Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computadores

Proyecto #1

Arquitectura de Computadores I

Profesores:

Luis Chavarría Zamora

Jefferson González Gómez

Por:

Emmanuel Esquivel Chavarría

Abril 13, 2025

Contents

1 Requerimientos del sistema

1.1 Requerimientos Funcionales

Interpolación Bilineal en Arm

El sistema debe incluir algoritmo de interpolacion de imágenes bilineal programacion en ISA ARM. El algoritmo está operando sobre la imagen en bloques definidos por cuadrantes (4x4) y crea píxeles intermedios transiciones suaves entre conocidos valores.

Proceso Exclusivo en Bajo Nivel

Toda la lógica de procesamiento de imagen (lectura, desencriptación,interpolación, escritura) debe existir exclusivamente en código ensamblador. No estará permitido elaborar cualquier cálculo relevante en lenguajes alta.

Selección de Cuadrante

La aplicación ha de incluir la posibilidad de seleccionar una de las 16 áreas de la imagen para hacer la interpolación. Esta selección ha de trasladarse tanto en el proceso de proceso como en la visualización del resultado.

Entrada y Salida a Archivos Binarios

Este conjunto de imágenes deben ser considerado como arreglos enteros de bytes en escala de grises ([0-255]) y leidos/grabados desde archivos binarios (cuadrante.bin y interpolado.bin).

Interfaz Gráfica Lenguaje de Alto Nivel

Aunque es en ensamblador el procesamiento, debe verse el sistema con una interfaz (por ejemplo, en Python) que muestre:

- La imagen original.
- El cuadrante seleccionado (mencionado)
- Y la imagen de interpolada generada por el programa en ensamblador.

1.2 Requisitos Ingenierías y Técnicos

Compatibilidad con arquitecturas ISA RISC ARM, x86 o RISC-V El código debe funcionar correctamente en los simuladores o los ambientes reales compatibles con los conjuntos de instrucciones mencionados. Para este caso se ha optado por ARM.

Estandarización de Tamaños

La imagen de entrada debe de tener un tamaño mínimo de 390x390 pixeles para poder divisarla en cuadrantes de al menos 97x97. Para hacer el desarrollo mas sencillo se usaran tamaños fijos como 500x500 y cuadrantes de 70x70.

Interpolación según Normas de Procesamiento Dígito

La interpolación debe seguir el modelo bilineal clásico descrito en el enunciado del proyecto, el cálculo de los nuevos píxeles se lleva a cabo mediante el método de promedio ponderado de la vecindad de píxeles horizontales y verticales.

Cifrado y Seguridad de Datos

La entrada (cuadrante.bin) debe incluir una simple capa de seguridad: los datos del cuadrante se han cifrado con XOR usando una clave fija. El sistema tiene que desencriptar bien antes de interpolar.

1.3 Requerimientos de Uso y Visualización

Interfaz Gráfica Amigable

La casilla de visualización tiene que dejar al usuario ver claramente el cuadrante seleccionado, y el resultado interpolado. Se recomiendan usar librerías como Tkinter, PIL o Matplotlib.

Marcado Visual del Cuadrante

En la imagen original debe haber una visión clara del cuadrante que se ha seleccionado.

Visualización Comparativa

La salida debe tener lado a lado la imagen original y la imagen interpolitada para la mejor visual comparativa.

1.4 Normas y Buenas Prácticas

Control de Versiones No Opcional (Git)

El desarrollo tiene que estar bien versionado creando la versión con ramas master y development, con commits incrementales y buena estructura.

Estándares de Codificación

El ensamblador del código debe ser clara con los comentarios, modular y no debe tener la replicación innecesaria. Debe estar bien estructurado para acomodar la validación del proceso.

Pruebas de Verificación Visual

La verificación final del sistema replica la verificación en alto nivel (Python) comparando visual el resultado que se obtiene mediante ensamblador con el resultado esperado.

2 Solución al problema

2.1 Opción 1: Procesamiento con binarios y visualización en Python

La primera solución alternativa fue implementada utilizando arquitectura que distingue el procesamiento de bajo nivel, encargado del núcleo computacional, de tareas de alto nivel como la interfaces y la visualisación. Esta opción asocia un flujo de trabajo enfocado en archivos binarios, donde se sigue estrictamente la lógica de procesamiento que se explicó en el enunciado del proyecto, ya que toda la lógica de interpolacion se realiza sólo en el lenguaje de ensamblado ARM sin delegar ningún cálculo relevante a lenguajes de más alto nivel. El proceso comienza con la creación de una imagen original en escala de grises, de 500x500 píxeles. La cual está dividida conceptualmente en 16 porcciones iguales, de 70x70 píxeles. Uno de estos cuadrantes es elegido por el usuario y extraído mediante un script en Python que al mismo tiempo cifra su contenido con una operación XOR con una llave fija. El cuadrante resultante, ya encriptado, se guarda en un archivo llamado `cuadrante.bin` que será la única entrada que procesará el código en Arm. Una vez que el archivo `cuadrante.bin` está preparado, se dispone del programa ejecutado en ensamblador ARM que posee varios módulos muy bien definidos. Primero se realiza la desencriptación del contenido aplicando nuevamente la operación XOR, recupera los valores originales de la imagen. Posterior a esto y con la ayuda de las fórmulas para el calculo de los nuevos valores de la matriz, el programa ejecuta la Interpolación bilineal sobre ese bloque creando nuevos valores para píxeles intermedios como una función de píxeles vecinos en las filas y columnas. Esta lógica de interpolación se ajusta al modelo clásico basado en el promedio ponderado de intensidades y necesita de control exacto sobre la aritmética, los desplazamientos en la memoria, manejo de registros en ARM. Finalmente, los nuevos datos generados se almacenan secuencialmente en otro archivo binario denominado `interpolado.bin`, el cual servirá como salida para el proceso de visualización.

Para completar el ciclo un script en Python lee la imagen original además del archivos `interpolado.bin`. Este script lo que hace es dibujar de forma gráfica el resultado para el usuario con una superposición visual que resalta el cuadrante seleccionado (con un recuadro rojo) y junto a ella, la imagen interpolada creada en ensamblador. Gracias a librerías como

PIL, Matplotlib, NumPy, este paso se srealiza de forma super sencilla. modo confortable y con finales precisos y estéticos. Además, un script en Bash automatiza todo el flujo, permitiendo que con una sola ejecución se procese la imagen, se ejecute el programa en ensamblador y se despliegue el resultado, minimizando la intervención manual, todo esto por medio de la ejecución en terminal.

Esta solución destaca por su rendimiento y su capacidad de escalamiento. Además, se adapta bien a prácticas automatizadas, facilita la integración de nuevas funciones y mantiene una separación muy marcada entre procesamiento y visualización. Su único problema radica en que el trabajo con archivos binarios dificulta la inspección directa de los datos intermedios, y cualquier error en el manejo de memoria o offsets puede resultar en comportamientos erráticos difíciles de depurar sin herramientas adicionales.

A continuación se ilustra el esquema general de esta solución:

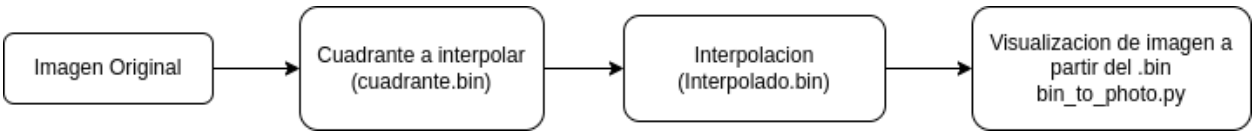

Figure 1: Diagrama de Bloques Solución Planteada

2.2 Opción 2: Archivos de texto plano y entorno Octave

La segunda opción plantea otro enfoque, el cual está basado en la transparencia de los datos y la facilidad de validación, especialmente útil en entornos educativos o durante el proceso de desarrollo temprano del proyecto. Distinto al caso anterior, en este se evita el uso de archivos binarios, muestra imágenes y sus transformaciones como matrices de números en archivos .txt, manejables manualmente. Este cambio formativo no sólo no modifica solo el flujo técnico de la solución, sino que también permite considerarlo de otra manera. Esta opción nos permite tener controlado y visible el flujo de datos a lo largo de todo su funcionamiento, aunque esta otra alternativa resulta más ineficiente. El procesamiento comienza con un script en Octave que carga una imagen y la convierte a una matriz de escala de grises. Esta imagen se redimensiona a dimensiones compatibles con el sistema, y se almacena. se lee como un archivo de texto plano (imagen.txt), donde una fila representa una linea de píxeles, y cada cifra es la intensidad luminosa de un píxel. Paralelamente, el usuario se indica el cuadrante a procesar mediante un archivo de texto, comúnmente llamado indice.txt, que almacena la posición numérica del bloque que estamos buscando.

Posteriormente, el programa ensamblador ARM lee los archivos de texto. Para esto, convierte texto ASCII en números enteros, lo cual hace que toda la lógica en bajo nivel sea mucho más completa en términos de implementacion. Una vez cargados los datos, el algoritmo de interpolación se aplica de la misma manera en que se hizo en la otra alternativa (esto se mantiene igual), generando una nueva matriz con los valores resultantes. Esta matriz se guarda también en formato `.txt`, lo que permite que el archivo pueda ser inspeccionado visualmente, línea por línea, esto con el fin de poder tener un mayor control de los datos.

Finalmente, un script en Octave elabora este archivo de salida y genera una visualización comparativa. Se muestran en la misma línea la imagen original, el cuadrante seleccionado y la versión interpolada. Como todos los datos están en formato texto, se pueden llevar a cabo pruebas unitarias específicas para checkear que el algoritmo funciona correctamente y, verdad, con matrices pequeñas con valores que se saben.

Este enfoque es en realidad muy útil para depurar y para validad el algoritmo paso a paso. Asimismo, haciendo uso de software gratuito como es el caso de Octave, se asegura

una gran portabilidad y accesibilidad. El procesamiento de archivos de texto es más lento y requiere más espacio en disco. Además, la lógica necesaria para interpretar estos datos en ensamblador es más extensa y susceptible a errores. En términos de eficiencia y rendimiento del progama a largo plazo, esta solución no es la más adecuada para flujos en tiempo real o imágenes de alta resolución.

A continuación se ilustra el esquema general de esta solución:

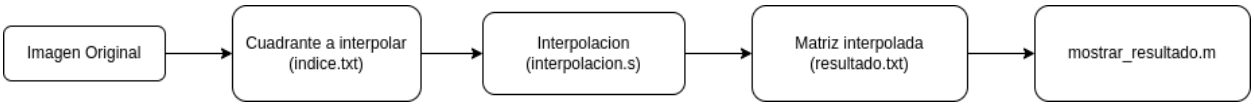

Figure 2: Diagrama de bloques de solución planteada

3 Comparación de opciones de solución

Ambas soluciones cumplen con el objetivo principal del proyecto, el cual es aplicar el algoritmo de interpolación bilineal en ensamblador ARM, respetando lo la restriccion que solicita el enunciado de realizar el procesamiento exclusivamente en bajo nivel. Sin embargo, cada enfoque presenta características distintas en términos de manejo de datos, facilidad de validación y adaptabilidad al entorno de trabajo. A continuación se detallan las ventajas y desventajas de cada una:

Opción 1

Ventajas:

- Eficiencia en espacio y velocidad: El uso de archivos .bin nos permite manejar volúmenes de datos grandes (imágenes completas) de forma rápida y compacta.
- Automatización fluida: Python, junto con Bash (proyecto_1.sh), facilita flujos automatizados sin intervención manual.
- Escalabilidad: Es fácil extender el sistema para procesar múltiples imágenes o realizar filtros adicionales si se desea.

Desventajas:

- Difícil trazabilidad del contenido: Los archivos binarios no son legibles directamente, lo que complica la depuración del contenido interno.
- Más propenso a errores en offsets: Cualquier mal cálculo en desplazamientos binarios puede generar errores silenciosos difíciles de detectar.

Opción 2

Ventajas:

- Legibilidad absoluta de datos: Todos los archivos intermedios son .txt, lo que permite inspeccionar manualmente cada matriz en cualquier editor.
- Facilidad en depuración: Ante un error, se puede revisar línea por línea la entrada y salida sin necesidad de decodificadores o binarios.
- Facilidad para pruebas de unidades: Se puede crear matrices pequeñas de prueba con valores específicos para validar regiones o cuadrantes concretos.

Desventajas:

- Archivos más grandes y lentos: Las matrices en texto ocupan más espacio y tardan más en cargarse y procesarse, especialmente en ensamblador.
- Mayor complejidad en el parser: Leer texto línea por línea y convertirlo a valores numéricos en ASM es más engorroso que cargar directamente datos binarios.
- No óptimo para producción o escalabilidad: El formato de texto no es adecuado para flujos en tiempo real o para imágenes de alta resolución.

4 Selección de la propuesta final

Con base en la valoración anterior, se decidió mantener la **Opción 1**. Esta elección se fundamentó en:

- La facilidad de validación del resultado mediante comparación visual.
- El fácil acoplamiento entre el archivo de entrada (cuadrante.bin) y la estructura interna de datos manejada por el ensamblador.
- La posibilidad de estructurar el trabajo como un sistema automatizado, lo cual es esencial para garantizar repetibilidad y facilitar la defensa del proyecto.

Otro factor muy importante es que esta estructura permite concentrar el esfuerzo de desarrollo en la implementación del algoritmo de interpolación como tal y no tanto en detalles menos significantes como formatos de imagen o interfaces de usuario, lo cual se alinea con los objetivos de aprendizaje del curso.