2026

Practica No 1.0
“edición de imágenes”

Contenido

Oscar D. De Jesús lucio

Análisis de Formatos de Codificación y Almacenamiento (BMP).

7-2-2026

Profesor; Sierra Romero Noe

Grupo; 3TM2

Objetivo

Analizar la estructura interna de un archivo de imagen de mapa de bits (BMP) para comprender sus formatos de codificación y almacenamiento, identificando la diferencia entre metadata (cabecera) y data (píxeles), tal como se establece en los contenidos de Multimedia Clásica.

Justificación

Al usar hexed.it para ver los bytes, no solo se está "viendo" la imagen; se está analizando su formato de codificación (cómo los bits representan colores) y su almacenamiento (cómo se estructura el archivo con cabeceras y datos).

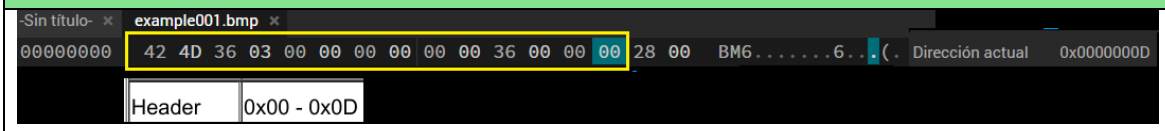
Procedimiento

1. **Preparación del Entorno de Análisis** Se accedió a la herramienta de edición hexadecimal basada en web **HexEd.it** para realizar la inspección de bajo nivel (byte a byte). Se procedió a la carga del archivo example001.bmp, asegurando que la visualización estuviera configurada en modo **Little-Endian** para la correcta lectura de enteros de 16 y 32 bits, conforme a la arquitectura estándar de archivos Windows.
2. **Identificación y Validación de la Estructura (Header)** Basándose en la especificación técnica del formato BMP (referencia: *The BMP File Format*), se localizaron los offsets iniciales para validar la integridad del archivo:
    2.1. **Firma del Archivo (Offset 0x00):** Se verificó la existencia de la secuencia ASCII BM (42 4D en hex), confirmando el tipo de fichero.
    2.2. **Cálculo de Tamaño (Offset 0x02):** Se extrajo la secuencia 36 03 00 00. Aplicando la conversión Little-Endian (00 00 03 36), se obtuvo el valor decimal **822 bytes**, el cual se contrastó exitosamente con las propiedades del archivo en el sistema operativo.
3. **Extracción de Metadatos (Info Header)** Se procedió a decodificar el bloque de información de la imagen (DIB Header) para obtener las dimensiones lógicas:
    3.1. **Ancho y Alto (Offsets 0x12 y 0x16):** Se interpretaron los valores hexadecimales 10 00 00 00 como **16 píxeles**, definiendo la matriz de la imagen como 16x16.
    3.2. **Profundidad de Color (Offset 0x1C):** Se identificó el valor 18 00 (24 decimal), estableciendo que la codificación de color es **True Color (24 bits)**, sin tabla de colores indexada.
4. **Segmentación del Mapa de Bits (Pixel Data Analysis)**
    Utilizando el valor del *Data Offset* encontrado en la posición 0x0A (36 00 00 00 → 54 bytes), se ubicó el inicio de la carga útil (payload) de la imagen.
    4.1. **Cálculo del Scanline:** Se determinó teóricamente el tamaño de cada fila mediante la fórmula: $Ancho (16) \times Bytes/Pixel (3) = 48 bytes$.
    4.2. **Inspección de Patrones:** Se aislaron los bloques de 48 bytes en el editor para correlacionarlos con las filas visuales de la imagen.
5. **Decodificación de Color y Verificación Visual** Se analizó la trama de datos siguiendo el estándar BMP (orden **BGR** y almacenamiento **Bottom-Up**):
    5.1. Se identificaron secuencias hexadecimales repetitivas (ej. FF 00 00) y se tradujeron a valores RGB decimales (ej. Azul: 0, 0, 255) para cotejar con los "cuadros" visuales de la imagen original.
    5.2. Se confirmó que la primera línea de datos en el archivo correspondía a la fila inferior de la imagen renderizada.
6. **Comprobación de Integridad Final** Se realizó una sumatoria de los componentes: *Tamaño del Header (14 bytes) + InfoHeader (40 bytes) + Datos de Imagen (16 filas $\times$ 48 bytes)*. El resultado (822 bytes) coincidió con el tamaño total del archivo, validando que no existen datos ocultos ni corrupción en la estructura.

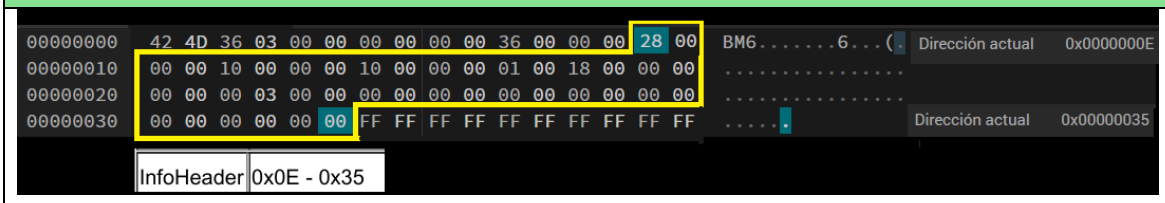
Evidencias del desarrollo de la practica

Explicación desde panorama general

Header --- contains information about the type, size, and layout of a device independent bitmap file

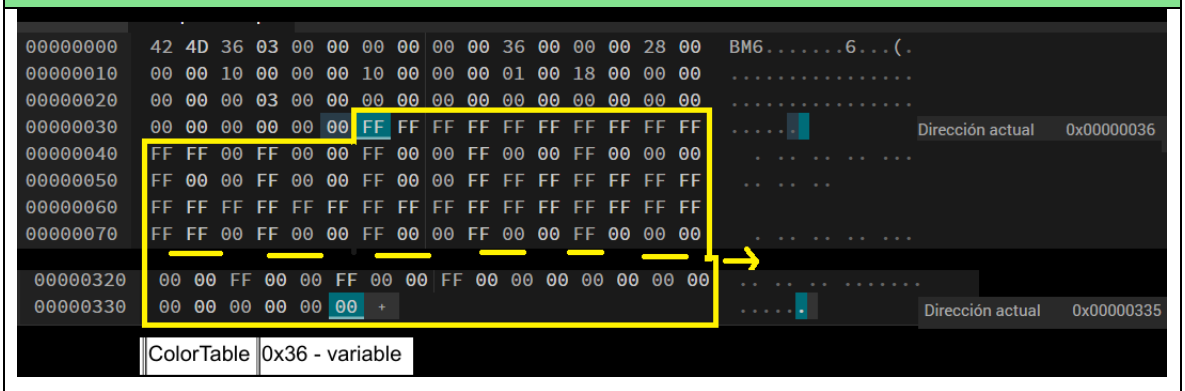


InfoHeader --- specifies the dimensions, compression type, and color format for the bitmap



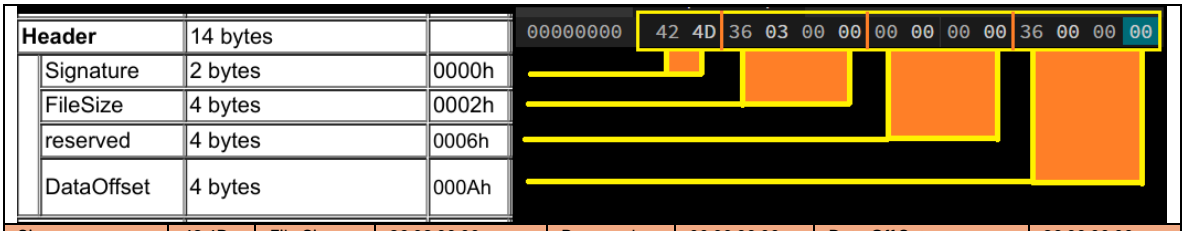
ColorTable -- contains as many elements as there are colors in the bitmap, but is not present for bitmaps with 24 color bits because each pixel is represented by 24-bit red-green-blue (RGB) values in the actual bitmap data area

Pixel Data --- an array of bytes that defines the bitmap bits. These are the actual image data, represented by consecutive rows, or "scan lines," of the bitmap. Each scan line consists of consecutive bytes representing the pixels in the scan line, in left-to-right order. The system maps pixels beginning with the bottom scan line of the rectangular region and ending with the top scan line



Explicación detallada de cada sección

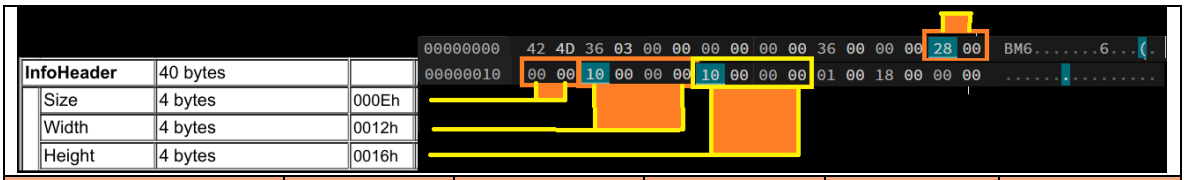
Header



Header	14 bytes	
Signature	2 bytes	0000h
FileSize	4 bytes	0002h
reserved	4 bytes	0006h
DataOffset	4 bytes	000Ah

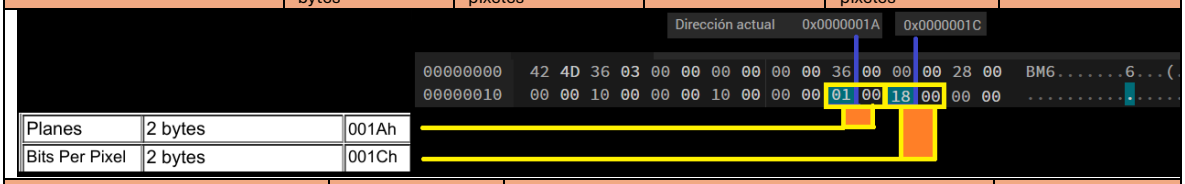
Signature	42 4D	File Size	36 03 00 00	Reserved	00 00 00 00	Data Off Set	36 00 00 00
'BM'	B M	File size in bytes	00000336 = 822 bytes	unused (=0)	0 bytes	Offset from beginning of file to the beginning of the bitmap data	00000036 = 54 empieza archivo

Info Header



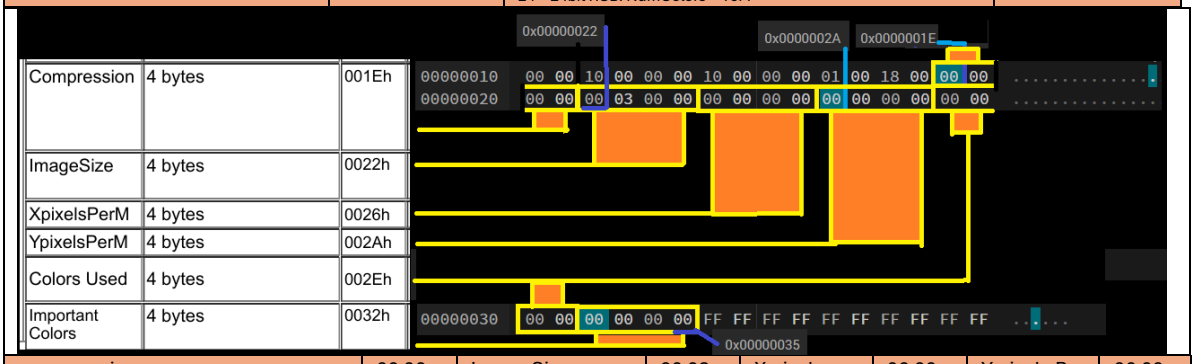
InfoHeader	40 bytes	
Size	4 bytes	000Eh
Width	4 bytes	0012h
Height	4 bytes	0016h

Size	28 00 00 00	Width	10 00 00 00	Height	10 00 00 00
Size of InfoHeader =40	00 00 00 28 = 40 bytes	Ancho horizontal en pixeles	00 00 00 10 = 16 px	Alto vertical en pixeles	00 00 00 10 = 16 px



Planes	2 bytes	001Ah
Bits Per Pixel	2 bytes	001Ch

Planes	01 00	Bits Per pixel	18 00
Number of Planes (=1)	00 01 = 1 bytes	Bits per Pixel used to store palette entry information. This also identifies in an indirect way the number of possible colors. Possible values are: 1 = monochrome palette. NumColors = 1 4 = 4bit palletized. NumColors = 16 8 = 8bit palletized. NumColors = 256 16 = 16bit RGB. NumColors = 65536 24 = 24bit RGB. NumColors = 16M	00 18 = 24 bytes



Compression	4 bytes	001Eh
ImageSize	4 bytes	0022h
XpixelsPerM	4 bytes	0026h
YpixelsPerM	4 bytes	002Ah
Colors Used	4 bytes	002Eh
Important Colors	4 bytes	0032h

compression	00 00 00 00	Image Size	00 03 00 00	X pixels Per M	00 00 00 00	Y pixels Per M	00 00 00 00
Type of Compression 0 = BI_RGB no compression 1 = BI_RLE8 8bit RLE encoding 2 = BI_RLE4 4bit RLE encoding		(compressed) Size of Image It is valid to set this =0 if Compression = 0	00 00 03 00 =	horizontal resolution: Pixels/meter		vertical resolution: Pixels/meter	

Color Used	00 00 00 00	Important Colors	00 00 00 00
Number of actually used colors. For a 8-bit / pixel bitmap this will be 100h or 256.		Number of important colors 0 = all	

Conclusión

Mediante la realización de esta práctica, se logró desmitificar el concepto de "archivo multimedia", comprendiéndolo no como una unidad abstracta, sino como una **sucesión estructurada de bytes**. Se comprobó experimentalmente que la **metadata** (cabeceras) actúa como el "contrato" de interpretación: sin las instrucciones correctas de ancho, alto y profundidad de color contenidas en los primeros 54 bytes, la **data** (payload) carece de sentido y no puede ser reconstruida visualmente.

Si bien la visualización en un editor hexadecimal no sustituye a las herramientas de diseño para la creación artística debido a su complejidad, se demostró que es una herramienta insustituible para el **análisis forense digital** y la **ingeniería de datos**. Permite una auditoría completa de la integridad del archivo y revela atributos que los editores convencionales ocultan (como el espacio de relleno o *padding*).

Finalmente, desde la perspectiva de la **Ingeniería Telemática**, visualizar la "trama" cruda de la información es fundamental. Al entender que una imagen BMP no es más que un flujo de bits sin comprimir con alta redundancia, podemos dimensionar mejor los retos que implica su transmisión. Esto valida la necesidad de algoritmos de compresión (como en JPEG) para optimizar el uso de memoria y ancho de banda en canales de comunicación limitados.

Bibliografía

Documentación del Formato: Liesch, N. (s.f.). *THE BMP FILE FORMAT* (Compiled by Nathan Liesch of Imperium Accelero 9000) [Archivo PDF]..

Herramienta de Software: HexEd.it. (2025). *HexEd.it - Client-side Hex Editor for the browser*. Recuperado de https://hexed.it

Referencia del Estándar (Opcional pero recomendado): Microsoft. (s.f.). *Bitmap Storage - Win32 apps*. Microsoft Learn. Recuperado de https://learn.microsoft.com/en-us/windows/win32/gdi/bitmap-storage

Contexto Académico: Instituto Politécnico Nacional. (2012). *Programa Sintético: Multimedia* (Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas)

ANEXOS

Color table

ColorTable		4 * NumColors bytes	0036h	present only if Info.BitsPerPixel less than 8 colors should be ordered by importance
	Red	1 byte		Red intensity
	Green	1 byte		Green intensity
	Blue	1 byte		Blue intensity
	reserved	1 byte		unused (=0)
	repeated NumColors times			
Pixel Data		InfoHeader.ImageSize bytes		The image data

Paleta de Colores usada

Color Visual	Muestra	Valor Hex en Archivo (BGR)	Valor Decimal (R, G, B)	Tamaño
Blanco		`FF FF FF`	(255, 255, 255)	3 bytes (24 bits)
Negro		`00 00 00`	(0, 0, 0)	3 bytes (24 bits)
Rojo		`00 00 FF`	(255, 0, 0)	3 bytes (24 bits)
Verde		`00 FF 00`	(0, 255, 0)	3 bytes (24 bits)
Azul		`FF 00 00`	(0, 0, 255)	3 bytes (24 bits)

Conceptos necesarios, apéndice

Archivo BMP; Bitmap (Mapa de bits) o DIB (Device Independent Bitmap).

Nacimiento (1987): Fue creado por Microsoft e IBM juntos. Lo introdujeron con Windows 2.0 y el sistema operativo OS/2.

El Problema que resolvió: En los años 80, había muchas tarjetas gráficas distintas (CGA, EGA, Hercules, VGA) y cada una manejaba el color de forma diferente. Si hacías un dibujo en una, se veía mal en otra. El formato BMP (específicamente la versión DIB) se inventó como un "traductor universal" para que una imagen se pudiera ver igual en cualquier monitor, independientemente del hardware.

Por qué sigue vivo: Aunque es viejo y pesado (porque casi no usa compresión, como notaste con el tamaño de archivo grande para tan pocos píxeles), es el formato más simple de leer para un programador. Por eso se sigue usando en prácticas de ingeniería como la tuya: no necesitas librerías complejas para decodificarlo, solo leer los bytes directamente.

Orden BGR los colores no se guardan como RGB, sino al revés; azul – verde - rojo

Bottom-up el primer archivo que se lee corresponde a la esquina inferior izquierda

METADATA es información que sirve de instructivo, la manera en como será tomada y tratada la información que le sigue

DATA La Data carece de estructura semántica propia; sin los parámetros del Header, es imposible determinar las dimensiones o profundidad de color para reconstruir la imagen.

Conversión de hexadecimal a decimal "para dos dígitos ab"

$$(a * 16) + b = "\#\ decimal"$$

Un par (2 dígitos hex) = 1 Byte (8 bits).

Definiciones del Volcado Hexadecimal Canónico (Canonical Hex Dump).

Origen (Años 70): Este formato específico nació con los primeros sistemas operativos para microcomputadoras, como CP/M (creado por Gary Kildall en 1974). El comando DUMP mostraba los archivos exactamente así para que los programadores pudieran depurar errores. El Estándar de los 80: Se popularizó masivamente con MS-DOS y su famosa herramienta DEBUG.EXE. ¿Por qué ese diseño? Se diseñó así para ajustarse perfectamente a las pantallas de 80 columnas (el estándar de los monitores de fósforo verde y tarjetas perforadas de la época).

"Estamos visualizando el archivo mediante un Volcado Hexadecimal (Hex Dump), una representación estándar heredada de herramientas de depuración de los años 70 como DEBUG de MS-DOS. Esto nos permite inspeccionar la estructura interna del formato BMP, creado por Microsoft en 1987 para Windows 2.0 con el objetivo de intercambiar gráficos entre diferentes dispositivos de hardware sin perder calidad."