Universidad de los Andes
Facultad de Ingeniería

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN
ISIS1304
FUNDAMENTOS DE INFRAESTRUCTURA TECNOLÓGICA

Proyecto 2 - 2017-2

Este proyecto vale 15% de la nota del curso.
Debe ser elaborado en grupo (3 integrantes).
No se permite ningún tipo de consulta entre grupos.
Se debe entregar por Sicua a más tardar el 14 de mayo a las 23:50

A. OBJETIVOS

- Practicar con el lenguaje ensamblador.
- Practicar con programación mixta C-ensamblador.
- Aplicar lo anterior en un programa que permite la codificación de un archivo binario a uno de texto

El proyecto consiste en completar un programa en C mediante ensamblador embebido siguiendo las instrucciones que se dan a continuación.

B. DESCRIPCIÓN DEL PROBLEMA

Utilizando como base el proyecto anterior se reescribirá el programa usando el lenguaje ensamblador. Sin embargo, existe una diferencia respecto a la entrada que recibe el proyecto: ahora será necesario indicar mediante parámetros en el método main la siguiente información:

1. La ruta completa del archivo `.bmp`
2. Indicar si se está ejecutando el programa para insertar o leer un mensaje.
3. El número de bits por Byte que se utilizaron o que se utilizarán para guardar el mensaje.
4. El mensaje que será guardado en la imagen o la longitud del mensaje a ser leído.

Para indicar que se leerá un mensaje, se utiliza el argumento `'r'`, y para el caso contrario se utilizará el argumento `'w'`.

A continuación se ejemplifica el uso de los parámetros en ambos casos:

- `/path/to/image.bmp w 4 thisIsATestMessage`
- `/path/to/image.bmp w 2 “this test message has spaces between words”`
- `/path/to/image.bmp r 5 18`

PROCEDIMIENTOS

Complete los procedimientos según lo indicado en el esqueleto del programa adjunto. En

particular, note que hay tres tipos de restricciones sobre los procedimientos:

- Los que no deben ser modificados (se deben dejar tal como están).
- Los que deben ser escritos en ensamblador pero pueden utilizar nombres simbólicos de variables. Esto quiere decir que pueden declarar variables locales en C y usarlas en el código ensamblador, igual que los parámetros de las funciones.
- Los que deben ser escritos sin usar nombres simbólicos, es decir, que acceden a los parámetros y variables locales a través de desplazamientos en la pila.

Procedimientos que no deben ser modificados:

- `main, cargarBMP24 y guardarBMP24`.

Procedimientos para escribir en ensamblador usando nombres simbólicos:

- `void decidir(char nomArch1[], char op, int bitsPorByte, char msg[], Imagen * img):`

Esta función se encarga de hacer el llamado al procedimiento correspondiente para insertar o leer un mensaje. Recibe como parámetros: un vector en el que se guarda la respuesta, el argumento usado para el modo de ejecución (leer/escribir), el número de bits por Byte, el mensaje que se va a colocar en la imagen y un puntero a la imagen.

Nota: Es evidente que el método `decidir` recibe parámetros que no serán usados dependiendo de la acción que se quiera ejecutar.

El parámetro `op` no se reconoce si no es `'r'` ni `'w'` o si tiene más de un carácter.

- `void leerMensaje(Imagen * img, char msg[], int l, int n):`

Esta función se encarga de leer un mensaje guardado previamente en la imagen. Recibe como parámetros: un puntero a la imagen, un vector en el que se guarda la respuesta, la longitud del mensaje que se codificó y el número de bits por byte usado.

Procedimientos para escribir en ensamblador sin usar nombres simbólicos:

- `void insertarMensaje(Imagen * img , char mensaje[], int n):`

Esta función se encarga de colocar un mensaje en una imagen. Recibe como parámetros un puntero a la imagen, el mensaje que se va a colocar en la imagen y el número de bits que se usan en cada componente de color para almacenar el mensaje.

Nota: la notación "`char mensaje[]`" es equivalente a "`char * mensaje`": es un apuntador a una cadena de caracteres.

Nota: Puesto que el proyecto se basa es su proyecto anterior, para que funcione debe funcionar también la parte anterior; en caso de que no sea así, debe arreglar el proyecto

anterior para esta entrega.

C. ESPECIFICACIONES

- Los programas se deben escribir en C (en Visual Studio). Nota importante: los programas se calificarán únicamente usando el ambiente de visual; si el programa no compila en este ambiente, se considerará que no corre (así compile en otros ambientes).
- Legibilidad del programa: indentar el programa; escribir comentarios explicando el código; nombres dicientes de variables.
- Debe respetar la estructura y requerimientos del código entregado. En particular, debe usar los procedimientos y variables del esqueleto, y no pueden crear procedimientos adicionales.

D. CONDICIONES DE ENTREGA

- Entregar el código fuente junto con el ejecutable en un archivo *.zip. **Al comienzo del archivo fuente escriba los nombres de los miembros, sus códigos y correos, de lo contrario no será evaluado (ver esqueleto)**. Si su programa no funciona o si su solución tiene particularidades, puede enviar un archivo .doc explicando por qué cree que no funciona o qué fue lo que hizo.
- El trabajo se realiza en grupos de **3** personas. No debe haber consultas entre grupos.
- El grupo responde solidariamente por el contenido de todo el trabajo, y lo elabora conjuntamente (no es trabajo en grupo repartirse puntos o trabajos diferentes).
- Se puede solicitar una sustentación a cualquier miembro del grupo sobre cualquier parte del trabajo. Dicha sustentación puede afectar la nota de todos los miembros.
- El proyecto debe ser entregado por Sicua por uno solo de los integrantes del grupo.
- **Se debe entregar por Sicua a más tardar el 14 de mayo a las 23:50.**

E. CASOS DE PRUEBA

Para hacer las pruebas, adjunto al proyecto, se encuentran tres carpetas, cada una con una imagen original (.bmp), el texto a insertar o leer (.txt) y la imagen modificada (.bmp). Al insertar en la imagen original se debe obtener la imagen modificada. Al leer de la imagen modificada se debe obtener el mensaje. Realice cada una de las pruebas con los siguientes parámetros:

1. Prueba 1 (12 caracteres, bits por Byte: 8)

   Nota: este caso no tiene sentido puesto que reemplaza todo el byte de cada componente (se pierde la imagen); se pone por completitud, para probar el programa en un caso extremo fácil de verificar (puesto que en el vector de información queda literalmente la cadena de caracteres).

2. Prueba 2 (794 caracteres, bits por Byte: 4)
3. Prueba 3 (2130 caracteres, bits por Byte: 2)

F. CRITERIOS DE CALIFICACIÓN PARA LOS PROGRAMAS

La calificación consta de dos partes:

- Ejecución (50%). Para las funciones propuestas se harán cuatro pruebas: dos de los casos de prueba entregados (seleccionados al azar) y otros dos nuevos.
- Inspección del código (50%). Se consideran tres aspectos:
  - ✓ 10% - legibilidad (nombres dicientes para variables, comentarios e indentación)
  - ✓ 20% - direccionamiento en ensamblador
  - ✓ 20% - uso del lenguaje ensamblador (evaluación de expresiones y control).

G. RECOMENDACIONES

- Recuerden que los trabajos hechos en grupo y entregados individualmente (o en grupos diferentes al original) son una forma de fraude académico.