4 ranas de la especie (a) están ubicadas en la parte izquierda de las posiciones, dejando así un único espacio vacío. El objetivo del juego es intercambiar las 4 ranas de cada lado, se debe tener en cuenta que las ranas no pueden retroceder y que solo pueden moverse un espacio a la vez, las ranas de distinta especie pueden saltarse otra rana si existe un espacio vació adelante, sin embargo, ranas de la misma especie no puede saltarse entre ellas. Represente cada especie de rana con un carácter diferente.

(a)	(a)	(a)	(a)	( )	( )	( )	(b)	(b)	(b)	(b)
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)

2. **Fuga de palabras.** Este juego consiste en que completar la vocal que falta 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. En cada juego se muestran un total de diez palabras diferentes, si el jugador acierta con la vocal que forma 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). Gana el jugador que haya obtenido más puntos.
   NOTA: su programa deberá tener un banco de 40 palabras que serán seleccionadas aleatoriamente.

   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

3. **Cuatro en línea modificado.** El juego consiste en formar líneas de 4 fichas, en forma horizontal, vertical o diagonal en una matriz de 4x4. Se tienen dos jugadores que juegan por turnos, indicando la columna de la matriz donde dejarán caer su ficha. Gana el jugador que primero arme una línea de cuatro fichas. Utilice un carácter ASCII diferente para cada jugador

Ejemplo de ejecución:

```
********MATRIZ 4x4******
|          |          |          |          |
|          |          |          |          |
|          |          |          |          |
|          |          |          |          |
(1)    (2)  (3)  (4)
Ingrese columna jugador 1 (1, 2, 3, 4): 1
********MATRIZ 4x4******
|          |          |          |          |
|          |          |          |          |
|          |          |          |          |
| x        |          |          |          |
(1)    (2)  (3)  (4)
Ingrese columna jugador 2 (1, 2, 3, 4): 1
********MATRIZ 4x4******
|          |          |          |          |
|          |          |          |          |
| o        |          |          |          |
| x        |          |          |          |
(1)    (2)  (3)  (4)
```

4. **Tragamonedas.** La computadora es una máquina tragamonedas. Genera tres dígitos al azar. Cada dígito puede ser un número de 1 a 3. la máquina tragamonedas inicia con 20 monedas en su depósito (esta cantidad puede ser modificada). El jugador debe ingresar la cantidad de monedas que desea apostar y luego girar la palanca de la máquina. 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 aun existan en el depósito de la máquina (lo que sea menor). Si el jugador pierde, entonces sus monedas ingresan al depósito de la máquina. El juego termina cuando el jugador o la máquina se queden sin monedas. Gana quien tenga las monedas.
   NOTA: en cada juego el jugador iniciará con 20 monedas (esta cantidad puede ser modificada). Si se inicia de nuevo el juego se le pondrán 20 monedas en el depósito a la máquina (esa cantidad puede ser modificada).

5. **Carrera con obstáculos.**

El objetivo del juego es llegar en primer lugar a la meta. Lo pueden jugar dos personas o más (uno de los cuales puede ser la computadora). La meta está localizada a 50 pasos de la línea de inicio (esta cantidad de pasos puede ser modificada). En cada turno, el jugador lanza dos dados y avanza la cantidad de pasos indicados por los dados. Si llega a la meta o la sobrepasa entonces concluye su carrera. Si él es el primero en cruzar la meta, entonces es el ganador del juego. 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.

6. **Juego Mayor o Menor.**

Se juega entre dos o más jugadores. La computadora puede ser uno de los jugadores. En el juego se tienen dos dados y se alternan turnos para que los jugadores tiren los dados. Al inicio el primer jugador tira los dados y se mira la cantidad mostrada en sus caras. Este es el valor inicial para el juego. Por turnos, cada jugador indica que el próximo número que muestren los dados será **mayor o menor** que la última cantidad obtenida por el jugador previo. Luego tira los dados, y si acierta se le otorga un punto. Si la cantidad mostrada por los dados es **igual** a la cantidad obtenida por el jugador previo, entonces vuelve a tirar los dados, sin acumular ningún punto. Gana el jugador que primero complete 10 puntos (esta cantidad de puntos puede ser modificada).

7. **2 fotos 1 palabra (2 Pics 1 Word)**

Se juega entre dos jugadores, la computadora no puede ser uno de ellos. Cada jugador contará con tres turnos de forma alternada; en cada turno se muestran en pantalla dos imágenes (diseñadas con ASCII Art) que tienen algún tema en común; debajo de las imágenes habrá un tablero con letras en desorden, para formar una palabra (la palabra está relacionada a las imágenes). El jugador deberá ingresar por teclado la palabra que responda al acertijo; si el jugador responde correctamente, entonces obtiene tres puntos. Gana el jugador que obtenga más puntos al final de cada ronda (una ronda = 6 turnos, 3 de cada jugador).

NOTA: su programa deberá tener un banco de al menos 18 imágenes que serán seleccionadas aleatoriamente.

8. Battleship

Se juega entre dos jugadores y el objetivo es hundir los barcos del otro jugador. En un tablero de 4x4, cada jugador elegirá la posición donde colocará un barco, seleccionando las posiciones contiguas para este. En un primer prompt, el jugador 1 selecciona de forma consecutiva las 3 posiciones contiguas horizontales o verticales del barco. Luego de seleccionadas, el juego debe desplegar un tablero con el barco (indicado +) en la posición seleccionada. Si el jugador 1 no confirma la posición de su primer barco, se reinicia el proceso. Si confirma la posición, se borra el tablero del jugador 1 y se repite todo el proceso para el jugador 2. En cada turno, se despliega el tablero de juego de cada jugador con el estado exitoso o no se sus elecciones anteriores. El jugador elige 3 posiciones en cualquiera de las casillas de su tablero para tratar de hundir el barco de su oponente. Luego de la selección, se mostrará su tablero con una falla con una "o" y un acierto con una "x" y este se borrará luego de 5 segundos, dejando la pantalla sin tablero y con un mensaje invitando al siguiente jugador. El juego termina cuando un jugador hunde el barco del oponente. EXTRA: usar un tablero con mayor número de casillas y mayor número de barcos de 3 posiciones.

	+		
	+		
	+		

Tablero con la posición para barco jugador 1

	o	o	
		o	
	x		
o			

Tablero jugador 2 con sus aciertos y fallas

9. Minesweep

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 aleatorias. 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. Si la posición vecina a la elección tiene una mina, mostrará distancia 0. Si la selección cae en una casilla sin minas, se marcará la posición con (#), pero si existe una mina en esa posición, se marca la posición con (X) y se acaba el juego. EXTRA: Si el jugador despeja todas las posiciones, puede pasar al siguiente nivel o concluir el juego pulsando (X). En el siguiente nivel del juego se puede: 1) cambiar la posición de las minas o 2) agregar una mina más y cambiar la posición. El máximo número de minas permitidas es 5, se acumulan 10 puntos en cada nivel y se agregan 10 puntos al puntaje del nivel anterior para el siguiente nivel.

2	2	1	
0	#	1	
2	1	0	

Tablero mostrando posición elegida y distancia a las minas

			X
	#		
		X	

Tablero habiendo elegido una mina

.

10. Rabbit Chase

El objetivo del juego es atrapar un "conejo" antes que escape del tablero de 5x5. El jugador (H) selecciona como punto de partida cualquier posición de inicio de las filas o columnas externas. El conejo (B) aparece en el centro del tablero. Para su siguiente movimiento, el jugador puede elegir únicamente casillas vecinas a su última posición. En su siguiente movimiento, el conejo elige de forma aleatoria cualquiera de los 9 cuadros vecinos y decide una dirección. En el tercer movimiento, el conejo solamente podrá elegir una de las 3 casillas vecinas en la dirección elegida. El jugador gana si elige la misma casilla elegida por el conejo en el siguiente movimiento (X). Se acaba el juego si el conejo sale del tablero (B!) alcanzando cualquiera de las posiciones en las filas o columnas externas. EXTRA: aumentar el tamaño del tablero y permitir que el conejo se mueva en cualquier dirección.

		H		
		B		

Primer movimiento de jugador (H) y conejo (B)

	H			
	B<			

Segundo movimiento de jugador (H) y conejo (B)

X				

Conejo atrapado

H				
B!				

Conejo escapó

11. 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. En cada turno el jugador elegirá su opción: par o impar. 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, de lo contrario, retrocede una casilla. 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.

12. GO

Se juega entre dos jugadores. El objetivo del juego es convertir todas las fichas de un jugador a fichas del otro jugador. En un tablero de 5x5 cada jugador coloca una ficha por turno. A un jugador le corresponde la ficha (O) y a otro la ficha (X). Si un jugador coloca las fichas del otro entre dos fichas suyas, todas las fichas atrapadas se convierten a fichas del jugador. Esto se pude hacer de forma vertical u horizontal. El juego termina cuando no queden más posiciones por jugar y gana el jugador con más fichas en el tablero. EXTRA: extender la función de captura de fichas a posiciones diagonales.

X	O			
	O			
	X			

X	X	X		
	O			
	X			

13. ARM-Ztype

El juego requiere de 1 jugador. Consiste en “destruir” las palabras que son enviadas por la computadora en forma de ticker tape (texto que se desplaza horizontalmente) de texto que se mueven de izquierda a derecha. Al iniciar el juego aparece la primera palabra con su(s) espacios en blanco y cada 5 segundos o luego de que el jugador intente adivinar, aparece una nueva palabra. La destrucción de palabras se realiza ingresando la letra o letras faltantes (máximo 2) en el orden de ausencia en la palabra. Cada palabra está precedida por un numeral entre paréntesis, por lo que el jugador debe indicar primero el número de la palabra a eliminar y luego la cadena de caracteres faltantes, todo en la misma línea y sin espacios en blanco. Al eliminar cualquier palabra, se debe recalcular el orden de las demás dentro de la secuencia, para poder desplegarla al jugador. El juego acaba cuando cualquier palabra no adivinada llega al lado derecho de la pantalla. EXTRA: El juego debe tener al menos dos niveles de dificultad, los cuales se diferenciarán por la velocidad de avance de las palabras y por el número de letras faltantes. El jugador acumula puntos según el número de letras de la palabra. Para mayor referencia del juego, dirigirse a: https://zty.pe/

T= -0 s	Debes “destruir” las palabras completándolas. Selecciona el orden de la palabra seguido de sus caracteres faltantes: 1ER Listo para jugar? (Y/N) Raspi:> Y
T = 0 s	(1)_rnitorr_nco Raspi:>
T = 5 s timer	(2)ba_l (1)_rnitorr_nco Raspi:>2L
T= 8 s Intento de jugador	¡Intenta de nuevo! (3) t_levisi_n (2)ba_l (1)_rnitorr_nco Raspi:>2U
T= 11 s Intento de jugador	Correcto: BAUL (2) t_levisi_n (1)_rnitorr_nco Raspi:>
T = 16 s timer	(3) z_p_to (2) t_levisi_n (1)_rnitorr_nco Raspi:>

14. ARM-Google Feud

El juego requiere de un único jugador. El juego está basado en la API de Google; consiste en adivinar la palabra que completa oración de búsqueda realizada en Google. Inicialmente, se debe seleccionar 1 categoría entre cultura, personas y preguntas; posterior a seleccionar la categoría, automáticamente se dirigirá a la pantalla a la pantalla de juego, en donde se muestra una oración incompleta, el jugador debe adivinar cuál es la palabra para tres búsquedas distintas. cada categoría debe tener por lo menos 5 elementos distintos de búsqueda. El juego finaliza si el jugador falla 3 veces al adivinar las palabras con mayor búsqueda. Para mayor referencia del juego, dirigirse a: https://www.paisdelosjuegos.es/juego/google-feud.html

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 del mismo**. *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 poder 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 doble del viernes 26 de abril.

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 (2) subrutinas que utilizan el formato ABI. Las subrutinas tienen en el encabeado el nombre del estudiante que las diseñó. **10%**	El programa es casi completamente estructurado con mínimo dos (2) subrutinas que utilizan el formato ABI. **6%**	El programa no está estructurado o las subrutinas no utilizan el formato ABI. **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%**