Universidad Simón Bolívar
Vicerrectorado Académico
Departamento de Computación y Tecnología de la Información
CI3815 – Organización del Computador

Informe Proyecto 1

Autores:
Franco Murillo – 1610782
Alejandro Zambrano – 1710684

Profesor:
Fernando Torre Mora

Sartenejas, junio de 2024

Introducción

El proyecto trata de la implementación de un módulo que permite crear, ver y reproducir melodías para el teléfono celular Nokia 3310. Aun cuando el objetivo de Nokia era poder contar con 32 espacios para melodías, por los momentos el módulo contará con 8 espacios.

El programa carga las melodías (tanto las que tienen notas, como las melodías vacías) desde un archivo .asm, y luego presenta al usuario un menú interactivo donde se puede, o bien agregar una melodía nueva, como ver las notas de la melodía, como reproducir una melodía.

Dado que cada melodía puede tener una longitud variada de notas, se decidió usar una lista enlazada para cada melodía. Y como el tamaño total de espacios para las melodías es fijo (8 espacios), se decidió guardar las melodías (las listas enlazadas) en un arreglo de 8 posiciones. Inicialmente en cada posición irá una cabeza de lista iniciada en 0.

El presente informe explicará cómo funciona el módulo cuáles estructuras de datos se decidieron utilizar para implementar el módulo, qué limitaciones fueron encontradas durante el desarrollo del proyecto, el estado actual del módulo y las lecciones aprendidas durante el desarrollo de la implementación.

Estructuras Utilizadas

Lista Enlazada:

La lista enlazada implementada consta de nodos que contienen 3 campos:

- Un string, que contiene la nota musical (en formato acústico científico).
- Un entero positivo, que contiene la duración de la nota.
- El apuntador al siguiente nodo.

En la sección .data del módulo, el nodo de la lista es declarado como “node”, y contiene un .space de 17 bits, donde 12 bits corresponden al string, 1 bit corresponde al entero, y 4 bits corresponden al apuntador al siguiente nodo. Además, se declara la cabeza de la lista como “head”, que contiene un .word de 0, que va a ser usado para indicar si la lista está vacía.

Métodos de la lista enlazada

- agrega_nota: Primero asigna 20 bytes de espacio para el nuevo nodo, y verifica si la asignación de espacio fue exitosa; si no es así, salta a **alloc_error**. Luego hace una copia del buffer del string para ingresado al nodo, y también agrega el entero (la duración) al nodo. Posteriormente verifica si el nodo a agregar es el primero de la lista, y en caso de que así sea, salta a **update_head**. Si no es el caso, se agrega el nodo al final de la lista usando las funciones **find_last** y **agregar**.

- alloc_error: en caso de que hubiera algún error al momento de asignar el espacio, imprime un mensaje de error, y sale del programa.

- update_head: en caso de que el nodo a ingresar sea el primero de la lista, establece dicho nodo como la nueva cabeza de la lista.
- find_last: recorre la lista verificando el siguiente nodo al nodo actual. En el momento de que se llegue a un nodo que apunte a NULL, se encontró al último nodo de la lista, por lo que salta a agregar.
- agregar: Ya con el último nodo encontrado, se establece como siguiente de ese nodo al nodo que se quiere agregar.

- ver_melodia: primero carga la dirección de la cabeza de la lista, y verifica si es cero o no, es decir, si la lista está vacía o no. Si está vacía, salta a **error_ver_vacia**. Luego entra en un ciclo llamado **imp**.

- error_ver_vacia: si la lista está vacía, imprime un mensaje de error, y vuelve al menú de selección de melodía para ver.
- imp: primero verifica si el nodo actual tiene valor cero, si es así, se llegó al fin de la lista, y se sale a través de fin_imp, la cual hace un salto de vuelta al menú principal. Luego carga la dirección del valor del string del nodo, y realiza un syscall para imprimir la nota. Luego pasa al siguiente nodo a imprimir.

- reproduce_melodia: similar a ver_melodia, con el manejador de error llamado **error_vacia**, y con el ciclo llamado **reproduccion**.

- En reproduccion, imprime la nota igual que en imp (de ver_melodia), y posteriormente, transforma el entero del nodo (que representa la duración, en fusas) a milisegundos para poder usar el syscall 32, el cual se encarga de “dormir”

temporalmente la ejecución del programa, lo cual da la duración de la nota a reproducir. En esta implementación, se decidió usar como fusa el valor de 31 milisegundos.

Arreglo

Es usado para almacenar las 8 melodías (las 8 listas). Al momento de seleccionar un espacio para agregar una melodía, reproducir una melodía o ver las notas de la melodía, se hacen recorridos a través de este arreglo para verificar espacios disponibles, o melodías disponibles para ver o reproducir.

Funcionamiento del Módulo

El módulo consta de 3 archivos .asm:

- Uno con las melodías a precargar (melodies.asm).
- Uno con múltiples macros que son usados en el programa principal (macros.asm).
- Otro con el programa principal de reproducción (nokia.asm).

Melodías

El archivo “melodies.asm” contiene las melodías a precargar en el programa principal, las cuales se encuentran en labels que van desde M1 hasta M8, donde cada una contiene un string con las melodías. Las etiquetas M1 y M2 contienen melodías por defecto, mientras que de M3 a M8 contienen strings vacíos.

Macros

El archivo “macros.asm” contiene diversos macros que son usados para facilitar un poco la escritura en el programa principal. Los macros son:

- print_int: dado un registro correspondiente a un entero, imprime dicho entero.
- print_label: dado un label (declarado en el .data), lo imprime.
- read_int: dado un registro correspondiente a un entero ingresado por el usuario, lo lee.
- read_string: dada una dirección de un string ingresado por el usuario, y una longitud, lee el string.
- terminate: finaliza el programa.
- len_label: dado un label, retorna su longitud.

- pow: dada una base y una potencia, devuelve la potenciación.
- music_note: dado un string (correspondiente a una nota musical) realiza la verificación de si es una nota válida. Retorna 1 si es válida, o 0 si no lo es.
- reestablecer: restablece los valores de los registros utilizados en music_note.
- concat: concatena strings dada una posición deseada en múltiplos de 8. Hace overflow con más de 4 dígitos.

- string_to_int: cambia un string almacenado en memoria por un entero, dada la longitud deseada, y lo retorna. Soporta máximo 9 dígitos.

Programa Principal

El archivo “nokia.asm”, el cual contiene el programa principal, importa los archivos “melodies.asm” y “macros.asm” mediante el operando .include. Luego cuenta con un .data, y de un .text (que contiene los métodos que usa el programa)

En el .data se declara una letra “M”, la cual usaremos para el menú, específicamente para seleccionar un espacio disponible para agregar una melodía, y para ver o reproducir una melodía. Posterior a eso, se declara el nodo de la lista enlazada, la cabecera de la lista, el arreglo que contendrá las listas, y todos los mensajes que se usarán en el programa.

En el .text, se cuenta con un .main, en el cual primero carga a las listas las melodías que se encuentran en el archivo “melodies.asm” de la siguiente manera:

- Carga la primera melodía (M1).
- Inicializa un iterador.
- Entra en el ciclo conseguir_head.
- El ciclo primero carga la base del arreglo, e inicializa la posición del arreglo de listas en 0.
- Entra en otro ciclo llamado espacios_disponibles1, el cual. recorre todas las posiciones del arreglo, carga la dirección de la cabeza de la lista de cada posición, y verifica si la lista está vacía o no, y se queda con la primera que encuentre libre.
- Al encontrar un espacio vacío, salta a continue_espacio1, el cual calcula la dirección exacta de la cabeza de la lista.
- Posteriormente entra al ciclo recuperador, el cual carga cada bit del string de M1, y al final del ciclo recupera la nota y la duración de cada línea.
- Ya con los valores correspondientes a la cabeza de la lista, y del nodo y duración de la línea, salta a agregar_nota (que se explica en la sección de estructuras utilizadas), y vuelve al ciclo, a recorrer la siguiente línea
- Al finalizar el ciclo recuperador, aumenta en uno el iterador, y carga la melodía M2.
- Luego usa un condicional para verificar si el valor del iterador es igual a dos. Si ese es el caso, ejecuta otra vez conseguir_head, pero esta vez en la segunda posición, y ahora con M2.
- Esto lo repite para el resto de 8 melodías en “melodies.asm”.

Luego de haber precargado las melodías, el programa muestra un menú interactivo con 4 opciones, y el usuario debe ingresar por consola la opción deseada, donde la opción 1 permite agregar una melodía, la opción 2 permite visualizar una melodía, la opción 3 permite reproducir una melodía, y la opción 4 termina el programa.

La opción 1 hace que el programa salte a la función **selecciona_espacio**, la opción 2 hace que salte a la función **ver**, la opción 3 hace que salte a la función **selecciona_melodia**, y la opción 4 hace que salte a la función **exit**.

selecciona_espacio

- Imprime el mensaje de selección de espacio, el cual mostrará aquellos espacios disponibles para agregar una melodía. Luego carga la base del arreglo de listas, inicializa una bandera (que se usará para indicar si todos los espacios están ocupados) en 1, e inicializa un iterador en 0. Con esto, hace un recorrido por el arreglo, verifica aquellas listas que están vacías, e imprime sus posiciones por pantalla.

- Para verificar si una lista está vacía, se hace uso de un ciclo llamado espacios_disponibles, en el cual primero se verifica si el valor es igual a 8; en tal caso, ya recorrimos todas las posiciones del arreglo, por lo que sale del ciclo por medio de fin_espacios_disponibles. Luego de esta verificación, se calcula la dirección de la cabeza de la lista, a través de un shift lógico a la izquierda, y sumándole a este resultado el valor de la dirección base del arreglo.

- Luego carga el valor de la cabeza de la lista usando la dirección calculada. Si este valor es cero, la lista está vacía, por lo que salta a imprime_posicion_e, que imprime el string

"M", seguido de la posición del arreglo. Dado que las posiciones de los arreglos se enumeran a partir de 0, para mayor comodidad visual, primero se incrementa en uno el índice de la posición antes de imprimirlo. Luego se cambia el valor de la bandera a 0, indicando que existe al menos un espacio disponible; y vuelve al ciclo. Si el valor de la cabeza no es cero (el espacio en la posición está ocupado), salta a next_posicion_e, el cual incrementa en uno el iterador, y vuelve al ciclo.

- Al finalizar el ciclo, se verifica el valor de la bandera. Si el valor es 0, salta a la continuación del programa; si es 1, imprime un mensaje de error de que no hay espacio disponible, y vuelve al menú principal.

- En la continuación del programa, se lee la posición ingresada por el usuario. Como se incrementó en 1 la posición al imprimirla, aquí se disminuye en 1. Si se ingresa un espacio ya ocupado o fuera del rango de espacios, se muestra un mensaje de error, y vuelve al menú principal. Si se ingresa un espacio válido y disponible, salta a la función agrega_melodia, la cual pide al usuario la nota a ingresar, y la duración de la nota.

- En esta sección se cuenta con un contador de notas, el cual, al llegar a 30 notas ingresadas por el usuario, imprime un mensaje de máximo de notas, y vuelve al menú principal.

- Al ingresar la nota, primero elimina los saltos de línea que ésta pueda tener, mediante el macro no_next_line.

- Posteriormente verifica si se ingresó un “0”, el cual es el indicador de que se llegó al fin de la melodía, por lo que se sale al menú principal. Si no se ingresó un “0”, se verifica que la nota ingresada sea válida a través del macro music_note.

- Si se ingresó una nota inválida, salta a nota_invalida, que muestra un mensaje de error, y vuelve a agrega_melodia. Si se ingresó una nota válida, establece el último valor del string como cero (nulo), y se procede a pedir al usuario que ingrese la duración de la nota. Aquí también se verifica que la nota sea válida (mayor que cero, y menor que noventa y nueve).

- Si todo está bien, se llama a la función agrega_nota (que se explica en la sección de estructuras utilizadas)

ver

- Imprime las posiciones disponibles para ver la melodía.
- La impresión es similar a la vista en selecciona_espacio, con una modificación.
- En vez de verificar listas vacías, se verifica listas **no** vacías.
- La bandera se inicializa en 1, indicando que todas las listas están vacías.
- Si se encuentra una lista no vacía, se imprime su posición en el arreglo, y se cambia el valor de la bandera a 0, indicando que existe al menos una lista no vacía.
- Luego de recorrer las posiciones, se verifica el valor de la bandera (parecido a selecciona_espacio) para continuar el programa o volver al menú.
- Si se continúa el programa, se lee la posición (igual que en selecciona_espacio) y se llama a la función ver_melodia (que se explica en la sección de estructuras utilizadas).

selecciona_espacio

- Imprime las posiciones con melodías disponibles y lee la posición ingresada por el usuario (igual que en la función ver) y se llama a la función reproduce_melodia (que se explica en la sección de estructuras utilizadas)

exit

- Hace uso del macro terminate para finalizar el programa.

Limitaciones

- Por pantalla se imprimen las posiciones disponibles para ver o reproducir una melodía, como M1, M2, etc., pero al momento de ingresar la opción, sólo se puede ingresar el número.

- El código cuenta con varias secciones de código que se repiten (que se reutilizan). No se pudo hacer funcionar el código sin reutilizar de esta manera las secciones de código.

- Después de corrido el programa, al correrlo sin limpiar el I/O se llega a un punto en el que colapsa; es necesario cerrar y volver a abrir el programa (incluso varias veces) antes de que vuelva a funcionar. Se recomienda limpiar el I/O antes de volver a correr el programa. Se desconoce la causa de este comportamiento.

- La sección que se encarga de agregar las melodías precargadas del archivo “melodies.asm” no es la manera más eficiente, pero fue la manera en la que se pudo hacer funcionar el programa.

- El macro concat solo puede concatenar hasta 4 caracteres o 4 dígitos máximo.

- El macro str_to_int puede convertir un número de máximo 9 dígitos, siendo 999999999 el límite. Y no puede convertir números negativos.

Estado Actual y Conclusiones

Actualmente el programa carga correctamente las melodías del archivo “melodies.asm” haciendo uso de la importación con .include, y permite crear correctamente una melodía y almacenarla en el arreglo de listas, así como imprimirla para visualizar las notas, y reproducir las melodías usando el tiempo de duración de cada nota (todo esto, con las limitaciones mencionadas). El proyecto fue bastante interesante, permitiendo aprender más en profundidad acerca de MIPS, de cómo crear macros, cómo importar otros archivos, cómo recorrer arreglos, cómo declarar una lista enlazada, cómo agregar elementos a una lista, y cómo recorrerla, lo cual puede tener muchas aplicaciones al momento de tener otros proyectos a futuro.