Universidad de los Andes
Departamento de Ingeniería de Sistemas y Computación
ISIS 1311 – Tecnología e Infraestructura de Cómputo

Caso 1 - 20252

Arquitectura del Computador y Representación de Datos

Objetivo

- Usar el lenguaje ensamblador para escribir una función de baja complejidad.
- Comprender la estructura y funcionamiento de la pila para paso de parámetros y manejo de variables locales.
- Comprender la representación binaria de imágenes (en formato BMP)

MUY IMPORTANTE: El entregable de este caso debe ser 100% de su autoría. No está permitido usar ayudas no autorizadas (incluyendo chatbots o tecnologías similares). El incumplimiento de lo anterior resultará en una calificación de cero (0.0) para el caso 1.

Tenga en cuenta que los casos están diseñados para apoyar **su proceso de aprendizaje e identificar y resolver sus dudas**, sobre los temas estudiados, antes del parcial.

A. Descripción del contexto

En el laboratorio #1, trabajamos con imágenes BMP de 2 colores manipulando directamente los bits. En esta ocasión, se extenderán esos conocimientos para procesar imágenes BMP de 256 colores, usando una paleta y modificándola para obtener una nueva imagen, pero en escala de grises.

En el laboratorio 1 se vio el formato para una imagen BMP con paleta de dos colores. Para manejar 256 colores el formato cambia en dos aspectos principales:

- La paleta tiene 256 entradas, cada una de las cuales sigue las mismas convenciones de la paleta de dos entradas (R,G,B y reservado, para un total de 1024 bytes).
- Por cada pixel de la imagen se guarda 1 byte que indica la entrada en la paleta que tiene la representación binaria del color correspondiente a ese pixel (su color en RGB).

B. Actividades

Su tarea es traducir a ensamblador un fragmento de código de un programa en C que procesa archivos BMP de 256 colores para convertirlos en imágenes en grises. Para esto se entregan, con este enunciado: un archivo fuente en C y cuatro archivos BMP de 256 para pruebas.

Para hacer la traducción, siga los pasos que se presentan a continuación.

A partir del código fuente proporcionado (`bitmap.c`), realice las siguientes tareas:

1. Comprensión y Ejecución

- Compile y ejecute el programa sobre una imagen BMP de 256 colores. Use alguna de las imágenes entregadas. Verifique que se genera correctamente el archivo salida.bmp en escala de grises.
- Observe y explique qué hace la función `convertir_a_grises` (en C)

2. Análisis de la pila

- A partir del código C de la función `convertir_a_grises` identifique las variables locales y parámetros.
- Pinte la pila justo antes de la ejecución de la primera instrucción de `convertir_a_grises`, y justo después de la ejecución de la última instrucción (justo antes del retorno). Justifique los desplazamientos asociados con variables locales y parámetros.

3. Traducción a Ensamblador

- Traduzca a ensamblador la función `convertir_a_grises`. La función debe seguir el algoritmo planteado en la versión en C. No trate de optimizar el código o mejorar la función, solo traduzca el código entregado.
- Solo debe traducir la función indicada (no modifique el resto del programa).
- No use nombres simbólicos. Es decir, para referirse a parámetros o variables locales debe usar desplazamientos a partir del ebp.
- Incluya comentarios para las instrucciones en ensamblador, explicando las transformaciones que se hacen a “R”, “G” y “B”.

C. RECOMENDACIONES

Al momento de hacer la traducción a ensamblador dentro del programa en C, tenga en cuenta las siguientes recomendaciones:

- No modifique las estructuras de datos ni el main del programa.
- Use solo imágenes de 256 colores. Para evitar conflictos use los ejemplos suministrados.
- Puesto que estamos dentro de un procedimiento de C, **el compilador se encarga de salvar el ebp.** En consecuencia, NO es necesario hacer:

  ```
  push ebp
  mov esp, ebp
  ```

  Tampoco es necesario recuperar ebp al finalizar la rutina.
  Atención: de hacerlo dañará el ebp verdadero y no podrá tener acceso a los parámetros y variables locales (leerá posiciones erróneas).
- Dado que las variables locales se declaran en C, el compilador se encarga de separar el espacio en la pila. En consecuencia, **NO es necesario hacer**:

  ```
  sub esp, <tamaño-locales>
  ```

  Tampoco es necesario devolver este espacio en la pila al finalizar la rutina.
  Atención: lo importante con las variables locales es determinar su desplazamiento con respecto al ebp para poder direccionarlas. Para eso, revise las recomendaciones para determinar el desplazamiento en la pila.

- En cuanto a la salvaguarda de registros, puede necesitarla o no; depende de cómo sea su programa.

  Dado lo anterior, en el epílogo NO son necesarias las acciones correspondientes: ni recuperar el ebp, ni devolver el espacio de las variables locales.
  Atención: TAMPOCO es necesario escribir el ret; ese también lo genera el compilador.

Para determinar los desplazamientos en la pila:

- Se recomienda hacer la traducción a ensamblador en dos pasos: primero traduzca a ensamblador usando nombres simbólicos (de variables y parámetros), compile, ejecute y verifique el resultado. Después de verificar que este primer paso funciona, reemplace los nombres de variables y parámetros por desplazamientos a partir del ebp.
- ANTES de hacer CUALQUIER cambio use la función printf para identificar las direcciones de TODAS las variables locales y la dirección del primer parámetro. Compile y ejecute.
- Con la información impresa por el printf calcule los desplazamientos de TODAS las variables locales (y los parámetros).
- Reemplace TODOS los nombres simbólicos por sus desplazamientos (¡este es un juego de todo o nada!), compile y ejecute. Deje el printf de las direcciones (sirve para determinar las direcciones, pero también para impedir que el compilador elimine las variables locales).
  Atención: puede ocurrir que en diferentes ejecuciones las direcciones cambien, sin embargo, los desplazamientos se mantienen, así que no hay problema con esto ya que nosotros usamos los desplazamientos y no las direcciones.
- Si ejecuta y el resultado no es correcto, recalcule los desplazamientos de los parámetros y las variables locales, reemplace los desplazamientos con los nuevos cálculos (y SOLO eso), compile y ejecute de nuevo.

D. Condiciones de entrega

- En un archivo .zip entregue el código fuente del programa y un informe con el nombre de los integrantes. El nombre del archivo debe ser: caso1_login1_login2.zip
- El trabajo se realiza en grupos de 2 estudiantes. 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).
- En el parcial se incluirá una pregunta sobre el caso.
- El proyecto debe ser entregado por Bloque Neón por uno solo de los integrantes del grupo. **Al comienzo del documento PDF, deben estar los nombres y códigos de estudiante de los integrantes del grupo.** Si un integrante no aparece en el documento entregado, el grupo podrá informarlo posteriormente, sin embargo, habrá una penalización: la calificación asignada será distribuida (dividida de forma equitativa) entre los integrantes del grupo.
- Tenga en cuenta las reglas establecidas en el programa del curso sobre la obligatoriedad del trabajo en grupo.
- **Se debe entregar por Bloque Neón <u>a más tardar</u> el 27 de agosto a las 11:59 p.m.**
- **La fecha de entrega no será movida.**
- **Política de entrega tarde.** Para las entregas tarde, se aplicará la siguiente política: por cada 30 minutos o fracción de retraso, con respecto a la hora de entrega establecida en Bloque Neón, habrá una penalización de 0,5/5.

E. Entregables

- Código en C con el fragmento de código traducido a ensamblador (bitmap_asm.c).
- Incluya al comienzo del programa C modificado un comentario con el nombre y código de los autores de la modificación.
- El programa debe compilar y correr en la máquina virtual (haga las pruebas necesarias antes de la entrega)
- Informe en formato PDF que incluya los nombres de los integrantes del grupo y sus respectivos códigos de estudiante, así como la descripción de la pila del procedimiento traducido a ensamblador. El informe debe estar debidamente estructurado y presentado de manera profesional, evitando entregas apresuradas o carentes de formato.

Cronograma Propuesto

Ago. 14-15: Lectura del enunciado. Definición de los grupos, definición de canales de comunicación entre los integrantes del grupo, identificación de franjas de trabajo disponibles.

Ago. 18-19: Definición de la estrategia de solución. Implementación del primer paso (traducir a ensamblador con nombres simbólicos)

Ago. 20-24: Implementación del segundo paso (traducir a ensamblador con desplazamientos a partir del ebp) + pruebas

Ago. 25-26: Construir el Informe

Ago. 27: Entrega