Explicación del código.

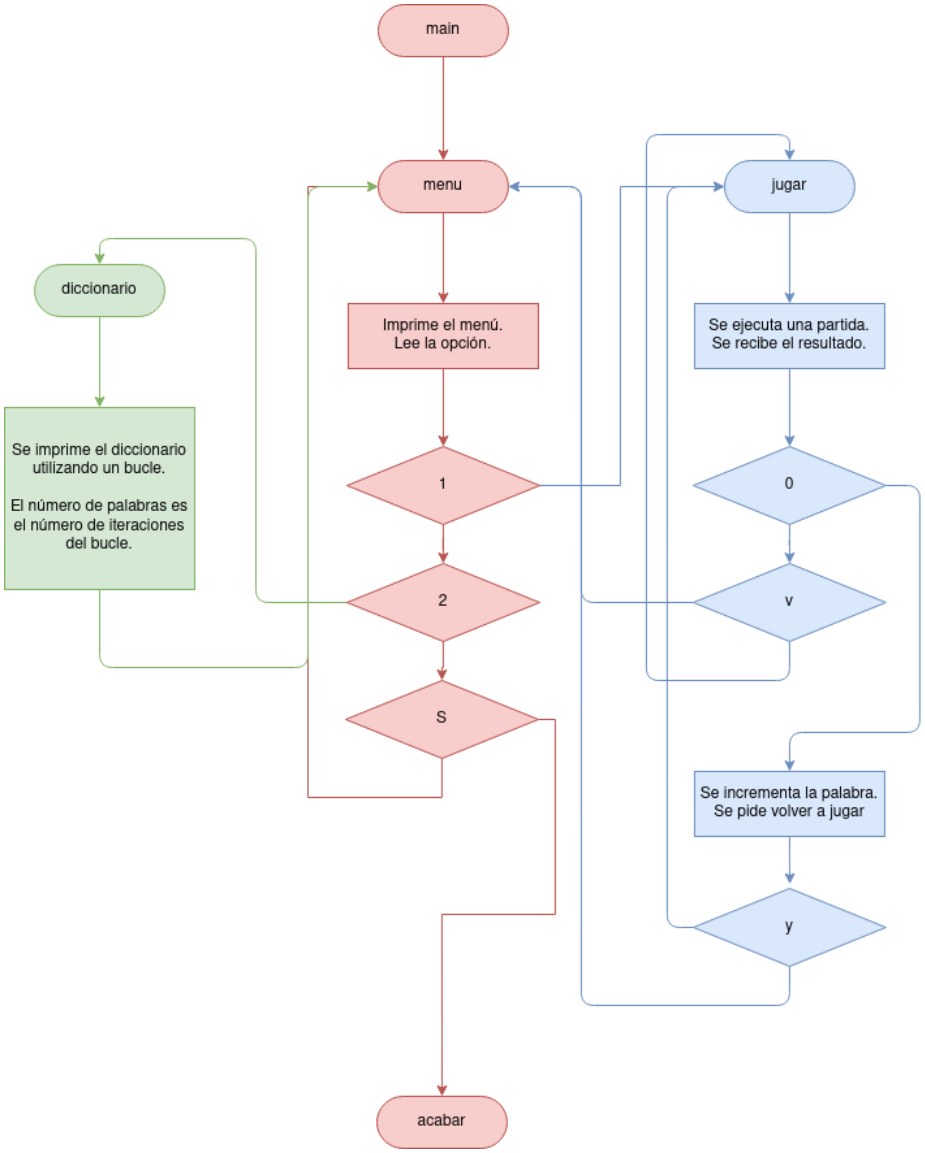
1.- Descripción general:

El código de todo el programa se encuentra dividido entre 4 documentos:

- **main.asm:** Fichero con el área absoluta. El menú, presentación del diccionario e inicio y control de las partidas se realiza aquí.
- **game.asm:** Contiene el código para ejecutar una partida de Wordle.
- **lib.asm:** Biblioteca de subrutinas que se utilizan en los dos archivos anteriores. Las funciones suelen ser de lectura y escritura de cadenas o funciones específicas de wordle.
- **diccionario.asm:** Lista de palabras que se utilizan en wordle.

Los archivos ***main.asm*** y ***game.asm*** se describirán con diagramas de manera algo más superficial, puesto que una gran parte de estos son tareas simples como imprimir cadenas o realizar comprobaciones simples. Por otro lado, el archivo ***lib.asm*** se explicará función a función de manera algo más detallada. ***diccionario.asm*** carece de lógica, así que no se explicará.

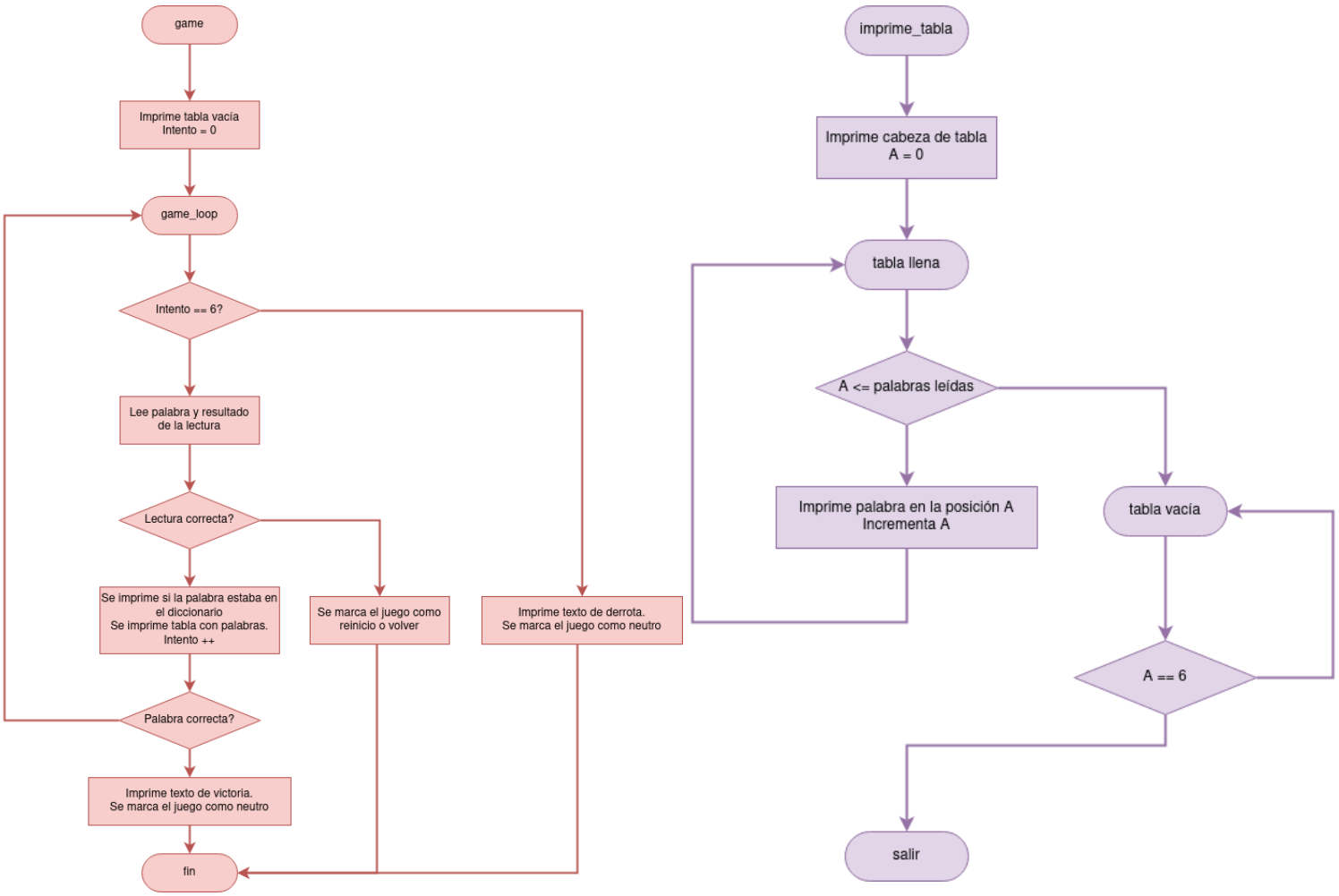
2.-**main.asm**



El archivo ***main.asm*** es el controlador principal del programa. Contiene el menú que permite el paso al wordle, al diccionario o a salir. Cosas que destacar:

- El menú de opciones no utiliza una tabla de saltos, puesto que las comprobaciones para que funcionase son las mismas que para comprobar cada caso particular.
- Las direcciones elegidas para **S** e **Y** son: *0xFF00* y *0xF000*.
- El programa comienza en la dirección *0x1000* puesto que utilizar *0x100* no deja suficiente espacio para el código y las variables.
- Cada vez que comienza un juego o se muestra el menú se limpia la pantalla.
- Puesto que la función ***daa*** es defectuosa, se utiliza una función propia para imprimir el número de palabras en el diccionario.
- Al incrementar la palabra, se tiene en cuenta el final de la lista.
- El texto del terminal se pone en negrita al comenzar el programa y vuelve a normal al acabar

3.- **game.asm**



El archivo ***game.asm*** contiene el bucle principal del juego, así como una subrutina para imprimir la tabla de palabras anteriores y palabra actual. Características destacables:

- Hay que establecer A a -1 para imprimir la tabla vacía, puesto que la comprobación para imprimir una línea existente es beq, por lo que 0 no valdría.
- En *game_end_mal*, usamos "*leas 4,s*" para sacar 4 bytes del stack **S** sin guardarlos en variables para poder recuperar los valores iniciales.
- El almacenamiento de palabras pasadas se hace moviendo el registro **Y** a lo largo de una serie de palabras sin separar. Para eso es necesario el registro **A** guardando la palabra actual.
- Para imprimir las palabras pasadas, el registro **Y** debe apuntar al comienzo de una palabra. Para esto, se multiplica la dirección de **A** por 5.

4.- **lib.asm**

Las funciones de ***lib.asm*** se explicarán principalmente por palabra en el orden en el que aparecen en el archivo.

- **imprime_valor_decimal:** Puesto que la función **daa** no funciona correctamente en el emulador, necesitamos buscar una manera distinta de imprimir el valor de los registros en decimal por pantalla. En este caso, dividimos el valor del registro **A** entre 10. El cociente es el primer dígito y el resto el segundo.

- **imprime_cadena:** Se hace uso del registro **X** para mostrar las cadenas por pantalla. (Esta es la razón por la que se realizan muchos *"exg x,y"*, para imprimir el contenido de **Y** por pantalla sin necesidad de una función nueva)

- **imprime_cadena_color:** Para imprimir una cadena con color, se utilizan algunas secuencias de escape. Se imprime la cadena de cambio de color, el texto a imprimir y la cadena de cambio a color blanco. Para apuntar a la cadena de color correcta, se "siftea" el valor de registro **A** (el color a imprimir) tres veces (para que sea un múltiplo de 8) y luego se imprime la cadena en **a,x**. De ahí la existencia de los *\0* que se ven al definir las cadenas de color, para que se ajusten a 8 bytes.

- **imprime_cadena_wordle:** La función se compone de un bucle principal que recorre la palabra introducida (en **Y**). En cada una de estas iteraciones se recorre la palabra correcta (en **X**). Si las letras son iguales, se suma 6 o 1 al registro a, dependiendo de si las letras están en el mismo sitio o no. Una vez terminada la segunda iteración, se comprueba el color que debería tener la letra: verde si **A** >= 6, amarillo si 6>**A**>0 y blanco si **A** = 0. Finalmente, se imprime con la subrutina *imprimir_caracter_color*.

- **imprime_palabra:** Funciona de manera similar a imprime_cadena, excepto que en vez de para una vez encontrado un \0, para después de imprimir 5 caracteres.

- **imprime_caracter_color:** Funciona igual que *imprime_cadena_color* con la diferencia que en vez de llamar a *imprime_cadena* se imprime el carácter en **B** directamente.

- **compara_palabras:** Recorre ambas palabras a la vez. Si se encuentra un par de caracteres distintos, se devuelve falso. Si se recorre todo el bucle, 1.

- **palabra_en_diccionario:** Se recorren las palabras del diccionario hasta encontrar un \0. En cada iteración se compara la palabra del diccionario con la dada, si son iguales se devuelve uno. Si nunca se encuentra una palabra igual se devuelve 0.

- **lee_palabra:** Inicialmente, se escribe \0 en las 6 primeras posiciones de **Y** para evitar problemas al escribir la palabra por pantalla. Posteriormente entramos en un bucle que no termina hasta que haya 5 letras leídas. Cada vez que se lee una letra se comprueba que esté entre A y Z. En ese caso se añade una letra a la cadena. En caso contrario, se comprueba si es una 'v', 's' o un espacio. En los primeros dos casos, se sale de la subrutina con dichos valores en el registro **A** para marcar un reinicio o vuelta al menú. En el tercer caso, se decrementa el número de letras introducidas y se escribe \0 en la última (evitando que se imprima en la próxima iteración). Finalmente se escribe 0 en **A** significando una correcta lectura y se sale.