Header	14 bytes		Windows Structure: BITMAPFILEHEADER

Signature	2 bytes	0000h	'BM'

```
42 4D 36 03 00 00 00 00 00 00 36 00 00 00 28 00   BM
```

Se toman los primeros dos del ejemplo de la imagen y se puede observar que son los 2 bytes que deben abarcar.

FileSize	4 bytes	0002h	File size in bytes

```
42 4D 36 03 00 00 00 00 00 00 36 00 00 00 28 00   BM6...
```

En este apartado se seleccionan 4 elementos, pero se aplica el de invertir el hexadecimal y nos quedaría 336 si eso se pasa a decimal nos da 822 bytes que es lo que pesa el documento.

reserved	4 bytes	0006h	unused (=0)

```
42 4D 36 03 00 00 00 00 00 00 36 00 00 00 28 00   BM6.....
```

Como es el reservado su valor va a ser de 0.

DataOffset	4 bytes	000Ah	Offset from beginning of file to the beginning of the bitmap data

```
42 4D 36 03 00 00 00 00 00 00 36 00 00 00 28 00   BM6.......6...
```

Aquí igual aplicamos la de invertir el numero hexadecimal que nos quedaría 36 y eso nos da 54.

InfoHeader	40 bytes		Windows Structure: BITMAPINFOHEADER

Size	4 bytes	000Eh	Size of InfoHeader =40

```
42 4D 36 03 00 00 00 00 00 00 36 00 00 00 28 00   BM6.......6...(.
00 00 10 00 00 00 10 00 00 00 01 00 18 00 00 00   ................
```

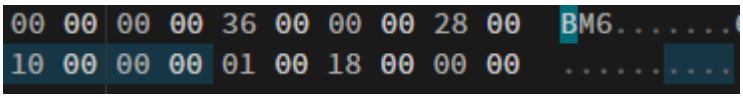
Se aplica el mismo método para que nos queden 28 hexadecimales y en decimal eso nos da 40.

Width	4 bytes	0012h	Horizontal width of bitmap in pixels

```
42 4D 36 03 00 00 00 00 00 00 36 00 00 00 28 00   BM6...
00 00 10 00 00 00 10 00 00 00 01 00 18 00 00 00   ......
```

Pasamos 10 hexadecimales a decimal dando un resultado de 16

Height	4 bytes	0016h	Vertical height of bitmap in pixels

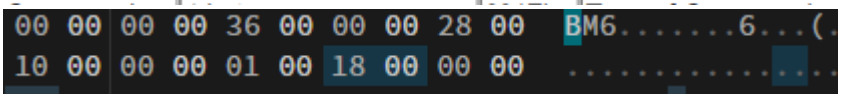


Al igual que en el anterior nos da 16

Planes	2 bytes	001Ah	Number of Planes (=1)

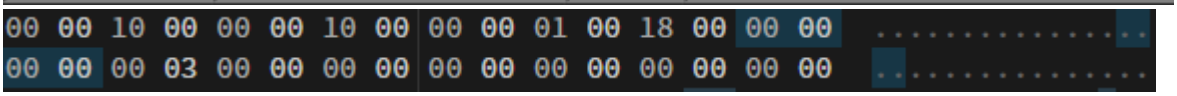
Transformando en este caso nos da un total de 1

Bits Per Pixel	2 bytes	001Ch	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



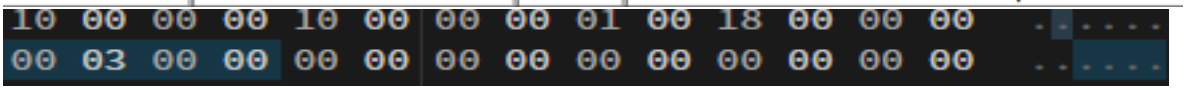
Al convertir el número hexadecimal nos da un resultado de 24 que en este caso sería la última opción

Compression	4 bytes	001Eh	Type of Compression 0 = BI_RGB no compression 1 = BI_RLE8 8bit RLE encoding 2 = BI_RLE4 4bit RLE encoding



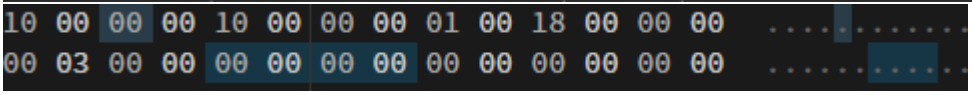
Se observa que son ceros entonces el archivo no tiene compresión.

ImageSize	4 bytes	0022h	(compressed) Size of Image It is valid to set this =0 if Compression = 0



En la imagen se dice que si hay comprensión el valor dará o es valor que se le asigna es 0 y es lo que está pasando

XpixelsPerM	4 bytes	0026h	horizontal resolution: Pixels/meter



Aquí se ve que la resolución horizontal da 0

YpixelsPerM	4 bytes	002Ah	vertical resolution: Pixels/meter

```
10 00 00 00 10 00 00 00 01 00 18 00 00 00
00 03 00 00 00 00 00 00 00 00 00 00 00 00
```

Se puede ver que para la verificación da un total de 0

Colors Used	4 bytes	002Eh	Number of actually used colors. For a 8-bit / pixel bitmap this will be 100h or 256.

```
00 00 00 03 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 FF FF FF FF FF FF FF FF FF FF
```

Aquí en los colores usados nos dicen que es 0

Important Colors	4 bytes	0032h	Number of important colors 0 = all

```
00 00 00 00 00 00 FF FF FF FF FF FF FF FF FF FF
```

Aquí se muestra que todos los bloques son importantes

ColorTable	4 * NumColors bytes	0036h	present only if Info.BitsPerPixel less than 8 colors should be ordered by importance

```
00 00 00 00 00 00 FF FF FF FF FF FF FF FF FF FF
FF FF 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 00
```

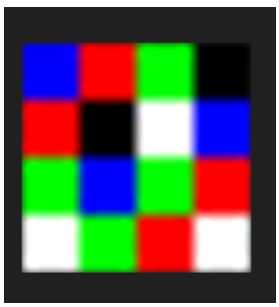
Se toman los primeros hexadecimales y viendo el código nos damos cuenta del color blanco

```
00 00 00 00 00 00 FF FF FF FF FF FF FF FF FF FF
FF FF 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 00
```

Cuando se toman los otros hexadecimales no damos cuenta que es el color verde

```
FF FF 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 00
FF 00 00 FF 00 00 FF 00 00 FF FF FF FF FF FF FF
```

Cuando se toman los otros hexadecimales no damos cuenta que es el color azul



```
FF FF FF FF FF FF 00 FF 00 00 FF 00 00 FF 00 00
FF 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 00 FF
```

En esta sección se encuentra el color rojo

```
00 00 FF 00 00 FF 00 00 FF 00 00 00 FF 00 00 FF
00 00 FF 00 00 FF 00 FF 00 00 FF 00 00 FF 00 00
FF 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 00 FF
00 00 FF 00 00 FF 00 00 FF 00 00 00 FF 00 00 FF
00 00 FF 00 00 FF 00 FF 00 00 FF 00 00 FF 00 00
FF 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 00 FF
00 00 FF 00 00 FF 00 00 FF 00 00 00 FF 00 00 FF
00 00 FF 00 00 FF 00 FF 00 00 FF 00 00 FF 00 00
FF 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 00 FF
00 00 FF 00 00 FF 00 00 FF 00 00 00 FF 00 00 FF
00 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 FF 00
00 FF 00 00 00 00 00 00 00 00 00 00 00 00 FF FF
FF FF FF FF FF FF FF FF FF FF FF 00 00 FF 00 00
FF 00 00 FF 00 00 00 00 FF 00 00 FF 00 00 FF 00
00 FF 00 00 00 00 00 00 00 00 00 00 00 00 FF FF
FF FF FF FF FF FF FF FF FF FF FF 00 00 FF 00 00
FF 00 00 FF 00 00 00 00 FF 00 00 FF 00 00 FF 00
00 FF 00 00 00 00 00 00 00 00 00 00 00 00 FF FF
FF FF FF FF FF FF FF FF FF FF FF 00 00 FF 00 00
FF 00 00 FF 00 00 00 00 FF 00 00 FF 00 00 FF 00
00 FF 00 00 00 00 00 00 00 00 00 00 00 00 FF FF
FF FF FF FF FF FF FF FF FF FF FF 00 00 FF 00 00
FF 00 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 FF
00 00 00 00 FF 00 00 FF 00 00 FF 00 00 FF 00 FF
00 00 FF 00 00 FF 00 00 FF 00 00 00 00 00 00 00
00 00 00 00 00 00 FF 00 00 FF 00 00 FF 00 00 FF
00 00 00 00 FF 00 00 FF 00 00 FF 00 00 FF 00 FF
00 00 FF 00 00 FF 00 00 FF 00 00 00 00 00 00 00
00 00 00 00 00 00 FF 00 00 FF 00 00 FF 00 00 FF
00 00 00 00 FF 00 00 FF 00 00 FF 00 00 FF 00 FF
00 00 FF 00 00 FF 00 00 FF 00 00 00 00 00 00 00
00 00 00 00 00 00 FF 00 00 FF 00 00 FF 00 00 FF
00 00 00 00 FF 00 00 FF 00 00 FF 00 00 FF 00 FF
00 00 FF 00 00 FF 00 00 FF 00 00 00 00 00 00 00
00 00 00 00 00 00 +
```

Se van repitiendo los códigos hexadecimales de los colores previamente encontrados hasta que se llega al último que representa el color negro.